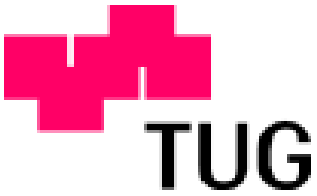
DESIGN AND DEVELOPMENT OF A DATABASE AND RETRIEVAL SYSTEM FOR RESEARCH IN CELLULAR AGING

MARTINA PITZL



MASTER THESIS

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

Evaluator:
Univ.-Prof. Dipl.-Ing. Dr. techn. Zlatko Trajanoski

Graz, September 2007

For my parents
Für meine Eltern

Acknowledgments

I would like to thank Robert Rader, Hubert Hackl and Thomas Burkhard for their advice and support during the development and implementation of this application.

Abstract

The underlying processes that lead to premature senescence of an organism and apoptosis of its cells are still not fully understood. But it is believed that oxidative stress is related to these processes. The National Research Network tries to uncover those mechanisms.

The objective of this thesis was the development and implementation of a centralized, web-based database for storage and retrieval of information about genes associated with cellular aging, oxidative stress and apoptosis. The database should ease the exchange of data throughout the network and provide access to formerly unavailable data.

A web-based database system based on the Java Enterprise Edition 5 platform, the open source MDA tool AndroMDA and the Struts framework, called GiSAO.db (Genes in Senescence, Apoptosis and Oxidative Stress), was developed. The database stores gene expression data gained from microarray experiments together with ortholog data. Annotation data is stored in both an experiment centered and gene centered way. Favorite gene lists, that can be shared amongst users, can be created. In addition tags can be added to genes and microarray experiments. Also a search for KEGG pathways was integrated into the database.

GiSAO.db allows to display gene expression from different experiments based on several adjustable criteria and thus to easily gain information about genes with specific expression patterns. Favorite gene lists allow the focus on genes of special interest to research groups and through sharing of those lists the finding of genes of common interest. With the addition to add tags to experiments, classes and genes, not only annotation data in text format can be added, but they can also be assigned to one or more hierarchically ordered categories providing further fine grained classification.

Keywords: database, microarray experiments, aging research, Java EE, MDA

Zusammenfassung

Die Prozesse, die zur vorzeitigen Alterung eines Organismus und zur Apoptosis dessen Zellen führen, sind für Forscher noch nicht vollends nachvollziehbar. Man glaubt aber daran, dass oxidativer Stress mit diesen Prozessen in Verbindung steht. Das National Research Network arbeitet daran, ebendiese Mechanismen zu erforschen.

Das Ziel dieser Masterarbeit war die Entwicklung und Implementierung einer zentralisierten, web-gestützten Datenbank für die Speicherung von und Suche nach Informationen über Gene, die in Verbindung mit zellulärem Altern, oxidativem Stress und Apoptosis stehen. Die Datenbank sollte den Austausch von Daten über das gesamte Netzwerk vereinfachen und Zugung zu davor nicht verfügbaren Daten gestatten.
Es wurde ein web-gestütztes Datenbanksystem, basierend auf der Java Enterprise Edition 5 Technologie, dem Open Source MDA Tool AndroMDA und dem Struts Framework, namens GiSAO.db (Genes in Senescence, Apoptosis and Oxidative Stress), entwickelt. Die Datenbank speichert Genexpressionsdaten, die durch Microarray Experimente generiert wurden, zusammen mit Orthologen Daten. Annotationsdaten werden dabei sowohl basierend auf Experimenten als auch basierend auf Genen gespeichert. Listen mit speziellen Genen von Interesse (Favorite Gene Lists), die auch für andere Forschungsgruppen zugänglich gemacht werden können, können kreiert werden. Außerdem können Tags sowohl zu Genen als auch zu Microarray Experimenten hinzugefügt werden. Zusätzlich wurde eine Suche nach KEGG Pathways in die Datenbank integriert.
Die GiSAO Datenbank erlaubt die Möglichkeit, Geneexpressionen von verschiedenen Experimenten basierend auf mehreren anpassbaren Kriterien darzustellen und ermöglicht somit das Gewinnen von Informationen über Gene mit speziellen Genexpressionsmustern. Favorite Gene Lists erlauben das Fokussieren auf Gene von speziellem Interesse für bestimmte Forschungsgruppen. Außerdem können dadurch, dass diese Listen auch für andere Gruppen zugänglich sind, Gene von gemeinsamen Interesse gefunden werden. Mit dem Hinzufügen von Tags zu Experimenten, Klassen und Genen können diese nicht nur mit zusätzlicher Information in Textform ausgestattet werden, sondern es kann auch eine feine Klassifikation erreicht werden durch die Zuordnung von Tags zu einer oder mehreren hierarchisch aufgebauten Tag Kategorien.

Schlüsselwörter: Datenbank, Microarray Experimente, Altersforschung, Java EE, MDA

Contents

List of Figures

List of Tables

Glossary

A

AAS	Authentication and Authorization System
ACL	Access Control List
API	Application Programming Interface

C

CCI	Client Connector Interface
cDNA	Complementary Deoxyribonucleic acid
CORBA	Common Object Request Broker Architecture

D

DAO	Data Access Object
DBMS	Database Management System
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Management Language
DNA	Deoxyribonucleic acid
DTO	Data Transfer Object

E

EIS	Enterprise Information System
EJB	Enterprise Java Bean
EJB-QL	EJB Query Language
ENC	Enterprise Naming Context
EST	Expressed Sequence Tag

H

HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol

J

Java EE	Java Enterprise Edition
JDBC	Java Database Connectivity
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
JSF	Java Server Faces

JSP	JavaServer Pages
JSTL	JavaServer Pages Standard Tag Library

K

KEGG	Kyoto Encyclopedia of Genes and Genomes

L

LDAP	Lightweight Directory Access Protocol
LIMS	Laboratory Information and Management System

M

MARS	Microarray Analysis and Retrieval System
MAGE-ML	Microarray Geneexpression Markup Language
MDA	Model Driven Architecture
MDB	Message Driven Bean
MIAME	Minimum Information About a Microarray Experiment
mRNA	messenger Ribonucleic acid
MVC	Model-View-Controller

N

NRN	National Research Network

O

OMG	Object Management Group

P

PCR	Polymerase Chain Reaction
PIM	Platform Independent Model
PSM	Platform Specific Model

R

RMI	Remote Method Invocation
ROS	Reactive oxygen species
RPC	Remote Procedure Call
RT-PCR	Reverse transcription polymerase chain reaction

S

SQL	Standard Query Language
SOAP	Simple Object Access Protocoll
SPI	Service Provider Interface

SSO	Single Sign-On
T	
TLD	Tag Library Description
U	
UML	Unified Meta Language
W	
WSDL	Web Service Description Language
X	
XML	eXtendensible Meta Language
XMI	XML Metadata Interchange

1 Introduction

1.1 Background

The main research goal of the Aging Research group of the National Research Network (NRN) [1], a research network funded by the Austrian Science Funds [2] since 2004, is to uncover the mechanisms that lead to premature senescence and apoptosis in an organism. It is assumed that oxidative stress plays a great role as trigger of aging processes at both a cellular and organismic level (see Figure 1). It can cause damage to biological macromolecules and induce specific signalling cascades, leading to the death of a cell. Unfortunately the underlying processes are not yet completely understood. The experiments currently undertaken should enable scientists to discover relationships between oxidative stress, senescence and apoptosis. The organisms included in the research projects of the NRN are human cells, mice and yeast [3].

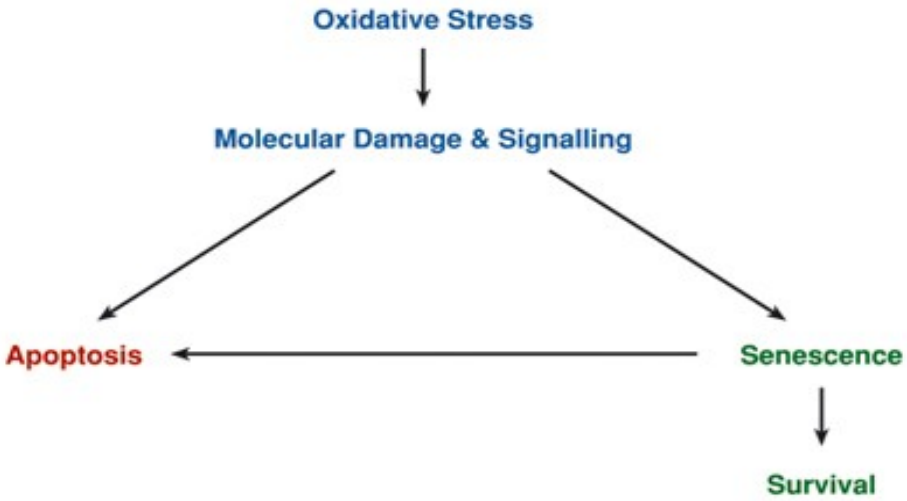


Figure 1: possible relationship between oxidative stress, senescence and apoptosis; taken from [1]

Main goals of the research are [3]:

- new insights into the process of cellular senescence in various species and tissues
- a distinction between unique and common pathways leading to the senescent phenotype
- a clarification of the role of apoptosis for cellular aging
- the elucidation of molecular pathways that are triggered by ROS (reactive oxygen species) generation, which lead to molecular damage and changes in signal transduction to cause senescence and eventually apoptotic cell deatch.

The research in the NRN is divided into three projects areas: senescence, apoptosis and oxidative stress. The senescence area explores the changes in the physiological functions that occur when an organism ages. Of special interest are changes in cell proliferation, survival and apoptosis, cell differentiation and gene expression. Genes that are contributing to the aging process should be identified. Other goals are the discovery of evolutionary conserved mechanisms of aging and the revelation of factors that lead to accelereated aging of an organism. In the apoptosis research focus is put on identifying molecular mechanisms involved in programmatic cell death and finally, the oxidative stress research area deals with how far oxidative stress generated intracellularly contributes to the rate of aging of an organism [3].

1.2 Objectives

The objective of this thesis was the development and implementation of a centralized, web-based database for storage and retrieval of information about genes associated with cellular aging, oxidative stress and apoptosis. The gene expression data stored in the database is based on microarray experiment data and subsequent bioinformatic evaluation. This database should provide an integrated, up-to-date view on previously unavailable data for the NRN. The gene information that will be stored in the database is concerned with:

- experimental details about genomic profiling experiments regarding all aging models (human, mouse and yeast) so far
- follow-up validation of singled-out candidates by employing state-of-the-art bioanalytics such as quantitative RT-PCR as well as functional bioassays
- related bioinformatic analysis either available from open-source relational databases or through individualized bioinformatic research on selected candidate genes
- tertiary annotation about biological pathways and processes
- gene orthology between organisms in order to concomittantly address questions regarding the functional role of the respective gene regarding aging in other model organisms as well as human cellular systems

In addition to gene expression datasets and orthology data, the database should also allow the creation of favorite gene lists containing genes of special interest to a specific group allowing to add further proprietary information. Users should be able to share these lists with other members of the NRN. As another feature users should be enabled to add tags to microarray experiments, their classes and to specific genes. All these tags are assigned to a tag category,

making it possible to further classify experiments, classes and genes.

In short, this database should provide information about experimental data gathered by the NRN as well as information about genes and gene families, thus ease the exchange of data and speed up the workflow within the NRN.

2 Methods

2.1 Gene Expression and Microarray Technology

Gene expression is the term that describes transcription of DNA into messenger RNA (mRNA) which in turn is then translated to proteins. Proteins perform a lot of critical functions of a cell, like providing structure or acting as enzymes, antibodies and as a transport medium[4]. Looking at gene expression patterns, at how much mRNA is produced and which genes are over- or underexpressed, gives scientists an insight into how a organism works, how it can adapt to stimuli from its environment and what changes when an organism is deseased. Since there are thousands of genes expressed simultaneously, there is a need for a technology that enables scientists to look at many genes at the same time to gain an understanding of the global gene expression pattern. For this purpose, microarrays were developed [5].

Microarrays make it possible to explore the expression of thousands of genes simultaneously. They can be used both to determine the gene expression of a single sample or to compare gene expressions in two different cell types or tissue samples, for example in healthy and diseased tissue. There are two main types of DNA microarrays, spotted arrays and oligonucleotide arrays.

