DESIGN AND DEVELOPMENT OF A BIOINFORMATICS PLATFORM FOR LARGE-SCALE GENE EXPRESSION PROFILING

MICHAEL MAURER

DOCTORAL THESIS

Graz University of Technology
Institute for Genomics and Bioinformatics
Petersgasse 14, 8010 Graz, Austria

Graz, October 2004
Abstract

The identification and functional annotation of genes involved in the development and progression of complex diseases is a difficult and non trivial task. DNA microarrays allow generating a composite picture of the expression profile of the cell and are widely used in basic research as well as in clinical medicine and pharmacogenomics. However, to the best of our knowledge there is no freely available bioinformatics platform for multi color microarray experiments allowing an efficient storage and platform and application independent analysis.

We have developed a bioinformatics platform based on the Java 2 Enterprise Edition platform called MARS (Microarray Analysis and Retrieval System) that is MIAME (Minimum Information About a Microarray Experiment) compliant and fully integratable into the microarray workflow. The web based user interfaces allow to store information regarding the sample preparation, RNA extraction, labeling and hybridization procedure, to upload array designs, hybridization images, raw and normalized datasets, and to define experiments where the experimental design of a set of hybridizations can be annotated. The SOAP web service and the External Application Connector Interface (EACI) provide mighty interfaces for the integration of external or third party applications. An additional web application MARS-QM which is connected to MARS using the EACI was initiated. MARS-QM integrates several quality measurements performed during the microarray production as well as during the sample preparation, extraction, and hybridization process to assure high-quality data, to understand or optimize lower value data and to prove the concept of the EACI. Finally, the ability to export all the gained data into MAGE-ML allows to share the valuable results with other researchers.

Additionally to understand the function of the cell and to gain knowledge about biological pathways, their components, and the interaction between the components we have developed an interactive drawing tool called PathwayMapper that allows to model pathways, to import them from other pathway databased and map gene expression data form microarray experiments onto available pathways.

Finally, in order to avoid unauthorized access in a multi-user environment and to control user access we have developed an Authentication and Authorization System (AAS) which provides one username-password combination for all applications and allows single sign-on to increase usability. The developed AAS affords to manage authorization and authentication for any number of applications.

Summarizing, the MARS database design, state-of-the-art software technology, well designed user interface, and its powerful application interfaces provide a capable tool for storing, retrieving and analyzing multi color microarray data. The unique affiliation of using web-based and standalone applications connected to the latest powerful application server technology facilitate MARS users to transform microarray data into valuable knowledge.

Keywords: microarray database, multi color microarray experiments, MIAME, MAGE-ML, J2EE
Publications

This thesis was based on the following publications, as well as upon unpublished observations:

Papers


Book Chapters

Alexander Sturn, Michael Maurer, Robert Molidor, and Zlatko Trajanoski. Systems for Management of Pharmacogenomic Information. Pharmacogenomics Methods and Protocols Humana Press, Totowa, USA 2004 in press

Conference Proceedings and Abstracts

Maurer M, Molidor R, Sturn A and Trajanoski Z. MARS: Microarray Analysis and Retrieval System, 6th International Meeting of the Microarray Gene Expression Data Society (MGED6), Aix en Provence, France 2003


# List of Figures

1. Microarray Technology ................................................................. 7
2. Multi-tiered Applications Architecture .......................................... 13
3. J2EE Three Tier Architecture ......................................................... 14
4. The Model-View-Controller Architecture ......................................... 19
5. Struts Workflow Diagram ................................................................. 20
6. Session Facade Sequence Diagram ..................................................... 24
7. MAGE Object Model ........................................................................ 30
8. Functional Genomics Environment ..................................................... 33
9. Microarray Workflow ....................................................................... 35
10. MARS User Interface ...................................................................... 36
11. MARS Sample Annotations and Extract Manipulation Pages ................. 37
12. MARS Hybridization User Interface .................................................. 38
13. MARS Experimental Design Interface ............................................... 38
14. MARS Raw Dataset Page .................................................................. 39
15. MARS External Application Connector Interface Example ................. 42
16. MARS MAGE-ML Export Possibility ............................................... 44
17. MARS-QM User Interface ................................................................. 44
18. PathwayMapper User Interface .......................................................... 46
19. Authentication Process ..................................................................... 48
20. Quality of the microarray data .......................................................... 52
21. GO distribution for physiological process, metabolism genes ................ 54
22. Androgen and Estrogen Metabolism Pathway ..................................... 55
List of Tables

1. Different Types of Enterprise Java Beans ............................................. 15
2. Packages of the MAGE Object Model ..................................................... 29
3. Classes of genes regulated more than 40% in at least one class. ................. 53
4. Genes mapped to specific Pathways ....................................................... 57

Listings

1. Sample code: External Application Connector ............................................ 43
2. Sample code: Client Connector Authentication ............................................ 49
# Contents

1 Introduction ........................................... 1  
1.1 Background ........................................... 1  
1.2 Objectives ........................................... 4  
2 Methods ............................................... 6  
2.1 Microarray Technology ................................. 6  
2.2 The Java 2 Enterprise Edition (J2EE) .................... 12  
2.3 Relational Databases .................................. 25  
2.4 User Authentication and Authorization ................... 26  
2.5 Web Services ......................................... 27  
2.6 Microarray Gene Expression Markup Language ............ 27  
2.7 MGED Ontology ....................................... 30  
3 Results ............................................... 32  
3.1 Overview ............................................ 32  
3.2 MARS ................................................ 34  
3.3 MARS-QM ........................................... 42  
3.4 GOLD.db ............................................ 45  
3.5 Authentication and Authorization System .................. 47  
3.6 Transcription profiling of NCI-H295R cells treated with fat cell-conditioned medium .................. 51  
4 Discussion ............................................. 58  
5 Conclusion and Outlook .................................. 62  
References .............................................. 63  
Glossary ............................................... 71  
Acknowledgement ......................................... 74  
Publications .......................................... 75
1 Introduction

1.1 Background

The identification of candidate genes for complex diseases such as diabetes mellitus, mental and neurological disorders, or infectious diseases has been fairly unsuccessful so far. These diseases have been ranked by the World Health Organization as the leading causes of disability worldwide. For example, twenty-five percent of individuals develop one or more mental or behavioral disorders at some stage in life, in both developed and developing countries. Other recently compiled data show that approximately 150 million people have diabetes mellitus worldwide, and that this number may well double by the year 2025 [1]. Much of this increase will occur in developing countries and will be due to population growth, aging, unhealthy diets, obesity, and sedentary lifestyles. Some of the reasons for the failure of current approaches to identify biochemical markers have been small sample sizes and poorly defined phenotypes. Genotyping errors in the data have also been blamed, to a lesser extent. The task, to identify a few genes which cause a disease, within the human genome, from approximately 30,000 [2, 3] genes, is a difficult one. It is made even more difficult by the, virtually totally unknown, complexity of the diseases in question, so that modelling of the disease becomes impossible, blunting the weapons of statisticians. Methods that are less model dependent, such as sibpair studies, have also failed [1].

In the past decade bioinformatics has become an integral part of research and development in biomedical and biological sciences. When in the early 1980s methods for DNA sequencing became widely available, molecular sequence data expeditiously started to grow exponentially. After the sequencing of the first microbial genome in 1995, the genomes of more than 100 organisms have been sequenced and large-scale genome sequencing projects have evolved to routine, though still non-trivial, procedures [4, 5]. The imperative of efficient and powerful tools and databases became obvious during the realization of the human genome project, whose completion has been established several years ahead of schedule. The accumulated data was stored in the first genomic databases such as GenBank [6], European Molecular Biology Laboratory Nucleotide Sequence Database (EMBL) [7], and DNA Data Bank of Japan (DDBJ) [8]. For further analysis of the collected data novel computational methods such as sequence similarity searches and functional and structural predictions had to be developed. One of the first breakthroughs in the area of bioinformatics was the introduction of the rapid sequence database search tool BLAST [9], which nowadays has become a valuable and indispensable tool in the
everyday life of biomedical research. The advent of the genome project has vastly increased our knowledge of the genomic sequences of several organisms, as well as the genes they encode. To exploit this growing body of data various techniques have been developed. Automatic sequencing was the first forerunner and had a major impact on the high throughput generation of various kinds of biological data such as single-nucleotide polymorphisms (SNPs) and expressed sequence tags (ESTs). Subsequently, other novel high-throughput methods such as SAGE (serial analysis of gene expression) [10] and DNA microarrays [11] have been developed to analyze the transcriptional program of a cell, tissue or organism at a genomic scale. These transcriptional profiling techniques promise a wealth of data that can be used to develop a more complete understanding of gene function, regulation, and interactions. However microarray analysis has become the most widely used technique for the study of gene-expression patterns on a genomic scale [12, 13]. Beside the study of mRNA [13] microarrays have been utilized to study protein levels [14], to decipher protein-DNA interactions [15], to analyze the DNA copy number [16], to detect methylated sequences [17], and to analyze gene phenotypes in living mammalian cells [18]. Thus many laboratories have adopted the microarray technology. Microarray analysis is a very complex, multi step technique involving array fabrication, labeling, hybridization and data analysis. A microarray workflow may start with the production of spotting slides. Robotic arrayers provide a reproducible and precise mathematical map from spots on the microarrays to wells in the microtiter plates, and therefore to the cDNA clones and the genes they represent. State of the art microarrays can have from several hundred up to 60,000 reporters annotated by dozens of parameters ranging from accession numbers to quality control descriptions. Once a collection of microarrays is produced, each slide represents a potential experiment. The arrayed genes are probes that can be used to query pooled, differentially labeled targets derived from RNA samples from different cellular phenotypes to determine the relative expression level of each gene. Most laboratories, that are using spotted arrays, are employing the classical method of hybridizing two labeled samples to an array, but several applications using three color microarrays have been established [19, 20]. After hybridization, slides are scanned and independent images for the control and query channels are generated. These images must then be analyzed to identify the arrayed spots and to measure the relative fluorescence intensities for each element. After image processing it is necessary to normalize the relative fluorescence intensities to identify and remove any systematic bias in the measured fluorescence intensities, arising from variation in the microarray process [21, 22].
All these steps leave a lot of room where errors may occur or protocols might need optimization to improve results. Moreover, information on details of the bench work, typically kept in lab notebooks or scattered files, as well as information regarding spotting, reliable tracking of the spotted molecules, scanning, and image quantification settings, is very relevant to the computational analysis and to reproduce experiments. All these information must be archived according to accepted scientific standards, which allow scientists to share common information and to make valid comparisons among experiments. For this reason the Microarray Gene Expression Data Society (MGED) [23] is focusing on establishing standards for microarray data annotation and exchange, facilitating the creation of microarray databases and related software implementing these standards, and promoting the sharing of high quality, well annotated data within the life sciences community. In particular the MGED society is working on three different projects: 1) Minimum Information About a Microarray Experiment (MIAME) [24], that aims to describe the information that researchers should provide to explain the procedures and biological purpose of their microarray data in adequate detail; 2) the MGED Ontology [25] to describe microarray experiments, biological samples and their manipulations using a large set of controlled vocabularies and ontologies; and 3) MAGE-ML (MicroArray and Gene Expression Markup Language) [26] which aims to create a common data format so that data can be shared easily. MAGE-ML is also used to deposit microarray experiments in ArrayExpress [27], a public repository for microarray gene expression data. These initiatives will maximize the value of microarray data by permitting greater opportunities for sharing information and thus for discovery, and will ultimately affect the description, analysis, and management of all high throughput biological data [25].

The well described ‘list of genes’ resulting from microarray analysis is not the end of a microarray experiment. The major challenge is to facilitate the search for biological meaning and to generate new hypotheses and/or to find new functions of genes. The simplest way to find genes of potential interest through several related experiments is to search for those that are consistently either up or down regulated. However, identifying patterns of gene expression and grouping genes into expression classes might provide much greater insight into their biological function and relevance. Techniques like hierarchical clustering, self-organizing maps, and principal component analysis have been used and already described for the analysis of gene expression data [28, 29, 30]. Other ways to extract functional information from microarrays are to incorporate functional annotations that are provided by the Gene Ontology (GO) Consortium [31] in the data analysis [32] or to map clusters or complete datasets to pathways [33].
Functional genomics, the study of gene function through parallel expression measurements of a genome, can give information about the function of uncharacterized genes. Examining gene expression patterns of biological processes and molecular pathways as well as transcriptional profiling in development and differentiation gives insights into molecular mechanisms and can lead to the generation of new hypothesis for further investigations [36].

Several different applications and databases have been developed by the bioinformatics community that address particular needs of a functional genomics environment, such as laboratory information management systems (LIMS) [37], web-based lab notebooks [38], microarray databases and repositories [39, 40, 41, 27], normalization [42, 43], clustering [29, 44], pathway [33, 45, 46] and GO mapping [47, 48] tools. However, there is a demand for a platform that allows to store and annotate microarray data in order to facilitate data analysis and functional annotation by providing the integration of sundry applications.

1.2 Objectives

The main objective of this thesis was to develop a scaleable and extensible platform that allows to store, query, and analyze large scale gene expression data gained from cDNA microarray experiments. Therefore a system should be developed to enable a flexible integration of heterogeneous data types, data sources, and applications. Furthermore this system should provide well defined user and data interfaces and fine grained user access levels.

Consequently, the specific aims were:

- Design and development of a Microarray Analysis and Retrieval System (MARS) including:
  - an integrated laboratory notebook to store the necessary information during biomaterial manipulation
  - a laboratory information management system to keep track of the information that accrues during the microarray production
  - well defined data interfaces for importing, exporting, and handling data
  - an External Application Connector Interface (EACI) to connect other web applications and link to its data without amending the MARS code
DESIGN AND DEVELOPMENT OF A BIOINFORMATICS PLATFORM FOR LARGE-SCALE GENE
EXPRESSION PROFILING

– a web-service interface to allow external applications such as normalization and
clustering tools to query and read the stored data and to write back results.

◼ Designing and implementing an application that is using the EACI to store and manage
quality parameters gathered during a microarray experiment (MARS-QM)

◼ Initiation, design and implementation of a user management system that provides libraries
and interfaces which can be integrated in any application to facilitate user authentication
and authorization.

◼ Initiating a database and web portal to upload biological pathways and microarray datasets
in order to analyze the gene expression levels in the context of several biological path-
ways.

◼ Evaluating the platform by studying the transcription profile of NCI-H295R cells treated
with fat cell-conditioned medium.
2 Methods

2.1 Microarray Technology

2.1.1 Introduction

DNA microarray technology has become an important tool in biomedical research during the last years. Microarray technology enables researchers to investigate the expression of several thousand genes simultaneously. The whole transcriptional response of these genes in normal cells or tissue, in disease condition, as an response to biological, genetical or chemical stimuli or during normal biological processes such as cell cycle or embryonic development can be investigated. The great potential of DNA microarrays lies not only in viewing the technology as a collection of individual expression measurements, but also in generating a composite picture of the expression profile of the cell.

The two major platforms for microarrays are 1.) spotted arrays, where the probes are deposited on modified glass slides by contact or inkjet printing, and 2.) in situ arrays [13], where oligo probes with a length of 20 to 25 nucleotides are synthesized via photolithography and combinatorial chemistry techniques [49]. In the latter approach, each gene or an expressed sequence tag (EST) is represented multiple times on the array by using different sequences designed to hybridize to different regions of the same RNA. An additional level of redundancy is implemented by the use of mismatch (MM) control probes that are identical to their perfect match (PM) partners except for a single base difference in a central position. The mismatch probes act as specificity controls that allow the direct subtraction of both background and cross-hybridization signals, and allow discrimination between real signals and those due to non-specific or semi-specific hybridization [50, 51].

In contrast to in situ arrays spotted microarrays permit the spotting cDNA clones with an inserted cDNA element representing an EST or a gene. These clones are in general polymerase chain reaction (PCR) products with 1000-1500 base pairs in length. The other possibility is to spot oligonucleotides designed for specific genes. Methods based on synthetic oligonucleotides do not require time-consuming handling of cDNA resources [52]. In addition, the elements can be designed to represent the most unique part of a given transcript, enabling the detection of closely related genes or splice variants. Spotted arrays allow a greater degree of flexibility in the choice of arrayed elements, particularly for the preparation of smaller, customized arrays for specific investigations. Additionally, arraying of unsequenced clones from cDNA libraries
or clones for ESTs not similar to characterized genes can be useful for gene discovery and functional annotation. Aliquotes of these probes are printed on coated glass microscope slides using a high precision robot.

Total RNA from test cells (e.g. treated cells) and reference cells (e.g. untreated cells) is reverse transcribed to cDNA and fluorescently labeled with different dyes. Commonly Cy3 and Cy5 are used. This is in contrast to in situ arrays, where only one labeled RNA sample is used. The fluorescent targets are pooled and allowed to hybridize under stringent conditions to the elements on the array. After hybridization, slides are scanned by a laser or CCD scanner and independent images for the reference and test channels are generated. The resulting monochrome images must then be analyzed to identify the arrayed spots and to measure the relative fluorescence intensities for each element [22]. The basic principle of the cDNA microarray technology is illustrated in figure 1.

All biological conclusions and predictions resulting from microarray data rely on the quality of the data. Consequently, it is important to focus on the key factors that affect the quality of the result during the experiment design phase.

Good microarray experimental design should comprise at least four elements [53]:

1. A clearly defined biological question and/or hypothesis;
2. Treatment, perturbation, and observation of the biological materials, as well as the microarray experimental protocols should be as little affected by systematic and experimental errors as possible;

3. A simple, sensible, and statistically sound microarray experimental arrangement that will give the maximal amount of information given the cost structure and complexity of the study;

4. Compliance with the standards of microarray information collection;

The key principles to provide statistically sound microarray experimental arrangements have been described in [53, 54, 55, 56]. Generally the following issues should be considered:

- **Biological Replicates**: Perform repeated hybridizations with RNA samples from independent sources to consider biological variability.

- **Technical Replicates**: Microarray experiments offer two ways for replicated measurements

  - **Replicated Features**: Repeated positioning of the same element on the array. Additionally the use of internal control features (e.g. features for genes of other organisms) can help to ensure the quality of the data.

  - **Replicated Hybridizations**: Repeat the hybridization with the same RNA. Additionally dye-swap can be used to reduce the systematic bias.

- **Pooling Samples**: Pooling all available samples together would minimize the biological variance, but would also eliminate replication. Thus using several pools and fewer technical replicates is preferable.

- **Control versus reference RNA**: When comparing several RNA samples (controls) with one reference, a reference with a broad coverage of genes would be desirable. This can be achieved for example by pooling cell lines.

### 2.1.2 Data Mining

After analyzing the monochrome images using software tools like GenePix [57] or TigrSpotFinder [42] the resulting raw data set has to be filtered for low intensity, saturated or inhomogen spots. Additionally, background correction can be performed by estimating and subtracting the local
background. To correct different types of systematic and random errors that are associated with microarray experiments it is advisable to normalize the filtered raw data set [58]. Normalizing is known as removing of all non-biological variation introduced in the measurement and mining the random error to get reliable results [12, 59]. Depending on the experiment, normalization is used in different ways. It has to be distinguished between within-slide normalization, paired-slides normalization for dye-swap pairs, or multiple-slides normalization (scaling between slides). In each case one can use all genes on a slide, constantly expressed genes, or a set of control genes as the set of genes used for normalization [60, 12, 61, 62, 63, 64].

The resulting transformed dataset can be used for further analysis. The data for each gene are typically reported as an expression ratio or as the logarithm of the expression ratio. The expression ratio is simply the normalized value of the expression level for a particular gene in the query sample divided by its normalized value for the control.

At this point in the analysis we typically look for genes that are differentially expressed. To define differential expression one could either define a cut-off level for increase or decrease in measured expression or alternatively use statistical tests [28, 65, 66]. The true power of microarray analysis does not come from the analysis of a single hybridization, but rather from the analysis of many hybridizations under different experimental conditions to identify common patterns of gene expression. Based on our understanding of cellular processes, genes that are contained in a particular pathway, or that respond to a common environmental challenge, should be co-regulated and consequently should show similar patterns of expression [28, 67]. For the identification of genes showing similar patterns a vast repository of statistical methods is available that is generally referred as **Cluster Analysis**. Basically there are supervised and unsupervised clustering techniques [28]. The most important methods for the latter are hierarchical clustering, self-organizing maps, k-means clustering, and principal component analysis. As a representative for supervised clustering support vector machines can be named [28, 29, 30, 68, 69, 70].

In order to gain biological understanding, it may be necessary to analyze the functional annotations of all genes or of gene clusters. The Gene-Ontology database (GO) provides a useful tool to annotate and analyze the functions of a large number of genes. Thus generating statistics of annotations which are overrepresented in the analyzed list of genes is facilitating the annotation of new genes [71, 72].

