2012

RegulonIT - A web based tool for regulon, gene, and co-expression data

Aravindh kumar Balakrishnan

Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/etd
Part of the Bioinformatics Commons, and the Computer Sciences Commons

Recommended Citation
RegulonIT - A web based tool for regulon, gene and co-expression data

by

Aravindh kumar Balakrishnan

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Leslie Miller, Co-major Professor
Eve Wurtele, Co-major Professor
Simanta Mitra

Iowa State University
Ames, Iowa
2012

Copyright © Aravindh kumar Balakrishnan, 2012. All rights reserved.
DEDICATION

I would like to dedicate this thesis to my sister, Anitha Balakrishnan and to my parents for their unconditional support and encouragement throughout my work.
# TABLE OF CONTENTS

**LIST OF FIGURES** .................................................. vi

**ACKNOWLEDGEMENTS** ................................................ x

**ABSTRACT** ........................................................... xi

**CHAPTER 1. GENERAL INTRODUCTION** ......................... 1

1.1 Overview ......................................................... 1
1.2 Features ......................................................... 1
   1.2.1 Search ....................................................... 2
   1.2.2 Sort .......................................................... 2
   1.2.3 View Correlation matrix .................................. 2
   1.2.4 Search by multiple genes ................................. 2
   1.2.5 Data management ........................................... 3
   1.2.6 Integration with external websites ...................... 3
1.3 Implementation resources ..................................... 3
1.4 Thesis organization ............................................ 3

**CHAPTER 2. REGULONIT - A WEB BASED TOOL FOR REGULON,**

**GENE AND CO-EXPRESSION DATA** .............................. 4

2.1 Background ....................................................... 6
   2.1.1 Gene co-expression and the significance of clustering 6
   2.1.2 Existing web based tools for gene analysis ............ 7
2.2 Implementation .................................................... 8
   2.2.1 System Overview and Integration with External Databases 9
   2.2.2 Database design ........................................... 12
BIBLIOGRAPHY
## LIST OF FIGURES

| Figure 2.1 | RegulonIT - System overview. The RegulonIT framework provides features like search by regulon, search by gene, search by multiple genes and viewing correlation matrix. It also provides integration with external web services like KEGG(11)(12), MetNet(13), TAIR(19), yeast genome(20), gene cards(21), and MetaOmGraph(7). The administrator of this tool will have the privilege to add new species and to add new datasets for existing species.   | 10 |
| Figure 2.2 | RegulonIT - Database schema. Species can be *Arabidopsis thaliana*, *Saccharomyces cerevisiae* or *Homo sapiens*. The probeid column in the genes_species table acts as the primary key for the table. This column is referenced by columns in the pearsonmatrix table. | 13 |
| Figure 2.3 | Search by regulon page for *Arabidopsis thaliana*. This page prompts the user to enter a regulon, choose a dataset, and view information about the entered regulon at the selected correlation. | 15 |
| Figure 2.4 | Validation for search by regulon feature. When an invalid entry is entered, an error message is displayed. | 16 |
| Figure 2.5 | Regulon and gene Information for regulon 21. The search result shows that regulon 21 has 56 genes at a correlation of 0.7. | 16 |
| Figure 2.6 | Table containing detailed information about all the 56 genes in regulon 21 at a correlation of 0.7. | 17 |
Figure 2.7 Search by gene page for Saccharomyces cerevisiae (Yeast). This page prompts the user to select a dataset, enter a gene and view information about the entered gene at the selected correlation. In the case of *Saccharomyces cerevisiae*, the search could be performed by systematic name, gene symbol or probeid.

Figure 2.8 Table containing information about the gene with systematic name - ‘YKL204W’ at a correlation of 0.6. The table provides information like gene symbol, probeid, swissprot, GO terms, etc. It also displays the number of neighbours of the gene at the selected correlation, which in this case is 0.6.

Figure 2.9 Table containing information about the neighbours of gene - ‘YKL204W’. This table provides detailed information about all the neighbours of the ‘YKL204W’ at a correlation of 0.6.

Figure 2.10 Additional search features once the results are generated. These features can be used by selecting the rows (genes) in the table returned by the search results.

Figure 2.11 Correlation matrix for the selected genes. This correlation matrix is created by selecting all 8 neighbours of ‘YKL204W’ shown in Figure 2.9.

Figure 2.12 Table generated when probeid ‘4087at’ is selected from Figure 2.11. The table provides detailed information about the gene selected and also displays the neighbours of the gene at the selected correlation.

Figure 2.13 MetNet pathway information. The search results displays the pathway information from the MetNet database for the genes present in the table. The pathways have hyperlinks on them, which when clicked, displays detailed information about the pathways in the MetNet online website.

Figure 2.14 MetNet pathway information from MetNet online website.

Figure 2.15 Different features available with KEGG database.
Figure 2.16  Get KEGG Pathways for selected genes for *Arabidopsis thaliana*. This option prompts the user to select the genes for which KEGG pathway information has to be displayed. Once the genes are selected and the search button is clicked, the KEGG pathways are displayed in a new page as shown in the figure. ........................................ 25

Figure 2.17  Get KEGG common pathways for *Arabidopsis thaliana*. This option allows the user to view KEGG pathways that are common to a set of genes. The above figure shows pathways that are common to locusids AT5G37510, AT5G08530 and AT3G12260. ............................... 25

Figure 2.18  Hierarchical representation of selected genes in XML file for Regulon Analysis. This option allows the user to analyze accumulation levels of genes, regulon-wise, in MetaOmGraph. ............................... 26

Figure 2.19  Imported list from RegulonIT in MetaOmGraph. Once the xml file is imported into MetaOmGraph, it displays all the regulons in a list as shown in the figure above. ............................... 27

Figure 2.20  Plot in MetaOmGraph showing accumulation level of genes in regulon 1 for various samples. ........................................ 27

Figure 2.21  Hierarchical representation of selected genes in XML file for gene analysis. This creates a single list with all the selected genes. ....... 28

Figure 2.22  Viewing correlation matrix from text area for Saccharomyces cerevisiae. This feature prompts the user to enter the genes manually in the text area and view correlation matrix for all the entered genes. ........ 29

Figure 2.23  Correlation matrix table for the set of genes entered. ............... 29

Figure 2.24  Search results when Regulon 48 is used as the input query. ........ 31

Figure 2.25  Pearson correlation matrix for all 9 genes present in Regulon 48. ... 31

Figure 2.26  Kegg Pathway information for all 8 important genes in Regulon 48. ... 32

Figure 2.27  Search by multiple genes feature used for looking up information about POU2AF1, CD79A and IGHM in the Homo sapiens dataset prepared by Feng et al. (2) ........................................ 33
Figure 2.28 Gene information for POU2AF1, CD79A and IGHM. It could be seen that POU2AF1 and CD79A belong to regulon 47 and IGHM belongs to regulon 10. .......................................................... 33

Figure 2.29 Pearson Correlation Matrix generated for all three genes, POU2AF1, CD79A and IGHM shows that Regulon 47 and Regulon 10 are interconnected. .......................................................... 34

Figure 3.1 Flow pattern of the Struts MVC framework .......................................................... 38
Figure 3.2 web.xml .......................................................... 39
Figure 3.3 Example of action path configuration .......................................................... 39
Figure 3.4 Example of request view component .......................................................... 41
Figure 3.5 The retrieveURL Javascript function call from the body tag .............. 41
Figure 3.6 AJAX implementation .......................................................... 43
Figure 3.7 Functionality of the Search button .......................................................... 44
Figure 3.8 Example of a Controller class .......................................................... 45
Figure 3.9 Example of a Model class .......................................................... 47
Figure 3.10 Database configuration in context.xml file .................................................. 47
Figure 3.11 Establishing database connection using InitialContext and DataSource objects .......................................................... 48
Figure 3.12 Example of a response view component .......................................................... 49
Figure 3.13 Example of a Javascript function .......................................................... 50
Figure 3.14 Code for retrieving pathways from MetNet database .................................................. 51
Figure 3.15 Code for retrieving pathways from KEGG database .................................................. 52
Figure 3.16 Example of a view component created using struts tags .................................................. 53
Figure 3.17 Example of an ActionForm .......................................................... 54
ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to everyone who helped me throughout my Master’s degree and during the course of my thesis. Firstly, I would like to thank my advisors, Dr. Leslie Miller and Dr. Eve Wurtele for their guidance, encouragement and support throughout the period of this research work. I would also like to thank my committee member, Dr. Simanta Mitra for taking the time to review my thesis and giving suggestions for improvement. Further, I would like to thank Dr. Yaping Feng, Jonathan Hurst and Dr. Ling Li for providing biological data for my research work and for testing the RegulonIT web application. I would also like to thank Nick Ransom, Dr. Yi Wang and Manhoi Hur for helping me integrate the RegulonIT tool with external web services. I would also like to warmly thank my sister, parents and my roommates for their continuous encouragement, advice, and support during my education. Finally, I would like to express my gratefulness to the Virtual Reality Application Center for the research assistantship that provided support for me during my graduate studies at Iowa State University.
ABSTRACT

One of the greatest challenges in the field of biology today is the determination of unknown gene function. A number of web applications are currently being developed which aim at providing gene related information, such as gene expression and co-expression data for individual genes. Integrating gene expression networks with external data sources may give researchers additional information about these genes.