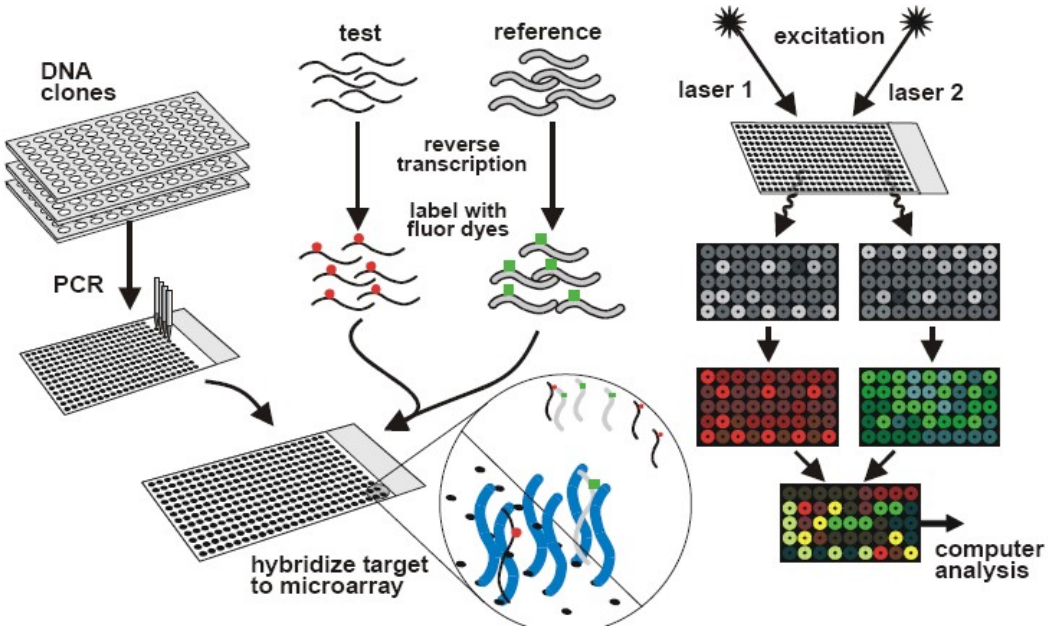


Figure 2: Microarray experiment: Test and reference samples are reverse transribed, labeled with different dyes and hybridized to the array. A laser then scans the array and creates to different images which are overlayed to gain relative intensities. These images are then further processed; taken from [6]

Spotted arrays use probes that are oligonucleotides, cDNA or small fragments of PCR products. When using cDNA, RNA is reverse transcribed to cDNA and fluorescently labeled. Usually two different dyes for the two samples to be compared are used. The samples then hybridize to the elements on the array. After hybridization, the array is scanned by a laser and independent images for the test and the reference channels are generated. Those independent images are overlayed for getting a color image where the relative intensities are used to identify up- and downregulated genes (see Figure 2). The second possible probe type, oligonucleotides, are designed for specific genes and can be designed to represent a unique part of a transcript.

On in situ arrays, only one labeled sample is used, thus the result gives an estimate of the absolute value of the gene expression. Genes or ESTs are represented multiple times on an array. Each time a different sequence designed to hybridize to different regions of the same RNA is used. Furthermore mismatch control probes are used. They differ from the probes on a single base in a central position. These mismatch probes allow to subtract background and cross-hybridization signals and the disrimination between real signals and those caused by non-specific or semispecific hybridization [5],[6].

2.2 Database Systems

Data can be defined as some information stored in a computer on a hard disk, a DVD or any other storage medium, but without meaning unless put in a certain context. Databases provide this context by storing data as a collection of related facts that are associated with real world objects. A database has a structure, also called schema, modelling data objects and the relationships between them. A database should be able to store data without redundancy and should assure data integrity [7]. Different models, like hierarchical and network models, have been proposed over the years, but the only widespread one is the relational database model which has been introduced by the british scientist E.F. Codd [7],[8] and is now available in many implementations like MySQL [9], Oracle [10] or IBMs DB2 [11].

Database systems try to separate application programs from data management and are usually comprised of three tiers, an external or application tier, a logical tier, the Database Management System (DBMS), and an internal tier, the actual database (see Figure 3) [8].

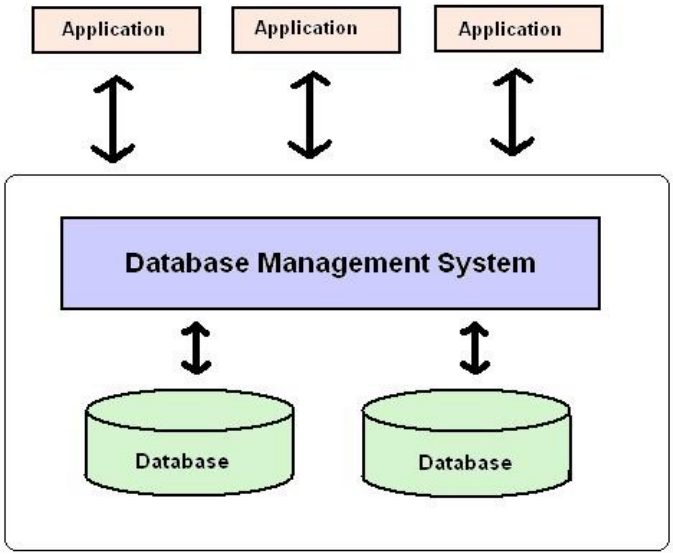


Figure 3: Database System constisting of application tier, database management system and multiple databases

A Database Management System manages the creation, access and manipulation of data of multiple databases. Management is achieved with three main languages, the Data Definition Language (DDL) for the creation of database schemas, the Data Manipulation Language (DML) for the manipulation of contained data and the Data Control Language (DCL) for access control [7],[8].

2.2.1 Relational Databases

Relational databases model data objects as two-dimensional tables. Each table is called an entity and represents a real world object of interest. Entities have relationships, logical associations, with each other and consist of a number of attributes. Instances of entities differ in the values contained for certain attributes and are represented as rows (also called tuple) in the two-dimensional table. Each instance is identified uniquely by a primary key, relationships are modelled by foreign keys, references of primary keys of the linked table. To modify database structure and manipulate data, the Structured Query Language (SQL), which is based on relational algebra, has been developed [8]. Table 1 shows an overview of often used SQL commands.

Language	Commands
Data Definition Language	`CREATE, DROP, ALTER,`
Data Manipulation Language	`INSERT, UPDATE, DELETE`
Data Control Language	`GRANT, REVOKE`

Table 1: Selection of SQL commands

2.3 Java Enterprise Edition 5

The Java Enterprise Edition is the standard technology „for developing portable, robust, scalable and secure server-side Java applications“ [12]. Enterprise applications are often comprised of many tiers to separate parts of the application into independent pieces. Such a multi-tier application extends the classical client -tier application by adding an additional layer, an application server, between a client machine and the data storage. Thus, an enterprise application is usually comprised of the following tiers [13],[14] (see Figure 4):

- Client Tier
- Web (or Presentation)Tier
- Business Tier
- Enterprise Information System (EIS)

Client tier components are running on the client machine, Web and Business Tier components run on a Java EE application server, e.g. Jboss [15], BEA WebLogic [16] or IBM WebSphere [17], and EIS components run on an EIS server [13]. The EIS system constitutes the persistence tier, typically a database like Oracle is used, but also flat files or XML files are possible as storage medium [18].

Java EE consists of several packets and specifications to implement the web and business tier of an application. Those specifications include Java Servlets, JavaServer Pages and Enterprise Java Beans. Java EE provides a standard way for the interaction of all these different technologies by using the Java Naming and Directory Interface Enterprise Naming Context (JNDI ENC), so that different components are able to look up and inject resources, and eXtensible Markup Language (XML) as file format for deployment descriptors that are used by application servers [19] .

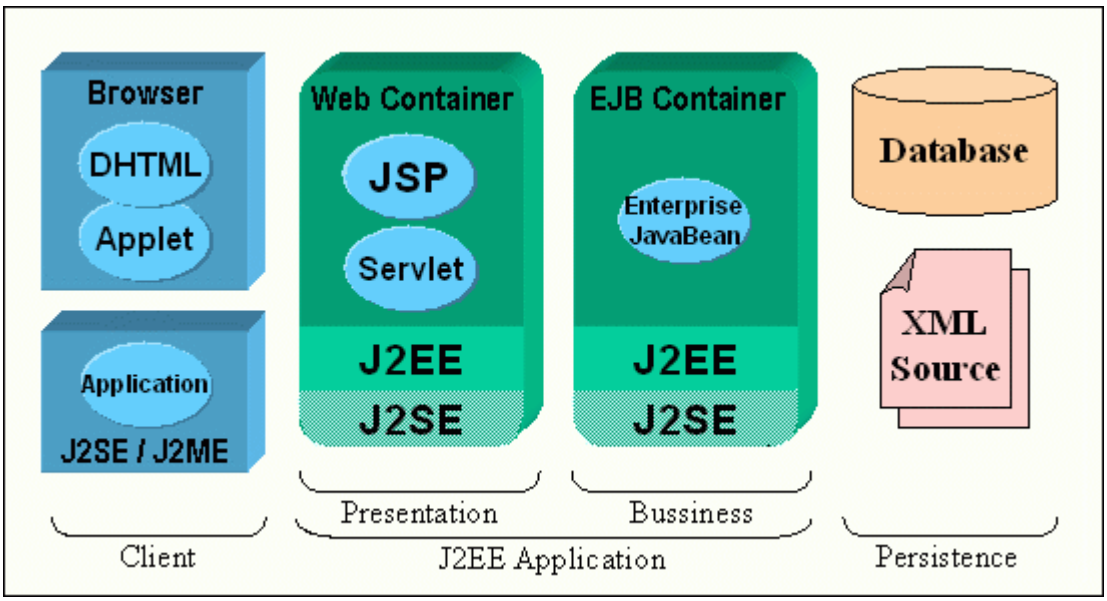


Figure 4: Different tiers of an Java EE application; taken from [18]

2.3.1 Servlets

In a typical Client – Server architecture a client sends a request to the server, the server processes this request and sends back a response. The most common used protocoll for communication is HTTP and the response of the server is typically presented in a web browser. The request can be for a static resource, like a simple HTML page, but it can also be a form which needs to be processed dynamically depending on the values handed over to the server [20].

Servlets have been developed for dealing with HTML request and response objects. Servlets are Java classes that implement the `javax.servlet.http.HttpServlet` Interface and receive the HttpRequest object as a parameter. With special methods like `doGet()` or `doPost()` the HTTP POST and GET requests can be processed. Servlets can implement a workflow like writing data from a form into a database or fetch data from it. Thus with servlets dynamic return content can be created. The result of the processing is sent back to the client via the Http response object [13],[20].

2.3.2 JSP

Servlets are suited for implementing a taskflow, but it is not practical to create HTML output with them. In theory they could be used to write HTML tags, but putting those into a string is complex and error-prone. Program code developers considered that it will be easier the other

way round, including Java code into HTML pages. The development of JavaServer Pages was the result. JavaServer Pages (JSPs) are used for adding dynamic content into otherwise static HTML pages. They are basically servlets and are compiled to Java class files and then executed before the HTML page is sent back to the client. The resulting HTML page contains the result of the execution of the JSP class file [14],[20].

2.3.3 Tag Libraries

Tag Libraries extend the JSP specification and make it possible to reduce the necessity for Java code in JavaServer pages [21]. A Tag Library consists of Tag Handlers, Java classes implementing the tag functionality, and a Tag Library Description (TLD), an XML file describing tags and their possible attribute values. Each tag has a name and a prefix. The prefix uniquely ties the name to a specific library, allowing tags with the same name but belonging to other libraries being used in one JSP file [20] .