A complementary approach is to view the data at the level of known biological processes or pathways. Identifying those groups of biologically related genes that are showing a large number of gene-expression changes will create an informative description of the biology that is
occurring in a particular dataset, making it possible to generate new hypotheses and identify those specific areas of biology that warrant more detailed investigation [33, 73]. Tools like Genesis [29] are combining cluster analysis with GO annotation whereas MAPPFinder is uniting pathway information with GO data [47].

2.1.3 Transcription Profiling of NCI-H295R Cells treated with Fat Cell-Conditioned Medium

Recent evidence suggests that human fat is a highly active endocrine tissue [74, 75, 76] and it has been shown that secretory products from isolated human adipocytes strongly stimulated steroidogenesis in human adrenocortical cells (NCI-H295R) with a predominant effect on mineralocorticoid secretion. This stimulation was comparable to maximal stimulation of these cells with forskolin [77].

For further investigations of these findings we have conducted three microarray experiments to analyze the expression profile of NCI-H295R cells stimulated with 1) fat cell-conditioned medium (FCCM), 2) Forskolin and 3) Angiotensin II.

NCI-H295R Cells: NCI-H295R adrenocortical cells were kindly provided by T. Skurk (German Diabetes Center, Düsseldorf, Germany). Cells were grown in DMEM F12 supplemented with insulin (66 nM), hydrocortisone (10 nM), 17β-estradiol (10 nM), transferrin (10 µg/ml), selenite (30 nM), penicillin (100 units/ml), streptomycin (100 µg/ml), and 2% FBS. NCI-H295R cells were grown in 75cm² flasks (Becton Dickinson) at 37°C in a humidified atmosphere of 5% CO₂ 95% air. The medium was changed every 3 days, and cells were subcultured every 7 days by using Accutase (PAA Laboratories, Cölbe, Germany) for cell detachment. Cells used for experiments were subcultured from 70% confluent stock cultures into 24-well culture plates (Falcon) at a density of 70,000 cells per cm² for 96 h. Cells were then treated as described below.

Incubation of NCI-H295R Cells: NCI-H295R cells were incubated with the respective stimulation medium for 24 h or for 3-36 h to evaluate the time dependency of the effect. FCCM was supplemented with insulin (66 nM), hydrocortisone (10 nM), β-estradiol (10 nM), transferrin (10 µg/ml), selenite (30 nM), penicillin (100 units/ml), and streptomycin (100 µg/ml). After
incubation, culture medium was collected and kept frozen at $-20^\circ C$ until assayed.

**RNA Extraction:** After cell stimulation cells were harvested by prior aspirate culture medium and subsequent wash step using PBS. RNA isolation was performed with the Trizol reagent (Invitrogen) according to the manufacturer’s protocol. Briefly, cells were homogenized by passing the cell lysate several times through a pipette. After incubation for 5 min. at room temperature and chloroform addition, samples were centrifuged for 15 min. at 12000 x g at 4 °C. RNA was precipitated from the aqueous phase by mixing with isopropyl alcohol. After centrifugation and alcohol washing steps RNA was air dried and redissolved in DEPC water.

**Microarray experiments:** The used labeling and hybridization procedures were based on those developed at The Institute for Genomic Research [22] and can be viewed at http://gold.tugraz.at. Briefly, 20 µg of total RNA was indirectly labeled with Cy3 and Cy5, respectively. The Random Hexamer (Invitrogen) primed first strand cDNA synthesis was carried out using SuperScript Reverse Transcriptase II (Invitrogen) in the presence of amino allyl dUTP (Sigma), dATP, dGTP, dCTP, dTTP (Invitrogen), DTT, and 1X first strand buffer overnight at 42 °C. cDNA was purified with QIAquick columns (Qiagen) according manufacturer’s directions, but using potassium phosphate wash and elution buffer instead of supplied buffers PE and EB. N-hydroxysuccinimide (NHS) esters of Cy3 and Cy5 (Amersham) were coupled to the amino allyl dUTPs incorporated in the cDNA. Coupling reactions were quenched by 0.1 M sodium acetate (pH=5.2) and unincorporated dyes were removed using QIAquick columns (Qiagen). Slides were prehybridized in 1% BSA, 5xSSC, 0.1 %SDS for 45 min at 42°C and washed in MilliQ water and 2-Propanol and dried in a centrifuge. Fluorescent cDNA samples were dried in a SpeedVac, resuspended in 12 µl hybridization buffer (50 % formamide, 5XSSC, 0.1 % SDS) and pooled. 1 µl containing 20 µg human Cot1 DNA and 1 µl containing 20 µg poly(A) DNA were added, denatured at 95°C for 3 min and snap cooled on ice for 30 sec. Sample with a final sample volume of 26 µl was applied to the prehybridized slide, covered with a glass cover slip (Roth) and hybridized in a humidified chamber for 20 hours at 42°C in the dark. Slides were washed 2 min in a 2xSSC, 0.1 % SDS solution (42 °C), 5 min in 2xSSC (30 °C pre heat - whash at room temperature), 0.1% SDS, 5 min 1xSSC and 5 min in a 0.5xSSC (30 °C pre heat - whash at room temperature), dipped twice in MilliQ water and dried in a centrifuge at 1500 rpm for 2
minutes.
Slides were scanned with a GenePix 4000B microarray scanner (Axon Instruments) at 10 μm resolution. Photo multiplier voltages (PMT) were selected in order that the histogram of the red channel (635nm) and the green channel (532nm) were overlapping to a large extend and few spots were saturated. Identical settings were used for the scanning of the corresponding dye-swapped hybridized slides. The resulting TIFF images for each of the two fluorophors were analyzed with GenePix Pro 4.1 (Axon Instruments) to get relative gene expression levels for each gene. Data were filtered for low intensity, inhomogeneity, and saturated spots.

**Normalization:** To correct the different sources of systematic (sample effect, array effect, dye effect and gene effect) errors associated with microarray experiments [58] the data has been normalized. As method of choice dye-swap normalization was applied using ArrayNorm [43]. The expression ratio $T$ for gene $i$ at each time point in relation to the reference was calculated by

$$T_i = \sqrt{\frac{R_{i1} \times G_{i2}}{G_{i1} \times R_{i2}}}$$

where $R_{i1}$ refers to the red signal of the first hybridization and $G_{i2}$ for the green signal of the second hybridization for gene $i$ [62]. Between hybridization one and two the the assignment of the dyes was reversed. Replicated spots were averaged and the resulting ratios were log$_2$ transformed and for each time point averaged over the dye swapped experiments.

### 2.2 The Java 2 Enterprise Edition (J2EE)

The Java 2 Enterprise Edition (J2EE) defines a standard for developing multi tier enterprise applications. Multi tiered applications are divided into the following tiers (Figure 2):

- **Client-tier** components run on a client machine
- **Web-tier** components run on a J2EE server
- **Business-tier** components run on a J2EE server
- **Enterprise information system (EIS)-tier** software runs on a EIS server

Although a J2EE application can consist of three or four tiers, J2EE multi tiered applications are generally considered to be three-tiered applications because they are distributed over three different locations:
Figure 2: Multi tiered applications: Application 1 depicts a typical four tiered web application, where the browser communicates with the web tier that fetches the information from the business tier. In application 2 a desktop application is communicating directly with the business tier, that can persist the data in the EIS tier. Multi tiered applications enforce the separation of logical tiers.

- Client machines
- J2EE server machines
- The database or legacy machines

Three-tiered applications that run in this way extend the standard two-tiered client and server model by placing a multithreaded application server between the client application and back-end storage [78]. Additionally a three tier architecture enforces the separation of presentation-, business-, and data logic. Thus this architecture is intended to allow any of the three tiers to be upgraded or replaced independently as requirements change.

The J2EE platform enables, employs, and simplifies the development of enterprise applications by basing them on standardized modular components like Enterprise JavaBeans (EJB), Java Servlets, Java Server Pages (JSP), and eXtensible Markup Language (XML) technology [79, 78]. Furthermore it is providing a complete set of services to those components and handling many details of application behavior automatically (Figure 3). Properly designed applications can be deployed and run on any J2EE compliant application server. Many established
companies offer their version of a certified J2EE compliant application server (eg. Oracle OC4J, Sun ONE, IBM Websphere, and BEA WebLogic). Additionally there are several open source application servers available that are not standing back in performance, scaleability, and implemented features (e.g. JBoss, JOnAS, and OpenEJB).

2.2.1 Enterprise JavaBeans (EJB)

Written in the Java programming language, an Enterprise JavaBean (EJB), or enterprise bean, is a server-side component that encapsulates the business logic, which is the code that fulfills the purpose of the application. One can think of an enterprise bean as a body of code having fields and methods that can be used alone or with other enterprise beans to execute business logic. Thus using EJBs can simplify the development of larger and/or distributed software applications, because they are designed to help a developer building distributed, scaleable, secure, portable, and transactional application-logic components. EJBs run within the context of an J2EE compliant application server, that provides a runtime environment and a relatively complete set of commonly used system-level services, such as life cycle management, security...
services, transaction management, and persistence mechanisms. Thus an enterprise bean developer can focus on solving business problems at hand, without having to worry about low-level system services [78].

Another advantage is that the enterprise beans and not the clients contain the business logic. This allows the client developer to concentrate on the presentation of data to the end user. As a result these clients are "thinner" (i.e. contain less code) and are therefore preferred to run on smaller client machines [80].

Given that enterprise beans are portable components, an application developer can rapidly assemble new applications from existing beans. Currently EJB 2.0 is the de facto standard implemented by the most important application server providers. This standard defines different types of enterprise JavaBeans summarized in table 1 [81].

<table>
<thead>
<tr>
<th>Type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Bean</td>
<td>Model business processes</td>
</tr>
<tr>
<td>Entity Bean</td>
<td>Model business data</td>
</tr>
<tr>
<td>Message Driven Bean</td>
<td>Similar to Session Beans, but can just be called by asynchronous messages</td>
</tr>
</tbody>
</table>

Table 1: Different Types of Enterprise Java Beans

**Session Beans** A session bean represents work being performed for client code that is calling it. Session beans are reusable business process objects. They implement business logic, business rules, algorithms, and workflow. For example a session bean can perform biosample entries, queries for plates, and more.

All enterprise beans hold conversations with clients at some level. A conversation is an interaction between a client and a bean and it is composed of a number of method calls between the client and the bean [81].

The two subtypes of session beans are *stateful session beans* and *stateless session beans*. Each is used to model different types of conversations [78]:

- **Stateful Session Beans**: Some business processes are naturally drawn-out conversations over several requests. Hence, a stateful session bean is designed to service business processes that span multiple method requests or transactions. To accomplish this, stateful session beans retain state on behalf of an individual client. If a stateful session bean’s state is changed during a method invocation, that same state will be available to that same client upon the following invocation.
Stateless Session Beans: Some other business processes lend themselves to a single request conversations. A single request business process is one that does not require state to be maintained across method invocations. When a method is invoked on a stateless session bean, it executes the method and returns the result without knowing whether other request have issued before or might follow. Stateless really means that there is no conversational state. However, stateless session beans can contain state that is not specific to any client, such as a database connection factory that all clients would use.

Entity Beans  Entity beans are persistent objects that can be stored in permanent storage. Thus they allow to model the underlying business data. Entity beans store data as fields, such as genBankID and have methods associated with them, such as getGenBankID() or setGenBankID(). Since entity beans map to a storage there are two ways to persist entity beans [81, 79]:

- **bean managed persistence entity bean (BMP)** is an entity bean that must be persisted by hand. In other words, the bean developer is entirely responsible for synchronizing the in memory fields (e.g. genBankID) with the underlying data store. Additionally the developer has to handle transactions and locks. The benefit is the full control over all actions pertaining the data store, allowing an access optimization.

- **container managed persistence entity bean (CMP)** lets the container supply and perform full synchronization between the entity and the persistence layer. The developer does not have to care how beans access their data. All transaction and locking mechanism are performed by the container. Additionally the container can provide entity caching and read ahead which may also improve performance.

Message Driven Beans  Messaging is an alternative to remote method invocations. The idea behind messaging is that a middleman sits between the client and the server. This middleman receives messages from one or more message producers and broadcasts those messages to one or more message consumers. Because of this middleman, the producer can send a message and then continue processing. This paradigm is called asynchronous programming. The Java Message Service (JMS) application programming interface (API) is a messaging standard that allows application components based on the Java 2 Enterprise Edition platform to create, send, receive, and read messages.

A **message driven bean** is a special EJB that can receive JMS messages. These consumed
messages are processed while the bean is decoupled from any clients. Summarizing, a message-driven bean is a stateless enterprise bean that allows J2EE applications to process messages asynchronously [78, 79].

2.2.2 Servlets and Java Server Pages

Servlets and Java Server Pages (JSP) are a subset of the overall collection of the Java server-side application programming interface J2EE. While EJBs are used to form the application and data layers, Servlets and JSPs together form the presentation layer of J2EE web applications. Servlets are the Java way to create web-enabled applications. When a Servlet receives an HTTP (Hyper Text Transfer Protocol) request, it returns an HTTP response and fortunately most of the underlying details of this process have been abstracted from the developer. Servlets can be compared to other technologies such as CGI (Common Gateway Interface) scripts that interfaces with a web server to produce dynamic web content.

JSPs perform the same task as Servlets (programmatically generating web content), but use a different development paradigm. While Servlets are created by writing Java code in classes, Java Server Pages are created by using a syntax very similar to HTML, unlike Microsoft’s Active Server Pages (ASP) technology. This so created JSPs are then, at run-time, automatically converted into Servlets.

While the functionality of Servlets and JSP are redundant, the two different development styles lend themselves quite nicely to effective web development [82]:

- **Java Server Pages** are ideal for creating dynamic web pages. HTML developers can use tools they are already familiar with to develop normal HTML. To add dynamic functionality it is just necessary to insert custom JSP tags here and there.

- **Servlets** on the other hand are ideal for the creation of highly programmatic content, such as images created on demand. Servlets are also often used as Controllers to implement a Model-View-Controller (see Chapter 2.2.4) architecture design pattern.

Thus, Servlets and JSP are complementing one another forming essential components to build powerful web applications [83]. Servlets are not stand-alone applications; they must be managed by a Servlet container. The Servlet container manages the lifecycle of a Servlet and handles the socket-level communication. This lets the Servlet and JSP developer focus on the content and the actions initiated by HTTP requests [78]. A Servlet container may function as a complete web server, or it may be in-
tegrated in a third-party web server. The Jakarta project Tomcat (http://jakarta.apache.org/tomcat/) is the Servlet container that is used as the official reference implementation for the Java Servlet and Java Server Pages technologies. The Servlet and JSP specifications are developed by Sun under the Java Community Process. Eventhough there are other vendors and projects providing Servlet containers (e.g. Jetty, IronFlare Orion, Caucho Resin and Sun ONE).

2.2.3 Custom Tags

A powerful feature of JSP is the ability to create custom tags in addition to the simple tags that are defined by the JSP standard. A custom tag is a user-defined JSP language element that provides a mechanism for encapsulating other types of dynamic functionality. When a JSP page containing a custom tag is translated into a Servlet, the tag is converted to operations on an object called a tag handler. The web container then invokes those operations when the JSP page’s Servlet is executed. Custom tags are usually distributed in form of a tag library, which defines a set of related custom tags and contains objects that implement the tags.

The major benefits of using custom tags and tag libraries are:

- tag libraries are a great way to group common functionality and methods
- tags make JSP easier to maintain
- tag libraries work on any JSP
- tags speed up web development

2.2.4 Jakarta Struts Framework

Struts is an open source framework useful for building web applications in Java Servlets and Java Server Pages technology that encapsulates best software design practices along with the power of custom tags. Struts implements a Model 2 JSP web application architecture, which is commonly associated with the Model-View-Controller (MVC) architectural software design pattern. [83, 84, 85] The MVC architecture divides applications into three layers – model, view, and controller – and decouples their respective responsibilities. Each layer handles specific tasks and has specific responsibilities to other areas (Figure 4).

- A model represents business data and business logic or operations that govern access and modification of this business data. Often the model serves as a software approximation to real-world functionality. The model notifies views when it changes and provides the
Figure 4: The Model-View-Controller Architecture: The controller is responsible for the application behavior. It maps user actions from the view to model updates and selects the next view for the response. The view can query the model to render it for the user. The model responds to state changes and notifies the view.

- A **view** renders the contents of a model. It accesses data from the model and specifies how that data should be presented. Data presentation is updated when the model changes. A view also forwards user input to a controller.

- A **controller** defines application behavior. It dispatches user requests and selects views for presentation. A controller interprets user inputs and maps them into actions to be performed by the model. In a stand-alone graphical user interface (GUI) client, user inputs include button clicks and menu selections, whereas in a web application, they are HTTP GET and POST requests to the web tier. A controller selects the next view.
to display based on the user interactions and the outcome of the model operations. An application typically has one controller for each set of related functionality.

Specifically, a typical workflow using the Struts components can be described as (see Figure 5):

Figure 5: Typical Struts workflow: Depicts the exact steps from the first request, to the query of the model, to the final response to the client. This figure was drawn by Dan Cancro for the Struts community.

When a request is sent to the Server (Step 1), the ActionServlet looks in its action-mappings for the Action and then in the form-beans for the Form Bean for the given request URL. It retrieves the Form Bean from a context container or creates one, resets its fields, and calls its validate()
method to validate the new request data (Step 2). The ActionServlet passes the Form Bean to the Action’s `perform()` method (Step 3). Then the Action invokes methods on business object(s). The business object(s) does some work (Step 4) and returns the result to the Action (Step 5). The Action updates the Form Bean, if necessary, with results from the Business Object(s). This is where the Action can set properties of the Form Bean using Value Objects (aka. Data Transfer Objects) (Step 6). The Action returns a success, failure, or other ActionForward object to the ActionServlet (Step 7). This looks in its ActionMappings and finds the appropriate page for the given Action and ActionForward, then forwards to that page (Step 8). The page is containing tag(s) that get their information from the ActionForm(s) and the ResourceBundles (Step 9). Finally, the response is returned to the client (Step 10) [84, 85, 86].

### 2.2.5 Cookies

When a user is connecting to a web server to view a web page, a relationship is formed between the browser and the web server. This relationship is of short nature. The browser generates a request for information that is passed to the server, which then replies to it. This is called the request/response paradigm. In the earliest implementation of HTTP, each request created a new socket connection to the server, sent the command, then read the response from the same connection. Although this was simple to specify and implement, it was also slow, especially in a high volume situation. To basically reduce the significant overhead of rapidly creating and closing socket connections for each new request, `keep-alives` were added to HTTP. However, `keep-alives` do not guarantee persistent relationships between the client and the server, because `keep-alive times` are typically very short. Thus, if a client is connecting to the same server after the `keep-alive time` has elapsed, the server is unaware that it has ever communicated with this particular client in the past. This concept is described by the term stateless and thus HTTP is said to be a stateless protocol. A certain lack of persistence is fine if a website’s sole functionality is serving up simple web pages. However, there is a need to persist some information and therefore Cookies have been introduced.

Cookies are pieces of information generated by a web server and stored in the user’s computer, ready for future access. Cookies are embedded in the HTML information flowing back and forth between the user’s computer and the servers. The main intention for the implementation of Cookies was to allow user-side customization of web information.

When a web application creates a Cookie, which is essentially a tagged string of text containing some relevant information, the web server transmits this Cookie to the user’s computer. If the
user has enabled Cookies in the web browser, the Cookie will be received and stored in a special file called a Cookie list. Whenever a user directs the web browser to display a certain page from this server, the server can read the Cookie from the client browser.

2.2.6 Design Patterns

The software community has known the value of design patterns for some time. In the 1970’s, a number of books have been published documenting patterns in civil engineering and architecture. The software community took note and began to refer to patterns found in their own work. Interest in software design patterns has peeked in the last years, especially in the Java community.

A pattern describes a proven solution to a recurring design problem, placing particular emphasis on the context and forces surrounding the problem, and the consequences and impact of the solution [87]. The most important reasons for using design patterns are [88, 89]:

- **They have been proven.** Patterns reflect the experience, knowledge, and insights of developers who have successfully used these patterns in their own work.

- **They are reusable.** Patterns provide a ready-made solution that can be adapted to different problems as necessary.

- **They are expressive.** Patterns provide a common vocabulary of solutions that can express large solutions succinctly.

Even though there are various different and very useful J2EE design patterns [90], this section is concentrating on the **Session Facade** and **Transfer Object** patterns which have been extensively used for this thesis.