A regulon is a collection of highly co-expressed genes which can be identified by clustering a network of genes obtained from transcriptomic analysis. Regulon analysis offers a potential way to evaluate functions of genes in a given gene family or for developing hypothesis about the function of unknown genes. Hence, it is important that information about regulons, genes, and co-expression data be available to all biologists in a user friendly manner. Currently, information about regulons and its associated genes are not easily accessible, and in order to view regulon information, manual look up into flat files such as text files is required. In order to avoid this, we come up with an application which aim at providing regulon, gene, and co-expression data for three important species in a user-friendly manner using a web interface.

RegulonIT is a web based tool that aims at providing information about regulons and other gene related information. Currently, the tool provides regulon, gene and co-expression information for three species (Arabidopsis thaliana, Saccharomyces cerevisiae and Homo sapiens) for very large transcriptomic datasets using a web interface. RegulonIT is a user friendly platform independent web tool and is available at http://metnetdb.org:9090/regulonit.
CHAPTER 1. GENERAL INTRODUCTION

1.1 Overview

RegulonIT is a software application which currently provides regulon, gene and co-expression information for three species (Arabidopsis thaliana, Saccharomyces cerevisiae and Homo sapiens) for very large transcriptomic datasets using a web interface. The tool includes features like search, sort, filter and viewing correlation matrix. It also provides integration with external websites and web services like TAIR (The Arabidopsis Information Resource), yeast genome, KEGG (Kyoto Encyclopedia of Genes and Genomes), MetNet and MetaOmGraph for enabling interactions with external data sources. The software has been designed for easy addition of new datasets in an user friendly manner. The web server is supported by a database containing experimental data, annotations and GO (Gene Ontology) terms for all three species for different datasets. The database has been designed to make it possible to store pre-calculated co-expression results for faster processing of user queries.

1.2 Features

This section gives a brief description about the different features that are available with the RegulonIT web tool. These features are available in a form like layout where the user is prompted to enter an input. The form takes input using text boxes, drop down list boxes, check boxes or radio buttons depending on the functionality of the page. When the input is entered and the submit button is clicked, the request is sent to the server and once the business logic is processed, a response is sent back to the user in the form of a table.
1.2.1 Search

This feature allows the user to enter a search criteria, select a dataset and view information about regulons or genes at a selected correlation co-efficient.

There are two main search features in this application

- Search by Regulon
- Search by Gene

1.2.2 Sort

The search feature returns information in the form of a table and the table usually contains a large number of rows. The Sort feature allows the user to sort these rows in the table by ascending or descending order.

1.2.3 View Correlation matrix

This feature allows the user to view correlation matrix for a set of genes. A correlation matrix provides information about the correlation co-efficient between all pair of genes in a given set. If there are ‘n’ genes under consideration, then the final output will be a ‘n*n’ matrix represented in the form of a table.

A correlation matrix can be viewed in our application by two ways:

- Using the Search feature for getting gene information in the form of a table, selecting the genes in this table, and then generating the matrix.
- Entering a set of genes manually in a text area and then generating the matrix.

1.2.4 Search by multiple genes

This feature allows the user to enter more than one gene and view information about these genes simultaneously.
1.2.5 Data management

It is also possible for a privileged user to add a new species or add a new dataset for an existing species. This can be done only by administrators of this application.

1.2.6 Integration with external websites

This feature enables the user to interact with external websites and web services. Currently, the tool has interactions with TAIR, yeast genome, gene cards, KEGG, MetNet and MetaOmGraph.

1.3 Implementation resources

The application has been implemented using open source technologies like Java EE, Struts 1.3, AJAX(Asynchronous JavaScript and XML, Javascript), JDBC(Java Database Connectivity) and Apache Axis. The database used is MySQL 5.0.77 which is available in the MetNet server and the web server used is Tomcat 5.5.

1.4 Thesis organization

The thesis is organized as follows:

Chapter 2 contains the manuscript for the journal paper. This chapter has been prepared according to the format specified by BMC Bioinformatics journal. Dr. Leslie Miller provided suggestions for choosing the resources to be used for developing the web based tool. Dr. Eve Wurtele helped with the manuscript by providing biological information. Dr. Yaping Feng provided regulon, gene and co-expression data for the species.

Chapter 3 discusses the project implementation in detail. It describes the components of the Struts framework and the different resources that have been used for developing the RegulonIT project.

Chapter 4 provides conclusive remarks for the thesis.
CHAPTER 2. REGULONIT - A WEB BASED TOOL FOR REGULON, GENE AND CO-EXPRESSSION DATA

A paper to be submitted to the BMC Bioinformatics Journal.

Aravindh kumar Balakrishnan, Yaping Feng, Leslie Miller, Eve Syrkin Wurtele

Abstract

Background

One of the greatest challenges in the field of biology today is the determination of unknown gene function. A number of web applications are currently being developed which aim at providing gene related information, such as gene expression and co-expression data for individual genes. Integrating gene expression networks with external data sources may give researchers additional information about these genes.

A regulon is a collection of highly co-expressed genes which can be identified by clustering a network of genes obtained from transcriptomic analysis. Regulon analysis offers a potential way to evaluate functions of genes in a given gene family or for developing hypothesis about the function of unknown genes. Hence, it is important that information about regulons, genes, and co-expression data be available to all biologists in a user friendly manner. Currently, information about regulons and its associated genes are not easily accessible, and in order to view regulon information, manual look up into flat files such as text files is required. In order to avoid this, we come up with an application which aim at providing regulon, gene, and co-expression data for three important species in a user-friendly manner using a web interface.
Results

RegulonIT is a software application which provides regulon, gene, and co-expression information for very large transcriptomic datasets using a web interface. It is currently focussed on *Arabidopsis thaliana*, *Saccharomyces cerevisiae*, and *Homo sapiens*. The tool includes features like viewing Pearson correlation matrix for genes in a regulon, analysis of gene expression by exporting data from RegulonIT into MetaOmGraph (www.metnetdb.org/MetNet_MetaOmGraph.htm), as well as standard features like search, sort, filter, and download. For instance, the user could search by regulon to find the number of genes present in the regulon, and view information about each and every gene in the regulon. The user could also generate a Pearson correlation matrix for all genes present in the regulon. The tool also allows the user to integrate the results with external data websites and web services like TAIR (The Arabidopsis Information Resource, www.arabidopsis.org), yeast genome (www.yeastgenome.org), KEGG (Kyoto Encyclopedia of Genes and Genomes, www.genome.jp/kegg), MetNet (www.metnetonline.org) and MetaOmGraph (www.metnetdb.org/MetNet_MetaOmGraph.htm) for enabling interactions with external data sources. The RegulonIT software has been designed to enable extensibility, and new species and datasets can be added in an user friendly manner. The web server is supported by a database containing experimental data, annotations, and GO(Gene Ontology) terms for all three species for different datasets. The database has been designed to store pre-calculated co-expression results for rapid processing of user queries.