Suns JavaServer Pages Standard Tag Library (JSTL) 1.2 consists of four libraries, Core (standard tasks like conditional processing or iterating over a collection), XML (XML processing), I18N & Formatting (Internationalization and Formatting) and Database Access [22]. To use tags from a tag library in a JSP file, it has to be included using a taglib directive. A taglib directive for including the core JSTL library would look like as follows:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
```

The prefix declares the namespace and the uri a logical name [21]. Apart from the JSTL tag library, a vast number of other free custom tag libraries have been developed. A list can be found at [23].

2.3.4 JDBC

The JDBC Application Programming Interface (API) is written in Java programming language amd was developed for the communication between a database and a Java program. It is comprised of the packages `java.sql` and `javax.sql`. The connection is set up using JDBC drivers which translate the requests from Java programs to database specific commands [24]. There exist special drivers for each database, a list from different vendors can be found at [25]. JDBC drivers not only make it possible to connect to relational databases but also to other datasources, like flat files containing data in tabular format [24].

The routines of the API make it possible to open a connection to a database, execute a SQL query or manipulate data in the database. A query returns a result set which then can be

processed in the Java class.

The steps necessary for executing a simple query are [26]:

- load the driver for the underlying SQL database
- get a connection
- create a statement
- execute the SQL query and get a result set
- close the connection
- do something with the result set

2.3.5 EJB 3.0

The Enterprise JavaBeans (EJB) architecture is „an architecture for the development and deployment of component-based business applications“[27]. Enterprise JavaBeans live in an EJB Container, which is part of an application server. Such an EJB container is responsible for controlling the life cycle of an EJB, for instance pooling, a naming and directory service, transaction management, a message service and persistence handling. With the introduction of version 3.0, the Enterprise JavaBeans technology has been extensively remodelled with the intention to simplify the development process for developers and make the use of this technology more intuitive. EJBs can now use annotions, added to the Java programming language in its version 5.0 [21], rendering it unnecessary to make all specification in a deployment descriptor file. Also the need to implement methods that were needed by the interface, but not by the application was eliminated [19].

2.3.5.1 Java Persistence and Entity Beans

Since EJB 3.0 , persistence now has its own specification, which is called Java Persistence 1.0 API, and is no longer part of the EJB specification. With Java Persistence, automatic mapping of Java Objects to and from relational databases is possible.

Entity Beans are plain Java objects that can be serialized and instantiated with the `new()` operator. Just like entities in a relational database, entity beans describe the state and behaviour of real life objects. They contain properties along with getter and setter methods corresponding to columns of the specific entity they represent. They provide a simple and reusable way to access the underlying database. Creation, modification and removal of entity beans and database queries are performed by the entity manager. The entity manager is responsible for the Object/Relational mappings. EJBs only become persistent when

interacting with the entity manager. Entities can either be managed or unmanaged where managed means that the entity bean is attached to the entity manager and its state is synchronized with the database. An entity becomes unmanaged for example, when it is sent back to a client as a return value. Persistence context is usually provided as long as a transaction takes place. Since entity beans can now be sent directly back to the client via serialization, technically, value objects are no longer needed [19].

2.3.5.2 Session Beans

Session Beans are used to implement the workflow of an application. This includes calculations using data in the database through entity beans and changing the content of the database again with the use of entity beans or direct access via the entity manager. There exist two types of session beans, stateful and stateless session beans, which differ in their lifecycles and how they maintain a conversation with a client [19].

2.3.5.2.1 Stateless Session Beans

Stateless session beans don't maintain a state from one method invocation to the next one, they are used for requests that only require a single method to perform this request. Each method is self-contained and represents a certain service which may or may not return a value to the client. A stateless session bean can be assigned to a new client with each method invocation. Its life cycle consists of only two states, either it does not exist or it is in the method-ready pool, meaning it can be assigned to any client needing a service from that bean [19].

2.3.5.2.2 Stateful Session Beans

In contrast to stateless session beans, stateful session beans maintain a conversational state to the client and are dedicated to this client through their whole lifetime. They hold data that is shared among all methods through various method calls, so one method invocation is dependent on the previous one. A stateful session bean can have three states: does not exist, method ready and passive. In the method ready state, methods can be invoked by a client. A stateful session bean may be passivated, for example when the client is not requesting any services for a longer period of time, and activated several times during its lifetime. During this transitions, the conversational state has to be preserved for future method invocations. How this is done depends on the Container, but one way to realize this is via standard Java

serialization [19].

2.3.5.3 JMS and Message Driven Beans

JavaMessage Service (JMS) is a vendor-independent API for sending asynchronous messages. To send a message both a connection to a JMS provider and a destination address for the message are required. Messages are either sent to topics or to queues, depending on the purpose of the message. JMS clients that send messages are called producers, that who receive messages are called consumers. All enterprise beans can send messages by using such a messaging service. When sending messages one has to be aware that security contexts and transactions are not propagated from the sender to the receiver.

“Message Driven Beans (MDBs) are stateless, server-side, transaction-aware components for processing asynchronous messages delivered via JMS”[19]. MDBs differ from other enterprise beans in the fact that they don't have a local or remote interface, a client cannot directly interact with a MDB. Since the execution of a MDB is completely decoupled from the client, he doesn't need to wait until execution is completed. Thus MDBs can be used for tasks where no reply value is needed and when the execution of a message would take a large amount of time, for example if a large file is being loaded into the database [19].

2.3.5.4 EJB-QL

EJB-QL is a query language for relational databases similar to SQL, but specifically developed to work with Java objects and is thus independent of the underlying database implementation. EJB-QL does not directly work on tables and relationships but with properties of entity beans and their respective relationships. It uses the abstract persistence schema of entities to identify beans and the relationship properties to navigate across relationships. The entity manager is responsible to translate the query to the specific native SQL dialect of the database in use. The native SQL query is then executed through a JDBC driver [19].

2.3.6 JNDI

The Java Naming and Directory Interface (JNDI) provides naming and directory services to Java applications [28]. The responsibility of a naming system is to provide a way to associate names with objects and find them by their names. Those associations are achived using so called bindings. A context is a set of name-to-object associations. All the bindings in one context have the same naming convention, meaning that all obejcts (or references thereof) are

named according to a specific syntax. Operations on a context include the lookup of an object, binding and unbinding of names and the listing of bound names. Directory services extend a naming service with the ability to allow objects to have attributes and to search them based on those attributes [28],[29]. JNDI only provides an interface and relies on existing naming services like CORBA, LDAP or RMI. It consists of an Application Programming Interface (API) and a Service Provider Interface (SPI). With the use of the SPI, different naming services can be plugged into JNDI [28].

2.3.7 Design Patterns

When developing software systems, programmers are likely to be confronted with recurring problems. Especially when working with multi-tier applications, one is assigned with the solution of several tasks like how to increase performance and throughput, how to make a system modular, flexible, maintable and extensable and how to hide the complexity of communication between different tiers. Over the years programmers have found solutions to exactly those problems, they are called Design Patterns. Design Patterns are thoroughly tested and refined solutions to common problems. They are language-independent and can be applied to any object oriented programming language. The next section will present an overview of some important and often used Design Patterns [20],[30].