**Session Facade Pattern**  Application clients need access to business objects to fulfill their responsibilities and to meet user requirements. These business objects are created by the business logic that is typically located in the EJB container and is represented by session, entity, and message driven beans. Many business processes involve complex manipulations of business classes, which often participate in multiple business processes or workflows. Clients can directly interact with these business objects when they expose their interfaces. The client must understand and be responsible for the business data object relationships, and must be able to handle business process flow properly.
However, direct interaction between the client and the business objects leads to tight coupling between the two, and such tight coupling makes the client directly dependent on the implementation of the business objects. Direct dependence means that the client must represent and implement the complex interactions regarding business object lookups and creations, and must manage the relationships between the participating business objects as well as understand the responsibility of transaction demarcation. The resulting complex relationships between low-level business components make clients difficult to write and the resulting code lacks on reusability. Additionally, fine-grained access through remote interfaces is inadvisable because it increases network traffic and latency. The "before" diagram in figure 6 shows a sequence diagram of a client accessing fine-grained business objects through a remote interface. The multiple fine-grained calls create a great deal of network traffic, and performance suffers because of the high latency of the remote calls.

The Session Facade pattern defines a higher-level business component that contains and centralizes complex interactions between lower-level business components. Thus, it hides from the client’s view the complex interactions between the participants. A Session Facade is typically implemented as a session enterprise bean. This session bean also manages the life cycle of these participants by creating, locating (looking up), modifying, and deleting them as required by the workflow. It provides clients with a single interface for the functionality of an application or application subset [90, 88, 91].

The main forces of the Session Facade pattern are:

- Provide a simpler interface to the clients by hiding all the complex interactions between business components.

- Reduce the number of business objects that are exposed to the client across the service layer over the network.

- Hide from the client the underlying interactions and interdependencies between business components. This provides better manageability, centralization of interactions (responsibility), greater flexibility, and greater ability to cope with changes.

- Provide a uniform coarse-grained service layer to separate business object implementation from business service abstraction.

- Avoid exposing the underlying business objects directly to the client to keep tight coupling between the two tiers to a minimum.
Figure 6: Sequence diagram before and after adding Session Facade: The before diagram shows that there is a permanent data flow over the network boundary caused by several fine-grained function calls. The session facade reduces the traffic by reducing the client-exposed business objects.

- Enforce a clear and strict separation of business logic from presentation and data logic.

**Transfer Object Pattern**  Some entities contain a group of attributes that are always accessed together. Accessing these attributes in a fine-grained manner by invoking a business object’s get-method multiple times through a remote interface causes network traffic and high latency, and consumes server resources unnecessarily. Therefore, using multiple calls to get methods that return a group of attributes is inefficient for obtaining data values from an enterprise bean. To reduce the number of remote calls and to avoid the associated overhead, it is best to use *transfer objects* to transport data from the enterprise bean to its client. A transfer object is a serializable class that groups related attributes, forming a composite value. This class is used as the return type of a remote business method. Clients receive instances of this class by calling coarse-grained business methods, and then locally access the fine-grained values within the transfer object. Because the transfer object is passed by value to the client, all calls
to the transfer object instance are local calls instead of remote method invocations. Fetching multiple values in one server roundtrip decreases network traffic and minimizes latency and server resource usage [90, 88, 91].

A detailed description of other design patterns used for this project can be found at [92].

2.3 Relational Databases

Since functional genomics deals with a great many of data there is a need to efficiently store, query, and retrieve it. A relational database management system (DBMS) is a collection of programs that enables to store, modify, and extract information from a relational database by providing data integrity, scaleability, security, and concurrency. Relational databases are rested upon the theory of relational mathematics based on the set theory and was conceived by E. F. Codd in 1969 [93]. The basic idea behind the relational model is that a database consists of a series of unordered tables (or relations) that can be manipulated using non-procedural operations that return tables. This model was in vast contrast to the more traditional database theories of the time that were much more complicated, less flexible and dependent on the physical storage methods of the data. Relational databases have a much more logical structure in the way data is stored. Tables are used to represent real world objects; with each field acting like an attribute. The set of rules for constructing queries is known as a query language. Different DBMSs support different query languages, although there is a semi-standardized query language called SQL (structured query language) [94]. The SQL command set can be divided into three sections:

- **Data Definition Language (DDL)** allows the creation and deletion of tables in the database as well as the definition of indexes and constraints.

- **Data Manipulation Language (DML)** includes the syntax for complex queries as well as for updates, insertions and deletions of data records.

- **Data Control Language (DCL)** include actions, such as granting privileges to users, and defining when proposed changes to a databases should be irrevocably made.

One major advantage of the relational model is that if a database is designed efficiently according to Codd rules [95], there should be no duplication of any data, which helps to maintain database integrity and to save storage space. Anyhow, sometimes there is a need to disregard some of Codd´s rules and to de-normalize the database to improve performance or data handling.
2.3.1 Java Database Connectivity (JDBC)

JDBC is a low-level application programming interface (API) written in Java programming language which allows to establish a connection with any SQL database to query, update, and insert data. It provides library routines that support the integration of direct SQL calls into the Java programming environment. Thus it facilitates a very easy access to a database by opening a connection and sending SQL code to the database engine which executes the demanded commands. Having accomplished the request, the Java program closes the connection and continues with its execution [96, 97]. The combination of JDBC with J2EE becomes an extremely useful tool in generating web based database applications [96].

2.4 User Authentication and Authorization

To avoid unauthorized access in a multi-user environment the control of user access is a crucial criterion for the acceptance of web based applications storing sensitive data. Authentication is normally a prerequisite for authorization (unless everybody is authorized to do something), but they are separate and distinct concepts [98, 99]:

- **Authentication** establishes who someone is.
- **Authorization** establishes what someone is allowed to do.

**User authentication** is the process of reliably verifying the identity of someone. Authentication is the most difficult from the perspective of network security. Classically, there are several different ways that someone authenticate himself or a computer to another computer system by providing a username and password, a digital certificate, a card key, a smart card, or even more sophisticated a fingerprint, a retina scan or voiceprint analysis [100].

**User authorization** is verifying that the person is really allowed to do what it is requesting to do. Usually authorization is checked after user authentication. Authorization is achieved by assigning access controls such as read, write, or delete, for users or groups to the resources (EJBs, HTML pages, Servlets or functions) being accessed. These controls, along with the authorized users or groups, can be maintained in an access control list (ACL) associated with each resource.

Maintaining ACLs for every user to be controlled can quickly become prohibitively expensive. A common way to keep the maintainace of ACLs easy is to introduce the concept of groups.
Instead of specifying all the individuals on a resource, it is better to specify groups on a resource and add users to this group. Offering users of a group additional rights is also possible by assigning them extra access rights in addition to the one gained through their group memberships. [100, 99].

### 2.5 Web Services

Web services is an emerging technology driven by the will to expose business logic beyond a firewall in a secure and interoperable way. Thus it allows to interconnect web services of different server platforms, including the .NET Framework, J2EE, Perl, Python, and C++. This is potentiated because web services are platform agnostic; in other words, the medium used to communicate is not specific to any programming language, operating system, and hardware [101].

Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across the web. They perform functions, which can be anything from simple requests to complicated business processes. Once a web service is deployed, other applications (and other web services) can discover and invoke the deployed service. Web services communicate using HTTP and XML and interact with any other web service using standards like Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description Discovery and Integration (UDDI) services, which are supported by major software suppliers [102]. Alongside the mentioned benefits, network speed and round trip time latency are the main limitations of web services. An additional limitation is the use of SOAP as the protocol, since it is based on XML and HTTP, which degrades performance compared to other protocols like CORBA [103, 104].

A more detailed description on web services can be found at [92].

### 2.6 Microarray Gene Expression Markup Language

Microarray experiments are generating a wealth of gene expression data, providing important insights into a variety of biological processes [105]. Among the vast challenges microarray technology present to both bioinformaticists and biologists, datacommunication is one of the most significant. In comparison to biological sequences, microarray data requires data structures that are both multidimensional and varied. This applies to the underlying gene expression data and the descriptive biological annotations that provide context for gene expression measurements [26]. To maximize the use of these data, a community infrastructure for sharing these data has been established [106, 107, 108].
MIAME describes the Minimum Information About a Microarray Experiment that is needed to enable the interpretation of results of an experiment unambiguously and potentially to reproduce the experiment [24, 109]. Currently some journals have begun to endorse and encourage MIAME compliance for papers describing results of microarray experiments. However, it is essential, if MIAME is to be useful, that there is a standard transmission format for the data. Several groups started to develop their own XML-based data communication syntaxes for microarray experiments, but finally they committed to work together to design a common data structure for communicating microarray-based gene expression data that is flexible and robust [26, 25, 23].

XML (eXtensible Markup Language) is a set of rules whereby new vocabularies (tags) may themselves be defined. These tags do not indicate how a document should be formatted, but instead provide semantic context to the content of the document. Thus XML allows to hold information in a way such that the information can be understood. Usually an XML document is not a stand-alone document, but will refer to another document, called the document type definition, or DTD. The DTD contains a set of rules, or declarations, that specify which tags can be used, and what they can contain [110, 111, 97].

Finally, the results of this cooperation, which are collectively referred to as MAGE (MicroArray Gene Expression), are:

- MAGE-OM, an object model to store MIAME compliant data
- MAGE-ML, the XML representation of MAGE-OM
- MAGE-STK, a software toolkit facilitating the adoption of MAGE

Through the participation in the OMG [112], MAGE is an adopted specification [23] that can be found at [113].

**MAGE-OM** The MAGE-OM is a data centric object model to hold augmented MIAME compliant microarray data. Currently MAGE-OM is divided into 17 packages that are used to organize classes which share a common purpose. For example the array package contains classes that describe individual arrays, including detailed information on relevant manufacturing processes. The key components of MAGE-OM reflect many of the core requirements of MIAME, specifically see table 2 and figure 7:

While the MAGE model is not a laboratory information management system (LIMS), such information does have a critical role in understanding microarray data, and much of this in-
DESIGN AND DEVELOPMENT OF A BIOINFORMATICS PLATFORM FOR LARGE-SCALE GENE EXPRESSION PROFILING

<table>
<thead>
<tr>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment</td>
<td>Stores experiment goals and design</td>
</tr>
<tr>
<td>BioMaterial</td>
<td>Holds the used biological materials and description of their creation</td>
</tr>
<tr>
<td>ArrayDesign, BioSequence</td>
<td>Manages the arraydesign and its purpose</td>
</tr>
<tr>
<td>Array</td>
<td>Records array manufacture details</td>
</tr>
<tr>
<td>BioAssay</td>
<td>Accumulates hybridization, wash, and scan information</td>
</tr>
<tr>
<td>BioAssayData</td>
<td>Saves gene expression data</td>
</tr>
<tr>
<td>Utility packages</td>
<td>Shared by the above packages, like information on people, organizations, protocols, free text descriptions, and the ability to link to ontologies such as those provided by MGED</td>
</tr>
</tbody>
</table>

Table 2: Packages of MAGE-OM [23]

information can be stored in MAGE-OM (for example, protocols and sources for clones used in manufacturing microarrays). However, MAGE-OM provides a structure for the logical flow of experiments.

**MAGE-ML** MAGE-ML (MicroArray Gene Expression Markup Language) is a language designed to describe and communicate information about microarray based experiments. MAGE-ML is based on XML and the DTD is automatically generated out of MAGE-OM. Due to the large overhead of the text based format caused by the recurrent content describing tags, exporting measured bioassay signals (raw data) would blow up the XML file. Thus this data is stored in a separate file as a three-dimensional matrix (or cube) of values whose axes are labeled by DesignElements (the genes), BioAssays (experimental samples), and QuantitationTypes (parameters from the scanning software). MAGE-ML has a flexible design so that it can be used in a wide variety of technical settings like spotted two-color cDNA arrays or Affymetrix arrays. These extend well beyond gene-expression experiments, for without modification, data from all DNA microarray experiments and technologies can be stored. Presently there are initiatives to adopt MAGE-ML to support experiments that use arrays made of proteins, cells, or tissues [26].

**MAGE-STK** MAGE-STK is a suite of software tools based on the MAGE-OM to access MAGE-ML. Currently the MAGE group provides APIs for MAGE-ML reader and writer supporting two implementations: MAGE-Perl, and MAGE-Java. Using the MAGE-STK allows to fill the MAGE-OM by instanciating MAGE-OM classes and to serialize the objects to a MAGE-
ML file.
All MAGE software is open source for academic and commercial use. The MAGE documentation and software can be found at [23]. For this thesis the Java MAGE-STK has been used.

2.7 MGED Ontology

The primary purpose of the MGED Ontology is to provide standard terms for the annotation of microarray experiments [107]. This controlled vocabulary enables structured queries of certain elements of experiments. Furthermore, standard terms allow unambiguous descriptions of how an experiment was performed. Providing such terms in the form of an ontology means that the terms are organized into classes with properties. The current version of the MGED Ontology has 226 classes and 107 properties [114]. To make the MGED Ontology available, the MGED Ontology Group uses the DARPA Agent Markup Language (DAML). DAML is being developed as an extension to XML and the Resource Description Framework (RDF) [115] to provide ontologies in a machine readable and understandable format [116].

There are a several projects providing tools to create, query, and parse DAML files [117]. Jena is an open source Java framework that has grown out of work with the HP Labs Semantic Web Programme and provides the following components:
A RDF API
- Reading and writing RDF in RDF/XML
- RDQL - a query language for RDF

Thus we use the JENA framework to query the MGED Ontology and enable users to apply the MGED vocabulary on their data.
3 Results

3.1 Overview

We have composed a bioinformatics platform for large-scale gene expression profiling comprising the following components (see Figure 8):

- **Microarray Analysis and Retrieval System (MARS)** is a web-based MIAME compliant database that allows to store and retrieve data generated during microarray production, sample preparation, hybridization, and analysis.

- **Microarray Analysis and Retrieval System Quality Management (MARS-QM)** is tightly connected to MARS using the External Application Connector Interface (EACI) and the MARS API. This web application allows to generate and manage quality parameters that are important for microarray experiments.

- **Genomics of Lipid Associated Disorders Database (GOLD.db)** focuses on providing biological pathways and allows to map gene expression data individually to each pathway.

- **Authentication and Authorization System (AAS)** provides an API and a set of JSP custom tags that allow other applications to implement user authentication and authorization with minimal coding effort.

All above mentioned components are based on a three tier architecture using the Java 2 Enterprise Edition platform (see 2.2). After evaluating several application servers we have decided to deploy our developed EJBs, JSPs, and Servlets in JBoss [118], because it has a reasonable performance and it is an open source project with a large active developer community whose participation ensures both continued innovation and product stability. Additionally, Tomcat [119] the industry’s leading and freely available Servlet container, provides the standards-based Servlet and JSP container for JBoss. All components provide a common look and feel and are accessible via a standard web browser like Netscape, Mozilla, Internet Explorer, or Safari (see figure 10(a)). We are using the Struts framework (see 2.2.4) to build these web applications, since it encapsulates best software design practices along with scaleability. To avoid both tight coupling between the client and the business objects and increased network traffic and latency, we have in most instances built our projects on the session facade and transfer object design patterns (see
Figure 8: Functional genomics environment. It comprises 1) a central authentication and authorization system (AAS); 2) a microarray database for storage, retrieval and analysis (MARS); 3) an application for managing microarray quality controls (MARS-QM) and 4) a web portal for biological pathways (GOLD.db). All the applications are web based and provide a similar look and feel. Furthermore they are all connected to the AAS facilitating a single username per user and single sign-on to all mentioned applications. Mars-QM and MARS are connected via the External Application Connector Interface (EACI). Additionally to the web interface, applications can connect to MARS using a SOAP interface.
2.2.6), although other patterns like Value List Handlet patterns have been implemented [92]. Moreover MARS, MARS-QM and GOLD.db share Oracle [120] as their EIS to persist data. Currently, the AAS uses the freely available MySQL [121] database as its data backend, but it would be possible to use Oracle or other DBMSs instead.

### 3.2 MARS

The objective was to develop a web-based and MIAME compliant microarray database that allows several institutions the acquisition, management, and retrieval of all microarray production and experiment data in a scaleable and performant way. The analysis of stored data should be facilitated by well established applications that can connect to MARS through web-services and application interfaces.

Based on this objective we first have carefully elaborated the microarray workflow that is typically carried out. Every step in the workflow was enhanced with the parameters necessary for reproduction. Quality control milestones have been introduced to the workflow at critical positions to ensure high quality data. Additionally possible shunt pathways have been entered. The complete microarray workflow is shown in figure 9.

This elaborated workflow built the basis for the design of a database model to store MIAME compliant microarray data. The design was influenced by already existing microarray databases such as BASE [40], Tigr Madam [42], and the repository ArrayExpress [27]. To address the needs of many laboratories which spot their own microarrays, MARS includes a fully functional and generic array production LIMS. It manages microtiter plates, slide types and their coating, array design spotted onto an array batch, and slides pertaining to an array batch. The flexible and generic database design facilitates to map the steadily changing laboratory plate handling workflow by simply tagging plates with *types* and *events*. Additionally, each plate can be assigned to a library, which designates the organism and contains details about the cloning vector, forward and reverse primer in addition to the standard molecule annotations like gene-name, accession-number, unigene-number, sequence and sequence-length (see figure 10(b)).

To use the array production LIMS a user has first to upload microtiter plates into the system and enter certain manipulations such as PCR amplification of plasmid stocks or purification of PCR products. After all necessary plates have been entered, they can be chosen to set up a spotting run. Therefore a file is generated and prepared for download which includes a list of all spotting plates comprising one spotting run and their molecules per well. This file is utilized by the spotting robot software to generate an array design file. After the spotting run has been
DESIGN AND DEVELOPMENT OF A BIOINFORMATICS PLATFORM FOR LARGE-SCALE GENE
EXPRESSION PROFILING

Figure 9: Microarray workflow.
completed, the array design file has to be uploaded into MARS. Then an array batch has to be created in MARS, and all slides spotted by this spotting run have to be pooled to this array batch. Additionally, important parameters regarding the spotting run such as temperature, duration, or humidity can be assigned to this array batch. Barcode tracking is employed for plates as well as for arrays to reduce manual input errors. Laboratories that are using commercial arrays can upload the array design of their array instead and define an array batch afterwards.

Labor-intensive tasks such as uploading an array design file would block the user interface for several minutes. To avoid such latencies the asynchronous programming paradigm has been applied on this kind of jobs and implemented using message driven beans (see 2.2.1).

Besides using MARS as a LIMS it can also be used as a web based microarray lab book. Samples can be annotated in a user-customizable way. It allows to annotate biological descriptions such as the source and characteristics of a sample (e.g. tissue and disease), any genetic and chemical manipulation and stimulation. Performing such annotations in free text fields would cause problems like large undefined vocabularies and would make them difficult to query [39]. Thus we provide three different annotation types: 1) enumeration enabling the usage of defined vocabularies or ontologies, 2) numbers to allow scoring and counting and 3) free text (see figure 11(a)). These annotated samples will be linked to the extract, where the lab worker can annotate...
DESIGN AND DEVELOPMENT OF A BIOINFORMATICS PLATFORM FOR LARGE-SCALE GENE EXPRESSION PROFILING

(a) List of entered sample annotation types. The box above the table allows to select the viewable columns.

(b) Extract edit page

Figure 11: MARS sample annotations and extract manipulation pages.

The extraction method, protocol, concentration, purity and quantity (see figure 11(b)). The labeled extract stores information on the used extract quantity, the label and the labeling protocol. The hybridization page archives parameters regarding the hybridization tool and method and links to the used labeled extracts (see figure 12(a)). In comparison to several other popular microarray databases [40, 122, 123, 124] MARS can handle any number of labeled extracts and thus allows the storage of multi color experiments. The resulting images from hybridized scanned slides can be uploaded to MARS and added to a hybridization. It is noteworthy that a hybridization can have several image sets with images of different scanner settings. After analyzing the images the resulting raw datasets can be uploaded and added to the appropriate image set. Again it is possible to add several different raw datasets, that are analyzed with different program settings, to an image set (see figure 12(b)).

A set of hybridizations can be assembled to an experiment by using the experiment page. The detailed description and annotation can be conducted by using the MAGE Ontology to specify the perturbational, methodological, epidemiological design, and biological properties (see figure 13(a)). The exact experimental design and the relations within hybridizations can be described by defining experiment-classes and associating raw datasets to these classes. A class denotes a
Figure 12: MARS hybridization user interface.

Figure 13: MARS experimental design.
subset of a whole experiment. For example one single time-point out of a time-course experiment represents one class, containing all raw datasets belonging to this time-point. Within a class raw datasets can be paired and flagged as dye swapped hybridizations.

Raw datasets can be viewed and queried for all stored parameters. As the spot is the central element of a microarray experiment, the raw dataset page displays a thumbnail as well as a magnification of the spots if a false-color image is available. This feature allows at any time in the analysis process to go back and look at the spot easily.