Conclusions

RegulonIT is a user-friendly platform independent web tool which provides biologists an easy way to access regulon, gene, and co-expression information for very large transcriptomic datasets. This tool may help biologists in developing hypothesis about gene functions or associating genes which participate in the same biological process. The RegulonIT software is available at http://metnetdb.org:9090/regulonit.
2.1 Background

2.1.1 Gene co-expression and the significance of clustering

Genes which are highly correlated are biologically significant, since they may participate in the same biological processes or share similar functionalities\(^1\)(\(^2\))(\(^3\)). Gene co-expression networks derived from gene expression level data are being widely used for analysing the system-level functionality of genes\(^3\). Cluster of genes obtained by clustering a co-expression network, formed by using gene expression data from several experiments, are often functionally related\(^1\)(\(^2\))(\(^4\)).

The expression data obtained from transcriptomic analysis can be transformed into a co-expression network where genes are represented as nodes. For this, the Pearson correlation coefficient between all pair of genes is calculated using gene expression data, as the absolute value of Pearson correlation is often used in gene expression cluster analysis\(^1\)(\(^2\))(\(^3\)). Two nodes are connected by an edge if the correlation co-efficient between them is higher than a threshold correlation co-efficient. Bin Zhang et al.\(^3\) used the idea of soft thresholding for converting gene expression data into a co-expression network. Using simulated data, they provide evidence that weighted networks can yield better results than unweighted networks. Unweighted networks use hard thresholding, where the threshold correlation co-efficient is determined by analysing the density of networks. With increasing Pearson cut-off correlation, the number of edges decreases, and consequently the number of nodes in the network decreases. The correlation at which the network density is minimum is considered to be the threshold correlation co-efficient, as this would maximize the formation of densely connected sub-networks\(^2\).

The resulting network can be subjected to clustering in order to find regulons\(^1\)(\(^2\)). There are different techniques for clustering large scale transcriptomic networks, such as feature-based clustering\(^5\)(K-Means), similarity matrix-based clustering\(^5\)(H cluster), and graph-based clustering\(^5\)(MCL\(^5\)(Markov clustering))\(^5\)). Feng et al.\(^2\) and Mentzen & Wurtele\(^1\) used Markov clustering algorithm for clustering their co-expression network prepared from transcriptomic analysis, as they found it to be scalable to large graphs, and it yielded much better results when
compared to the other two clustering methods. Their evaluation was based on the determination of the best matches of over-representation of genes in GO (Gene Ontology) terms, and metabolic and regulatory pathways. It was concluded by them that MCL yielded better GS scores (global significance score) when compared to the other two clustering methods (1)(2).

Feng et al. (2) analysed a large human co-expression network formed by using transcriptomic data available in Array express (6) using the MetaOmGraph interface. High quality data from 18,637 Affymetrix chips encompassing over 500 experiments was used to form this co-expression network. Regulons formed by clustering this large transcriptomic network using Markov clustering (5) was analysed using over-representation of GO (Gene Ontology) terms and direct visualization of transcript levels (2). Similarly, Mentzen & Wurtele (1) created an Arabidopsis co-expression network by using 22,746 Affymetrix probes dataset derived from 963 micro-array chips from a wide variety of experiments using Array express (6) and MetaOmGraph (7). Markov clustering (MCL) (5) of the co-expression network resulted in 998 regulons and the significance of these regulons were evaluated by calculating the statistical over-representation of GO term and comparing them to randomly-generated sets of clusters (1). It was concluded in both cases that regulon analysis is statistically significant and it creates a framework for developing testable hypotheses about function of unknown genes or genes with no known physiological or developmental role. Hence, regulon analysis offers the potential to identify genes with unknown functionality, prevailing cellular processes and for associating genes which participate in the same biological processes (1)(2).

2.1.2 Existing web based tools for gene analysis

Gene expression data from a variety of experiments across different environmental, disease, and developmental stages, are currently publicly available for different species. Widely used public microarray and RNA sequence data repositories include ArrayExpress (6) and Gene Expression Omnibus (8). Substantial amounts of gene expression data are often generated by each transcriptomic experiment and it is often difficult for biologists to extract the information they seek. A number of software applications have been developed for biologists to query
large gene co-expression databases using a web browser interface. A few examples of such applications are Genevestigator(9), Arabidopsis Co-expression Tool(ACT)(10), KEGG(11)(12), MetNet(13), and GeneCat(14). Though these tools provide comparative gene analyses such as cis-element prediction, expression profiling, and co-expression analysis, there are currently no web tools which provide information about regulons generated by clustering transcriptomic networks.

RegulonIT is a software application which provides regulon, gene, and co-expression information. It is currently focussed on Arabidopsis thaliana, Saccharomyces cerevisiae, and Homo sapiens for very large transcriptomic datasets using a web interface. The datasets were prepared by Feng et al(2), Mentzen & Wurtele(1), and Chen, Xi(15) by using transcriptomic data from Array express using the MetaOmGraph interface.

2.2 Implementation

RegulonIT is an online application developed using the open source technologies: Java EE (Enterprise Edition), Struts (http://struts.apache.org/, version 1.3)(16) framework, Asynchronous JavaScript and XML (AJAX, http://www.zammetti.com/articles/xhrstruts)(17), JavaScript, Java Database Connectivity (JDBC), and Apache Axis(http://ws.apache.org/axis/)(18). The Apache web server is supported by a MySQL(version 5.0.77) database, which is available in the MetNet server, and a Tomcat(version 5.5) servlet container. The main idea behind using AJAX(17) is that re-building a web page for every user interaction is inefficient and should be avoided. When a form is submitted, AJAX helps retrieve results without the web page getting refreshed.

The performance of RegulonIT tool has been improved significantly by using client side scripting wherever possible. JavaScript has been used to achieve this, so that the business logic is processed in the client side, hence putting less load on the server. Also, in order to speed up user queries in the server side, database indexes have been created wherever possible in order to improve database retrieval time.
2.2.1 System Overview and Integration with External Databases

Figure 2.1 shows a schematic overview of the RegulonIT framework and its relationship with other databases. The features which include, search, sort, and filter are available in a form like layout where the user is prompted to enter an input. The form takes input using text boxes, drop down list boxes, check boxes or radio buttons depending on the functionality of the page. When user inputs a query, the request is sent to the server and a response is sent back to the user in the form of a table. Gene information like gene symbols, systematic names, GO (Gene Ontology) terms, and pathway information are retrieved from the RegulonIT database corresponding to the user query. The outcome of all search results are tables containing regulon, gene or co-expression data depending on the functionality of the page. During processing, results generated by RegulonIT are integrated with data from external websites. For example, in the case of Arabidopsis thaliana, the locus ids in the table generated are linked to TAIR (The Arabidopsis Information Resource)(19) in order to obtain more information about the gene function and characteristics. Similarly, systematic names for Saccharomyces cerevisiae are linked to the yeast genome(20) website and gene symbols for Homo sapiens are linked to gene cards(21) website. The application also allows the user to get pathway information from the KEGG(11)(12) database and the MetNet(13) database.

KEGG (Kyoto Encyclopedia of Genes and Genomes) is a database resource developed by Kanehisa Laboratories that integrates genomic, chemical, and systemic functional information for different species(11)(12). The RegulonIT tool integrates with the KEGG database for retrieving pathway information for genes. The KEGG integration has been implemented using the KEGG application programming interface (API) which consists of the SOAP/WSDL interface and the REST interface to the KEGG system. This allows searching biochemical pathways in processes in an easy manner. In order to use the KEGG API, the Apache Axis web services library has been imported and integrated into our project. Pathways from the MetNet database are also incorporated into the search results. MetNet is a publicly available software for analysis of genome-wide mRNA, protein, and metabolite profiling data(13). The MetNet integration is implemented using the MetNet application programming interface (API).
Figure 2.1 RegulonIT - System overview. The RegulonIT framework provides features like search by regulon, search by gene, search by multiple genes and viewing correlation matrix. It also provides integration with external web services like KEGG(11)(12), MetNet(13), TAIR(19), yeast genome(20), gene cards(21), and MetaOmGraph(7). The administrator of this tool will have the privilege to add new species and to add new datasets for existing species.
The MetNet API is a programming library that provides direct access to the MetNet database. The MetNet API is a JAR file which contains Java packages that has been used for retrieving pathway information from the MetNet database.