2.3.7.1 Business Delegate

The Business Delegate Pattern is used to decouple the presentation from the business layer [31]. Implementation details, like complicated JNDI lookups or remote method invocations (RMIs), are encapsulated and thus hidden from the presentation layer. This makes it easier to apply changes in the presentation or business layers without affecting the other one. The Business Delegate is responsible for handling business exceptions, like remote or JMS exceptions, and either transforming them into exceptions that are of meaning to the client or first trying to recover from those business exceptions without notifying the client about them. It also can cache results and references to remote business services, thus decreasing network traffic. Business Delegates work hand in hand with a Lookup Service, which can be implemented as a separate component using the Service Locator pattern (see Figure 5). This design pattern will be introduced in the next section [30].

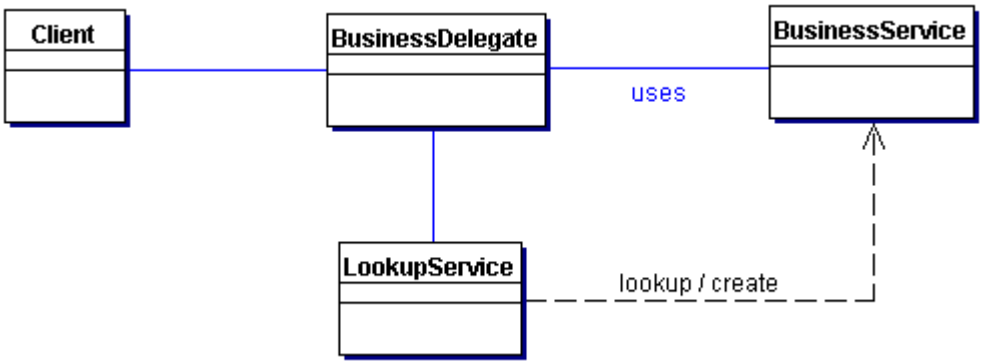


Figure 5: The client makes a request to the Business Delegate, which provides access to the underlying Business Service. The Business Service uses the Lookup Service to locate the Business Service components; taken from[30]

2.3.7.2 Service Locator

Before a client can make a call to remote enterprise beans or invoke a Java Message Service, the respective component has to be located. This operation is referred to as lookup. In Java EE, lookup is done with the JNDI Service. Service Locators are used to encapsulate vendor-specific JNDI lookup processes. They are often implemented as Singletons, meaning there is only one per application, providing a single access point for all operations (see Figure 6) [30].

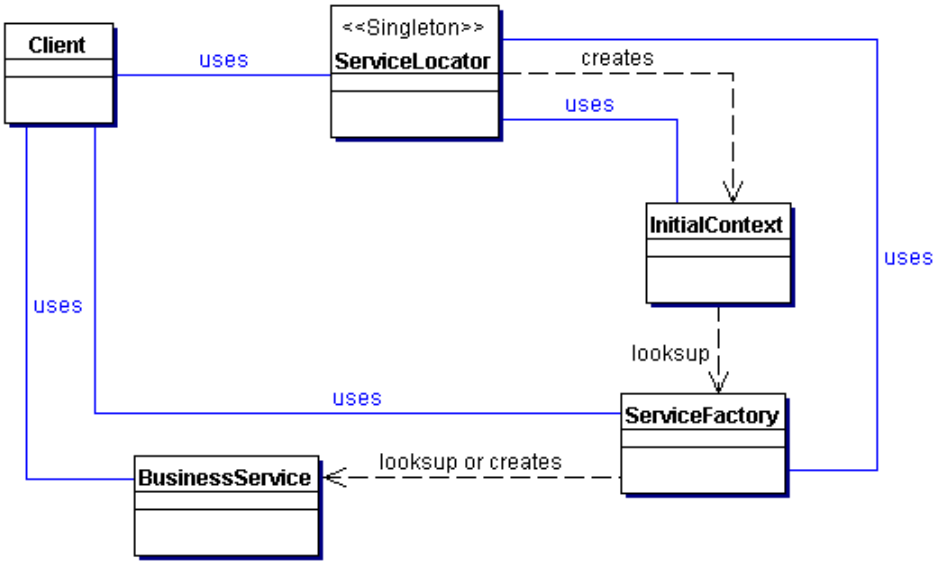


Figure 6: The Service Locator encapsulates the complex and vendor-dependent process of creating of JNDI Initial Context and lookup of Business Service methods.

2.3.7.3 Transfer Objects

Data Transfer Objects (DTOs), which are also known as Value Objects, are serializable Java classes that can be used for carrying data from one layer to another [31]. Those classes consist of properties and according getter and setter methods. DTOs make it possible to package data from multiple beans in one object, thus making repeated calls to retrieve the requested data unnecessary. This leads to a decrease in network traffic, because method calls to business methods may be remote calls, which require a lot of overhead. DTOs also provide an interface for interaction of different layers, so that there is no need for a layer to know implementation specifics of another layer (see Figure 7) [30].

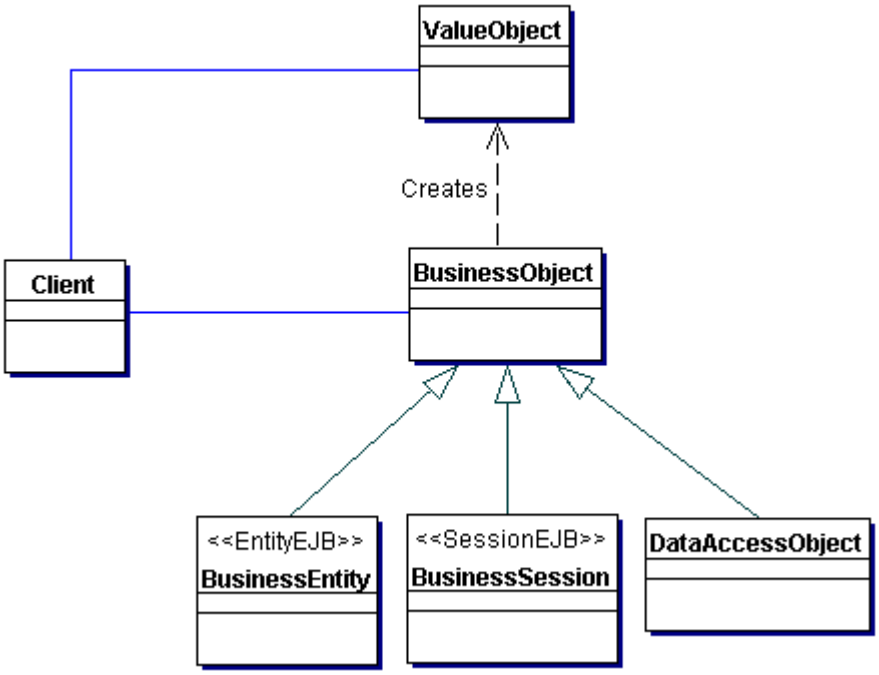


Figure 7: A Data Transfer Object, or Value Object, can be populated with data from different Business Objects, like Entity and Session Beans and Data Access Objects, and is then sent back to the client; taken from [30]

2.3.7.4 Data Access Objects

An application may need to access data from different data sources, like different database models or flat files. How this access is achieved is very dependent on the type of data storage. Data Access Objects (DAOs) are used as an abstraction to access data sources. They encapsulate the underlying storage mechanism and manage connections to store and query data by exposing interfaces to a business component. The DAO also may use a Data Transfer

Object to return data or receive data from a DTO to store it into a data storage (see Figure 8) [30].

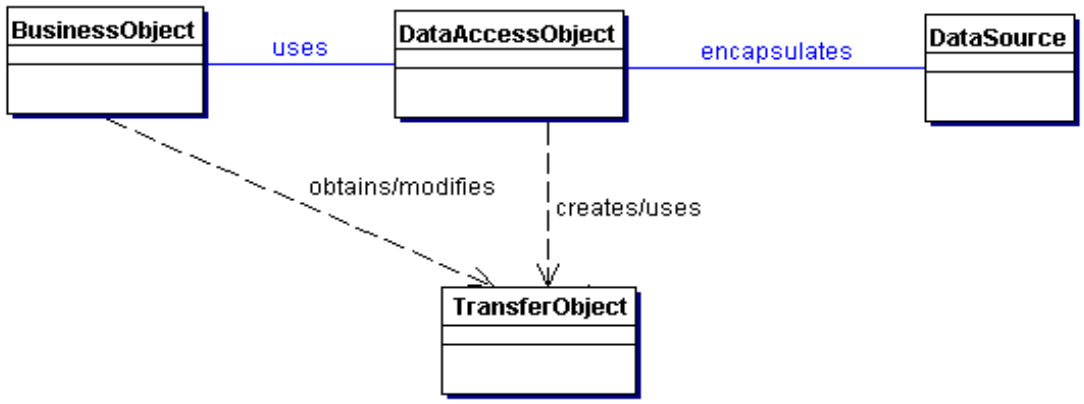


Figure 8: Business Objects use Data Access Objects to store data into or receive data from data storages. Transfer Objects may also be used to transfer data between a client and a Data Access Object; taken from [30]

2.3.7.5 Front Controller

A front controller represents a single entry point for requests to an application. When using only one entry point that manages all requests, control of the application can be centralized, which makes the application more manageable and secure. A front controller can manage authentication and authorization, global error handling, delegating calls to business methods and selecting a suitable view for the output. Without such a front controller navigation would have to be done in the view, leading to a mixture of view content and control code. Also each view would have to include code for different services, like authentification and so on. This would lead to duplicate code in several views, hence the overall application would be error-prone and it would be difficult to adapt to changes (see Figure 9) [30].

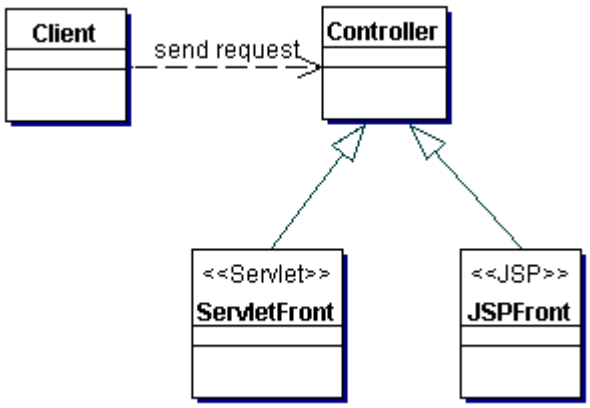


Figure 9: The client sends a request to the controller, which then delegates the business processing to a selected servlet and chooses a suitable view for output; taken from [30]

2.3.7.6 MVC

MVC is another design pattern whos goal it is to separate one application into different parts where each part has its own field of responsibility. It will then be possible to develop different parts independently from each other and to exchange one part with no effect on the other parts. As the name indicates, the MVC pattern consists of three components: the model, the view and the controller. The model contains the business logic, the view contains the presentation part of the application, for example JSP and HTML pages, and the controller is responsible for interconnecting and coordinating these two parts. Since a single controller is used for coordination, MVC makes use of the Front Controller design pattern. The three components communicate with each other over well defined interfaces, to make easy exchange of parts possible (see Figure 10) [20],[32],[33].

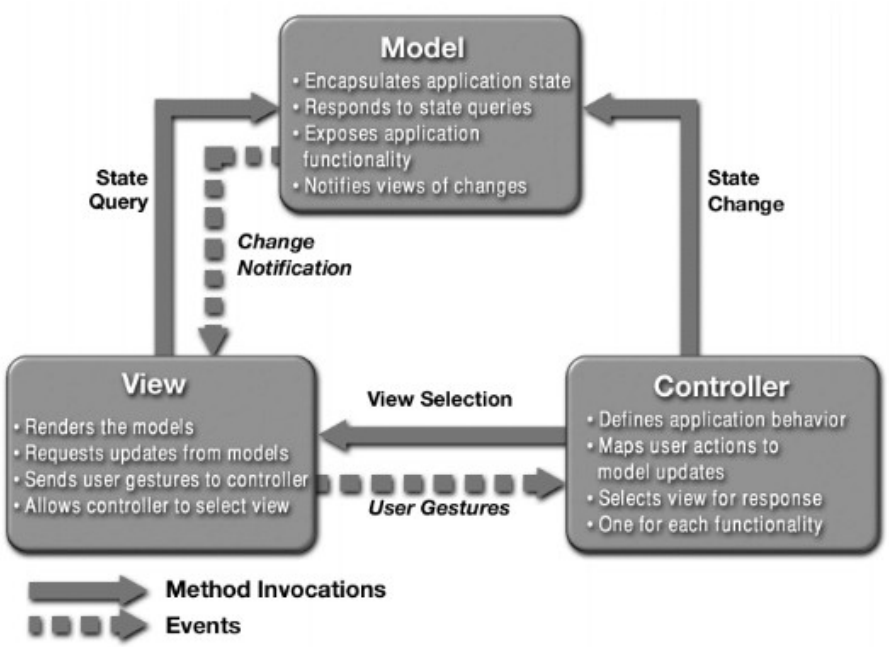


Figure 10: Diagram of the Model-View-Controller Pattern showing the different responsibilities of the three parts of an application; taken from [32]

2.4 Struts

Struts is an open source framework for developing web applications using the Java Enterprise Edition technology. It is based on the MVC design pattern and makes use of Java technologies like servlets, JSPs and Tag Libraries, but it is not part of the Java EE standard [33].

The core parts of a Struts framework are [34]:

ActionServlet: The ActionServlet is the central point for processing all requests coming from a client. It is the main component of the Struts controller and extends the `HTTPServlet` class. It intercepts the HTTP request coming from a client, chooses the right Action class for performing the business logic, fills the ActionForm with content and chooses the right view for output.

Action: This class is used to access business classes, thus comprises the model part of the Struts framework, and provides an interface e.g. to EJBs or to a DBMS. The action class returns an ActionForward.

ActionForm: An ActionForm is a Java bean that is automatically populated with form content whenever a user submits a form. The input can be validated using the `validate()`

method of the ActionForm.

ActionMapping: ActionMappings associate an Action name with an Action, thus telling the ActionServlet which Action to call based on the path invoked by the user.

ActionForward: An ActionForward defines either the view for output or an Action class that should take over further processing.

web.xml file: This file configures the ActionServlet. The file contains a mapping for this servlet including path and extension name, init parameters and the default welcome page. Also other servlets used by the application are configured in this file.

struts-config.xml: This is the configuration file that stores ActionMappings, one or more ActionForwards for each Action and associates Actions with the ActionForms used by this Action. Here the workflow of the whole Struts framework is defined.

Struts also provides its own **custom tag library,** which consists of four parts [35]:

- Bean Tags: with Bean Tags beans can be defined and rendered to an output response
- HTML Tags: included are tags used to create HTML-based user interfaces. Those tags are for example used to create forms, tables, submit buttons, selection lists, checkboxes and links.
- Logic Tags: included are tags for condition-based flow control and iteration over collections
- Nested Tags: those tags extend the other Struts tags so they can relate to each other in a nested manner.

Figure 11 shows the process flow in the Struts architecture: When a user requests a resource from the server, the ActionServlet intercepts this request and looks up the mapping for the requested path. If an ActionForm exists for this mapping, it is created if not existing and its properties are automatically filled with form content. Using the `validate()` method, the input can be checked for certain restraints (if validation is set to true in the ActionMapping). If the input is not valid, an error message object is created and control is returned to the input form. Otherwise the controller (ActionServlet) chooses the right Action class based on the struts-config.xml file and the `perform()` or `execute()` method of the Action is called. The Action class can then use a Java bean or an EJB to process the business logic or can

directly contact a DBMS to gather data. After processing the business logic, the Action class returns an ActionForward, which either determines the appropriate view for output or can transfer the control to another Action class. The view, e.g. a JSP, can be chosen based on the fact if the processing of the business logic generated an error or not. The output is finally returned to the client via an HTTP Response object [33],[34].

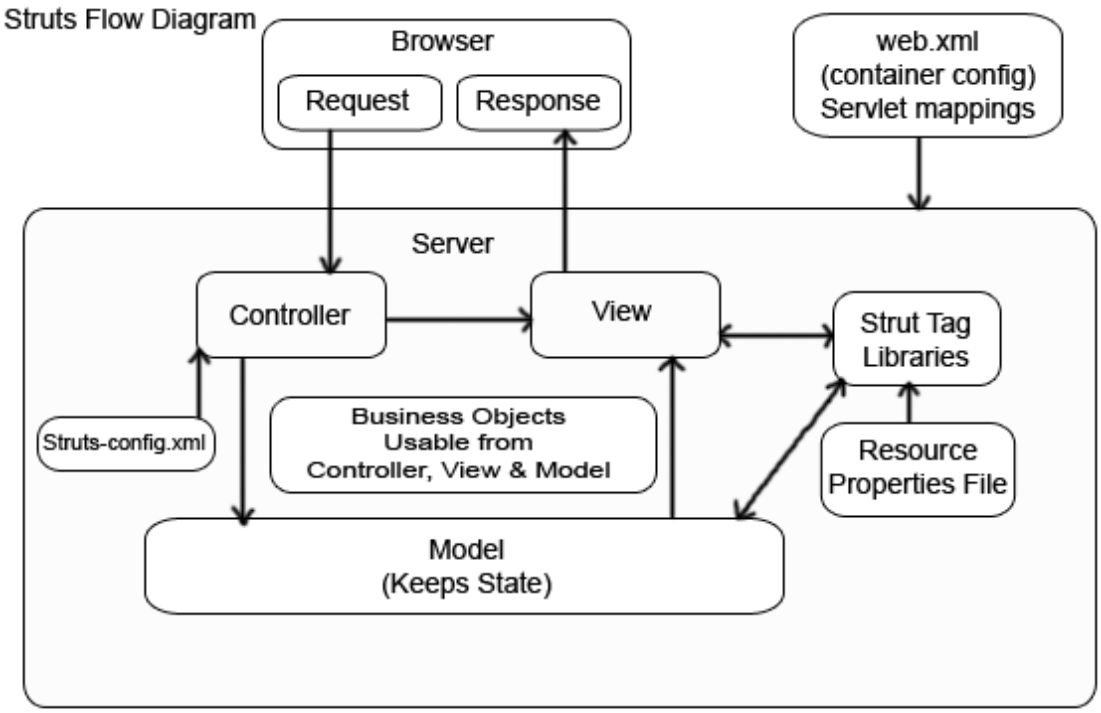