Figure 14: MARS raw dataset including the query tool. The spots are cut out from the uploaded false color image and can be zoomed by positioning the mouse cursor on the desired spot. The left frame provides a tree to navigate through the current experiment.

Anyhow, building up an experiment must not necessarily be an physically conducted experiment. The user can assemble any hybridizations, place them into classes and analyze these so called in-silico experiments.
Any file that has to be imported, linked, or used has to be uploaded to MARS first. Thus every user can upload data to the database at the place where this data has been generated. Afterwards this data can be analyzed by the users at their accustomed work place without having to use another central storage system. These uploaded files are stored in the local file system of the operating system where MARS has been installed. Additionally links to these files are stored in the relational database to prevent the deletion of already imported, linked, or used files.

Besides filing typical microarray data such as arraydesign, hybridization data, and raw datasets MARS allows and solicits to store protocols that have been used during microarray production or analysis. When uploading a protocol, the user interface is requesting two types of protocol files. First a text file, which is easily exportable via MAGE-ML is requested. Additionally the user is allowed to upload the same protocol in a pdf or word processing file format. Several pages like sample-, extract preparation or hybridization allow to link to these protocols.

To complete the list of possible user interactions MARS also allows to store used hardware like scanners or microarray robots and software like image analysis, normalization, or clustering software. Finally there is the possibility to store providers of PCR products, plates, slides, slide coating etc. All the entered data is stored user and/or institute centric.

3.2.1 Generic File Parser

One of the most important parts for the acceptance of a database is the data import interface. To allow not only the import of proprietary file formats from software packages like GenePix [57] or Scanalyze [125] we have implemented a generic and user defineable parser that allows to read and upload any tab delimited file into MARS. Therefore the user has to assign the file columns to the appropriate database fields. This can be easily done using the web interface. The user has to navigate to Define File Format and upload a template file. The generic fileparser reads the first couple of lines and presents them to the user. Now the user has to define the data header of that file. The data header is the line that describes the contents of the columns. Afterwards the columns have to be correctly assigned to the data fields to which the file data has to be mapped. This mapping is stored in the database and every time a user uploads a file for parsing the file is scanned for all available data headers. If a header is found the appendant mapping is fetched and the file can be parsed. MARS allows to define file formats for importing plates, raw and transformed datasets, and array designs.

This generic file parser has been developed as an separate project and can thus be used in other applications, too. Additionally, the source of mapping the data fields must not necessarily be
DESIGN AND DEVELOPMENT OF A BIOINFORMATICS PLATFORM FOR LARGE-SCALE GENE EXPRESSION PROFILING

a database, but can also be defined in Java classes or other sources like XML files. Another feature of this file parser is the ability to parse file header information. Technically, all data that is read from the file is given to an Java object that has been previously designed by the application that is using the parser. Subsequently the parser stores the data into this object by using reflection [126].

3.2.2 External Application Interfaces

Building a platform for functional genomics does not mean building a self-contained application. It is important to facilitate the communication or export of data with other applications. MARS provides three different types of interfaces that permit communication.

**External Application Connector Interface (EACI):** We have developed a JSP custom tag library and a Java API that allows other web based applications to create dynamic links to their data. Therefore an application has to lookup the EACI session bean and register itself. Then this application has to define the source of data (e.g. plates) from where the link should be created and must define the exact HTTP link to its data. Additionally the external application is allowed to define separate icons for already connected links and for those that can be created. From now on, this application can easily create dynamic links from MARS to its data by invoking the `setItem` method from the EACI session bean (see example code 1 for implementing the EACI in other applications).

The MARS user interface is dynamically displaying the links to all former registered applications (see figure 15). The dynamic links are generated by the developed JSP custom tag library on the JSP.

**Webservices:** In order to provide users access to MARS with software they are familiar with (e.g. BioConductor [44], Matlab [127]), MARS provides a well defined Simple Object Access Protocol (SOAP) interface and a JAVA library software developers can use, to extend their programs with data access functionality. These interfaces allow programs after minor software adaptations to authenticate against MARS, to browse own and shared datasets, filter and download raw data, and to upload the transformed datasets into MARS. If there is no firewall
between the client software and MARS, the applications can call public accessible methods via the Remote Method Invocation (RMI) interface.

**MAGE-ML Export:** The Microarray Gene Expression Markup Language (MAGE-ML) (see 2.6) has emerged as a language to describe and exchange information about microarray based experiments [107]. MAGE-ML is based on XML (eXtensible Markup Language) and can describe microarray designs, microarray manufacturing information, microarray experiment setup and execution information, gene expression data and data analysis results. Using the Java MAGE-STK (see 2.6) MARS is able to export samples, extracts, labeled extracts, arraydesigns, raw data sets, hybridizations (figure 16), or whole experiments including several hybridizations. The resulting files can be used to submit the microarray data to ArrayExpress or other microarray databases that provide MAGE-ML import.

### 3.3 MARS-QM

We have developed a powerful quality management system to ensure high quality data and to allow the detection of possible sources of errors. This system is based on the standard quality control procedures conducted during the microarray production as well as during the sample preparation, RNA extraction and hybridization process. To control the quality of PCR and purified PCR products generated during probe production, authorized users can upload gel images
and analyze the bands according to a predefined schema (see figure 17(a)). Based on this schema
good and bad PCR products can be identified later as the source of bad or missing spots on a
slide and their quality annotation can be viewed by any user. Another feature is the evaluation
of the quality of array batches or single arrays. Slides can be scanned after fixation and/or after
staining and parameters like spot walking, or the number of missing spots are used to determine
the slide quality.

In addition to the array production quality controls, it is also necessary to check the quality
of samples and its extracts. Data gained from the Agilent Bioanalyzer\textsuperscript{TM} or gel images can be
uploaded and analyzed either automatically (Bioanalyzer\textsuperscript{TM} file) or manually (gel images) (see
figure 17(b)). The labeled extract can be measured with a spectrophotometer to assess the effi-
ciency of dye incorporation. The results of these measurements can be entered in MARS and
the corresponding efficiency will be calculated automatically.

Finally the quality of a hybridized slide is analyzed by extracting and displaying several statisti-
cal parameters from the raw data result file and by examining positive and negative controls
printed on the slide.

MARS-QM has been also developed using the J2EE platform, but the implementation was
based on a Model Driven Architecture (MDA) using the Unified Modelling Language (UML).
The OMG’s UML [128] specification defines UML as a graphical language to specify, visualize,
Figure 16: MARS hybridizations page, red circle shows a link to export a hybridization into MAGE-ML format.

(a) A gel image can be scored and associated to a plate.

(b) Bioanalyzer analysis for a given extract.

Figure 17: MARS-QM User Interface
and document models of software systems, including their structure and design. In addition to business modelling, UML could be used for modelling other non-software systems [129, 130]. Although MARS-QM is not tightly integrated into MARS, the user is not aware of running a separate application. This became possible due to the very similar user interface, look and feel, and links that are navigating the user to the appropriate quality management data. These links are build dynamically by the EACI and allow therefore the extension of the quality management without amending the MARS code. Due to the fact that MARS-QM is using the same central usermanagement as MARS, the user gets logged in automatically. MARS-QM is an excellent example for the usage of the EACI.

3.4 GOLD.db

The GOLD.db (Genomics of Lipid-Associated Disorders Database) was developed to address the need for integrating disparate information on the function and properties of genes and their protein products that are particularly relevant to the biology, diagnosis management, treatment, and prevention of lipid-associated disorders. The database provides a reference for pathways and information about the relevant genes and proteins in an efficiently organized way. The main focus was to provide biological pathways with image maps and visual pathway information for lipid metabolism and obesity-related research. For each element in the pathway, specific information exists including structured information about a gene, protein, 3D-structure, gene regulation, function, literature, and links. Addressing the need to draw or import pathways from KEGG we have developed a platform independent Java application called PathwayMapper.

3.4.1 PathwayMapper

PathwayMapper is a visual application for drawing and annotating biological pathways. It integrates the possibilities of charting elements with different attributes (size, color, labels), drawing connections between elements in distinct characteristics (color, structure, with, arrows) as well as adding links to molecular biology databases, promoter sequences, information on the function of genes or gene products and references [46]. To simplify the tedious process of drawing, it allows to import pathways which are supported by KEGG. KEGG provides pathways in the form of an XML standard called KGML (KEGG Markup Language) which is an exchange format of the KEGG graph objects. Due to a certain incompleteness regarding the interconnection of pathway elements some additional changes have to be performed on the imported pathways.
Afterwards, the imported KEGG pathways can be updated with organism specific gene identifiers depending on the enzyme or locus identifiers provided by KEGG. Finally the pathway can be stored in a database. Therefore PathwayMapper connects to the application server via RMI (Remote Method Invocation) or via SOAP, which offers the possibility to tunnel through firewalls using the HTTP protocol, and executes the business methods to store the pathway (see figure 18).

Data from microarray experiments can be mapped onto a specific or onto all pathways. PathwayMapper can map either single or timecourse experiments. When mapping a dataset to all pathways, PathwayMapper will display a list with all pathways containing mappable genes sorted by their occurrence.

### 3.4.2 Web Interface

Because biologist prefer to use web applications rather than applications they have to install locally on their computer, we have enhanced the PathwayMapper with a web interface. Building
on the business methods in the application server, the Struts framework provides an easy to use interface to upload expression data and consequently map it to a pathway. Basically the web interface provides the same features as the stand-alone application besides drawing new pathways.

Additionally to the pathway database analytic and data mining tools, reagents, protocols, videos, references, and links to relevant genomic resources were included in GOLD.db. [131].

3.5 Authentication and Authorization System

Developing a usermanagement for every application is a tedious and error-prone work. Besides potentiating the possibility of severe security holes, the user has to remember several different usernames and passwords. Therefore a central usermanagement is simplifying account management and minimizing the risk of security holes.

We have developed an authentication and authorization system (AAS) that integrates a central management for users, applications and application based user access levels. The whole administration can be accomplish by a web based user interface. This user interface allows to create users, groups, resources, access control lists (ACL) and enables to assign users and groups to ACLs. Furthermore it allows to assign administrators for distinct applications. Consequently, these administrators can create users and groups and assign these to their application. Moreover they can create resources and ACLs for their application.

The developed AAS is based on the open source project OpenSymphony [132].

3.5.1 Client Connector Interface

Newly developed or existing applications can easily integrate the AAS by using the Client connector interface (CCI). The CCI consists of a custom tag library and a Java API that allows applications to connect to the user management system in order to validate user authentications and to administer application specific data. The connection to the AAS can be established by using HTTP, HTTPS, or RMI. However, HTTPS should be used in the production environment.

**Authentication:** Users need to provide their username and password during an authentication process. This information is compared with the stored values in the database. Additionally authentication rules can be configured for every application. These rules contain all precepts a user must accomplish before access is granted. After a successful login process, users get the
status of authenticated users and a unique identifier is generated for each user and sent back to the client application. Each client stores the received user associated id and transmits it back with every request to a protected resource. The AAS server compares the client and the server-side generated authentication id to prove an earlier valid login process. Only if both id’s are equal user access rights are processed. Schematically a typical authentication process can be described as (see figure 19):

Applications can implement the AAS authentication as shown in listing 2.

![AAS Authentication Diagram](image)

**Figure 19:** The schematic authentication process of the AAS system. The client sends an authentication request to the AAS server. After the correct authentication the server transmits the generated authentication id to the client. With every request to a restricted resource the client has to send the authentication id. Based on the given rights the server can grant or deny the access.

In order to guarantee that a user has logged in successfully before displaying a JSP, the developer can use the

```xml
<login:checkLogin applicationName='<%=WebConstants.MYAPPLICATIONNAME%>'/>
```

tag from the provided `login` tag library (see section 2.2.3). To check a successful login in a Java application the method call
Listing 2: Sample code: Client Connector Authentication

```java
import at.tugraz.genome.usermanagement.serverconnection.*;
import at.tugraz.genome.usermanagement.cookie.*;
ServerConnection serverConnection = null;

try {
    // connecting to the AAS server
    serverConnection = ServerConnectionManager.getInstance().getConnection(WebConstants.MYAPPLICATIONNAME);
    String user="["+loginForm.getUsername()+"] " *;
    if (serverConnection != null) {
        AuthenticationToken atoken = null;
        // authenticating against the AAS
        atoken = serverConnection.authenticateUser(WebConstants.MYAPPLICATIONNAME,
                                                loginForm.getUsername(),
                                                loginForm.getPassword(),
                                                WebConstants.MYAPPLICATIONKEY);

        // checking authentication result
        if (!atoken.isError()) {
            log.debug(user+" Authenticated correct");
            ExtendedUserVO vo = serverConnection.getUserVOExtended(loginForm.getUsername(),
                                                                    atoken.getAuthenticationID(),
                                                                    WebConstants.MYAPPLICATIONNAME,
                                                                    WebConstants.MYAPPLICATIONKEY);

            // setting the cookie
            CookieManager.createCookie(request, response, vo,
                                        loginForm.getUsername(),
                                        atoken.getAuthenticationID());

            // setting results in session
            request.getSession().setAttribute(UserManagementConstants.USER, vo);
            request.getSession().setAttribute(UserManagementConstants.LOGGEDUSER, vo);
            request.getSession().setAttribute(UserManagementConstants.APPLICATIONNAME, WebConstants.MYAPPLICATIONNAME);
            request.getSession().setAttribute(UserManagementConstants.AUTHENTICATION_ID, atoken.getAuthenticationID());
        } else {
            log.error(user+" Not authenticated: " + atoken.getErrorMessage());
        }
    }
}

catch (Exception ex) {
    log.error("error: no server connection"+ex);
}
```
import at.tugraz.genome.usermanagement.login.Login;
Login.checkSuccessfullLogin(request, WebConstants.MYAPPLICATIONNAME);

can be used instead.

**Authorization:** The developed AAS provides a custom tag library to grant or deny access to parts of web pages. This technology enables web developers to grant access to web pages according to the given access rights of users. Examples for the usage of the hasPermission tag are:

<!-- Protecting code chunks based on the access levels -->

```xml
<permission:hasPermission resourceKey="plates" accessLevel="R">
    Here is the protected part of the webpage.
    This body is only accessible to users with proper access rights.
    In this case users having read (R) permissions on resource plates will see the enclosed part,
    the others will skip this part in a JSP.
</permission:hasPermission>

<!-- Disabling a button based on the access levels -->