RegulonIT also allows the search results to be integrated with MetaOmGraph(7). The genes present in the table can be selected and exported to an xml file in a format that is compatible with MetaOmGraph. This file can be imported into MetaOmGraph and the accumulation levels of the genes can be viewed in a plot for different samples. This functionality has been implemented in the client side using Javascript and hence makes the process really fast. Javascript libraries from Downloadify(22) have been used to make it possible to save the file in the users local machine.

The RegulonIT tool has been developed using the Model-View-Controller(MVC) framework. MVC is a framework or a design pattern that is used for developing web applications by providing a clean separation of software architecture into three distinct elements.

1. Model: Model manages behaviour and business data of an application.

2. View: Views are used for displaying Model data in a form suitable for interaction. Views in our design contain HTML components, JavaScript, AJAX and Struts tags for displaying response retrieved from the server.

3. Controller: Controllers are for handling events and they act as an interface between the model and view components. The Controller receives the request from the browser (View), invokes a business operation and coordinates the view to return to the client.

Struts(16), a MVC web framework, is an open-source solution for creating Java web applications. This framework has been used for designing the RegulonIT tool as it provides a clean separation between the Model, View and Controller components. It provides a primary Controller component called the ActionServlet which handles requests from the View component and calls the corresponding business logic component. Struts also includes a set of custom tag
libraries for the View Component, which can be used for creating user interfaces that provide graceful interaction with the user.

2.2.2 Database design

Currently, the database is designed to provide information for three species: *Arabidopsis thaliana*, *Homo sapiens* and *Saccharomyces cerevisiae*. For a given species, new datasets can be added by an administrator. Each species has two tables for a given dataset which are:

- A genes table which contains gene information such as gene symbol, gene title, GO terms, etc.
- A pearsonmatrix table which contains correlation co-efficient for all pair of genes in the genes table.

For different datasets for a given species, the information in the genes table will remain the same except that a new column is added for regulon for each dataset. But, the pearsonmatrix information is different for different datasets. Hence a species will have one gene information table and many pearsonmatrix tables for different datasets.

In the tables shown in Figure 2.2, species can be *Arabidopsis thaliana*, *Saccharomyces cerevisiae* or *Homo sapiens*. Currently, for *Arabidopsis thaliana*, the database has three datasets and for *Homo sapiens* and *Saccharomyces cerevisiae*, the database has one dataset each. The *Arabidopsis thaliana* datasets were prepared by Mentzen & Wurtele, Feng et al.(2) and Chen(15) by using gene expression level data from MetaOmGraph. This expression data was converted into a co-expression network by choosing a Pearson correlation co-efficient of 0.7 and the resulting network was subjected to Markov clustering(MCL)(5) algorithm. The *Homo sapiens* dataset was prepared by Feng et al.(2) in a similar manner.

Currently, the datasets_information table contains dataset name and dataset description information for all five datasets. Each time a new organism or a new dataset is added, the datasets_information table gets updated automatically. The probeid column in the genes_species table acts as the primary key for the table. This column is referenced by columns in the
RegulonIT - Database schema. Species can be *Arabidopsis thaliana*, *Saccharomyces cerevisiae* or *Homo sapiens*. The probeid column in the genes_species table acts as the primary key for the table. This column is referenced by columns in the pearsonmatrix table.
2.3 Results and discussion

The RegulonIT web application consists of a number of features which are used for searching and analysing information for *Arabidopsis thaliana*, Saccharomyces cerevisiae and *Homo sapiens*. Most of these features work the same way for all 3 species, so the example of *Arabidopsis thaliana* and Saccharomyces cerevisiae have been used to explain how these features work.

2.3.1 Search by regulon

The search by regulon feature is used for searching regulon and gene information by regulon. It prompts the user to enter a regulon, select a correlation co-efficient, and select a dataset for viewing regulon information as shown in Figure 2.3. When the search page for *Arabidopsis thaliana* is opened, the dataset drop-down list box lists the datasets currently available in the RegulonIT database. Currently, we have regulon information for 3 different datasets for *Arabidopsis thaliana*. When additional datasets are added for a species by the administrator, the drop-down list box gets populated automatically with the new dataset, the next time the page is loaded. When a dataset is selected from the list, the form makes a request to the server, which fetches a detailed dataset description from the database and displays it just below the drop-down list box. This has been implemented using AJAX. The page also allows the user to select the gene attributes to be displayed once the ‘Search’ button is clicked. The attributes must be selected using a check box in order to be displayed. By default, all gene attributes are checked.

When the input is entered and the ‘Search’ button is clicked, the following validations are done in the client side using JavaScript before the request is sent to the server.

- Validate if an input has been entered.
- Validate if the input entered is a number.
Regulon Presentation for *Arabidopsis thaliana*

Enter Regulon:  
Correlation Co-efficient: 0.7  
Select Dataset: List of datasets  
Select columns to be displayed:  

- Validate if the number is greater than 0
- Validate if the number is an integer
- Validate if a dataset has been selected from the drop down list box.

If the input entered fails to meet any one of the conditions, then an alert box is displayed notifying the user of the error as shown in Figure 2.4. When a valid entry is entered, the search returns the following results:

- Information about the number of genes present in that regulon at the selected correlation co-efficient.

Figure 2.5 shows that regulon 21, has 56 genes present in it at a correlation co-efficient of 0.7 for the selected dataset.

- A table containing information about each gene in the regulon and the number of neighbours for each gene. The table shown in Figure 2.6 provides a detailed description of all 56 genes in regulon 21. The entry in the number of neighbours column in the table
Figure 2.4  Validation for search by regulon feature. When an invalid entry is entered, an error message is displayed.

<table>
<thead>
<tr>
<th>Gene Symbol</th>
<th>LocusID</th>
<th>SwissProt</th>
<th>Description</th>
<th>Pathway from METNET Database</th>
<th>Gene Title</th>
<th>GO Biological Pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSO2</td>
<td>AT3G27060</td>
<td>O95609</td>
<td>ribonucleotide reductase small subunit, putative similar to RIBONUCLEOSIDE-DEPHOSPHATE</td>
<td>purine nucleotide metabolism ( phosphotransfer and nucleotide modification)</td>
<td>TSO2 (TSO meaning ugly in Chinese; oxidoreductase/ribonucleoside-diphosphate reductase/transition metal ion binding)</td>
<td>0006260 II DNA replication</td>
</tr>
<tr>
<td>AIAUR3</td>
<td>AT2G45490</td>
<td>064629</td>
<td>putative protein kinase domain profile (PDOC00100)</td>
<td>No pathway information found</td>
<td>AIAUR3 (AIAUR3A3); ATP binding/histone kinase (H3-S10 specific) / protein kinase</td>
<td>0006468 II protein acid phosphorylation</td>
</tr>
</tbody>
</table>

Figure 2.5  Regulon and gene Information for regulon 21. The search result shows that regulon 21 has 56 genes at a correlation of 0.7.
Figure 2.6 Table containing detailed information about all the 56 genes in regulon 21 at a correlation of 0.7.

has a hyper-link on them, which when clicked displays a detailed description of all the neighbours.

- Additional search features are also displayed (Figure 2.10). These features can be used by selecting rows in the table thus generated and viewing more information. The first column of the table has check-boxes which are used for selecting the rows.

2.3.2 Search by gene

The search by gene feature allows the user to search for gene information by three different criteria for S.cerevisiae - systematic name, symbol and probeid. When an option is selected and the ‘Submit’ button is clicked, it displays more input boxes for providing input data. For example, when the systematic name option is selected and the ‘Submit’ button is clicked, the page displays a text box for entering the systematic name of the gene for which we want to view information and two drop-down list boxes; one for selecting the correlation co-efficient and the other one for selecting the dataset information as shown in Figure 2.7. The page also allows the user to select the gene attributes to be displayed when the ‘Search’ button is clicked.
Figure 2.7  Search by gene page for Saccharomyces cerevisiae(Yeast). This page prompts the user to select a dataset, enter a gene and view information about the entered gene at the selected correlation. In the case of *Saccharomyces cerevisiae*, the search could be performed by systematic name, gene symbol or probeid.

The attributes must be selected using a check box in order to be displayed. By default all gene attributes are checked.

When a valid systematic name is entered and the ‘Get Gene Information’ button is clicked, the page returns the following:

- A table containing detailed information about the gene entered. Figure 2.8 shows gene information when systematic name ‘YKL204W’ is entered.