Figure 11: Struts process flow; The controller receives a request from a user, reads the struts-config.xml file and chooses an approprate action. Business logic is processed in the action class and using an Action Forward, the right view for output is chosen. The view component uses the Struts Tag Library to output its content that is finally sent as response to the user.;taken from [33]

2.5 Web Services

The Internet has grown quite enourmously over the last decades. Unfortunately the computers connected to the Internet are comprised of various hardware systems and there also exist a lot of different software products for the provided services. Achieving interaction of computers under those circumstances has turned out to be a great challenge.

Web Services were developed with the goal to create a platform and software independent format for the interchange of data of various content that every hardware and software system should be able to understand. To achieve this, a widespread format has to be used together

with an established communication protocoll.

There exist different standards for Web Services of which XML-RPC was the first one. Forms in XML format were used to formulate remote procedure calls (RPCs). The XML file defined method names and parameters and HTTP was used as communication protocoll. Unfortunately XML-RPC had a few drawbacks like strong limitations for datatypes, complex coding for binary data and lack of definition of meta data. A new standard, the Simple Object Access Protocol (SOAP), tries to attend to those drawbacks and has now taken the place of the most commonly used Web Service standard [14].

SOAP provides a standard XML framework for packaging application data that is exchanged between different hardware and software systems [19]. The root element of the SOAP message is the `<soap:Envelope>` which contains an optional `<soap:Header>` element and a mandatory `<soap:Body>` element. The envelope element defines the XML format as SOAP message, the header is used for meta information like security handling and routing. The body contains the data to be sent. Inside the body element, the `<soap:Fault>` element contains error messages if something went wrong [36]. Even though SOAP does not depend on a special protocol or network type for transport the most common used protocoll is HTTP because of its widespread support and the possibility to bypass firewalls [19].

When wanting to use a Web Service, one has to know where this service is located and what operations can be performed by this special Web Service. This is what the Web Service Description Language (WSDL) is for. It is an XML file used for the description of a Web Service. A WSDL file is made of five main parts which are all surrounded by the `<definitions>` elment. The `<message>` elements describe parameters and return values, the `<portType>` element describes the abstract interface of a web service, the `<binding>` element describes which protocols and encoding styles are used, the `<types>` element defines data types used by the web service and the `<service>` element provides the Internet address where the service can be found [37].

2.6 Model Driven Architecture

In contrast to other engineering disciplines, software development is a relatively new technology. Although the situation has improved in the last years, software systems are still error-prone, sometimes poorly documented and cost more than anticipated. The result of many studies came to the conclusion that the cause of these problems is the fact that there

don't exist uniform processes for creating software and that the human being is the main source for errors [38]. Thus the conclusion was that there is a need for automated methods for analysing and designing software and for the development of tools to automatically transform those designs to program code. This is the concept of Model Driven Architecture (MDA). The MDA was stardardisized by the Object Management Group (OMG) in 2003 [39]. The intention was to eventually generate 100% of the code with code generators. MDA specifies different models and rules to transform one model into the other. It starts with a platform independent model (PIM), which describes the structure and functionality of a system without containing any platform specific context. Such a model can eventually be transformed into every technology possible, e.g. J2EE, CORBA, .NET or Web Services. The next step in the specification is the platform specific model (PSM), which contains additional information about the conversion of the model into code of one specific technology. The third and last step is the generated code. Normally the models are designed with the Unified Meta Language (UML) and there already exist lots of tools to generate code from these models, sometimes omitting the PSM layer by creating code directly from the PIM [39],[40].

2.6.1 UML

UML is a graphical notation for modelling technology-independent specifications. It was also developed by the OMG and is now existent in version 2.0. UML is often used for modelling object oriented software systems, but in fact can be used for other, non technical systems, too. The specification contains a graphical notation, 13 diagrams divided into three groups of models [41]. One group are structural models, e.g. the class diagramm, which are static models, describing components and their relation with each other. The other two groups are behavioural and interaction models, describing how different parts can communicate to other parts at specific times or for example showing the systems behaviour from the perpective of the user (use case diagram). Also part of the specification are CORBA IDL interfaces and a XML Metadata Format (XMI) for the interchange of UML models, thus having the possibility of using tools from different providers for modeling [42].

Diagram Type	Diagram Name
Structure Diagrams	Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Package Diagram, Deployment Diagram
Behavior Diagrams	Use Case Diagram, Activity Diagram, State Machine Diagram
Interaction Diagrams	Sequence Diagram, Communication Diagram, Timing Diagram, Interaction Overview Diagram

Table 2: The 13 UML diagrams

2.6.2 AndroMDA

AndroMDA is an open source, template-based code generator. It consists of a core, using Velocity [71] as template engine, and a set of cartridges, pluggable units, which contain the rules for converting the model into code of a specific technology [43],[44]. The input for AndroMDA is an UML model in XMI format. It does not provide means for modeling the system, this has to be done with third party UML modeling tools like MagicDraw [45] or ArgoUML [46].

AndroMDA relies on the Maven Plugin, a Java-based Build Management Tool composed of a plugin architecture [47], to generate the code. AndroMDA is loaded as a plugin into Maven and then executed. It also contains a validation for input models before generating the code. AndroMDA currently provides cartridges for Spring, EJB, .NET, Hibernate, Struts and others [43]. The generated code consists of two parts. One is completely generated by AndroMDA and must not be altered and the other part are code frames which have to be filled with business logic by the programmer (see Figure 12) [44].

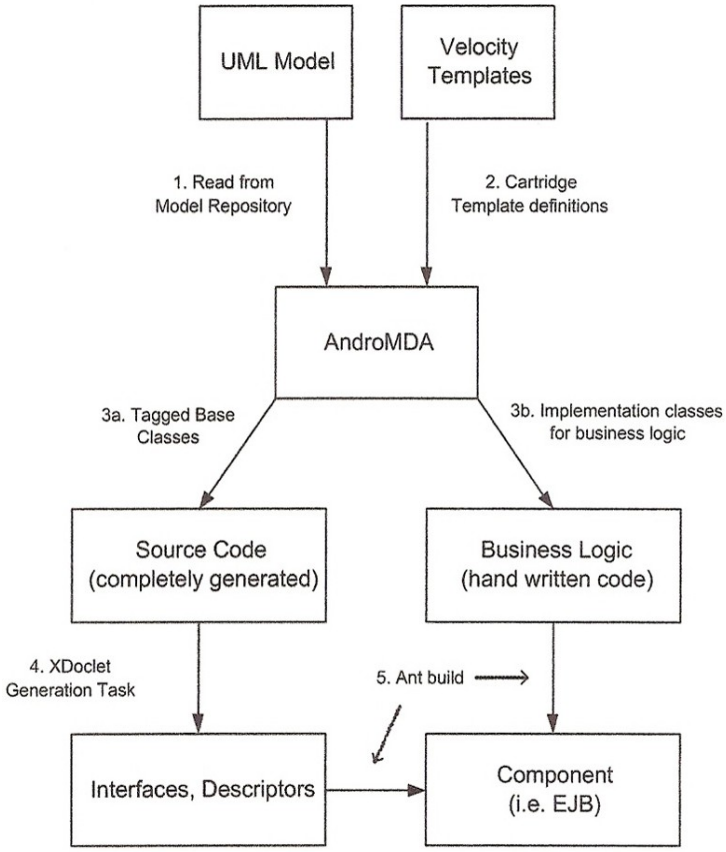


Figure 12: AndroMDA code generation; taken from [44]

2.6.2.1 EJB3 Cartridge

Models designed with AndroMDA have to use specific UML stereotypes and tagged values for different cartridges. Stereotypes are used for the classification of model elements, for example a class with the stereotype <<Service>> becomes a session bean with the EJB3 cartridge, <<Entitiy>> a Java Persistence Entity Bean and <<Identifier>> can be used to denote the primary key of an entity class. Each stereotype has also its own specific set of tagged values. For example the the tagged value @andromda.persistence.table assigns a name to the table in the database used (see Figure 13). Tagged values can also be used to denote an operation as a database query. The datatypes used are generic UML datatypes, the assignment of real datatypes is done in special mapping files.

AndroMDA uses the concept of Design Patterns during code creation. A Service Locator for JNDI lookup and Business Delegates with the ability to invoke the avalaible service methods are created automatically, as are Data Access Objects for communication with the underlying database. In addition, Data Transfer Objects can be defined using the <<ValueObject>> stereotype [48].

Figure 13: A sample Entity bean annotated with stereotypes and tagged values.

2.7 User Management

2.7.1 Authentication and Authorization

Authentication is a mechanism to ascertain the identity of an user trying to gain access to an application. This can be done with simple methods like a username/password combination or more sophisticated methods like fingerprint or retina scans. All these methods however rely on some information that only one person can provide. If the user is able to give the correct information, the system consideres him as authentificated.

Closely linked to authentification is the mechanism of user authorization. Authorization takes place after a user was sucessfully authenticated, it is responsible for determining which access rights the authenticated user has to different parts of the application. This information is usually stored in Access Control Lists (ACLs) [49].

2.7.2 User Management System