```xml
<input type="button"
    <permission:hasPermissionForButton resourceKey="plates"
        accessLevel="N"
        trueValue=""falseValue="DISABLED"/>
        value="Create" />
```

In analogy to the tag it is possible to grant permissions to Java functions or classes using:

import at.tugraz.genome.usermanagement.serverconnection.cache.PermissionCache;
PermissionCache.hasPermission(userName, authenticationID, applicationname, resourceKey, accessLevel, applicationkey));

**Single Sign-on:** Single sign-on (SSO) is a mechanism whereby a single action of user authentication and authorization can permit a user to access all web applications where he has access permission, without the need to enter the password multiple times.

The AAS implements SSO for applications that are located in the same subdomain. After a user has successfully authenticated a cookie (see 2.2.5) is stored with the encrypted username and authentication id. Everytime a user navigates to a web application he has not previously logged in, the AAS reads the cookie and authorizes the user to enter the web application if the according rights have been assigned. When a user logs out the cookie will be destroyed and the
3.6 Transcription profiling of NCI-H295R cells treated with fat cell-conditioned medium

The RNA has been extracted from the treated cell by the Department of Endocrinology, University Medical Center, Heinrich Heine University, Düsseldorf, Germany. This RNA has been aliquoted into four equal parts and labeled to hybridize two dye swap experiments using previously spotted human oligo chips (HOC) with 37632 elements. The production of these arrays has been facilitated by MARS. Especially the array production LIMS has been used for tracking the PCR products and for preparing the spotting run file containing all plates and their corresponding wells in a correct order. The MARS laboratory note book has been utilized to record all steps conducted during labeling and hybridization. After the scanned images have been evaluated, the resulting raw dataset and the images have been uploaded into MARS. ArrayNorm 1.6 has been connected to MARS using the SOAP web service to download and normalize the datasets.

Since thousands of elements are analyzed in parallel it is very important to check the overall quality of the microarray data. As representative example the results of 1st experiment for FCCM treated cells were used to show the consistency and quality of the data in several ways (see figure 20). After normalization there was no intensity dependency of the log ratios, the distribution of the log ratios was centered around 0 and in a certain range related to a normal distribution. The consistency between technical replicates (dye swap) became evident by analyzing the regression plot.

After filtering, normalization, and averaging over 2 experiments data were screened for genes that had no missing values and had at least 40% regulation in one of the experiments. The resulting 2919 genes were subject of further analysis and can be described as follows: 1386 genes were regulated exclusively by ANGII, 714 exclusively by FSK, and 136 exclusively by FCCM. Also, there were 446 genes that were regulated by FSK and ANGII, 77 genes regulated by FCCM and ANGII and 77 regulated by FCCM and FSK. A group of 120 genes was regulated by all three experiments. The given abbreviation code for the classes and the number of regulated genes is summarized in table 3. A preliminary study for the functional annotation was undertaken to derive the involvement in specific biological processes. For 1231 genes out of the 2919 selected genes a gene ontology assignment for biological processes could be
Figure 20: Visualization of the distribution and the quality of the microarray data after normalization. As representative example the results of 1st experiment for FCCM treated cells are shown.
DESIGN AND DEVELOPMENT OF A BIOINFORMATICS PLATFORM FOR LARGE-SCALE GENE EXPRESSION PROFILING

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Nr of regulated genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>genes regulated just in the ANGII experiment</td>
<td>1386</td>
</tr>
<tr>
<td>010</td>
<td>genes regulated just in the FSK experiment</td>
<td>714</td>
</tr>
<tr>
<td>100</td>
<td>genes regulated just in the FCCM experiment</td>
<td>136</td>
</tr>
<tr>
<td>011</td>
<td>genes regulated in the FSK and ANGII experiment</td>
<td>446</td>
</tr>
<tr>
<td>101</td>
<td>genes regulated in the FCCM and ANGII experiment</td>
<td>77</td>
</tr>
<tr>
<td>110</td>
<td>genes regulated in the FCCM and FSK experiment</td>
<td>40</td>
</tr>
<tr>
<td>111</td>
<td>genes regulated in all experiments</td>
<td>120</td>
</tr>
</tbody>
</table>

Table 3: Classes of genes regulated more than 40% in at least one class.

found. They could be divided in physiological process (44.7%), cellular process (28.8%), regulation of biological process (12.5%), and development (10.2%). Looking a level deeper into physiological process revealed that 686 genes (38.8%) were in the group of metabolism, 451 (28.5%) in cellular physiological process, 195 (11.0%) in regulation of physiological process, 177 (10.0%) in response to stimuli, 162 (9.2%) in organismal physiological process, and 66 (3.7%) in death. Finally the distribution in the metabolism group can be described as following: nucleobase, nucleoside, nucleotide and nucleic acid metabolism contained 286 genes (20.2%), protein metabolism 266 genes (18.8%), biosynthesis 137 genes (9.7%), phosphorus metabolism 91 genes (6.4%), lipid metabolism 74 genes (6.2%) and, catabolism 74 genes (5.2%). The detailed distribution of the physiological process, the metabolism and lipid metabolism can be seen in figure 21. Mapping the genes of the seven classes individually to the GO, rendered in general the same percentage distributions, but no genes mapped to lipid metabolism in the classes 101 and 110. In addition to GO analysis, microarray data were analyzed in the context of some representative pathways. The current version of PathwayMapper contained 113 pathways containing 3195 unique mappable RefSeq Ids and onto this 331 genes of the dataset could be uniquely mapped. Relative gene expression levels were mapped to corresponding elements (enzymes) in all available pathway diagrams. The used pathway diagrams were derived from the KEGG database and adapted for PathwayMapper. The mapping summary showed that for example 50% of 12 possible mappable genes could be mapped to the C21-Steroid Hormone Metabolism pathway or 9 out of 35 mappable genes mapped to the Androgen and Estrogen Metabolism pathway. The largest amount of genes (45) mapped to the MAPK Signaling pathway that contains 444 mappable RefSeq Ids. The elements in the pathways were color coded according to the log ratios for every experiment. In figure 22 the mapping of relative expression levels to the Androgen and Estrogen Metabolism pathway is illustrated schematically. For example in the Androgen and Estrogen Metabolism pathway a induction of the expres-
Figure 21: Detailed distribution of genes mapped to the Gene Ontology classes for physiological process, metabolism and lipid metabolism
Figure 22: Androgen and estrogen metabolism pathway with the relative gene expression levels mapped to corresponding elements.
sion of a number of enzymes reactions including the steroid delta-isomerase, steroid 11 beta-
monooxygenase, alcohol sulftotransferase, arylsulfatase, glucuronosyltransferase, and steroid sulftotransferase was observed. Additionally and as described for the GO annotation we mapped
the classes separately to all the available pathways. This revealed for example that in class 111 4
genes out of the total 6 mapped to the C21-Steroid hormone metabolism pathway. The summary
of all mappings is shown in table 4.
DESIGN AND DEVELOPMENT OF A BIOINFORMATICS PLATFORM FOR LARGE-SCALE GENE EXPRESSION PROFILING

Table 4: Number of elements mapped to a specific class in a pathway
4 Discussion

The functional annotation and identification of genes involved in the development and progression of complex diseases is a cumbersome and non-trivial task. DNA microarrays allow generating a composite picture of the expression profile of the cell and are widely used in basic research as well as in clinical medicine and pharmacogenomics. Therefore the development of a powerful and easy to use bioinformatics platform for storing and analyzing microarray data was the primary objective of this thesis.

There are already several academic and commercial software products available that claim to provide most of the required functions necessary for a functional genomics environment, but after evaluating most of these packages several drawbacks became evident. On the one hand commercial packages lacked the possibility of extending them with programs and features developed by the bioinformatics community and on the other hand open-source academic packages were written in programming languages like PHP or Perl that lack scalability. Moreover some packages were based on a fat client architecture and open source packages were not provided by concurrent versions system (CVS) for continuous code updates or contributions. Based on these evaluations we decided to develop our own system using the Java 2 Enterprise Edition (J2EE) platform and composing our database design on already established ideas and designs. The J2EE platform enables the development of scaleable multitier enterprise applications (see 2.2). As data tier the well established and very performant Oracle database management system has been chosen. The middle and web tier has been built on the open-source application server JBoss and the open-source framework Struts, because these projects have a large active developer community whose participation ensures both continued innovation and product stability. Building the development of such a large system on the newest technologies bears some risks, but the recent rollout of Java Server Faces [133] from Sun, which architecture is very similar to Struts, reinforced the decision. MARS has been developed by modeling just the relations of EJBs and coding the data logic, business logic, and web interface in a custom way for every bean and web page. In contrast MARS-QM has been developed using a model driven architecture (MDA) design which generates most of the code by using previously defined templates. Of course, there are several advantages for justifying both software designs, but a great advantage of the MDA is that technology changes can be easily adopted by revising the template instead of changing the whole code.
Fundamental for the acceptance of a database or an application are its data interfaces. In principle two types of data interfaces for human computer interactions can be distinguished. Standalone applications allow better program-user interactions while having the drawback that several or even very old versions can be in use. Additionally the user has to have a powerful desktop machine if calculations or algorithms become more complicated. On the other hand web based applications can be easily used on every computer without any installation effort and the same and the newest version are provided to all users. Less powerful computers can be used, because all application and business logic is executed on the server. To ensure data integration as well as pleasing usability the core data manipulation and storing functions have been developed in a web based technology and for data analysis it was harked back on already developed mature standalone applications providing excellent usability. This fusion of using web based and standalone applications connected to the latest application server technology brings about the usability advantages of both technologies and additionally the advantages of the J2EE environment, namely data integrity, multi user accessibility, and scaleability.

Excellent usability does not only account for primely data interfaces, because the ability to easily import data and the availability of well defined application interfaces is also crucial. Typically, every institution has its own favorite, mostly self tailored, data analysis applications with proprietary and varying data formats. To meet this requirements, MARS provides several data and application interfaces. For importing data, users can define parsers for any tab delimited file. Every time a user is uploading data, MARS tries to find an appropriate parser based on the previous user defined parser settings (see 3.2.1). Once the data is uploaded and stored in the database tables, the data can be analyzed using applications like ArrayNorm, Genesis, and PathwayMapper. Even though these tools are very powerful, there are of course scientist who would like to analyze their data with software they have developed themselves or they are familiar with. These scientists can after some slight adaptations of their software authenticate and down or upload data using the same SOAP web service data interface the earlier mentioned applications are using (see 2.5 and 3.2.2). Some of the evaluated microarray database applications provide the possibility to write plug-ins and thus allow to access and use the stored data, but this approach assumes that the user has to have knowledge about the programming language used for the development of the database. In contrast, providing a web service interface allows through its widespread and platform independence to be implemented in well-established programming languages and even in tools like Matlab or R [134]. Additionally to the web service interface, existing web applications can be plugged-in using the EACI (see 3.2.2) that enables the linkage
between data provided by the plugged-in application and data stored in MARS. Moreover this interface facilitates an enhancement of MARS without amending the MARS core code. In order to prove this concept the development of a self-contained web application called MARS-QM has been initiated. MARS-QM integrates several quality measurements performed during the microarray production as well as during the sample preparation, extraction, and hybridization process to assure high-quality data, to understand or optimize lower value data and to provide the ability to trace back all conducted quality control steps (see 3.3).

Besides sharing microarray experimental data among MARS users, MARS facilitates the export of array designs, hybridizations and experiments into the common exchange format MAGE-ML (see 2.6 and 3.2.2) by simply clicking the appropriate button in the web interface. This feature capacitates to easily share and publish high quality, well annotated data within the life sciences community by uploading these generated files into public repositories like ArrayExpress. In order to provide a standardized annotation of the data the MGED Ontology is used (see 2.7).

All the above mentioned interfaces provide the basis that enables MARS to be fully integrated into the microarray workflow. The web based user interface allows to store information regarding the sample preparation, RNA extraction, labeling and hybridization procedure, to upload array designs, hybridization images, raw and normalized datasets, and to define experiments where the experimental design of a set of hybridizations can be annotated. The SOAP web service and the EACI provide mighty interfaces for the integration of external or third party applications. MARS-QM provides the tracking of conducted quality control steps to assure high-quality data microarray data. Finally, the ability to export all the valuable data to MAGE-ML allows to share the gained results with other researchers.

For understanding the function of the cell knowledge about biological pathways, their components, and the interaction between the components is crucial. Consequently, modeling, editing, and annotating biological pathways is an important issue for the organization of knowledge as well as for pathway analysis and computation. Basically, there are three types of pathway drawing approaches: auto-layout, manual (interactive) drawing, or a hybrid of these two approaches [34,135,136,137,138]. PathwayMapper (see 3.4.1) was implemented as an interactive drawing tool, because this type fits best for the construction of pathway diagrams in a visual way based on available knowledge, and the annotation of the components and interactions between them. In addition to model pathways, PathwayMapper allows to import pathways from
KEGG and expression data form microarray experiments can be mapped onto a specific or onto all available pathways.

Avoiding unauthorized database access in a multi-user environment and controlling user access is a crucial criterion for the acceptance of databases. Additionally it is important that a platform uniting various applications provides one username-password combination for all applications. Furthermore single sign-on (SSO) increases the usability of such a platform. SSO means that a user who has successfully logged into an application can change to another application (for example by following a link) without having to enter the username-password combination again. Our developed Authentication and Authorization System (AAS) (see 3.5) affords to manage authorization and authentication for any number of applications and provides SSO to all web based applications. Single sign-on for standalone applications was also considered, but this would mean that every computer has to install and run a small program in the background in order to provide an appropriate SSO interface. Hence, to keep the AAS simple, this considerations were not put in action. Currently the consolidation of applications and Windows and Unix accounts in the AAS has been finished. The importance and necessity of the AAS is made clear by the fact that Sun Microsystems has released the Sun Java System Access Manager [139] in the same year. All applications developed at the Institute for Genomics and Bioinformatics have currently integrated the AAS.

A preliminary microarray study (see 3.6) showed that the developed platform potentiates functional annotations of genes by mapping the dataset either to the GO or to biological pathways. As an example for the biological validity of the experiment melanocortin 2 receptor which was highly upregulated in all three experiments can be named. The up-regulated adrenocorticotropin receptor (MC2-R) by both adrenocorticotropin and angiotensin II through the activation of protein kinase A and protein kinase C pathways has been described in [140].
5 Conclusion and Outlook

The MARS database design, state-of-the-art software technology, well designed user interface, and its powerful application interfaces provide a capable tool for storing, retrieving and analyzing multi color microarray data. The unique combination web-based and standalone applications connected to the latest powerful application server technology facilitate MARS users to transform microarray data into valuable knowledge.

Since no software package is complete and satisfies all the needs, we are still working on several new features. Because information attached to molecules is changing quickly, the possibility to update and enhance the information tagged to a molecule is currently implemented. Changing this information on the molecule level may affect already existing results. In order to avoid such precarious alterations, the user should be able to update the molecule information for each experiment separately instead of replacing the initial molecule information. In order to expand the microarray information to the protein level we are planning to automatically map ESTs to proteins. A page where all quality control steps can be viewed together in a quality control matrix will be implemented, to gain an overview of the quality of spotted arrays, biological samples and hybridizations. Currently the Authentication and Authorization System (AAS) does not provide libraries for other programming languages like Perl or C++, therefore the development of such client connector libraries would enhance the potential of the AAS.

Summarizing, an extensible platform that is fully integrated in the microarray workflow has been developed. It allows to manage slide production, to store data ranging from sample preparation to hybridization, track the conducted quality control measurements, and to analyze the gained results using either the provided applications or connecting any custom applications or suits like R-Bioconductor with the aid of the webservice interface. Finally, the ability to export data to MAGE-ML allows to share the gained results with other researchers. The current functionality and the ongoing and planned extensions will make this platform indispensable for large scale gene expression profiling and ultimately functional genomics.
References


### Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAS</td>
<td>Authentication and Authorization System</td>
</tr>
<tr>
<td>ACL</td>
<td>Access control list</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASP</td>
<td>Active Server Pages</td>
</tr>
<tr>
<td>BLAST</td>
<td>Basic local sequence alignment tool</td>
</tr>
<tr>
<td>BMP</td>
<td>Bean Managed Persistence</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge coupled device</td>
</tr>
<tr>
<td>cDNA</td>
<td>complementary DNA</td>
</tr>
<tr>
<td>CGI</td>
<td>Common Gateway Interface</td>
</tr>
<tr>
<td>CMP</td>
<td>Container managed persistence</td>
</tr>
<tr>
<td>CVS</td>
<td>Concurrent versions system</td>
</tr>
<tr>
<td>DAML</td>
<td>DARPA Agent Markup Language</td>
</tr>
<tr>
<td>DBMS</td>
<td>Database management system</td>
</tr>
<tr>
<td>DCL</td>
<td>Data control language</td>
</tr>
<tr>
<td>DDBJ</td>
<td>DNA Data Bank of Japan</td>
</tr>
<tr>
<td>DDL</td>
<td>Data definition language</td>
</tr>
<tr>
<td>DML</td>
<td>Data manipulation language</td>
</tr>
<tr>
<td>DNA</td>
<td>Deoxyribonucleic acid</td>
</tr>
<tr>
<td>DTD</td>
<td>Document type definition</td>
</tr>
<tr>
<td>EACI</td>
<td>External application connector interface</td>
</tr>
<tr>
<td>EIS</td>
<td>Enterprise Information System</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise Java Bean</td>
</tr>
<tr>
<td>EMBL</td>
<td>European molecular biology laboratory</td>
</tr>
<tr>
<td>EST</td>
<td>Expressed sequence tag</td>
</tr>
<tr>
<td>GO</td>
<td>Gene Ontology</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical user interface</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Secure Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>J2EE</td>
<td>Java 2 Enterprise Edition</td>
</tr>
<tr>
<td>JDBC</td>
<td>Java database connectivity</td>
</tr>
<tr>
<td>JMS</td>
<td>Java Message Service</td>
</tr>
<tr>
<td>JSP</td>
<td>Java Server Page</td>
</tr>
<tr>
<td>KEGG</td>
<td>Kyoto encyclopedia of genes and genomes</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>LIMS</td>
<td>Laboratory information management system</td>
</tr>
<tr>
<td>mRNA</td>
<td>Messenger ribonuclein acid</td>
</tr>
<tr>
<td>MAGE-ML</td>
<td>Microarray gene expression markup language</td>
</tr>
<tr>
<td>MAGE-OM</td>
<td>Microarray gene expression object model</td>
</tr>
<tr>
<td>MAGE-STK</td>
<td>Microarray gene expression software toolkit</td>
</tr>
<tr>
<td>MARS</td>
<td>Microarray Analysis and Retrieval System</td>
</tr>
<tr>
<td>MARSQM</td>
<td>Microarray Analysis and Retrieval System Quality Management</td>
</tr>
<tr>
<td>MDA</td>
<td>Model Driven Architecture</td>
</tr>
<tr>
<td>MGED</td>
<td>Microarray gene expression data consortium</td>
</tr>
<tr>
<td>MIAME</td>
<td>Minimum information about a microarray experiment</td>
</tr>
<tr>
<td>MM</td>
<td>Mismatch</td>
</tr>
<tr>
<td>MVC</td>
<td>Model View Controller</td>
</tr>
<tr>
<td>PCR</td>
<td>Polymerase Chain Reaction</td>
</tr>
<tr>
<td>PM</td>
<td>Perfect match</td>
</tr>
<tr>
<td>RMI</td>
<td>Remote Method Invocation</td>
</tr>
<tr>
<td>RNA</td>
<td>Ribonuclein acid</td>
</tr>
<tr>
<td>SAGE</td>
<td>Serial analysis of gene expression</td>
</tr>
<tr>
<td>SNP</td>
<td>Single-nucleotide polymorphisms</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured query language</td>
</tr>
<tr>
<td>SSO</td>
<td>Single sign.on</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal Description Discovery and Integration</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Service Description Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible markup language</td>
</tr>
</tbody>
</table>
Acknowledgment

Major parts of this work were supported by the Austrian Academy of Sciences and the GENAU:BIN, Bioinformatics Integration Network. I would like to express my deepest gratitude to my mentor Zlatko Trajanoski for his encouragement, visions, and believing in me. I want also express my appreciation to my colleagues and friends Robert Molidor and Alexander Sturn for their assiduousness and fervor in developing MARS with me. Further thank go to all previous members of the Bioinformatics group and people at the Institute of Genomics and Bioinformatics for fruitful discussions and support. A special acknowledgment is dedicated to the people, that have contributed to this work: Thomas Truskaller, and Christoph Thumser for the development of MARS-QM, Elmar Trost and Bernhard Mlecnic for working on the PathwayMapper, Jürgen Hartler and Dieter Zeller for assisting me in developing the AAS, Hubert Hackl and Andreas Prokesch their valuable comments during the design and development of MARS, Marcel Scheideler, Monika and Stefan Bornstein for realizing the FCCM study, and last but not least Gernot Stocker for the perfect management of our computing facility and his valuable comments regarding security. I’m indebted to my parents and Irene for accompanying me and for their support.
Design and Development of a Bioinformatics Platform for Large-Scale Gene Expression Profiling

Publications

Journals


GOLD.db: Genomics of Lipid-Associated Disorders Database

HUBERT HACKL, MICHAEL MAURER, BERNHARD MLECNIK, JÜRGEN HARTLER, ELMAR TROST, GERNOT STOCKER, DIEGO MIRANDA-SAAVEDRA AND ZLATKO TRAJANOSKI*

Institute of Biomedical Engineering and Christian Doppler Laboratory for Genomics and Bioinformatics, Graz University of Technology, Krenngasse 37, 8010 Graz, Austria

ABSTRACT

The GOLD.db (Genomics of Lipid-Associated Disorders Database) was developed to address the need for integrating disparate information on the function and properties of genes and their products that are particularly relevant to the biology, diagnosis, management, treatment, and prevention of lipid-associated disorders. The database provides a reference for pathways and information about the relevant genes and proteins in an efficiently organized way. The main focus was to provide biological pathways with image maps and visual pathway information for lipid metabolism and obesity-related research. The GOLD.db provides also the possibility to map gene expression data individually to each pathway. Gene expression at different experimental conditions can be viewed sequentially in context of the pathway. Related large scale gene expression data sets were provided and can be searched for specific genes to integrate information regarding their expression levels in different studies and conditions. Additionally, analytic and data mining tools, reagents, protocols, videos, references, and links to relevant genomic resources were included in the database. GOLD.db is available at http://gold.tugraz.at.

INTRODUCTION

The excessive consumption of high calorie, high fat diets and the adoption of a sedentary life style have made obesity and atherosclerosis major health problems in Western societies. In the USA, over 50% of the population are over-weight (BMI>25) and close to 25% are considered obese (BMI>30) (1, 2). As a consequence, a large fraction of the population is at risk to develop a broad range of common, life-threatening diseases including non-insulin dependent diabetes, various hyper-lipidemias, high blood pressure and atherosclerosis.

Vascular disease including coronary heart disease and stroke is currently the major cause of death in the United States and in other industrialized nations. At the root of obesity and atherosclerosis is an excessive deposition of neutral lipids. Adipose tissue accumulates predominantly triglycerides, whereas macrophages along the blood vessel wall mainly accumulate cholesterol and cholesteryl esters. Accordingly, a detailed understanding of the molecular mechanisms that govern the balance between lipid deposition and mobilization is fundamentally important for the prevention and improved treatment of disease. In addition to the apparent environmental components involved in the pathogenesis of disorders related to lipid and energy metabolism, a large number of studies have provided undisputed evidence that susceptibility genes contribute around 50% of the phenotype. These genes encode products involved in the cellular uptake, synthesis, deposition and/or mobilization of lipids. However, characterization of many if not most of these genes and their products remains rudimentary. Deficiencies in the current level of understanding extend to key enzymes such as important triglyceride hydrolases in adipose tissue (3) or cholesteryl ester hydrolases in macrophages, hormones, signal transduction pathways, and the regulation of the transcription of relevant genes.