- A table containing information about neighbours of the entered gene at the selected correlation co-efficient as shown in Figure 2.9. This table also contains the correlation co-efficient between the entered gene and all its neighbours at the selected correlation.

- Additional search features are also displayed (Figure 2.10). These features can be used by selecting rows in the table and viewing information. These search features are explained in detail in section 2.3.3.
Figure 2.8 Table containing information about the gene with systematic name - ‘YKL204W’ at a correlation of 0.6. The table provides information like gene symbol, probeid, swissprot, GO terms, etc. It also displays the number of neighbours of the gene at the selected correlation, which in this case is 0.6.

Figure 2.9 Table containing information about the neighbours of gene - ‘YKL204W’. This table provides detailed information about all the neighbours of the ‘YKL204W’ at a correlation of 0.6.
2.3.3 Additional features from search results

The Search features available in this web application provide additional features once the results are generated and displayed in the form of a table. These features, as shown in Figure 2.5 and Figure 2.10, can be used by selecting the rows in the table using the check boxes available in each row and viewing information about the selected rows (genes).

These features are:

- View correlation matrix
- Integration with MetNet database
- Integration with KEGG database
- Integration with MetaOmGraph

2.3.3.1 View correlation matrix

The Correlation Matrix of ‘n’ genes, X1...Xn, is a ‘n*n’ matrix whose (i,j)th entry is Corr(Xi,Xj) where Corr(Xi,Xj) is the correlation co-efficient between genes, Xi and Xj. When the ‘Get Correlation Matrix’ button is clicked, it will first prompt the user to select the rows for which correlation matrix needs to be viewed. Once the rows are selected and the Submit button is clicked, correlation matrix for all pair of selected genes is displayed. Figure 2.11 shows the correlation matrix created by selecting all 8 neighbours of ‘YKL204W’ shown in Figure 2.9.
Figure 2.11 Correlation matrix for the selected genes. This correlation matrix is created by selecting all 8 neighbours of ‘YKL204W’ shown in Figure 2.9.

When the hyperlink on ProbeID in the correlation matrix table (Figure 2.11) is clicked, then the following results are displayed in a new page:

- Information about the gene selected. This table also provides the number of neighbours at the given correlation co-efficient.

- Table containing information about all the neighbours of the selected gene at a correlation co-efficient that was used for generating the correlation matrix previously.

The table also displays the correlation co-efficient between the selected Probe ID and its neighbours at that correlation co-efficient. The correlation co-efficient for viewing the number of neighbours can be changed by using the ‘Change Correlation Co-efficient’ drop down list box available just below the Gene Information table as shown in Figure 2.12.

2.3.3.2 Integration with MetNet database

The Search results in RegulonIT are designed to display pathway information from the MetNet database using the MetNet API. The MetNet API consists of set of Java library
The Below Table Provides Information For VIR006C

<table>
<thead>
<tr>
<th>Probe ID:</th>
<th>4087_at</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region to which gene belongs to:</td>
<td>22</td>
</tr>
<tr>
<td>Symbol:</td>
<td>P6411</td>
</tr>
<tr>
<td>Gene Title:</td>
<td>Part of actin cytoskeleton-regulatory complex Faptp-Statp-Endp associates with actin patches on the cell cort of poly(A) ribonuclease</td>
</tr>
<tr>
<td>GeneProt:</td>
<td>P35521</td>
</tr>
</tbody>
</table>

**GO Biological Process:**
- 0000447: actin cortical patch assembly inferred from mutant phenotype
- 0005510: cytokinesis inferred from endocytosis inferred from mutant phenotype
- 00007120: axolemma mitochondrial selection inferred from mutant phenotype

**GO Cellular Component:**
- 0005534: nucleus inferred from direct assay
- 00055757: cytoplasm inferred from direct assay
- 00055757: cytoplasm inferred from electronic annotation
- 00005504: plasma membrane inferred from direct assay
- 0010008: endosome inferred from electronic annotation
- 0010020: membrane inferred from direct assay

**GO Molecular Function:**
- 0007779: actin binding inferred from direct assay
- 0007806: calcium ion binding inferred from direct assay
- 0007816: binding bridging inferred from physical interaction

**Number of neighbours at correlation 6.7:**
- 3

**Change Correlation Coefficient:** [0.1 0.9]  
[Get Correlation chart | Get Gene Information | Further database]

Figure 2.12  Table generated when probeid ‘4087at’ is selected from Figure 2.11. The table provides detailed information about the gene selected and also displays the neighbours of the gene at the selected correlation.

As seen in Figure 2.13, our application pulls out pathway information for each row in the table. If the pathway information corresponding to a particular row is not found, ‘No pathway information found’ is displayed. If more than one pathway information is available, then they are displayed one after the other separated by a line. The pathways have a hyperlink on them which when clicked; display a more detailed description of the pathway from the MetNet online website as shown in Figure 2.14.

### 2.3.3.3 Integration with KEGG database

Currently, our web application has three search options available with KEGG(11)(12) as shown in Figure 2.15.

- **Get Pathways** - When the ‘Get Pathways’ radio button is clicked, it prompts the user to select the rows for which KEGG pathway information has to be displayed. When the rows are selected and the Submit button is clicked, it pulls out pathway information from...
Figure 2.13  MetNet pathway information. The search results displays the pathway information from the MetNet database for the genes present in the table. The pathways have hyperlinks on them, which when clicked, displays detailed information about the pathways in the MetNet online website.

Figure 2.14  MetNet pathway information from MetNet online website.
Figure 2.15 Different features available with KEGG database.

the KEGG database and displays the information in a new page(Figure 2.16). This page also contains a detailed description about the genes selected.

- Enter probeid and view Pathway information - Clicking on this option displays a textbox and prompts the user to enter the probeid for which KEGG pathway information has to be displayed. Once the probeid is entered and the ‘Submit’ button is clicked, the KEGG pathway information is retrieved and displayed in a new page.

- Get Common Pathways - This option is used for selecting a set of genes and displaying the pathway information that is common to these genes(Figure 2.17).

Figure 2.17 shows common pathways when rows containing locusids AT5G37510, AT5G08530 and AT3G12260 are selected for fetching the results.

2.3.3.4 Integration with MetaOmGraph

When the MetaOmGraph(7) button shown in Figure 2.10 is clicked, it displays two options:

- Regulon analysis

- Gene analysis
Figure 2.16  Get KEGG Pathways for selected genes for *Arabidopsis thaliana*. This option prompts the user to select the genes for which KEGG pathway information has to be displayed. Once the genes are selected and the search button is clicked, the KEGG pathways are displayed in a new page as shown in the figure.

**Kegg Pathway Information For Arabidopsis Thaliana**

<table>
<thead>
<tr>
<th>Locus ID</th>
<th>Common Pathways</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT5G37510</td>
<td>Oxidative phosphorylation - <em>Arabidopsis thaliana</em> (thale cress)</td>
</tr>
<tr>
<td>AT5G08530</td>
<td>Metabolic pathways - <em>Arabidopsis thaliana</em> (thale cress)</td>
</tr>
<tr>
<td>AT3G12260</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.17  Get KEGG common pathways for *Arabidopsis thaliana*. This option allows the user to view KEGG pathways that are common to a set of genes. The above figure shows pathways that are common to lociids AT5G37510, AT5G08530 and AT3G12260.
This option allows the user to analyze accumulation levels of genes, regulon-wise, in MetaOmGraph.

Selecting the ‘Regulon analysis’ option will first prompt the user to select the rows in the table which has to be exported to MetaOmGraph. It will also prompt the user to enter a file name to store this regulon and gene information in a xml file. Once the genes are selected and a file name is specified, the ‘Save To Disk’ has to be clicked. This will open a dialog box to specify the path where the xml file has to be saved. Figure 2.18 shows the hierarchy of the xml file for a select set of genes for regulon analysis. The genes selected are arranged according to the regulon to which they belong. Hence, this allows the user to analyze the accumulation levels of the selected genes, regulon-wise in MetaOmGraph. Once the xml file is saved in the local machine, it could be imported into MetaOmGraph by using the ‘Import Lists’ option available in the Project menu bar in MetaOmGraph. Once the xml file has been imported, the list of regulons shows up in the lists section in the left side of MetaOmGraph as shown in Figure 2.19.