At the Institut of Genomics and Bioinformatics (IGB), a central user management system was developed that handles authentification and authorization for multiple applications. This authentification and authorization system (AAS) is based on the open source project OpenSymphony [50] . It provides a web-based interface where users, groups, resources and access control lists can be created, and users or groups can be assigned to those ACLs[6]. Applications can use this system via the client connector interface (CCI). It enables connection via HTTP, HTTPS or RMI and consists of a Java API and a custom tag library, whose most important tags are the `<login:checkLogin/>` tag to determine if the user is logged in and to deny him access to the page if not, and the `<permission:hasPermission/>` tag, which allowes or denies users to access part of a page based on ACLs [6],[51]. The AAS makes use of Single-Sign-On (SSO), a mechanism whereby users only need to log in once

and consequently gain access to all applications for which they have permission to. This mechanism is implemented for all applications in the same subdomain. Login is achieved by entering a combination of username and password. After a successful login attempt, a unique identification ID is assigned to the user, which is stored at the client and sent back at every attempt of gaining access to a restriced resource (see Figure 14) [6].

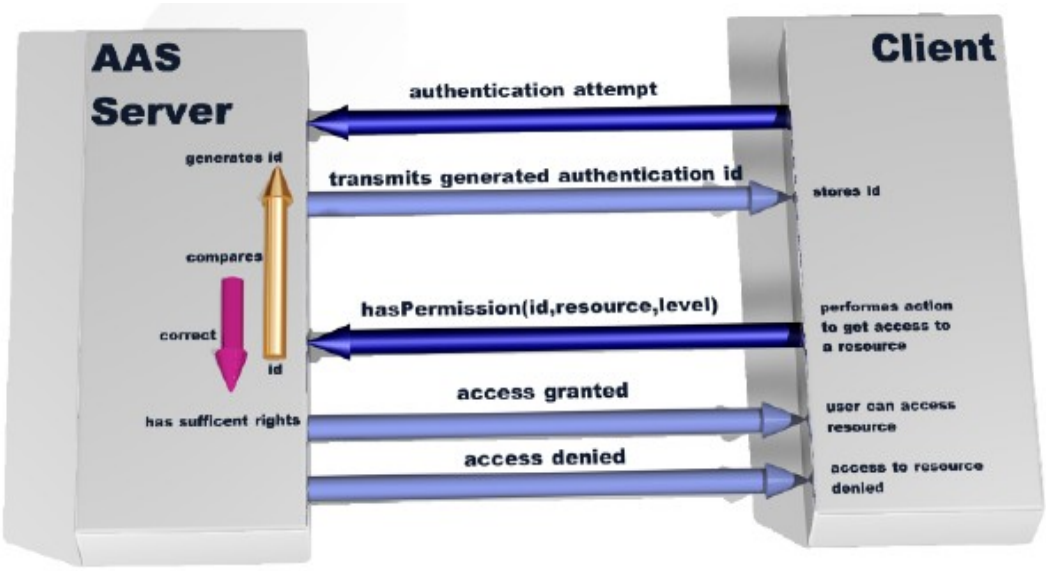


Figure 14: Authentification process of the AAS System. After a successful authetication attempt, the AAS Server sends an authentication ID to the client. This ID is stored there and sent back with every attempt to gain access to a specific resource; taken from [6]

2.8 KEGG

The Kyoto Encyclopedia of Genes and Genomes (KEGG) is a set of public databases that was initiated in 1995 by the Japanese human genome programme. It contains system, genomic and chemical information and is comprised of four main databases, PATHWAY, GENES, LIGAND and BRITE (see Figure 15). KEGG PATHWAY consists of manually drawn pathways representing the current knowledge about metabolism, genetic information processing, environmental information processing, cellular processes and human diseases. KEGG GENES relies mostly on data from NCBI RefSeq, but also from other public resources, and is an accumulation of gene catalogs for all complete and some partial genomes. KEGG LIGAND holds the current knowledge of chemical substances and reactions and KEGG BRITE holds functional hierarchies representing various aspects of biological systems [52].

KEGG also provides computational tools for reconstructing biochemical pathways and for predicting gene regulatory networks from gene expression profiles [53].

Besides providing a web-based database access, KEGG offers a Web Service using SOAP and WSDL, allowing different web applications to easily integrate the KEGG databases into their system. Details about the KEGG API can be found at [52].

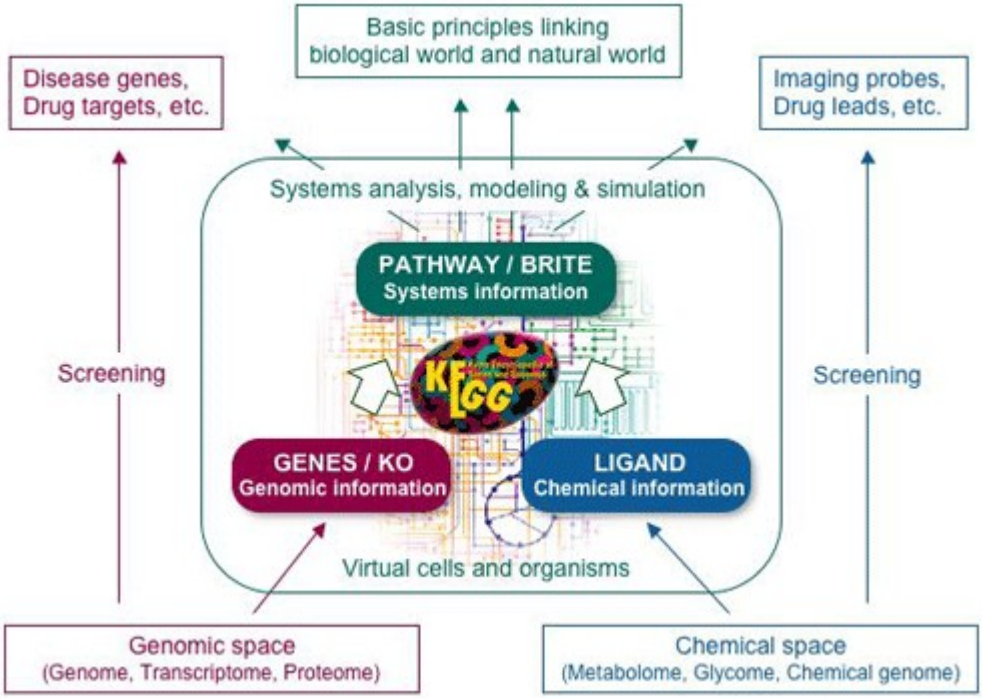


Figure 15: This picture shows the four main databases of KEGG. Also shown is the source of information and what can be discovered using the database system; taken from [52]

2.9 MARS

The Microarray Analysis and Retrieval System (MARS) is a web-based MIAME [54] compliant database for the storage and retrieval af data generated during microarray production, sample preparation, hybridization and analysis, developed using Java 2 EE technology. It includes a laboratory information management system (LIMS) for keeping track of information created during microarray production and biomaterial manipulation and a quality control management to ensure high quality of production and analysis results. MARS allows the annotation of samples and experiments. It also contains a generic file parser which allows the upload of plates, raw and transformed datasets in a generic file format. Data can be exported in MAGE-ML format, a XML–based file format developed for the exchange of

microarray based experiment information.

MARS exposes its functions through a SOAP-based Web service to which other applications can connect, or through Remote Method Invocation (RMI) if both applications are in the same subdomain and behind a firewall [6],[55].

3 Results

3.1 Overview

GiSAO.db (Genes in Senescence, Apoptosis and Oxidative Stress), a database for storage of normalized data gained through microarray analysis, was developed. The database includes the possiblity to store ortholog data, favorite gene lists and tags. For implementation mostly open source technology was used. After careful consideration it was decided to use the new EJB3 technology for implementing the business logic of the application. AndroMDA was used for automatic creation of source code with the help of MagicDraw for creating models in XMI format that provided the input for AndroMDA. The web part of the application was implemented using the Apache Struts framework, because it is an established framework with lots of documentation available. The Web service API provided by KEGG was used to connect to the KEGG database. For deployment it was decided to use the open source application server JBoss. The database used during development was MySQL, but now, in the production environment, an Oracle database is in use.

The usermanagement implemented a the IGB was used for authentication and authorization. By utilizing the single sign-on mechanism users also have access to the MARS database without having to login for a second time. The search for orthologs and pathways is open to guests as is part of the search for complete information of one gene. This free available data includes experiments declared as “for public” together with their gene expression data, information about orthologs, but not information about favorite gene lists or tags.

The upload of all files is done using Message Driven Beans, so that users don't have to wait until the whole file has been loaded into the database and are able to continue working with the application.

3.2 Database Model

Gene expression data is stored in both an experiment centered and a gene centered way. A gene in the database can be identified using the following IDs: Refseq [56], Unigene [57], MGI [58], AGI [59], Flybase [60], SGD [61], UniProt [62], EntrezGene, Gene Name and Gene Symbol (all [63]) . Gene Ontology (GO) [64] data is stored extra for experiment based annotation data. An experiment stored in the database is an accumulation of several microarray experiments based on a common goal. It contains multiple classes, where each class can either represent a time series or is used for the comparison of a healthy and a sick

tissue sample. Each class represents the result of one microarray experiment, so classes are linked to gene expression data, where each gene expression value represents the gene expression of one spot of a microarray. Each gene expression value is linked to its spot-based annotation data. Database experiments are also linked to the organism that was probed.

Gene centered annotation data is stored independently of experiments to allow the inclusion of genes which are not contained in microarray chips. Each entry is also linked to its specific organism.

Tags exist for experiments, classes and genes. Each tag is linked to a tag category. Gene tags are furthermore linked to the database table, containing the before mentioned IDs to indicate from which source the ID for the gene was derived.

Favorite lists are linked to favorite list genes, whos IDs are also linked to a specific database.

Ortholog data is stored separately in a table.

Information about which user has added or edited the respective data entries is stored in experiments, tags, favorite lists and list genes. Tags and favorite list genes also store their time of creation/editing (see Figure 16).

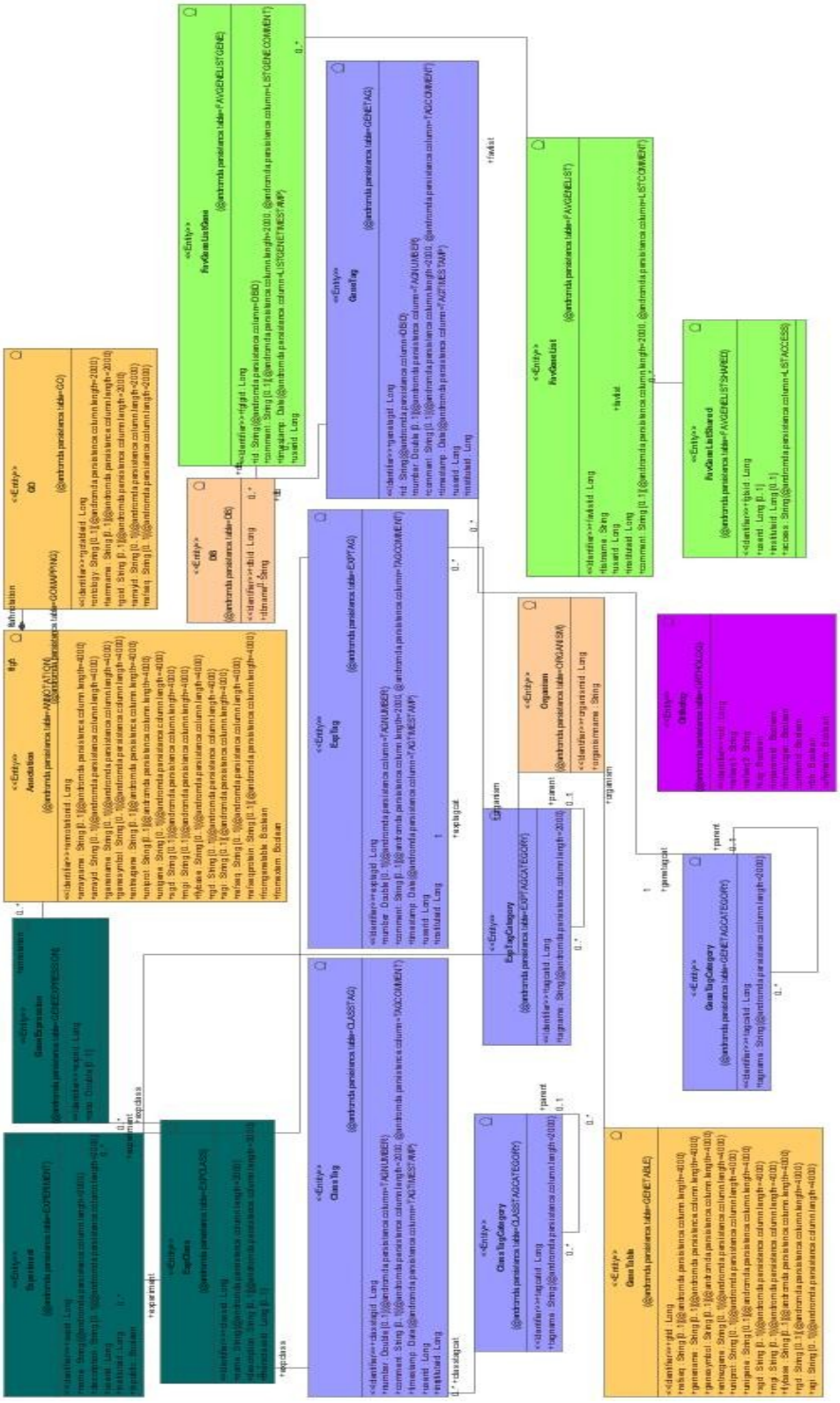


Figure 16: Database schema of the GiSAO database (dark green: experiment data, orange: annotation data, blue: tag data, light green: favorite gene list data, pink: databases and organisms)

3.3 Web Interface

Experiments Gene expression values can be displayed in several ways. One possiblity is to display the whole dataset. For this, the user first has to chose an experiment and the classes whos gene expression data he wants to see from a list (see Figure 17). In addition the database IDs that should be included in the display can be chosen. Furthermore gene expression values can be viewed as log2 values or delogarithmised. In the result set gene expression data is visualized using a color scheme, tags can be added to this genes and they can be added to a favorite gene list. A search for pathways can also be performed (see Figure 18).

The second possibility is map data from different experiments based on a specific ID. For this mapping additional criteria can be applied. Classes of different experiments can be chosen independently for viewing and as criterion. It can be specified how many percent of the chosen classes must fulfill the criterion and also the level of gene expression values included in the criterion can be defined.

The third alternative for viewing gene expression values is achieved through search for complete gene information, which will be discussed in one of the following sections.

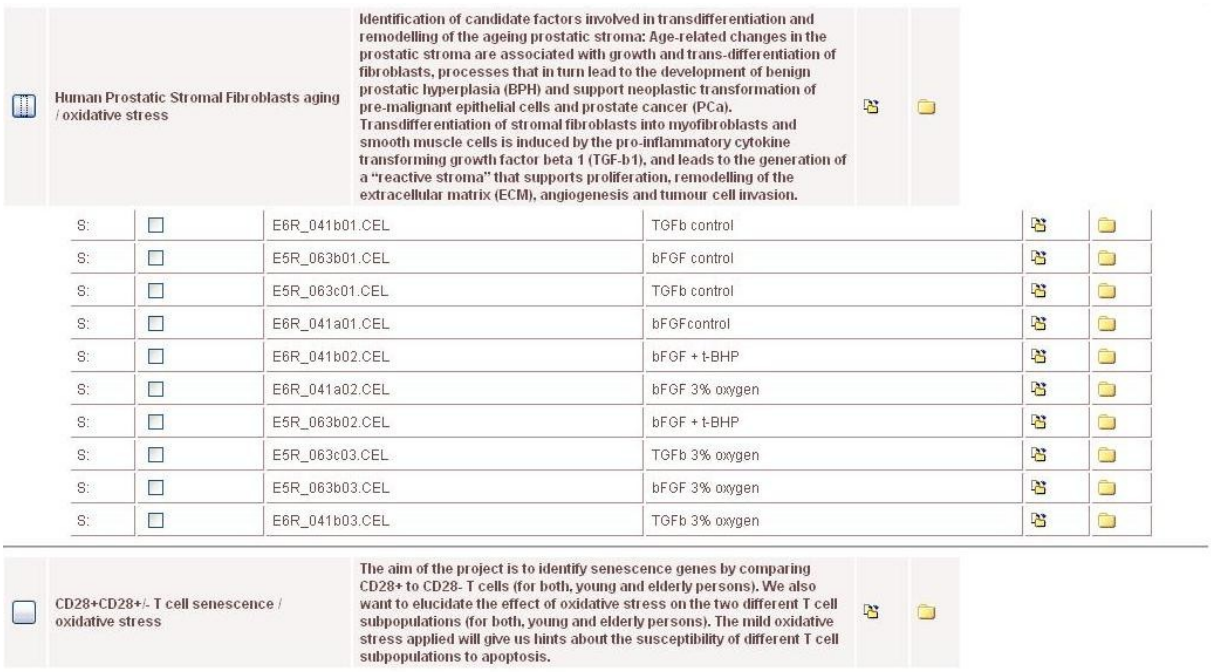


Figure 17: List of experiments and classes whos gene expression data can be displayed. Also Tags can be added to a specific experiment/class or its already existing tags can be viewed.

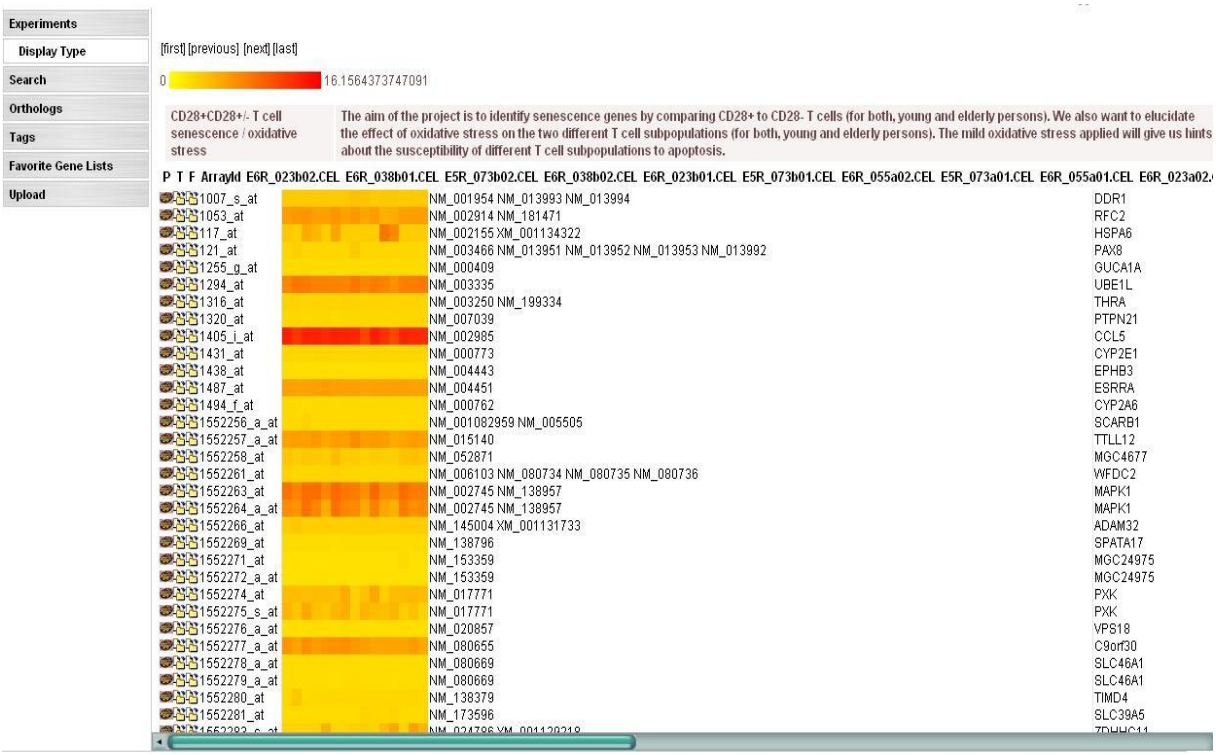


Figure 18: Display of gene expression data of an experiment. Gene expression values are visualized using a color scheme. To each id a tag can be added to the gene, or the gene can be added to a favorite gene list. Also a search for pathways can be carried out.