While medical molecular biology traditionally associates single genes and gene products with diseases, a growing body of evidence suggests that several common disease phenotypes arise from the delicate interaction of many genes as well as gene-environment interactions. To elucidate the development of obesity and atherosclerosis, it will be necessary to analyze patterns of gene expression and relate them to various metabolic states. To discover novel genes, processes and pathways that regulate lipid deposition and mobilization, a departure from hypothesis-driven research and turn to a discovery-driven approach is necessary. The application of high-throughput technologies and genome-based analysis will provide the tools for the analysis of gene-gene and gene-environment interactions in a systematic and comprehensive manner.

To facilitate genomic research we have initiated the development of a system for storing, integrating, and analyzing relevant data needed to decipher the molecular anatomy of lipid associated disorders. In
order to provide a reference for pathways and information of the relevant genes and proteins in an efficiently organized way, we have created the Genomics Of Lipid-Associated Disorders database (GOLD.db). The GOLD.db integrates disparate information on the function and properties of genes and their protein products that are particularly relevant to the biology, diagnosis management, treatment, and prevention of lipid-associated disorders. The main focus was to provide biological pathways with image maps and visual pathway information. For each element in the pathway, specific information exists including structured information about a gene, protein, 3D-structure, gene regulation, function, literature, and links. The GOLD.db provides also the possibility to map gene expression data individually to each pathway. Additionally, analytic and data mining tools, reagents, protocols, videos, references, and links to relevant genomic resources were included in the database.

**DATABASE DESCRIPTION**

**PATHWAYS**

In order to construct the biological pathways of interest, we have developed a pathway editor. This drawing tool provides the possibility to draw elements – typically representing a gene as part of the pathway – and the connection between those elements. The benefit of this tool is that information can be appended to each element via an input mask. This information can be accessed by clicking on the corresponding element in the image map, which was saved and uploaded to the web page. To design this pathway service as flexible as possible, features are provided for the remove, up- and download of relevant pathways (image maps) including the underlying additional information of the elements. However, this service is on a restricted basis to prohibit unauthorized access. Since some pathways tend to become very detailed an option to search for genes or gene accession number, respectively, within the pathway was built in. The pathway editor is executable as a standalone application and is available from http://genome.tugraz.at (4). Currently annotated pathways are the insulin signaling pathway, the IGF-I pathway, and the adipogenesis regulatory network. Other pathways of lipid metabolism will follow in the near future. Available KEGG pathways can also be adapted with the pathway editor based on the provided XML files (5) and uploaded in the same way. Several relevant KEGG pathways for different organisms are already provided.

For each element in the pathway a specific information field exists. The field includes structured information about a gene, protein, 3D-structure, gene regulation, function, literature, and links. The GenBank accession number of the respective gene (typically a RefSeq number) acts as the primary key for the database entries and therefore the declaration of this identity is compulsory. Besides the gene name, symbol name and GenBank accession number for the gene, protein identities for the NCBI, the SWISS-PROT database, and the 3D structures databases can be specified, and the accession numbers displayed and linked to the appropriate databases. The body of the query strings for these links can be changed for all entries of the pathway at once. Since in the case of transcriptional networks, the binding of transcription factors to the DNA is of interest, in the gene regulation field options were implemented to upload and display sequences upstream of the transcription start site (usually the promoter sequence) and transcription factors known to bind to these upstream activator sequences. The description, localization and classification of the factors are entered by the annotator in plain text and are accessed in the same format. The references used to generate the content of the database entries can be appended, including a link to the PubMed entry. There is also the possibility to create a list of all reference entries for the pathway or a list of all upstream sequences in FASTA format, in order to search for transcription factor binding sites. If a clone for a specific gene is available in the clone resources, the clone name will be displayed automatically and a link with optional information about this clone is provided.

**MAPPING OF GENE EXPRESSION DATA TO PATHWAYS**

Through the integration of several types of biological information deeper insights into the molecular mechanisms and biological processes can be gained than just by the analysis of one type of experimental results. In the GOLD.db it is possible to map gene expression data (for instance results of microarray studies) to the corresponding elements of the available pathways similar to previous efforts (6). Either an individual or a provided gene expression data set can be used to visualize the gene expression at different experimental conditions sequentially in the context of the pathways. If an element (gene) of the pathway is included in the data set, the related symbol in the image map is color coded according to the relative gene expression or the log ratio in two color microarray experiments, respectively. As key for the mapped relation the RefSeq number (7) is used. Hence, only those elements in the data set file are mapped, where the RefSeq number in the data set is specified. For the KEGG pathways each element classified by the enzyme classification number (EC) is virtually subdivided into different corresponding RefSeq entries, since one EC is represented by one or more RefSeq entries.

**GENE EXPRESSION DATA SETS**

Analysis of gene expression patterns in animal models for lipid-associated disorders will help to understand
the fundamental gene relations and regulatory mechanisms responsible for the development of obesity related diseases. The huge amount of data associated with the analysis of large scale gene expression analysis raises the demand of tools for storing, processing and retrieving complex information. Approaches to upload and retrieve gene expression data were pursued within the GOLD.db. Large scale gene expression data sets can be uploaded in form of tab delimited text files (Stanford file format) as used for cluster analysis programs together with additional information about the experimental conditions and the citation for already published data sets. Within those data sets the search for specific genes is possible to provide integrated visualization of gene expression levels in different studies and experimental conditions. Finally, pathways can be selected where the gene expression data can be mapped.

REAGENTS
We have developed a relational database for tracking the repository of the reagents like clone resources which can be used for microarray studies. Information about the vector, the sequence and length of the clone insert, primers for the PCR amplification, tissue, organism, accession number, library, container, storage information, date and person and access to other clone bases (e.g. IMAGE Consortium) can be stored. Users of the GOLD.db can list these clones and get all the information about each available clone. Clone information or clone lists can be uploaded and selection lists can be created and deleted by users with appropriate access. The input mask is designed in such way that the user can choose one of the elements of the created selection lists.

TOOLS
In order to deal with the huge amount of data associated with large scale studies and to perform sequence based analysis, several bioinformatics tools were integrated. Sequence similarity search against databases can be performed with BLAST (Basic Local Alignment Search Tool) (8), FASTA (9) or HMM (Hidden Markov Models) (10) on a 48-CPU PC cluster. The sequence retrieval system SRS (LION Bioscience AG, Heidelberg, Germany) was included to enable rapid, easy and user friendly access to the large volumes of diverse and heterogeneous data (11). The PathwayEditor can also be downloaded from the GOLD.db to create new pathways.

OUTREACH COMPONENTS
To establish an educational and outreach component heterogeneous sources of information have been made accessible through the GOLD.db. Video presentations of leading scientists in genomics and proteomics research can be streamed and experimental protocols can be uploaded in pdf-format. The included references are not intended to report all citations associated with a gene or its protein products. The goal is to provide a set of citations with background information. Either these citations or those included in the links, can then be used to find related publications in the PubMed. Finally, links are included to a bundle of functional genomics and computational biology resources.

IMPLEMENTATION
The GOLD.db was implemented in Java (http://java.sun.com/) technology. Hence, the pathway editor as well as the web application are platform independent. The web application of GOLD.db is build in Java Servlets and JavaServer Pages technology based on the Model-View-Controller Architecture. For the implementation, the struts framework (http://jakarta.apache.org.struts) was used. This code can be easily deployed in any Servlet Container. We used the Servlet Container Tomcat (http://jakarta.apache.org/tomcat/) which is accessible from all web browsers. Oracle 9i was used as database management system. The interface between the Java and the Database management system was established using Java database connectivity (JDBC) 2.0. Therefore, migration to other freely available DBMSs like mySQL can be easily done. For additional storage and communication between the pathway-editor components, the markup language XML containing structured, human readable information, was used.

CITING AND ACCESSING GOLD.db
The GOLD.db database should be cited with the present publication as a reference. Access to GOLD.db is possible through the World Wide Web at http://gold.tugraz.at. The pathway editor and the clone tracker are available free of charge to academic, government, and other nonprofit institutions.

FUTURE DIRECTIONS
The vast quantity of gene expression data generated in genomic studies presents a number of challenges for their effective analysis and interpretation. In order to fully understand the changes in expression that will be observed, we must correlate these data with phenotype, genotype, and other information including the tissue distribution and time course expression data gleaned from previous studies. An important goal of our work is the development of tools that allow researchers to efficiently analyze patterns of gene expression and to display them in a variety of useful and informative ways, allowing outside researchers to perform queries pertaining to gene expression results.

We are currently developing a system for visualization of the results of microarray experiments to display relative gene expression for a given gene under specified experimental condition in combination with
other genes at the same or other experimental conditions. This approach will allow addressing further questions by analyzing of these “virtual chip experiments”. Connection and integrating to a microarray database and several analysis tools like gene clustering applications (12) will raise new opportunities in understanding mechanisms of different applications and lipid-associated disorders in particular.

ACKNOWLEDGEMENTS

This work was supported by the Austrian Science Fund, Project SFB Biomembranes F718, the GEN-AU projects Bioinformatics Integration Network (BIN) and Genomics of Lipid-Associated Disorders (GOLD). Diego Miranda-Saavedra was supported by an EU Marie Curie Training Site program “Genomics of Lipid Metabolism”. Michael Maurer was supported by a grant from the Austrian Academy of Sciences. We would like to thank Alexander Sturn for valuable comments and support for mapping of gene expression data and Dietmar Rieder for help with specifying of enzyme classifications.

REFERENCES

Java editor for biological pathways

Elmar Trost, Hubert Hackl, Michael Maurer and Zlatko Trajanoski*

Institute of Biomedical Engineering and Christian Doppler Laboratory for Genomics and Bioinformatics, Graz University of Technology, Krenngasse 37, 8010 Graz, Austria

Received on August 30, 2002; revised on October 23, 2002; accepted on November 13, 2002

ABSTRACT
Summary: A visual Java-based tool for drawing and annotating biological pathways was developed. This tool integrates the possibilities of charting elements with different attributes (size, color, labels), drawing connections between elements in distinct characteristics (color, structure, width, arrows), as well as adding links to molecular biology databases, promoter sequences, information on the function of the genes or gene products, and references. It is easy to use and system independent. The result of the editing process is a PNG (portable network graphics) file for the images and XML (extended markup language) file for the appropriate links.

Availability: http://genome.tugraz.at
Contact: zlatko.trajanoski@tugraz.at

INTRODUCTION
The knowledge about biological pathways, their components, and the interaction between the components is crucial for understanding the function of the cell. With the advance of both, molecular biology technology and information technology, the information about molecular interactions is steadily increasing. Consequently, modeling, editing and annotating biological pathways is becoming an important issue for the organization of knowledge as well as for pathways analysis and computation. The importance of tools for editing pathways including metabolic pathways, signal transduction pathways, or gene regulatory networks was recognized earlier and a set of programs was developed for this purpose. Basically, there are three types of pathway drawing approaches: auto-layout, manual (interactive) drawing, or a hybrid of these two approaches (Kanehisa et al., 2002; Koike and Rzhetsky, 2000; Karp, 2001; Karp et al., 2002; Becker and Rojas, 2001). Of these, interactive drawing tools are useful for the construction of pathway diagrams in a visual way based on available knowledge, and the annotation of the components and interactions between them. However, to the best of our knowledge, there is currently no easy to use and platform independent interactive drawing tool available. Therefore, we have initiated the development of a Java tool to facilitate the representation, visualization and analysis of biological pathways.

PROGRAM OVERVIEW
The pathway editor we have designed represents a novel drawing tool which integrates the possibilities of: (a) charting elements with different attributes (size, colour, labels); (b) drawing connections between elements in distinct characteristics (colour, structure, width, arrows); (c) adding text; and (d) creating a legend and adding literature (Figure 1). The form of each element—typically representing a gene as a part of a pathway—can be edited independently in the drawing plane. The great benefit of this tool is that additional information can be appended to each element via an input mask.

For each element in the pathway a specific information field exists. The field includes structured information about a gene, protein, 3D-structure, gene regulation, function, literature, and links. The GenBank (Benson et al., 2002) accession number of the respective gene (typically an entry of the mRNA, including the feature CDS for the complete coding sequence) acts as the primary key for the database entries and therefore the declaration of this identity is compulsory. Besides the gene name, symbol name and GenBank accession number for the gene, protein identities for the NCBI, the SWISS-PROT (Wu et al., 2002) database, and the 3D structures databases can be specified, and the accession numbers displayed and linked to the appropriate databases. The body of the query strings for these links can be changed for all entries of the pathway at once. Since in the case of transcriptional networks, the binding of transcription factors to the DNA is of interest, in the gene regulation field options were implemented to upload and display sequences upstream of the transcription start site (usually the promoter sequence) and transcription factors known to bind to these upstream activator sequences. The description, localization and

*To whom correspondence should be addressed.

Bioinformatics 19(0) © Oxford University Press 2003; all rights reserved.
E.Trost et al.

Fig. 1. An example of the use of the Pathway Editor for the construction of the insulin signaling pathway. The information that can be entered for a certain element (p110α/β, represented as a rectangle) is shown in the details-setting window and includes name, location, description and references.

classification of the factors are entered by the annotator in plain text and are accessed in the same format. The result of the editing process is a PNG (portable network graphics) file for the images and XML (extended markup language) file for the appropriate links and annotated information. Image maps can be easily created in a web page by parsing the XML files. An example of an image map constructed using this tool is the annotated pathway for insulin signaling (http://gold.tugraz.at).

The pathway editor was implemented in Java and is freely available.

ACKNOWLEDGEMENTS

This work was supported by the Austrian Science Fund, Project SFB Biomembranes F718. Michael Maurer was supported by a PhD fellowship grant from the Austrian Academy of Sciences.

REFERENCES


Mini-Review

New trends in bioinformatics: from genome sequence to personalized medicine

Robert Molidor, Alexander Sturn, Michael Maurer, Zlatko Trajanoski *

Institute of Biomedical Engineering and Christian Doppler Laboratory for Genomics and Bioinformatics, Graz University of Technology, Krenngasse 37, Graz 8010, Austria

Received 21 May 2003; received in revised form 26 June 2003; accepted 30 June 2003

Abstract

Molecular medicine requires the integration and analysis of genomic, molecular, cellular, as well as clinical data and it thus offers a remarkable set of challenges to bioinformatics. Bioinformatics nowadays has an essential role both, in deciphering genomic, transcriptomic, and proteomic data generated by high-throughput experimental technologies, and in organizing information gathered from traditional biology and medicine. The evolution of bioinformatics, which started with sequence analysis and has led to high-throughput whole genome or transcriptome annotation today, is now going to be directed towards recently emerging areas of integrative and translational genomics, and ultimately personalized medicine.

Therefore considerable efforts are required to provide the necessary infrastructure for high-performance computing, sophisticated algorithms, advanced data management capabilities, and-most importantly-well trained and educated personnel to design, maintain and use these environments.

This review outlines the most promising trends in bioinformatics, which may play a major role in the pursuit of future biological discoveries and medical applications.

© 2003 Elsevier Inc. All rights reserved.

Keywords: Bioinformatics; Genomics; Personalized medicine

1. Introduction

In the past decade bioinformatics or computational biology has become an integral part of research and development in biomedical sciences. In contemplating a vision for the future of this new branch of life sciences, it is appropriate to consider the remarkable path that has led to today’s status of the field. When in the early 1980s methods for DNA sequencing became widely available, molecular sequence data expeditiously started to grow exponentially. After the sequencing of the first microbial genome in 1995, the genomes of more than 100 organisms have been sequenced and large-scale genome sequencing projects have evolved to routine, though still non-trivial, procedures (Janssen et al., 2003; Kanehisa and Bork, 2003). The imperative of efficient and powerful tools and databases became obvious during the realization of the human genome project, whose completion has been established several years ahead of schedule. The accumulated data was stored in the first genomic databases such as GenBank, European Molecular Biology Laboratory Nucleotide Sequence Database (EMBL), and DNA Data Bank of Japan (DDBJ) and novel computational methods had to be developed for further analysis of the collected data (e.g. sequence similarity searches, functional and structural predictions).

One of the first breakthroughs in the area of bioinformatics was the introduction of the rapid sequence database search tool BLAST (Altschul et al., 1990), which nowadays has become a valuable and indispensable tool in the everyday life of biomedical research.

Automatic sequencing was the first forerunner and had a major impact on high throughput generation of various kinds of biological data such as single-nucleotide polymorphisms (SNPs) and expressed sequence tags (ESTs). Subsequently, other novel high-throughput methods such as serial analysis of gene expression (SAGE) (Velculescu et al., 1995) and DNA microarrays (Shalon et al., 1996) have been
developed to analyze the transcriptional program of a cell, tissue or organism at a genomic scale.

All this novel experimental procedures are associated with information technology in a symbiotic relationship. It is encouraging that the use of high throughput experimental procedures in combination with computational analysis so far has revealed a wealth of information about important biological mechanisms. This review will deliver insight to the current trends in bioinformatics that may help to bridge the considerable gap between technical data production and its use by both, scientists for biological discovery, and physicians for their daily routine.

2. From sequence to expression

The lifeblood of bioinformatics has been the handling and presentation of nucleotide and protein sequences and their annotation. With the advent of novel experimental techniques for large-scale, genome-wide transcriptional profiling via microarrays or gene chips, a new field of gene expression data analysis emerged (Slonim, 2002). This new momentum to the bioinformatics community has fueled the hope of getting more insight into the processes conducted in a cell, tissue or organism.

As more and more researchers adopted the microarray technology it soon became increasingly clear that simple data generation is not satisfactory and the challenges lie in storage, normalization, analysis, visualization of results, and most importantly in extracting biological meaningful information about the investigated cellular processes. Therefore, considerable progress has been made in the last couple of years to handle and analyze the millions of data points accumulated by state of the art microarray studies with tens of thousands of sequences per slide and maybe hundreds of slides (Brazma et al., 2003).

Several topics of the analytical pipeline, namely image analysis, normalization, and gene expression data clustering and classification have been addressed in numerous publications (Baxevanis, 2003; Brazma et al., 2003; Ermolaeva et al., 1998). Data interpretation, however, proliferated just recently and leaves still a lot of room for new tools to extract knowledge from the increasing amount of microarray data. A key challenge of bioinformatics in the future will be to bridge this considerable gap between data generation and its usability by scientists for incisive biological discovery.

The evolution of microarray data production to ever-larger and more complex data sets will enable bioinformati-cians to use this huge amount of information for developing innovative approaches to reverse engineer biological networks of molecular interactions, which may unravel the contribution of specific genes and proteins in the cellular context (D’haeseeleer et al., 2000). These new approaches of gene expression pattern analysis try to uncover the properties of the transcriptional program by analyzing relationships between individual genes. This will be the beginning of an exciting journey towards the ‘holy grail’ of computational biology: to generate knowledge and principles from large-scale data and to predict computationally systems of higher complexity such as the interaction networks in cellular processes and in the end to present an accurate and complete representation of a cell or an organism in silico.

The comparison of DNA sequences of entire genomes already gives insights into evolutionary, biochemical, and genetic pathways. Additionally, enabled by the increasing amount of public available microarray studies, comparative analysis of the transcriptome of different cell types, treatments, tissues or even among two or more model organisms promise to significantly enhance the fundamental understanding of the universality as well as the specialization of molecular biological mechanisms. The objective is to develop mathematical tools that are able to distinguish the similar from the dissimilar among two or more large-scale data sets.

Although new innovative procedures to analyze genomic data are still desirable, one problem during the analysis of gene expression data is not the lack of algorithms and tools, but the multiplicity of practices available to choose from. Moreover, these methods are difficult to compare and each method has its own implementation and frequently a different data format and representation. This diversity of methods makes it difficult and time consuming to compare results from different analyses. Therefore standardized data exchange and calculation platforms, which allow the straightforward and efficient application of different algorithms to the data one is interested in, are and will be highly welcomed by the research community (Box 1).

3. Integrative genomics

Genes and gene products do not function independently. They contribute to complex and interconnected pathways, networks and molecular systems. The understanding of these systems, their interactions, and their properties will require information from several fields, like genomics, proteomics, metabolomics or systematic phenotype profiles at the cell and organism level (Collins et al., 2003).

Database technologies and computational methods have to be improved to facilitate the integration and visualization of these different data types, ranging from genomic data to biological pathways (Diehn et al., 2003). The integration of pathway information with gene expression studies for instance has the potential to reveal differentially regulated genes under certain physiological conditions in a specific cellular component (Forster et al., 2002). Furthermore, connecting protein specific databases to genomic databases will be crucial to answer upcoming proteomic questions (Boguski and McIntosh, 2003).
Sophisticated computational technologies have to be developed to enable life scientists to establish relationships between genotype and the corresponding biological functions, which may yield to new insights about physiological processes in normal and disease states.

4. Translational genomics