Figure 2.20 shows a plot when regulon 1 is selected from the list and the ‘Plot’ option is used. The plot provides accumulation level information for all the genes in the list selected, in this case regulon 1, for different samples. Selecting the ‘Gene analysis’ option would put all
2.3.4 Get correlation matrix from text area

This feature allows the user to enter a set of genes and view the correlation matrix between them. For *Saccharomyces cerevisiae*, there are three ways for viewing the correlation matrix as shown in Figure 2.22. We could either view by entering systematic names, Probe IDs or Symbols. Once the genes are entered in the text area, a dataset must be selected using the
Hierarchical representation of selected genes in XML file for gene analysis. This creates a single list with all the selected genes.

drop down list box. When the ‘Submit’ button is clicked, the correlation matrix for the entered genes is retrieved and displayed(Figure 2.23). If ‘n’ genes are entered, then an ‘n*n’ matrix is displayed.

### 2.3.5 Search by multiple genes

This feature allows the user to enter more than one gene and view information. The form layout is somewhat similar to the ‘Get Correlation from Text area’ feature, where the genes to be analysed are entered in a text-area and the correlation co-efficient and the dataset are selected from the drop-down list box. When the dataset is selected from the drop down list box, a detailed description of the dataset is displayed just below the drop-down list box. When the ‘Submit’ button is clicked, the information about all the genes entered is retrieved from the database and is displayed in the form of a table. Detailed information about each of these genes can be viewed by clicking the hyperlink on the Probe ID column.

### 2.3.6 Search within table

So far, we have seen that most of the search results are displayed in the form of a table and some of these tables might contain a lot of rows that it might be difficult for the user to search for a particular entry in the table. In order to overcome this, we have a search feature which allows the user to enter a keyword and search the entire table for the entry. If the keyword is
Figure 2.22 Viewing correlation matrix from text area for Saccharomyces cerevisiae. This feature prompts the user to enter the genes manually in the text area and view correlation matrix for all the entered genes.

Figure 2.23 Correlation matrix table for the set of genes entered.
found, then the browser scrolls to that part of the page where the entry exists and highlights the entire row.

2.3.7 Sort

The sort feature is used to sort the data in the table. When a table needs to be sorted by a particular column, then the column header needs to be clicked once. When the header is clicked, the table gets sorted by ascending order of the column. When the column header is clicked again, the table gets sorted by descending order of the column. The sort feature is implemented by using the sorttable.js(23) library file.

2.3.8 Case Study 1 - Study of Regulon 48 in Human co-expression network prepared and clustered by Feng et al.(2)

RegulonIT tool was used for studying regulon 48 in the Human co-expression dataset prepared by Feng et al.(2). It can be seen from Figure 2.24 that the number of genes in regulon 48 is 9. When the Pearson correlation matrix is viewed for all the 9 genes, it could be seen from Figure 2.25 that 8 of the 9 genes are totally interconnected with each other. Further, the number of neighbours for each of these genes is 9 or 10. A closer look at each of the neighbours indicate that this regulon is isolated and has no interactions with other regulons as claimed by Feng et al. Figure 2.26 shows the Kegg(11)(12) pathway information for the 8 interconnected genes present in regulon 48. It could be seen that 7 of the 8 neighbours participate in the Mineral absorption pathway.

Feng et al.(2) analysis of Regulon 48 explains that, each of the genes encodes one of the eight metallothioneins. This is an isolated regulon, which means the genes present in this regulon are only connected to each other and there is no connection with any other genes in other regulons. It was concluded by Feng et al.(2) that regulon 48 is highly dense and contains all eight of the metallothionein (MT) antioxidant genes in the human genome.
Figure 2.24  Search results when Regulon 48 is used as the input query.

Figure 2.25  Pearson correlation matrix for all 9 genes present in Regulon 48.
2.3.9 Case Study 2 - Regulon inter-connectivity

The RegulonIT tool was used for looking up information about three genes, POU2AF1, CD79A and IGHM(Figure). It could be seen from the figure that POU2AF1 and CD79A belong to regulon 47 and IGHM belongs to regulon 10. A correlation matrix generated for these genes show that these genes are interconnected, which means regulon 47 and 10 are interconnected. Feng et al.(2) claim that Regulon 10 is almost entirely composed of immunoglobulins, and regulon 47 contains predominantly genes associated with immune signaling pathways, and this information could be used for developing testable hypothesis about the genes important in integration of immunoglobulin signaling and immune signaling(2).

2.4 Conclusions and future developments

The RegulonIT framework thus helps biologists for viewing regulon, gene and co-expression data in an user-friendly manner using a web interface. The application is available publicly and can be used by anyone with an internet connection and a web browser. The software tool has an user guide section which explains how to use the different features available with this
Figure 2.27  Search by multiple genes feature used for looking up information about POU2AF1, CD79A and IGHM in the Homo sapiens dataset prepared by Feng et al.(2)

Figure 2.28  Gene information for POU2AF1, CD79A and IGHM. It could be seen that POU2AF1 and CD79A belong to regulon 47 and IGHM belongs to regulon 10.
Correlation Matrix For Homo sapiens

Dataset: Human_regulons_1point1

<table>
<thead>
<tr>
<th>ProbeID</th>
<th>205049_s_at</th>
<th>205267_at</th>
<th>209374_s_at</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>CD79A</td>
<td>POU2AF1</td>
<td>IGHM</td>
</tr>
<tr>
<td>205049_s_at</td>
<td>CD79A</td>
<td>1</td>
<td>0.71</td>
</tr>
<tr>
<td>205267_at</td>
<td>POU2AF1</td>
<td>0.71</td>
<td>1</td>
</tr>
<tr>
<td>209374_s_at</td>
<td>IGHM</td>
<td>0.7</td>
<td>0.74</td>
</tr>
</tbody>
</table>

Figure 2.29 Pearson Correlation Matrix generated for all three genes, POU2AF1, CD79A and IGHM shows that Regulon 47 and Regulon 10 are interconnected.

application and helps the user to understand the tool better.

As with any software application, the RegulonIT application can be modified to include further enhancements. Some features suggested are:

- Capability to download tables generated as csv or text files.
- Integration with more external tools like Cytoscape, MetViz, etc.
- Adding micro-array data into the RegulonIT database and develop a feature that generates regulons on the fly.
- Adding more species data into the database.
- Developing a feature that enables the user to compare two or more regulons.

2.5 Availability and requirements

- Project name: RegulonIT
• Project home page: http://metnetdb.org:9090/regulonit/

• Operating system: Platform independent (web-based application)

• Programming language: Java

• Other requirements: A web browser.

• Any restrictions to use by non-academics: none

2.6 Abbreviations

TAIR, The Arabidopsis Information Resource; KEGG, Kyoto Encyclopedia of Genes and Genomes; GO, Gene Ontology; MCL, Markov clustering; ACT, Arabidopsis Co-expression Tool; AJAX, Asynchronous JavaScript and XML; JDBC, Java Database Connectivity; API, Application Programming Interface; MVC, Model-View-Controller; HTML, Hypertext Markup Language; URI, Uniform Resource Identifier.

2.7 Authors contributions

AB developed the RegulonIT web tool with constant guidance, support and advice from LM and EW. LM provided suggestions for choosing the resources to be used for developing the web based tool. EW helped in writing the manuscript by providing biological information. Biologists working in the Virtual Reality Application Center(VRAC) in the Iowa State University have also been helping by providing regulon, gene and co-expression data for all three species.

The next chapter explains the implementation in detail.
CHAPTER 3. ORGANIZATION AND IMPLEMENTATION

In this chapter, the details of the implementation are discussed. Java EE has been used extensively for developing this web application. Other notable technologies and API’s used are the Struts 1.3(16), MySQL 5.0.77, Ajax (Asynchronous JavaScript and XML)(17), JavaScript, Java database connectivity(JDBC) and Cascaded style sheets(CSS).

3.1 Project organization

The project is organized based on the model-view-controller framework. It contains three main java packages

- Controller - The Controller package contains Action classes. These are basically Java classes which extend the Action class available in the Struts library. The execute() method of the Action class is overridden and the business logic is implemented in this method.

- DAO - DAO stands for Data Active Objects. This package contains Java classes that help connecting to the database and executing SQL queries.