Tags Tags can be added to an experiment, a class or a specific gene. Each tag is assigned to a tag category and a comment can be added to provide further information. Links to third party information (e.g. another database) can be inserted into a comment. There exist different tag categories for experiments, classes and genes, all of which are hierarchically ordered. All tags present in the database are available for and can be added by logged in users. Users can also edit and delete tags which have been added by a member of their own institute. When a user adds a tag for an experiment or a class, he has to choose the category and can add a comment. When adding a tag to a gene represented by a specific ID representing this gene, also the database to which this ID belongs (e.g. Refseq or Flybase) has to be chosen. All tags for experiments, classes and genes can be searched independently. Experiment/class tags can be viewed ordered by experiment/class and tag category. It is also possible to view tags for a specific experiment or class separately. Gene tags can be displayed for a specific database ID. When a gene has not been tagged with the chosen ID, a lookup for the corresponding search

ID is performed in the table holding gene centered annotation data and the results are then displayed ordered by found IDs and tag category.

Favorite Gene Lists Favorite gene lists are a collection of genes of special interest of a certain group. Each user can add one or more favorite gene lists, which are then available for viewing, editing and deleting for all members of his institute (see Figure 19). Lists can also be shared and are then available for viewing to all logged in users, but editing and deleting of lists is not possible for users of other institutes. Genes can be either added individually to a gene list or multiple genes can be added through a file upload. As is the case with gene tags, when adding a gene to a favorite gene list, the database of the ID of the gene has to be specified. Comments can be added to both favorite lists and each individual gene, again with the possiblity to include hyperlinks. It is possible to view genes of a specific lists (see Figure 20) or to view genes of a specific ID. Again, if one gene is not of the same database ID as the search ID, a search in the database is made for the corresponding ID(s). Also a comparison of genes contained in lists of different institutes can be executed. The result shows a list of all found genes of the previously specified database ID and if they were found in lists of the selected institutes.

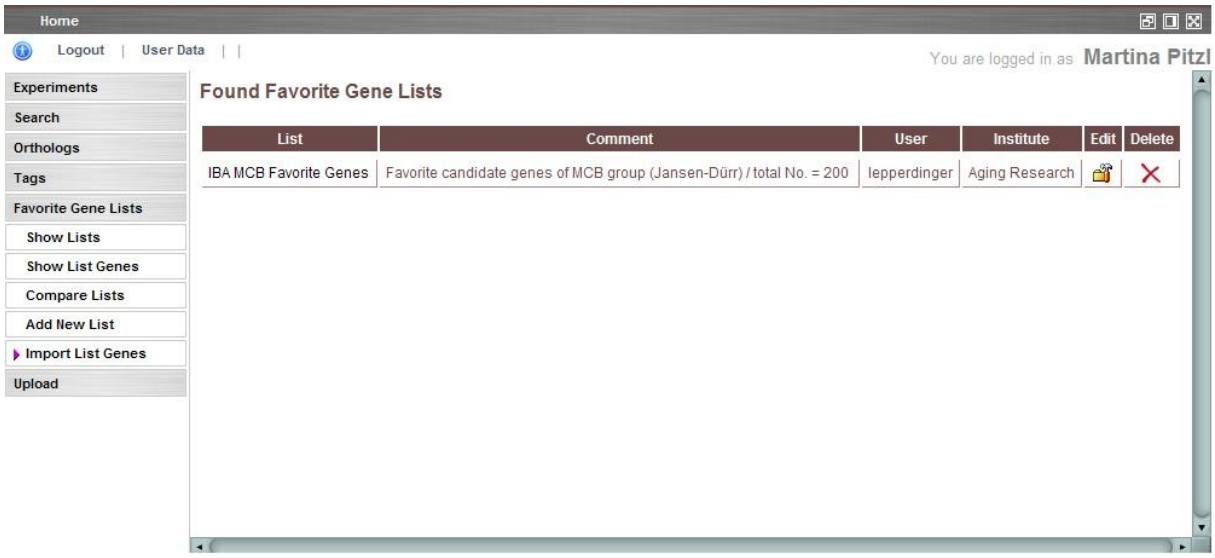


Figure 19: A favorite list, showing name, comment, the user who created the list and the institute to whicht the user belongs. Lists can be added and deleted.

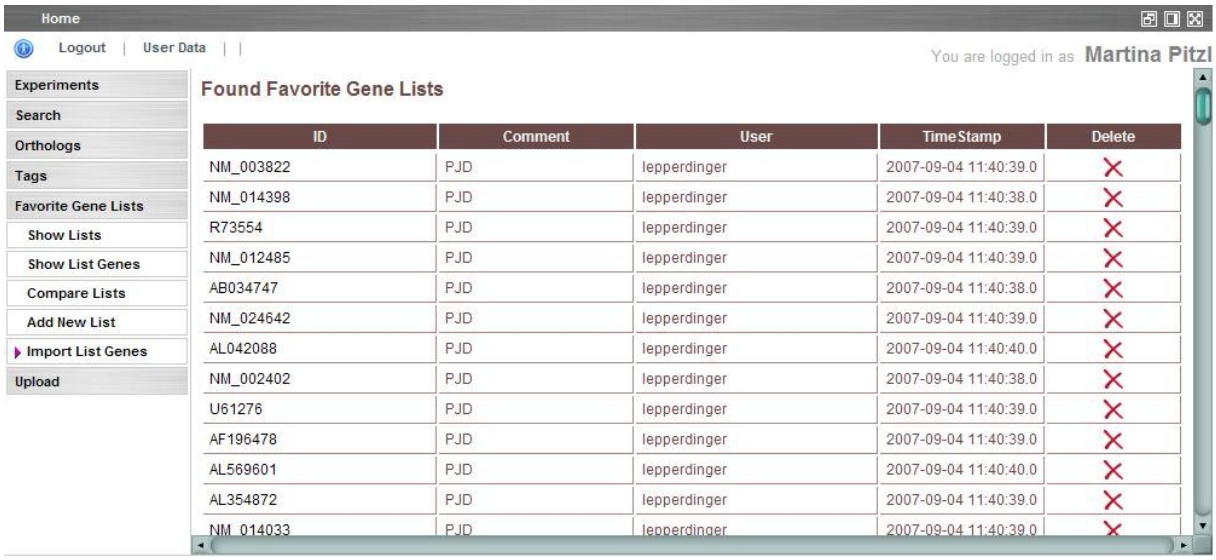


Figure 20: Genes of a favorite gene list. Shown is the id of the gene, a comment, the user who added the gene and the time of creation.

Orthologs The ortholog data saved in the database is based on Refseq IDs. Ortholog Refseq IDs are saved together with information where and how those orthologs were found. This information includes KOG [65], Inparanoid [66], RBH (Reciprocal Best Blast Hits), Affymetrix [67], HomoloGene [68] and OrthoMCL [69].

The database can be queried for a specific ID to find its orthologs. If an ID different than Refseq is chosen, a search for the corresponding Refseq IDs is executed in the in the table holding gene centered annotation data. The result lists the orthologs found for this ID together with the names of the organisms to which these IDs belong.

In addition, a search for all orthologs of two organisms can be carried out.

File Upload The upload of files is restricted to administrators with the exception of the upload of favorite genes to a favorite gene list. Uploadable data includes array annotation data, gene expression experiments, GO data, ortholog data and general annotation data which is experiment independent. It is possible to upload microarray annotation data from Affymetrix annotation files, but also annotation data for other arrays, if this data is in a certain predefined format. Experiment data can be uploaded from files or from the MARS database. Array annotation data and GO data is also updateable. When uploading experiment data, it can be specified if the experiment should be publically available or should only be accessible by logged in users.

Pathways The KEGG database can be queried directly by entering either a gene to find the pathways in which this gene is included or by entering a pathway and find all genes contained in this pathway. In addition, pathways for all genes included in the result of displaying a dataset, mapping experiment data or searching the complete information of a gene, can be searched.

Complete Gene Information In addition to the before mentioned search possibilities, a search for the complete data in the database related to a special ID can be performed. This ID can be a database ID, a GO term or an array ID (e.g. an Affymetrix ProbeSet ID). The result consists of two parts. One part contains experiment annotation together with experiments, classes and the respective gene expression values for the search ID. Tags for experiments and classes and pathways can be viewed through links (see Figure 21). The second part consists of gene centered annotation data together with ortholog data, gene tags and favorite genes. For this second part, the gene centered annotation table is queried for this ID, all related Ids for are gathered and gene tags and favorite list genes are searched for all those IDs. The found Refseq IDs are used for gathering ortholog data. All found results are ordered by organism.

Homo sapiens (human)

Experiment Annotation Data

RefSeq	GeneName	Symbol	EntrezGene	UniProt	UniGene	SGD	MGI	FlyBase	RGD	AGI	RefSeq Protein	Array Name	Array Id	GO
NM_152605	zinc finger protein 781	ZNF781	163115	Q2VPJ8 Q8N1U5 Q8N8C0	Hs.631565	---	---	---	---	---	NP_689818.2	HG-U133 Plus 2.0	1559789_a_at	Biological

Organism	Experiment Name	Description	View Tags
Homo sapiens (human)	Human Prostatic Stromal Fibroblasts aging / oxidative stress	Identification of candidate factors involved in transdifferentiation and remodelling of the ageing prostatic stroma: Age-related changes in the prostatic stroma are associated with growth and trans-differentiation of fibroblasts, processes that in turn lead to the development of benign prostatic hyperplasia (BPH) and support neoplastic transformation of pre-malignant epithelial cells and prostate cancer (PCa). Transdifferentiation of stromal fibroblasts into myofibroblasts and smooth muscle cells is induced by the pro-inflammatory cytokine transforming growth factor beta 1 (TGF-b1), and leads to the generation of a "reactive stroma" that supports proliferation, remodelling of the extracellular matrix (ECM), angiogenesis and tumour cell invasion.	

Class Name	Description	GeneExpression Value	View Tags
E6R_041b01.CEL	TGFb control	2.45490454645676	
E5R_063b01.CEL	bFGF control	2.39761987757904	
E5R_063c01.CEL	TGFb control	2.38873735309574	
E6R_041a01.CEL	bFGFcontrol	2.41626741169955	
E6R_041b02.CEL	bFGF + t-BHP	2.45523439344378	
E6R_041a02.CEL	bFGF 3% oxygen	2.41729656973594	
E5R_063b02.CEL	bFGF + t-BHP	2.40094556687599	
E5R_063c03.CEL	TGFb 3% oxygen	2.39749873302633	
E5R_063b03.CEL	bFGF 3% oxygen	2.39853149105966	
E6R_041b03.CEL	TGFb 3% oxygen	2.42844004186469	

Figure 21: Complete Gene Info: Experiment Annotation with Experiments, Classes and Gene Expression Values. Tags are availaible via links.

4 Discussion

The main goal of this thesis was the implementation of a centralized, web-based database to store gene expression and ortholog data and to provide means to gain information about this data in order to investigate mechanisms that lead to cellular aging and apoptosis.
The web application was developed using Java EE 5 technology, AndroMDA and the Struts framework. The Java Enterprise Edition is a widely used platform for implementing multi-tiered web-based applications. Java applications can run on any server because of its operating system independence. The new Java EE 5 platform, which was specified in May 2006, introduced some improvements to its specifications, most noticeable the completely remodeled Enterprise JavaBeans 3.0 specification together with the outsourcing of the persistence layer to its own specification. EJB3 makes working with Enterprise JavaBeans more intuitive because developers can now focus on implementing business logic and don't have to deal with the implementation of methods that were needed by the interface but not the application.

AndroMDA is an open source MDA framework that uses UML models in XMI format to create part of the source sode. The EJB3 cartridge of AndroMDA was used to implement the business logic part of this application. This tool facilitates the development process for programmers by automatically creating entity beans, session beans, message driven beans, data transfer objects, data access objects for the interaction with an underlying database as well as a service locator that implements lookup functionality and service delegates for interaction with the remote methods of session beans. What is left for the programmer to implement is the actual workflow of the application.
All file uploads have been implemented using Message Driven Beans, which are based on the asynchronous Java Message System. After the file is uploaded to the server and the message has been sent, users can continue their work and don't have to wait until the file data has been loaded into the database.
The Struts framework is also one of the most widespread Java frameworks worldwide. It is an established framework in contrast for example to the rather new Java Server Faces (JSF) [70] framework, and is well documented with a large developer community guaranteeing consistent refinements.
Furthermore AndroMDA as well as the Struts framework incorporate Design Patterns. Design