Genomic research is now entering an era where emerging data of sequencing projects and integrative genomics will help investigators to ultimately unravel the genetic components of common and complex diseases. The much anticipated complete sequence of the human genome, coupled with the emergence of the sequences of other animal, plant, and microbial genomes, now provides us with an incomparable source of information to address biological and medical questions. However, this advance in our knowledge accompanies the recognition that further progress in technology, information based systems for integrating genetic studies, large population based research, increased public awareness of ethical and legal issues, and education are mandatory (Collins et al., 2003).

A relatively new field employing innovative advances such as genome-wide array technology and the burgeoning field of computational biology is aptly entitled 'translational research'. The objective is to provide the data and tools necessary to identify genes that play a role in hereditary susceptibility to disease and additionally to discover genetic changes contributing to disease progression and resistance to therapy (McCabe, 2002; Rosell et al., 2002). Therefore it is crucial to integrate patient related data such as CT- and MRI scans, mammography, ultrasound, and the corresponding knowledge of their diagnostic parameters.

Achievements of this mission will be accelerated and empowered through the refinements and breakthroughs in research techniques that span biomedical and genomic methodologies, as well as computational biology. This will help to make a smooth translation of information from bench to bed and to better focus on the ongoing process of disease in the body.

5. Personalized medicine

The 20th century has brought us a broad arsenal of therapies against all major diseases. However, therapy often fails to be curative and additionally may cause substantial side effects. Moreover these drugs have, due to their widespread use, revealed substantial inter-individual differences in therapeutic response. Evidence has emerged that a substantial portion of the variability in drug response is genetically determined and also age, sex, nutrition, and environmental exposure are playing important contributory roles. Thus there is a need to focus on effective therapies of smaller patient subpopulations that demonstrate the same disease phenotype, but are characterized by distinct genetic profiles. Whether and to what extent this individual, genetics-based approach to medicine results in improved, economically feasible therapy remain to be seen. However, the realization of this will require new methods in biology, informatics and analytical systems that provide an order-of-magnitude increase in throughput, along with corresponding decreases in operating costs, enhanced accuracy and reduced complexity (Mancinelli et al., 2000; Collins et al., 2003).

6. Challenges

The challenges are to capitalize on the immense potential of bioinformatics to improve human health and well-being. Although genome-based analysis methods are rapidly permeating biomedical research, the challenges of establishing robust paths from genomic information to improved human health remain immense.

6.1. Data integration

The rapid expansion of biomedical knowledge, reduction in computing costs, spread of internet access, and the recent emergence of high throughput structural and functional genomic technologies has led to a rapid growth of electronically available data. Today, databases all around the world contain biomedical data, ranging from clinical data records for individual patients stored in clinical information systems to the genetic structure of various species stored in molecular biology databases (http://nar.oupjournals.org/cgi/content/full/31/1/1/DC1). The volume and availability of this kind of data has grown through a largely decentralized process, which has allowed organizations to meet specific or local needs without requiring them to coordinate and standardize their database implementations. This process has resulted in diverse and heterogeneous database implementations, making access and aggregation very difficult (Sujansky, 2001; Stein, 2003).

In molecular biology the data, which has to be managed, covers a wide range of biological information. The core data are collections of nucleic and amino acid sequences and protein structures. There are also many specialized databases covering topics like Comparative Genomics, Gene Expression, Genetic and Physical Maps, Metabolic Pathways and Cellular Regulation (Baxevanis, 2003). Although all of these resources are highly informative individually, the collection of available content would have more efficacies if provided in a unified and centralized context. The management and integration of these heterogeneous data sources with widely varying formats and different object semantics is a difficult task. This issue can be handled only by increasingly sophisticated electronic mechanisms to
store, manipulate, and communicate information. One possibility to facilitate the cross-referencing of disparate data sources is to introduce standardization of terms and data formats. For this reason, several efforts are underway to standardize relational data models and/or object semantics (Stein, 2002) (Box 1).

### 6.2. High-performance computing

With the introduction of high throughput technologies such as sequencing and microarrays the amount of data that has to be managed, compared and analyzed increased dramatically. Therefore, the analysis of large-scale genomic and proteomic data in reasonable time requires high-performance computing systems. The impressive and steady improvements of computational power contributed to the success of high throughput biological technologies and its research. This is depicted by the correlation of the exponential increase of GenBank entries and the number of transistors integrated on a single chip (Fig. 1). To ensure the steady progress of bioinformatics and its advantages even more powerful systems are required to be designed and implemented (Thallinger et al., 2002).

### 6.3. Ethical, legal, and social implications (ELSI)

The study of pointed questions of life-science and the desire to collect and disseminate data pertaining to biomedical research raise a number of important and non-trivial issues in ethics and patient confidentiality. The need to integrate information from various sources, such as hospital discharge records and clinical questionnaires, strengthens the problems related to this topic. Even if anonymization is enforced, a specific person could be traced back either exactly or probabilistically, due to the amount of remaining information available (Altman and Klein, 2002). Although the integration of additional clinical information would have the potential to dramatically improve human health, nonetheless, it is crucial to ensure that the availability of clinical phenotypic data or the like does under no circumstances lead to the loss of study-subject confidentiality or privacy. Researchers have to pay attention to these ELSI issues and should not view them as impediments (Collins et al., 2003; Oosterhuis et al., 2003).

### 6.4. Training and education

To be able to accomplish the diverse interdisciplinary challenges, which genomics and bioinformatics are facing nowadays and in the future, researchers with the expertise to understand the biological systems and to use the information efficiently are required. To widen the success of bioinformatics not only bioinformaticians themselves but also biologists and physicians using the computational tools need profound skills in bio- and computer sciences. To create and interpret results from bioinformatic approaches in a meaningful and responsible way, at least a fundamental understanding is required. The BioPathways Consortium is elaborating a standard data exchange format to enable sharing of pathway information, such as signal transduction, metabolic and gene regulatory pathways (http://www.biopathways.org).

In clinical settings SNOMED [http://www.snomed.org] or ICD [http://www.icd.org] have been established for a standardized classification of disease and health related problems (Liebman, 2002).

---

**Box 1. Standardization**

Given the increasing availability of biomedical information located at different sites and accessible mostly over the internet, researchers require new methods to integrate and exchange data. During the last years extensible markup language (XML) (http://www.w3.org/XML/) has emerged as a common standard for the exchange of data. XML consists of a set of rules whereby new vocabularies (tags) may be defined. These tags do not indicate how a document is formatted, but instead provide semantic context to the content of the document, as semantics require more constraints on the logical relationships between data items. e.g.: a tag for a SNP can only be located between the start- and end-tag of a coding region.

In the area of microarray databases for instance (Gardiner-Garden and Littlejohn, 2001; Anderle et al., 2003), the microarray gene expression data (MGED) society (http://www.mged.org) proposes with MAGE an object model and with minimum information about a microarray experiment (MIAME) (Brazma et al., 2001) a standard to describe the minimum information required to unambiguously interpret and verify microarray experiments. In adherence to MIAME, which is required by several journals for manuscript submission, the microarray gene expression-markup language (MAGE-ML) was designed based on XML (Spellman et al., 2002).

The human proteome organization (HUPO) is currently engaged to define community standards for data representation in proteomics to facilitate data comparison, exchange and verification. This organization is working on standards for mass-spectrometry, protein–protein interaction and on a general proteomics format (http://psidev.sourceforge.net).

The BioPathways Consortium is elaborating a standard data exchange format to enable sharing of pathway information, such as signal transduction, metabolic and gene regulatory pathways (http://www.biopathways.org).

In addition, the Gene Ontology Consortium (http://www.geneontology.org) provides a structured and standardized vocabulary to describe gene products in any organism (Gene Ontology Consortium, 2001).

In clinical settings SNOMED [http://www.snomed.org] or ICD [http://www.icd.org] have been established for a standardized classification of disease and health related problems (Liebman, 2002).
of the used technologies, algorithms, and methods is indispensable. Moreover, the interdisciplinary character of this field needs to be enforced by the incorporation of mathematics and theoretical foundations of physics and chemistry to detect basic architectures of complex biological systems. Therefore, adequate training and education has to be provided for bioinformatics specialists in such diverse and interdisciplinary fields as computer sciences, biology, mathematics, chemistry and physics (Collins et al., 2003).

Fig. 1. Base pairs (○) to transistors comparison (●): The number of base pairs in GenBank doubles every year (http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html), which correlates with the increasing packing density of transistors on a single chip (http://www.intel.com/pressroom/kits/quickreffam.htm). This emphasizes that the exponential growth of transistor integration on a chip and consequently the rapid development of information processing technologies have contributed to a great extent to the rapid growth of genomic data.

Fig. 2. Components of integrative and translational genomics, which are the building blocks of present and future bioinformatics applications. The heterogeneous character of bioinformatics is represented by diverse topics ranging form Genomics to Training and from High-Performance Computing to ethical, legal, and social implications (ELSI).
7. Conclusion

It is widely accepted that bioinformatics has led the way to the post-genomic era and will become an essential part in future molecular life-sciences. Nowadays, bioinformatics is facing new challenges of integrative and translational genomics, which will ultimately lead to personalized medicine. The ongoing investigations in these areas attempt to provide researchers with a markedly improved repertoire of computational tools that facilitate the translation of the accumulated information into biological meaningful knowledge. This virtual workbench will allow the functioning of organisms in health and disease to be analyzed and comprehended at an unprecedented level of molecular detail. To accomplish this, considerable endeavors have to be undertaken to provide the necessary powerful infrastructure for high-performance computing, sophisticated algorithms, advanced data management capabilities, and-most importantly well trained personnel to design, maintain, and use these environments (Fig. 2). The ultimate goal of this new field should be to evolve biology from a qualitative into a quantitative science such as mathematics and physics. Although there are still significant challenges, bioinformatics along with biological advances are expected to have an increasing impact on various aspects of human health.

Acknowledgements

This work was supported by the Austrian Science Fund (Grant SFB Biomembranes F718) and the bmbfwk, GENAU:BIN, Bioinformatics Integration Network. Michael Maurer and Robert Molidor were supported by a grant from the Austrian Academy of Sciences.

References

Differential Gene Expression Profile of Glucocorticoids, Testosterone, and Dehydroepiandrosterone in Human Cells

M. Maurer1,2, Z. Trajanoski1, G. Frey2, N. Hiroi2,3, J. Galon4, H. S. Willenberg3, P. W. Gold5, G. P. Chrousos2, W. A. Scherbaum3, S. R. Bornstein2,3

1 Institute of Biomedical Engineering, Graz University of Technology, Graz, Austria
2 NICHD, Bethesda, USA
3 Department of Endocrinology, University of Düsseldorf, Düsseldorf, Germany
4 Institut Curie Laboratoire d'Immunologie Cellulaire et Clinique Paris, France
5 NIMH, Bethesda, USA

Glucocorticoids are the major immunomodulating hormones in the human body. Recently, increasing interest in androgens as immunomodulators has emerged. In particular, Dehydroepiandrosterone (DHEA) has been suggested as beneficial in the treatment of some autoimmune disorders. However, the action and role of testicular and adrenal androgens on human immune cells remains unclear. This is the first study to provide large-scale gene expression data on the action of different steroids (DHEA, glucocorticoids, and testosterone) on human peripheral blood mononuclear cells using the recently developed genomic-scale technology of microarrays. Novel computational tools and techniques such as Principal Component Analysis (PCA) were used for analysis, clustering and visualization. We have demonstrated that each steroid has its distinct gene expression profile, although DHEA and testosterone co-regulated most genes in a similar direction while glucocorticoids frequently regulated the same genes in an opposite direction. Our data suggest an important and a complex regulatory role for androgens on human immune cells that should be considered in androgen replacement or treatment strategies.

Key words: Microarray – Gene Expression Profile – Dehydroepiandrosterone (DHEA) – Glucocorticoids – Testosterone – Principal Component Analysis (PCA) – Online PCR

Introduction

Androgens are the most abundant hormones in the human body. In addition to their critical role in sexual development, both testicular and adrenal androgens have been implicated in fat depot distribution, neurogenesis, age-related processes, and immune function [1,2].

Glucocorticoids are the major stress hormones in the human body, and are indispensable for survival. They have complex actions on metabolism, electrolyte homeostasis, bone, the cardiovascular and nervous system, and the immune system [3].

Their capacity to assist the innate and adaptive immunity in eliminating toxins, tumor cells, and tissue damage has led to the widespread use of synthetic corticosteroids in all fields of clinical medicine [4].

The ability of glucocorticoids to modulate immune function is one of the most striking and widely analyzed effects of these hormones. Glucocorticoids act on all nucleated cells of the immune system by binding to glucocorticoid or mineralocorticoid receptors that regulate transcription of genes in a cell-specific manner. However, the action and role of androgens on human immune cells remains unclear. Males have a well-established lower incidence of autoimmune diseases, and increasing interest is emerging in androgens as immunomodulating agents. The lack of testosterone in patients with androgen deficiency, such as in Klinefelter’s syndrome, enhances cellular and humoral immunity, while androgen replacement therapy suppresses it [5]. Males are known to be at increased risk of septic complications after traumatic injury, and recent data have suggested that this phenomenon is caused by the inhibitory effects of testosterone on immune function [6]. On the other hand, adrenal androgens, particularly dehydroepiandrosterone (DHEA), have recently been suggested to be beneficial in the treatment of some autoimmune disorders and may restore immune function following trauma-hemorrhage by a direct effect on T lymphocytes [7].

Opposing effects of DHEA and glucocorticoids on immune balance have been reported and refuted. It is still a matter of debate whether immune cells or any other cells possess specific receptors for DHEA. Recently, functional testosterone receptors have been described in the plasma membrane of T cells. But they may not be active in the genomic pathway [8]. So far, no comprehensive analysis of gene regulation on human immune cells by androgens has been performed. Recently developed high-throughput genomic techniques addressing these pharmacological questions should allow a better understanding of the molecular mechanisms of steroid action.
Therefore, we analyzed the effect of testosterone (TEST), DHEA, and dexamethasone (DEX) on the gene expression profile of human peripheral blood mononuclear cells (PBMC) from healthy donors using DNA microarrays. Single genes from different functional groups were confirmed by online TaqMan polymerase chain reaction (PCR).

Materials and Methods

Peripheral blood mononuclear cells were prepared from 6 healthy donors and stimulated with dexamethasone (DEX), dehydroepiandrosterone (DHEA) and testosterone (TEST) (10⁻⁷ M for 18 h). RNA extraction was performed (RNAGents, Promega, Madison, WI), and mRNA was purified with oligoTex mRNA isolation columns (Qiagen, Valencia, CA).

Microarray preparation

Sequences used for microarray fabrication were generated by PCR. PCR products were purified by gel filtration with Sephacryl-400 (Amersham Pharmacia Biotech, Inc., Piscataway, NJ) equilibrated in 0.2× SSC. The filtrate was dried down and rehydrated in one-tenth-volume dH₂O for arraying. The DNA solutions were arrayed by robotics on modified glass slides. After arraying, slides were processed to fix the DNA to the prepared glass surface and washed three times in dH₂O at room temperature. Slides were then treated with 0.2% I-Block (Tropix, Bedford, MA) dissolved in 1 x Dulbecco’s phosphate buffered saline (Life Technologies, Gaithersburg, MD) at 60 °C for 30 minutes. GEM microarrays were then rinsed in 0.2% SDS for two minutes followed by three one-minute washes in dH₂O.

Fluorescent labeling of probe

Isolated mRNA was reverse-transcribed with 5’ Cy3- or Cy5-labeled random 9-mers (Operon Technologies, Inc., Alameda, CA). Reactions were incubated for 2 h at 37 °C with 200 ng poly A RNA, 200 units M-MLV reverse transcriptase (Life Technologies, Gaithersburg, MD), 4 mM DTT, 1 unit RNase Inhibitor (Ambion, Austin, TX), 0.5 mM dNTPs, and 2 mg labeled 9-mers in 25 ml volume with enzyme buffer supplied by the manufacturer. The reactions were terminated by incubation at 85 °C for 5 min. The paired reactions were combined and purified with a TE-30 column (Clontech, Palo Alto, CA), brought to 90 ml with dH₂O, and precipitated with 2 ml 1 mg/ml glycogen, 60 ml 5 M NH₄OAc, and 300 ml EtOH. After centrifugation, the supernatant was decanted and the pellet resuspended in 24 ml of hybridization buffer: 5× SSC, 0.2% SDS, 1 mM DTT.

Hybridization

Probe solutions were thoroughly resuspended by incubating at 65 °C for 5 min with mixing. The probe was applied to the array and covered with a 22 mm glass cover slip, and placed in a sealed chamber to prevent evaporation. After hybridization at 62 °C for 6.5 hours, slides were washed in three consecutive washes of decreasing ionic strength.

Scanning

Microarrays were scanned in both Cy3 and Cy5 channels with Axon GenePix scanners (Foster City, CA) with a 10 mm resolution. The signal was converted to 16-bits-per-pixel resolution, yielding a 65536 count dynamic range.

Normalization and ratio determination

Incyte GEMtools software (Incyte Pharmaceuticals, Inc., Palo Alto, CA) was used for image analysis. The elements were determined by a gridding and region detection algorithm. The area surrounding each element image was used to calculate a local background and was subtracted from the total element signal. Background subtracted element signals were used to calculate Cy3:Cy5 ratios. The average of the resulting total Cy3 and Cy5 signal gave the ratio used to balance or normalize the signals. An absolute balanced differential expression greater than 1.4 was considered significant.

Principal component analysis

The microarray dataset was analyzed using the Principal Component Analysis (PCA) of a cluster-analysis program developed in our group (http://www.genome.tugraz.at) [15]. Principal component analysis (PCA), which is also known as singular value decomposition, is an exploratory multivariate statistical technique that identifies key variables (or a combination of variables) in a multidimensional dataset. Thus, the analysis of single genes juxtaposed to calculated patterns can be analyzed and transferred to a 3-dimensional space for improved visualization.

TaqMan PCR

To quantify mRNA expression, we applied the technique of real-time quantitative PCR (TaqMan PCR) using the 7700 Sequence Detector (Perkin Elmer Applied Biosystems, Foster City, CA) as described previously [16].

Results

Principal component analysis

The Principal Components (PC) of the microarray experiments showed that each component emphasizes a different experiment. PC1 shows that DEX-treated genes have a very strong influence, but DHEA and TEST treated genes had nearly no influence on this component. In PC2, TEST had the highest impact on this component and PC3 was dominated by DHEA.

Having calculated these 3 Components, all genes can be transformed into the 3-dimensional space (Fig. 1). PC1 is represented by the X-axis and corresponds to an expression pattern of about 73.6% of all the genes; PC2 is mapped onto the Y-axis and used 19.9%; PC3 utilized 6.5% of all available genes and was transferred to the Z-axis. At first glance, one can see that these figures consist mainly of straight lines and planes. Lines in PCA represent genes that were regulated by only one component (experiment), while planes are indicative of genes with two degrees of freedom (regulated by two components). For better visualization, we then color-coded all genes according to the scheme shown in Fig. 2. The large group of DEX regulat-
ed genes (dark yellow spots) is represented by a relatively short line, which can be explained by the fact that most of the genes had similar expression levels.

Based on the results and patterns from the PCA, we used the experiment classification scheme (Fig. 2) for further expression and functional analysis in conjunction with Access and Excel (Microsoft Corporation, Redmond, WA, USA) and the functional correlation gene dataset from GEMTools (Incyte Genomics Inc., St. Louis, MO, USA).

**Regulation of experimental and functional groups**

The signals of 4943 genes were detectable in all three experiments. Genes with regulation at less than 40% were considered unregulated. Thus, the total number of genes that were not regulated at all was 1945. The remaining 2998 genes can be described as follows: 1982 (66.1%) were regulated exclusively by dexamethasone (DEX), 261 (8.7%) exclusively by testosterone (TEST) and 119 (4.0%) exclusively by DHEA. Also, there were 237 (7.9%) genes that were regulated by DEX & DHEA, 242 (8.1%) genes regulated by DEX & TEST and just 77 (2.5%) were regulated by DHEA & TEST. A group of 80 (2.7%) genes was regulated by all three hormones. Looking more into the detail of co-regulated groups of genes, we could see that in the group of DEX & DHEA 97.0% were regulated in opposing directions, 50.4% of the genes in the DEX- & TEST-regulated group and 78.0% of the DHEA- & TEST-regulated group were regulated in opposing directions. The group representing gene expression in all three hormones showed directional homogeneity in 22.5% of the genes.

Considering all DEX regulated genes, we found that 56.3% were upregulated. This number changes to 40.4% in TEST and 7.4% in all DHEA-regulated genes. To get a better impression of this regulation, we divided the regulated genes into functional groups (Table 1). Hormone-specific regulation has been categorized by the following functional groups: enzymes, signal transduction and regulation, membrane transport, protein modification and maintenance, nucleic acid synthesis and modification, adhesion and molecular recognition, electron transfer, localized and structural proteins, metabolism, growth and development, kinesis, environmental responses and ecological interactions. Genes were classified into zero, one or more functional groups. Genes corresponding to no functional group were accumulated in the “unknown function” group.

In the DEX case, upregulation was more common in all of the functional groups, whereas downregulation was prevalent in functional groups of TEST- and DHEA-regulated genes. As mentioned before, there were many opposingly regulated genes between DEX & DHEA (230 out of 237) and DEX & TEST (122 out of 242). To complete the picture of functional regulation and co-regulation, we now analyzed the opposingly regulated genes on their functional behavior. The largest functional groups of opposingly regulated genes were the localized and structural proteins (17.6%), metabolism (16.2%) and signal
transduction and regulation (10.8%). Concerning DEX & DHEA and TEST & DHEA, the largest functional group was metabolism (16.6%, 25.0%). The group of DEX & TEST regulated genes had localized and structural proteins as the largest functional group for genes regulated in opposing directions (20.0%).

Single genes from these major groups were confirmed by quantitative PCR.

Discussion

DNA microarray analysis of gene expression is an emerging technology allowing a new, integrated view on physiological and pharmacological gene regulation [9–11]. Our study is the first to provide large-scale gene expression data on the action of different steroids (DHEA, dexamethasone, and testosterone) on normal human immune cells. Our data demonstrate that numerous genes are regulated by these steroids, and that each steroid has a distinct gene expression profile. Whereas the adrenal androgen, DHEA, and the testicular androgen, testosterone, co-regulate most genes in a similar direction, glucocorticoids frequently regulate the same genes in an opposite direction. The fact that each steroid elicits a distinct pattern of gene expression regulating a distinct subset of genes corroborates the notion that human immune cells have distinct receptors not only for glucocorticoids but also for testosterone and DHEA.

It is still a matter of debate whether immune cells possess specific receptors for DHEA or testosterone. It has been suggested that the effects of DHEA may not be exerted by DHEA-specific receptors, but rather via receptors for other active androgenic metabolites generated by immune cells. However, the capacity of DHEA and testosterone to induce a significantly different pattern of gene expression strongly suggests distinct receptor and signaling systems for each steroid. The idea that DHEA addresses specific receptors is supported by other reports suggesting the involvement of N-methylaspartate-excitatory amino acid receptors and the nuclear hormone-type receptor called CAR-β in mediating DHEA effects [11,12].

In our microarray analysis, all three steroids regulated genes in four primary groups, including enzymes, signal transduction, localized and structural proteins, and metabolism. Whereas androgens induced a downregulation in the majority of genes, it was unexpected that glucocorticoids that have well-established immunosuppressive actions upregulated more than 50% of all genes in these human immune cells.