- Model - The model package contains classes that are used for manipulating business data. These classes are Java Beans.

The struts-config.xml file and web.xml file are present in the WEB-INF folder and all the view files(JSPs) are stored in the /webapp folder of the project.

3.2 Struts overview

Figure 3.1 shows the flow pattern of the Struts MVC system architecture. JSP or Java Server Pages are compiled into Java servlets the first time a JSP page is requested. Tomcat 5.5
is a servlet container which provides the environment necessary for this translation, and servlets
in the controller component are designed to handle the requests made by the view component
from a web browser. Struts uses a special servlet called the ActionServlet, which is configured
in the web.xml file to direct requests to the appropriate servlet in the controller component.
The action servlet receives a forward request from the action and instructs Tomcat to send the
request to the forward’s URL. This architecture makes web applications much easier to design,
create, and maintain.

The web application has a deployment descriptor file, web.xml, which describes the con-
figuration of the web application. The configuration includes welcome pages, servlet mappings
and parameters to servlets. The Struts ActionServlet is configured in such a way that it will
handle all requests for a given mapping as shown in Figure 3.2.

The Struts configuration file, struts-config.xml, contains details as to how the model, view,
and controller components are tied together. It associates paths with the controller components
known as Action classes as shown in Figure 3.3. The mechanism by which Struts work is
discussed below.

When the Struts ActionServlet receives an incoming request, it invokes the corresponding
controller (Action) component configured in the struts-config.xml file. For each action, the re-
sulting page (View) that should be displayed once the action is complete can also be configured.
There could be more than one view as the result of an action. The Struts system using this
configuration file forwards the response to the appropriate page once the business logic has been
processed. The model part of the application is called from within the controller component.
A Java Bean is associated with an action in the struts-config.xml file and is used for storing
form data or display data. These beans are made visible to the controller component, using the
ActionForm class available with Struts, and any JSP (view) page that is associated with that
controller. The client submits the data from the web browser using POST/GET methods, and
the Struts system populates the bean with this data before calling the controller component.
The front end in our web application is developed using Java Server Pages (JSP) which uses
Figure 3.1 Flow pattern of the Struts MVC framework
Figure 3.2  web.xml

```xml
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>
    org.apache.struts.action.ActionServlet
  </servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
</servlet-mapping>

Figure 3.3  Example of action path configuration

```xml
<action path="/login"
  type="controller.loginauthenticationAction"
  name="loginauthenticationform">
  <forward name="success" path="/adminincapabilities.jsp"/>
  <forward name="failure" path="/index.jsp"/>
</action>
the Struts tag libraries for developing a seamless presentation layer. The Struts tag libraries includes Struts-specific tags that help display dynamic data in our view.

3.3 Implementation

The following sections use the example of search by regulon feature in Arabidopsis thaliana to explain how the application has been implemented. JSP (Java Server Pages), HTML (Hyper Text Markup Language), JavaScript, AJAX (Asynchronous JavaScript and XML), CSS (Cascaded Style Sheets) have been used for delivering dynamic data.

3.3.1 Request View

The view component is basically a JSP file which contains text boxes, radio buttons, drop-down list boxes and check boxes for obtaining input from the user. The code in Figure 3.4 shows implementation of some of these input boxes.

This creates a text box with name ‘regulonid’ and a drop down list box with name ‘correlationcoeff’. The search by regulon JSP page when loaded for the first time also displays the list of datasets available for Arabidopsis thaliana. This is achieved by using AJAX and the ‘onload’ option available with the body tag.

The function retrieveURL() in Figure 3.5 takes in two parameters:

- The request query string which contains the action path information, and
- The ‘datasets’ tag which will be used for displaying the response once the request has been processed. The retrieveURL() function is implemented as a part of AJAX.

3.3.1.1 AJAX

Ajax stands for Asynchronous JavaScript + XML. The main idea behind AJAX is that re-building a web page for every user interaction is inefficient and should be avoided. In other terms, when a form is submitted, AJAX helps retrieving results without the web page getting refreshed. Frank Zammetti’s(17) article on XHR Struts has been used for implementing this.
Figure 3.4 Example of request view component

<table>
<thead>
<tr>
<th>Enter RegulonId:</th>
<th>Co-relation Co-efficient:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;input type=&quot;text&quot; id=&quot;regid&quot; name=&quot;regulonid&quot; style=&quot;width:40px&quot;&gt;</td>
<td>&lt;select name=&quot;corelationcoeff&quot;&gt;</td>
</tr>
<tr>
<td>option value=&quot;0.7&quot;&gt;0.7&lt;/option&gt;</td>
<td>option value=&quot;0.75&quot;&gt;0.75&lt;/option&gt;</td>
</tr>
<tr>
<td>option value=&quot;0.8&quot;&gt;0.8&lt;/option&gt;</td>
<td>option value=&quot;0.85&quot;&gt;0.85&lt;/option&gt;</td>
</tr>
<tr>
<td>option value=&quot;0.9&quot;&gt;0.9&lt;/option&gt;</td>
<td>option value=&quot;0.95&quot;&gt;0.95&lt;/option&gt;</td>
</tr>
</tbody>
</table>

Figure 3.5 The retrieveURL Javascript function call from the body tag

<body onload="retrieveURL(’getlistofdatasets.do?organism=arabidopsis’, ’datasets’);”>
Ajax is based on a component called XMLHttpRequest. The XMLHttpRequest is a client side component and must be instantiated via scripting in the JSP using JavaScript.

The basic idea of the code in Figure 3.6 is simple. When the retrieveURL() function is called, it creates an instance of the XMLHttpRequest object depending on the web browser, and then sends a request to the URL provided using the GET method. The third parameter in the req.open is used to specify whether the call be asynchronous or not. Setting the value to true makes it asynchronous. Once the req.send() statement is executed, the request is forwarded to the struts-config.xml file which decides which Action class needs to be called. Once the business logic is processed in the servlet (Action class), a response is sent back. One important line in the above code is the req.onreadystatechange = processStateChange(tag) line. This line of code is for setting up an event handler. When the state of the request changes, a response is sent back and the processStateChange() function will be called.

We then question the state of the XMLHttpRequest object and respond appropriately. Next, we check the HTTP response code that was received. If the response code received is any number other than 200, it will result in an error. In the example shown in Figure 3.6, when a complete response is returned from the servlet, and if it has an OK response, the text we received is inserted into an HTML element whose identity is ‘datasetsdata’, since the variable tag has ‘datasets’ as its value. Hence, a call is made to the server without the web page getting refreshed and the results are displayed on the same page. The asynchronous nature of AJAX helps the user in making more than one request to the server without waiting for response from a given request.

Once the user enters a regulon, selects a correlation co-efficient and selects a dataset from the populated list, the ‘Search’ button needs to be clicked. The functionality of the ‘Search’ button is implemented as shown in Figure 3.7.

The code in Figure 3.7 calls a JavaScript function get_checked_values() by passing information about the regulon, correlation co-efficient, and dataset. This function gets the gene attributes to be displayed, once we get the response, and sends the request to the server using the retrieveURL() function in AJAX. Once the request is sent, the business logic is called by the Controller component is explained in the section 3.3.2.
```javascript
var req;
var which;

function retrieveURL(url, tag) {
    loadSubmit();
    if (window.XMLHttpRequest) { // Non-IE browsers
        req = new XMLHttpRequest();
        req.onreadystatechange = function() {
            processStateChange(tag);
        };
    }
    try {
        req.open("GET", url, true);
    } catch (e) {
        alert(e);
    }
    req.send(null);
} else if (window.ActiveXObject) { // IE
    req = new ActiveXObject("Microsoft.XMLHTTP");
    if (req) {
        req.onreadystatechange = function() {
            processStateChange(tag);
        };
        req.open("GET", url, true);
        req.send();
    }
}

function processStateChange(tag) {
    if (req.readyState == 4) { // Complete
        if (req.status == 200) { // OK response
            document.getElementById("progress").style.visibility = "hidden";
            document.getElementById(tag + "data").innerHTML = req.responseText;
        } else {
            alert("Problem: ", + req.statusText);
        }
    }
}
```

Figure 3.6 AJAX implementation
3.3.2 Controller

The Controller component forms the crux of our application. The controller component contains Action classes which process the business logic and sends back the response to the view component. For the search by regulon, using Arabidopsis thaliana as a specific example, the important segments in the code are explained in Figure 3.8.