Patterns are proven solutions to common problems that occur during programming software. Struts is a framework based on the MVC design pattern, AndroMDA allows the use of Data Transfer Objects and automatically creates Data Access Objects, a Service Locator and Business Delegates. By using those proven solutions, the software development process as well as the maintenance of the software is facilitated.

For usermanagement the Authorization and Authentication System developed at the Institute of Genomics and Bioinformatics was used. With the use of the Single Sing-On mechanism access to all other applications (if the user has access rights to them) in the same domain are possible. So, after loggin in, users not only have access to GiSAO, but also to the MARS database for uploading gene expresssion data. The AAS allows to deploy a fine grained user access scheme by allowing or denying access to pages or parts thereof based on user groups and access control lists. It is not only possible to deny a user access to a page if he is not logged in, but it is also possible to create different views based on access rights to resources.

Since the application is not dependent on a specific database management system (DBMS), every relational database can be chosen as Enterprise Information System. The application was developed using MySQL, the most widespread open source DBMS in use. Now, in the production environment, Oracle, a performent DBMS for enterprise applications, is in use.

GiSAO.db, an application that allows members of the National Research Network to gain access to previously unavailable data and to share information with other users, was designed. Gene expression data stored in log2 format can be displayed using a colour scheme, allowing fast detection of very high expressed genes. The possibility to map data from different experiments allows scientists to compare specific genes across multiple experiments based on several criteria.

Annotation data is stored in the database in both an experiment centered and gene centered way. This allows the integration of annotation data specifically composed for microarrays and on the other hand also allows the inclusion of genes which are not present on any microarray chip.

Information about experiments and classes can be added using descriptions, data in text format. They can also be further categorized by the assignment of special tags. Genes also can be tagged and through the use of annotation data, IDs from different databases can be retrieved for one gene.

Favorite gene lists allow users to compile lists of special meaning to a certain area of research as well as the facility to compare lists from different institutes, allowing to find common interests.
Links to third party information can be added to all comments from tags, favorite lists and list genes. This together with the integration of pathways from KEGG as well as the fact that all IDs shown in the result pages are links to their respective databases allows a tight coupling to outside information sources.
Furthermore the possibility to search all information in the database provides a simple and easy access to all relevant information of one special gene.

This database was developed specifically to store data for research in cellular aging, but basically all data gained from a microarray experiment, independent of its research goal, can be stored. With a few alterations, also data in log2ratio format would be able to be displayed together with the search possibility for over- and underexpressed genes. So in the future it will be possible to use this database design also for other research areas.

5 References

[1] Institute for Biomedical Aging Research - National Research Network (NRN) 093. WWW. http://www.iba.oeaw.ac.at/index.php?id=214

[2] FWF Austrian Science Fund. WWW. http://www.fwf.ac.at/en/index.asp

[3] National Research Network – Aging Research. WWW. http://www.iba.oeaw.ac.at/fsp/index.php

[4] Lodish H. et al. *Molecular Cell Biology*. W.H. Freeman and Company 2004

[5] NCBI Science Primer – Microarrays: Chipping away at the mysteries of science and medecine. WWW. http://www.ncbi.nlm.nih.gov/About/primer/microarrays.html

[6] Maurer M. *Design and Development of a Bioinformatics Platform for large-scale Gene Expression Profiling*. Doctoral thesis. TU-Graz, 2004. http://genome.tugraz.at/

[7] Scherbakov N.: Databases 1 – Lecture Scripts. WWW. http://coronet.iicm.tugraz.at/Dbase1/scripts/library.htm

[8] Rüttger M. *MySQL 5 Professionell*. mitp - Redline GmbH 2006

[9] MySQL. WWW. http://www.mysql.de/

[10] Oracle. WWW. http://www.oracle.com/global/de/index.html

[11] DB2. WWW. http://www-306.ibm.com/software/data/db2/

[12] J2EE at a glance. WWW. http://java.sun.com/javaee/

[13] The Java EE 5 Tutorial. WWW. http://java.sun.com/javaee/5/docs/tutorial/doc/

[14] Stark T. *Java EE 5*. Addison-Wesley 2007

[15] JBoss. WWW, 2006. http://labs.jboss.com/portal/

[16] BEA WebLgic. WWW. http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/weblogic&WT.ac=topnav_products_weblogic

[17] IBM WebSphere. WWW. http://www-306.ibm.com/software/websphere/

[18] Info 4 Java – Java Server & J2EE. WWW. http://www.info4java.com/?this=java_server_tutorials

[19] Burke B., Monson-Haefel R. *Enterprise JavaBeans 3.0*. O'Reilly 2006

[20] Basham B., Sierra K., Bates B. *Head First Servlets & JSP*. O'Reilly 2004

[21] Ullenboom C. *Java ist auch nur eine Insel - Programmieren mit der Java Standard Edition Version 5*. Galileo Computing 2006

[22] O'Reilly OnJava.WWW. http://www.onjava.com/pub/a/onjava/2002/08/14/jstl1.html

[23] Custom Tag library http://coldjava.hypermart.net/jsp.htm

[24] JDBC API introduction http://java.sun.com/j2se/1.5.0/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html

[25] JDBC driver types http://java.sun.com/products/jdbc/driverdesc.html

[26] Speegle G.D. *JDBC Practical Guide for Java Programmers.* Academic Press 2002

[27] JSR 220: Enterprise JavaBeansTM,Version 3.0 Specification. WWW. http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html

[28] The JNDI Tutorial. WWW. http://java.sun.com/products/jndi/tutorial/

[29] JavaWorld JNDI overview. WWW. http://www.javaworld.com/javaworld/jw-01-2000/jw-01-howto.html

[30] Core J2EE Patterns. WWW. http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html

[31] JavaCamp.org Design Pattern. WWW. http://www.javacamp.org/designPattern/index.html

[32] Java SE Application Design with MVC. WWW. http://java.sun.com/developer/technicalArticles/javase/mvc/

[33] Jakarta Struts Tutorial. WWW. http://www.roseindia.net/struts/

[34] Husted T. et al. *Struts in action*. Manning Publications Co. 2003

[35] Struts Tag Library. WWW. http://struts.apache.org/1.x/struts-taglib/

[36] W3School SOAP Tutorial. WWW. http://www.w3schools.com/soap/default.asp

[37] W3School WSDL Tutorial. WWW. http://www.w3schools.com/wsdl/default.asp

[38] Zeppenfeld K., Wolters R. *Generative Software – Entwicklung mit der MDA*. Spektrum Verlag 2006

[39] Object Management Group. WWW. http://www.omg.org/

[40] Andresen A. *Komponentenbasierte Softwareentwicklung mit MDA, UML2 und XML.* Hanser Verlag 2004

[41] Introduction to OMG's Unified Modeling Language. WWW. http://www.omg.org/gettingstarted/what_is_uml.htm

[42] UML Specifcation. WWW. http://www.omg.org/docs/formal/03-03-02.pdf

[43] AndroMDA. WWW. http://galaxy.andromda.org/index.php?option=com_frontpage&Itemid=48

[44] Truskaller T. *Data Integration into a Gene Expression Database*. Master Thesis, Graz University of Technology, 2003. http://genome.tugraz.at/

[45] MagicDraw. WWW. http://www.magicdraw.com

[46] ArgoUML. WWW. http://argouml.tigris.org/

[47] Apache Maven Project . WWW. http://maven.apache.org/

[48] AndroMDA EJB3 Cartrigde. WWW. http://web.aanet.com.au/persabi/andromda/index.html

[49] IBM Redbooks – The Need for Authentication and Authorization. WWW. http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/tips0266.html?Open

[50] OpenSymphony. WWW. http://www.opensymphony.com/

[51] Zeller D. *Design and Development of a user management system for molecular biology database systems*. Master thesis. TU-Graz, 2003. http://genome.tugraz.at/

[52] KEGG. WWW. http://www.genome.ad.jp/kegg/

[53] Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., and Kanehisa, M.; *KEGG: Kyoto Encyclopedia of Genes and Genomes.* Nucleic Acids Res. 27, 29-34 (1999).

[54] Minimum Information About a Microarray Experiment. WWW. http://www.mged.org/Workgroups/MIAME/miame.html

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

[56] National Center for Biotechnoloy Information RefSeq. WWW. http://www.ncbi.nlm.nih.gov/RefSeq/

[57] UniGene: An Organized View of the Transcriptome. WWW. http://www.ncbi.nlm.nih.gov/sites/entrez?db=unigene

[58] Mouse Genome Informatics. WWW. http://www.informatics.jax.org/

[59] Arabidopsis Genome Initiative- Fact Sheet. WWW . http://pgec-genome.ars.usda.gov/STRUCTURAL_DIR/AGI.html

[60] FlyBase – A Database of Drosophila Genes and Genomes. WWW. http://flybase.bio.indiana.edu/

[61] Saccharomyces Genome Database. WWW. http://www.yeastgenome.org/

[62] UniProt, the universal protein resource. WWW. http://www.expasy.uniprot.org/

[63] NCBI EntrezGene. WWW. http://www.ncbi.nlm.nih.gov/sites/entrez?db=gene

[64] Gene Ontology. WWW. http://www.geneontology.org/

[65] The KOG Browser. WWW. http://genome.jgi-psf.org/help/kogbrowser.html

[66] InParanoid: Eukaryotic Ortholog Groups. WWW. http://inparanoid.sbc.su.se/cgi-bin/index.cgi

[67] Affymetrix. WWW. http://www.affymetrix.com/index.affx

[68] HomoloGene. WWW. http://www.ncbi.nlm.nih.gov/sites/entrez?db=homologene

[69] OrthoMCL. Ortholog Groups of Protein Sequences. WWW. http://orthomcl.cbil.upenn.edu/cgi-bin/OrthoMclWeb.cgi

[70] Java Server Faces. WWW. http://java.sun.com/javaee/javaserverfaces/

[71] The Apache Velocity Project. WWW. http://velocity.apache.org/