The analysis of gene expression variation has recently been a new cornerstone in the analysis of human cells, particularly in the cells of the immune system. The peripheral blood monocytes are easily accessible, and recent studies using cDNA microarrays have allowed a genomic-scale view of gene expression in immune cells at many stages of differentiation, activation, and malignant transformation. Furthermore, genomic-scale gene expression studies have revealed a reliable relationship between the expression pattern of genes and function in many instances. Because of the intimate relationship between gene expression and gene function the information gained from these studies should help to define new clinical strategies [9–16].

The most commonly available DNA microarrays produce an ordered array of thousands of immobilized nucleic acids and serve to assay fluorescently labeled cDNA probes prepared from RNA. The fluorescent ratios are robust measurements of relative gene expression, and the method used in our study has previously been shown [13] to agree with high accuracy with more standardized methods of gene expression analysis, such as Northern blot hybridization and quantitative PCR, but also with Western blot analysis for the corresponding proteins.

The present study reveals interesting new aspects in the differential action of androgens on immune cells. It has previously been demonstrated that the lack of testosterone in patients
with hypogonadism enhances cellular and humoral immunity, and that androgen replacement therapy suppresses cytokine and immunoglobulin levels. Likewise, testosterone exerted a suppressive effect on cytokine levels in our study.

We did not see any opposing effects of androgens vs. glucocorticoids on the adaptive immune response, and there was no significant differential regulation of the Th1/Th2 balance. Testosterone and glucocorticoids have clear immunosuppressive actions on humoral and cellular immunity, whereas DHEA has only a mild-to-moderate immunosuppressive effect. In contrast to these findings, there is a significant differential regulation of the innate immune response between androgens and glucocorticoids that has not been described so far. The purpose of this study, however, was to give a first overview on the genomic-scale gene expression pattern of different steroids using a novel computational component analysis approach. Based on this preliminary analysis, further work needs to be carried out on examining the role of androgens on specific immune genes and the mechanism of their action.

In conclusion, this study suggests an important and complex regulatory role for androgens on human immune cells that need to be considered if androgens are to be administered on a broader basis. Furthermore, gene expression profiling will be an important part of pharmacogenomic strategies to develop more specific and efficient drugs [14]. Our study suggests that this approach seems to be particularly promising in developing and exploring the actions of new steroid compounds.

Acknowledgements

The work of M. Maurer was supported by the Austrian Academy of Science. It was also supported by DFG – grant BO 1147 – 7 to SRB. We would like to thank A. Sturn for developing Genesis and K. Bittner for technical assistance.

References


Requests for reprints should be addressed to:

Stefan R. Bornstein, M.D.
Department of Endocrinology
University of Düsseldorf
Moorenstrasse 5
40225 Düsseldorf
Germany

Phone: + 49 (211) 811-7810
Fax: + 49 (211) 811-7860
Systems for Management of Pharmacogenomic Information

Alexander Sturn, Michael Maurer, Robert Molidor, and Zlatko Trajanoski

Institute for Genomics and Bioinformatics and
Christian Doppler Laboratory for Genomics and Bioinformatics,
Graz University of Technology,
Krenngasse 37, 8010 Graz, Austria

To whom correspondence should be addressed:

Zlatko Trajanoski, PhD.
Krenngasse 37
A-8010 Graz
Austria
Phone: +43-316-873-5332
Fax: +43-316-873-5340
Email: zlatko.trajanoski@tugraz.at
Introduction

There is no doubt that the sequencing and initial annotation of the human genome, completed in April 2001, is one of the great scientific advancements in history [Lander 2001, Venter 2001]. This breakthrough in biological research was made possible by advances in high performance computing and the employment of a high sophisticated information technology infrastructure. High-speed computers are necessary to analyze the tens of terabytes of raw sequence data and correctly order the 3.2 billion base pairs of DNA that compose the human genome. The assembly and initial annotation is only the first step on a long road for understanding the human genome. Many companies, research institutes, universities and government laboratories are now rapidly moving on to the next steps: comparative genomics, functional genomics, proteomics, metabolomics, pathways, systems biology and pharmacogenomics [Collins 2003, Forster 2002]. Latter is the study of how an individual's genetic inheritance affects the body's response to drugs. Thus it holds the promise that drugs might one day be tailor-made for individuals and adapted to each person's own genetic makeup. Environment, diet, age, lifestyle, and state of health all can influence a person's response to medicines, but understanding an individual's genetic makeup is thought to be the key to creating personalized drugs with greater efficacy and safety [Mancinelli 2000]. Researchers are beginning the quest to determine exactly how each gene and protein functions and more important how they malfunction to trigger deadly illnesses such as heart disease, cancer, Alzheimer's and Parkinson's diseases.
Important prerequisites for pharmacogenomics or personalized medicine will be achieved by combining a person’s clinical data sets with genome information management systems. However, huge disparate data sources, like public or proprietary molecular biology databases, laboratory management systems, and clinical information management systems pose significant challenges to query and transform these data into valuable knowledge [Boguski 2003]. The core data are collections of nucleic and amino acid sequences stored in GenBank [Benson 1997] and protein structures in the Protein Data Bank (PDB) [Kanehisa 2003]. Additionally this core data is used to create secondary and integrated databases such as PROSITE [Falquet 2002] and InterPro [Mulder 2003]. Furthermore, integrating data collected from high throughput genomic technologies like sequencing, microarrays, SNP detection, and proteomics require the nontrivial development of information management systems [Stein 2002]. For their establishment, increasingly powerful computers and capacious data storage systems are mandatory. In the next paragraphs we will give an overview of the main and most important technologies needed for the management of pharmacogenomic information, namely database management systems, software, and hardware architectures.
Databases and Database Management Systems

Since pharmacogenomics deals with a great many of public and/or proprietary data there is a need to easily store, retrieve, and exchange it. The major problem is the integration of the steadily increasing heterogeneous data sources.

The most prominent ways to manage and exchange bioinformatics data are:

- Field/value based flat files
- ASN.1 (Abstract Syntax Notation One) files
- XML files
- relational databases

Field/value based flat files have been very commonly used in bioinformatics. Examples are the flat file libraries from GenBank, European Molecular Biology Laboratory Nucleotide Sequence Database (EMBL), DNA Data Bank of Japan (DDBJ), or Universal Protein Resource (UniProt). These file types are a very limited solution, because they lack referencing, vocabulary control, and constraints. Besides on the file level, there is no inherent locking mechanism that detects when a file is being used or modified. However these file types are primarily used for reading purposes.

ASN.1 is heavily used at the National Center for Biological Information (NCBI) as a format for exporting GenBank data and can be seen as a means for exchanging binary data with a description of its structure. The access concurrency is like flat files just manageable at file level, there is no support for queries, and it lacks on scalability. But since ASN.1 files convey the description of its structure, it thus provides the flexibility
that the client side does not necessarily need to know the structure of the data in advance [Steedman 1993].

XML (eXtensible Markup Language) documents are an emerging way to interchange data and consist of elements that are textual data structured by tags. Additionally XML documents may include a Document Type Definition (DTD) that describes the structure of the elements of an XML document. XML files are hence very flexible, human readable, and provide an open framework for defining standard specifications. For example the MGED (www.mged.org) and Gene Ontology Consortium (www.geneontology.org) have adopted XML to provide and exchange data. The weaknesses of XML are the file based locking mechanism and the large overhead of a text based format caused by the recurrent content describing tags. Although XML provides query mechanisms, it lacks scalability because it does not provide scalable facilities such as indexing [Achard 2001].

A relational database management system (DBMS) is a collection of programs that enables to store, modify, and extract information from a relational database. Such a relational database has a much more logical structure in the way data is stored. Tables are used to represent real world objects; with each field acting like an attribute. The set of rules for constructing queries is known as a query language. Different DBMSs support different query languages, although there is a semi-standardized query language called SQL (structured query language). One major advantage of the relational model is that if a database is designed efficiently according to Codd rules [Codd 1990], there should be no duplication of any data, which helps to maintain database integrity. DBMS
do also provide powerful locking mechanisms to allow parallel reading and writing without data corruption.

Needless to say, there are other ways to exchange data like the Common Object Request Broker Architecture (CORBA) [Hu 1998]. This standard provides an intermediary object-oriented layer which handles access to the data between server and client. Another recently emerging way to exchange data are web services [Stein 2003] which will be described later.

Data Warehouse and Federated Database System

Genomic management systems allow to query data assembled from different heterogeneous data sources. They are based on two different approaches:

- Data warehouse
- Federated database system

A data warehouse is a collection of data specifically structured for querying and reporting [Kimball 1996]. Therefore data has to be imported in regular intervals from sources of interest. These data constitutes and acts like a centralized repository. Applications can query these data efficaciously and create reports. Implemented data marts duplicate content in the data warehouse and allow faster responses due to much higher granularity of the information. The drawbacks of a data warehouse are that the timeliness of the content depends on the update interval of the external data sources. This updates can be very time consuming and may result in higher storage requirements and operating costs.
Federated database systems overcome these downsides by directly accessing external data through federated database servers [Sheth 1990]. Integration of external data can be complete (all data can be accessed) or partial (only information needed is available through the server). Shortcomings of federated databases are that queries spanning different data sources at different locations tend to be slow. Due to different query styles, dialects, and data formats federated database servers are quite complex.

The Sequence Retrieval System (SRS) [Zdobnov 2002] initially developed at EMBL and EBI uses an interesting approach by combining the features of data warehouses and federated database systems. SRS is on the one hand heavily indexing locally stored genomic flat file databases and on the other hand it allows to query database management systems on different sites. An example for a federated approach is the Mouse Federated Database of the Comparative Mouse Genomics Centers Consortium

http://www.niehs.nih.gov/cmgcc/dbmouse.htm

Software Architecture

To meet the requirements of pharmacogenomic data processing systems, a sophisticated software architecture has to be employed. Less complex tasks like microarray image analysis or gene expression clustering can be performed on a commonly used workstation. In this case applications are installed locally on a client machine where all computational tasks are performed. Required databases are either installed locally or can be accessed via the local area network (LAN) or the Internet. This kind of direct client-
server access is characteristic for two-tier systems (Figure 1). In a two-tier architecture the application uses the data model stored in the enterprise information system (EIS), but does not create a logical model on top of it. All the business logic is packed into the client application and therefore increased workstation performance is required as soon as the applications are getting more complex or computational intensive. Furthermore, applications and database clients have to be deployed and kept up-to-date in order to adapt to new interfaces on the server side or to add new business logic to the system. Although there is a technology provided by Sun Microsystems called Java Web Start to automate this cumbersome task, only a few software vendors are supporting it. In general, two-tier software application design is ideal for prototyping, for applications known to have a short life time, or for systems where the Application Programming Interfaces (APIs) will not change. Typically, this approach is used for small applications where development costs as well as development time are intended to be low.

Most of the drawbacks of two-tier architectures can be avoided by moving to a three-tier architecture (Figure 2) with an application server as central component. In a three-tier architecture the separation of presentation, business, and data source logic becomes the principal concept [Fowler 2002]. Presentation logic is about how to handle the interaction between the user and the software. This can be as simple as a command-line or text-base menu system, a client graphical user interface (GUI), or a HTML-based browser user interface. The primary responsibility of this layer is to display information to the user and to interpret commands from the user into actions upon the business and data source logic. The business logic contains what an application needs to do for the
domain it is working with. It involves calculations based on inputs and stored data, validation of data coming from the presentation layer, and figuring out exactly what data source logic to dispatch depending on commands received from the presentation layer. The data source logic or EIS is about communicating with other systems that carry out tasks on behalf of the application, like transaction monitors or messaging systems. But for most applications the biggest piece of data source logic is a database, which is primarily responsible for storing persistent data. The usage of a three-tier architecture leads to the following advantages:

- easier to modify or replace any tier without affecting the other tiers (maintenance)
- separating the application and database functionality leads to better load balancing and therefore supports an increasing number of users or more demanding tasks
- adequate security policies can be enforced within the server tiers without hindering the clients

The two major enterprise development platforms Java 2 Enterprise Edition (J2EE) and Microsoft .Net are supporting this kind of software architecture. They can be seen as a stack of common services, like relational database access, messaging, enterprise components, or support for web services, that each platform provides to their applications. With this knowledge in the back of one's mind, the question which platform to use can be answered based on the expertise of the team members, their preferences, and based on the existing hardware and software infrastructure.
The next step in the evolution of distributed systems are web services. The concept behind is to build applications not as monolithic systems, but as an aggregation of smaller systems that work together towards a common purpose. Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web [Thallinger 2002]. Web services communicate using HTTP and XML and interact with any other web service using standards like Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description Discovery and Integration (UDDI) services, which are supported by major software suppliers. Web services are platform independent and can be produced or consumed regardless of the underlying programming language. The main limitations of web services are the network speed and round trip time latency. An additional limitation is the use of SOAP as the protocol, since it is based on XML and HTTP, which degrades performance compared to other protocols like CORBA.

**Hardware**

Life science is becoming increasingly quantitative as new technologies facilitate collection and analysis of vast amounts of data ranging from complete genomic sequences of organisms to three-dimensional protein structure and complete biological pathways. As a consequence, biomathematics, biostatistics and computational science are crucial technologies for the study of complex models of biological processes. The quest for more insight into molecular processes in an organism poses significant challenges on the data analysis and storage infrastructure. Due to the vast amount of
available information, data analysis on genomic or proteomic scale becomes impractical or even impossible to perform on commonly used workstations. Computer architecture, CPU performance, amount of addressable and available memory, and storage space are the limiting factors. Today, high performance computing has become the third leg of traditional scientific research, along with theory and experimentation. Advances in pharmacogenomics are inextricably tied to advances in high-performance computing.

Parallel Processing Systems

The analysis of the humongous amount of available data requires parallel methods and architectures to solve the computational tasks of pharmacogenomic applications in reasonable time [Buyya 1999]. State of the art technology comprises three different approaches to parallel computing:

- Shared memory systems
- Distributed memory systems
- Combination of both systems

Shared Memory Systems

In shared memory systems multiple processors are able to access a large central memory (e.g. 16, 32, 64GBytes) directly through a very fast bus system (Figure 3). This architecture enables all processors to solve numerical problems sharing the same dataset at the same time. The communication between processors is performed using the shared memory pool with efficient synchronization mechanisms making theses systems very
suitable for programs with rich inter-process communication. Limiting factors are the relative low number of processors that can be combined and the high costs.

**Distributed Memory Systems**

In general, these systems consist of clusters of computers, so called nodes, which are connected via a high-performance communication network (Figure 4). Using commodity state-of-the-art calculation nodes and network technology, these systems provide a very cost efficient alternative to shared memory systems for dividable, numerical computational intensive problems that have a low communication/calculation ratio. On the contrary, problems with high inter-processor communication demands can lead to network congestion, which is decreasing the overall system performance. If more performance is needed, this architecture can easily be extended by attaching additional nodes to the communication network.

**Grid Computing**

Grid computing is an emerging technology, poised to help the life science community manage their growing need for computational resources. A compute grid is established by combining diverse heterogeneous high performance computing systems, specialized peripheral hardware, PCs, storage, applications, services, and other resources placed over various locations into a virtual computing environment. For every numerical problem the appropriate computing facility in a world wide resource pool can be harnessed to contribute to its solution. A computing grid differs from the earlier
described cluster topology mainly by the fact that there is no central resource management system. In a grid every node can have its own resource management system and distribution policy. Grid technologies promise to change the way complex life science problems are tackled and help to make better use of existing computational resources [Avery 2002]. Soon, a life scientist will look at the grid and see essentially one large virtual computer resource built upon open protocols with everything shared: applications, data, processing power, storage, etc, all through a network.

Partitioning

In order to use the parallel features of a high performance computing facility, the software has to meet parallel demands, too. A numerical problem that has to be solved in parallel must be divided into subproblems that can be subsequently delegated to different processors. This partitioning procedure can be done either with so-called domain decomposition (Figure 5) or functional decomposition (Figure 6).

The term domain decomposition describes the approach to partition the input data and to process the same calculation on each available processor. Most of the parallel-implemented algorithms are based on this approach dividing the genomic databases into pieces and calculating e.g. the sequence alignment of a given sequence on a subpart of the database. The second and simplest way to implement the domain decomposition on a parallel computing system is to take sequentially programmed applications and execute them on different nodes with different parameters. An example is to run the well known BLAST [Altschul 1990] with different sequences against one database by giving every
node another sequence to calculate. This form of application parallelization is called swarming and does not need any adaptation of existing programs.

On the other hand functional decomposition is based on the decomposition of the computation process. This can be done by discovering disjoint functional units in a program or algorithm and sending these subtasks to different processors (Figure 6).

Finally in some parallel implementations combinations of both techniques are used, so that functional-decomposed units are calculating domain-parallelized sub-tasks.

Data Storage

Drug discovery related data storage and information management requirements are doubling in size every six to eight months, more than twice as fast as Moore’s Law predictions for microprocessor transistor counts. For life science organizations, data is necessary, but not sufficient for organizational success. They must generate information – meaningful, actionable, organized, and reusable data. Data must be stored, protected, secured, organized, distributed, and audited, all without interruption.

State of the art storage architecture comprises the following solutions:

- Directly attached storage (DAS)
- Network attached storage (NAS)
- Storage area networks (SAN)
- Internet SCSI (iSCSI)
**Directly Attached Storage**

This historically first and very straightforward method can be seen today in every PC: hard disks, floppy disks, CD-ROM or DVDs are attached directly to the main host using short internal cables. Although in the mainframe arena storage devices, hard disks or tape drives are separate boxes connected to a host, this configuration is from a functional perspective equivalent to standard PC technology. DAS is optimized for single, isolated processor systems and small data volumes delivering good performance at low initial costs.

**Network Attached Storage**

NAS is defined as storage elements which are connected to a network providing file access services to computer systems. These devices are attached directly to the existing local area network (LAN) using standard TCP/IP protocols. NAS systems have intelligent controllers built in, which are actually small servers with stripped operating systems, to exploit LAN topology and grant access to any user running any operating system. Integrated NAS appliances are discrete pooled disk storage subsystems, optimized for ease-of-management and file sharing, using lower-cost, IP-based networks.

**Storage Area Networks**

A SAN is defined as a specialized, dedicated high-speed network whose primary purpose is the transfer of data between and among computer systems and storage
elements. Fibre Channel is the *de facto* SAN standard network protocol, although other network standards like iSCSI could be used. SAN is a robust storage infrastructure, optimized for high performance and enterprise-wide scalability.

**Internet SCSI (iSCSI)**

SCSI is a collection of standards which define I/O buses primarily intended for connecting storage subsystems or devices to hosts through host bus adapters. iSCSI is a new emerging technology and is based on the idea of the encapsulation of SCSI commands in TCP/IP (most widely used protocol to establish a connection between hosts and exchange data) packages and sending them through standard IP based networks. With this approach iSCSI storage elements can exist anywhere on the LAN and any server talking the iSCSI protocol can access them.
Conclusion

A pharmacogenomic data management system has to combine public and proprietary genomic databases, clinical data sets, and results from high-throughput screening technologies. Currently the most important public available biological databases require disk space in the magnitude of one Terabyte (1000 Gigabyte). Considering the exponential growth of data, it can be expected that the storage requirements for proteomics will claim Petabytes (1000 Terabyte). Even more, systems for personalized medicine will be in the range of Exabytes (1000 Petabyte). Assuming that the storage capacity doubles every year it is imaginable that in ten years working with Petabytes will be a standard procedure in many institutions. To facilitate the management, handling, and processing of this vast amount of data, such systems should comprise data mining tools embedded in a high performance computing environment using parallel processing systems, sophisticated storage technologies, network technologies, database and database management systems, and application services. Integration of patient information management systems with genomic databases as well as other laboratory and patient-relevant data will represent significant challenges for designers and administrators of pharmacogenomic information management systems. Unfortunately, the lack of international as well as national standards in clinical information systems will require the development of regional specific systems. Additionally all arising security issues concerning the sensitivity of certain types of information have to be
solved in a proper manner. To accomplish all this stated issues, considerable endeavors have to be undertaken to provide the necessary powerful infrastructure to fully exploit the promises of the postgenomic era.

Acknowledgments

The authors express their appreciation to the staff of the Institute for Genomics and Bioinformatics for valuable comments and contributions. This work was supported by bm:bwk, GEN-AU:BIN, Bioinformatics Integration Network.
References


Avery P. Data Grids: a new computational infrastructure for data-intensive science.


In a two-tier architecture the application logic is implemented in the application client, which directly connects to the Enterprise Information System (Database).
A three tier architecture enforces the separation of presentation, business, and data tier. This architecture is intended to allow any of the three tiers to be upgraded or replaced independently as requirements change.
A shared memory system consists of multiple processors that are able to access a large central memory directly through a very fast bus system.
In a distributed memory architecture the various computing devices (e.g. PCs) have their own local memory and perform calculations on distributed problems. Input data and results are exchanged via a high-performance inter-process communication network.
Domain or data decomposition is a computational paradigm where data to process is distributed and processed on different nodes.
Figure 6: Functional Decomposition

Functional decomposition divides the computational problem in functional units, which are distributed onto different working nodes processing the same data.