The Java class ‘searchbyregulonarabidopsisAction’ extends the Action class available in the Struts library. The execute method of the Action class is overridden and the business logic is implemented in this method. The execute method defines four parameters:

- **ActionMapping mapping**: The ActionMapping class contains a method called findforward() which is used for forwarding the response to the view that needs to be displayed once the business logic has been executed.

- **ActionForm inform**: The ActionForm is the Java Bean with set() and get() properties that are used for storing and retrieving form values. These are usually input values from the JSP file which has been entered by the user.

- **HttpServletRequest class**: This is used for handling requests from the JSP and the HttpServletResponse is for handling responses once the Action class has completed execution of its logic.

The request.getParameter() method of the HttpServletRequest class is used for getting data from the request query string sent from the JSP. By using this method, information about the regulon, correlation and dataset information is obtained and a call is made to the
public class searchbyregulonarabidopsisAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm inForm, 
        HttpServletRequest request, HttpServletResponse response) throws Exception {

        String regulon;
        regulon = request.getParameter("regulon_value");
        ...

        ResultSet result1 = searchbyregulonarabidopsisDAO.getResultSet(regulon, 
            corel, dataset);
        ArrayList<arabidopsisForm> list = new ArrayList<arabidopsisForm>();
        String AGI;
        int count = 0

        while (result1.next()) { // process results one row at a time
            ArrayList<arabidopsisForm> locusids = new ArrayList<arabidopsisForm>();
            count++;
            arabidopsisForm al = new arabidopsisForm();
            al.setRegulonid(Integer.parseInt(regulon));
            al.setDataset(dataset);
            al.setProbeid(result1.getString(1));
            al.setSymbol(result1.getString(2));
            ...
            al.setAGI(AGI);
            Scanner scanner1 = new Scanner(AGI);
            scanner1.useDelimiter(";");

            while (scanner1.hasNext()) {
                arabidopsisForm a2 = new arabidopsisForm();
                String locusid;
                locusid = scanner1.next();
                a2.setAGI(locusid);
                locusids.add(a2);
            }
            ...
        }
        result1.setNowhenIncreased(count);
        request.setAttribute("list", list);
        ...
        return mapping.findForward("success");
    }
}

Figure 3.8 Example of a Controller class
searchbyregulonarabidopsisDAO class in the DAO(Data Access Objects) package for retrieving data from the database for the input provided by the user. The data thus retrieved from the database is stored using a ResultSet object and is iterated and stored in a list using the user defined arabidopsisForm class and the ArrayList collection which is available in the Java.Util library package. The arabidopsisForm class acts as the Model component for storing data. The data containing regulon and gene information for the input entered by the user is now present in the form of a list and is sent back to the view component using the request.setAttribute() method.

3.3.3 Model

The model package contains classes that are used for holding data retrieved from the database. As shown in Figure 3.9, the model class can have list attributes for storing data that have multiple values. For example, a gene can have more than one locus id. Hence when the response is sent back, there will be a list(list of locus ids) within another list(the master list with all the information retrieved from the database for the entered input).

3.3.4 DAO

As discussed in section 3.3.2, database connections are handled by a separate package called the DAO package.

Database connectivity is achieved using the JDBC (Java Database Connectivity) API. The configurations are made in the context.xml file, as shown in Figure 3.10, where we can declare the characteristics of the resource to be returned for JNDI lookups of resource-ref and resource-env-ref elements in the web application deployment descriptor. We must also define the needed resource parameters as attributes of the Resource element, to configure the object factory to be used.

The driverclass name, username, password and the database url are specified and connection is established using the InitialContext and DataSource objects in the DAO class as shown in Figure 3.11.
public class arabidopsisForm extends OrganismForm{

    String swissprot;
    String locusid;
    ArrayList<arabidopsisForm> locusids;

    ...

    public String getSwissprot() {
        return swissprot;
    }

    public void setSwissprot(String swissprot) {
        this.swissprot = swissprot;
    }

    public String getLocusid() {
        return locusid;
    }

    public void setLocusid(String locusid) {
        this.locusid = locusid;
    }

    public ArrayList<arabidopsisForm> getLocusids() {
        return locusids;
    }

    public void setLocusids(ArrayList<arabidopsisForm> locusids) {
        this.locusids = locusids;
    }

    ...

    ...
}

Figure 3.9  Example of a Model class

<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
    maxActive="1000" maxIdle="30" maxWait="10000"
    username="***" password="***" driverClassName="com.mysql.jdbc.Driver"
    url="***"/>

Figure 3.10  Database configuration in context.xml file
InitialContext initContext = new InitialContext();
DataSource ds = (DataSource)initContext.lookup("java:comp/env/jdbc/TestDB");
java.sql.Connection conn = ds.getConnection();
Statement stm = conn.createStatement();
ResultSet result1 = stm.executeQuery(sqlquery);

Figure 3.11 Establishing database connection using InitialContext and DataSource objects

The Statement class can be used for executing any SQL statements and the result set data thus obtained after data is retrieved is sent back to the Action class which in turn stores the result set as a list and sends the response back to the JSP.

3.3.5 Response view

Once the logic is processed by the Action class in the Controller package, the response is forwarded to the appropriate JSP using the findforward() method of the Controller class. This JSP contains Struts tags and HTML tags for reading the response and displaying the results in a table format. In most of the search features, the response is in the form of a list which contains genetic information for the input entered by the user. For ‘Search by Regulon for Arabisopsis thaliana’ feature, each element in this list is an instance of the arabidopsisForm model class.

The first three lines of the code in Figure 3.12 include three main tag libraries available with Struts - the bean, html and logic tag libraries. A detailed explanation of these tag libraries is given in the section 3.3.8. The logic:present tag in the code is used to check if a list with name ‘list’ has been sent from the servlet. If the list is present, it is iterated using the logic:iterate tag, and the bean:write tag is used to retrieve the value of each attribute stored as shown in the code. The HTML table tag is used for displaying the list in the form of a table. The sortable.js(23) JavaScript library file included within the script tags is used for sorting the table thus obtained and the style tag is used for creating cascaded style sheets for formatting the page. Also, the script tag contains JavaScript functions which will be executed from the client side i.e the web browser. The code in Figure 3.13 shows an example of a JavaScript
Figure 3.12 Example of a response view component
function iteratetable(id) {

    searchid = id.replace(/\s+/g, "");
    searchid = searchid.replace(/\s+\$/g, "");
    var flag = 0;
    try {
        rows = document.getElementById("regulontable").getElementsByTagName("TR");
        for (var j = 1; j < rows.length; j++) {
            cells = rows[j].getElementsByTagName("TD");
            for (var l = 1; l < cells.length; l++) {
                temp = cells[l].textContent;
                temp = temp.replace(/\s+/g, "");
                temp = temp.replace(/\s+\$/g, "");
                if (temp.toUpperCase() == searchid.toUpperCase()) {
                    ScrollToElement(cells[l]);
                    for (var k = 0; k < cells.length; k++) {
                        cells[k].style.backgroundColor = '#AFF';
                    }
                    flag = 1;
                }
            }
        }
        if (flag == 0) {
            alert("" + searchid + "" + "not found in table!");
        }
    }
    catch (er) {
        alert(er.description);
    }
}

Figure 3.13 Example of a Javascript function

function which implements the ‘Search Within Table’ feature discussed earlier.

When the user enters a query and clicks the search button, the function in Figure 3.13 is called. The above code uses the HTML DOM(Document Object Model) for getting the table that needs to be searched. The HTML DOM defines a standard way for accessing and manipulating HTML documents. The document.getElementById('regulontable') first fetches the table which has ID as 'regulontable', and the getElementsByTagName("TR") method is then used for getting all the rows in the table. These rows are then iterated and for each row, the value entered by the user is checked in each column. If a match is found, then the browser scrolls to that part of the page where the match is found and highlights the entire row.

### 3.3.6 Getting MetNet pathways using MetNet API

The MetNet(13) API(application programming interface) is a programming library that provides direct access to the MetNet database. It offers flexible data retrieval methods for Java applications that use biological network information. The MetNet API is a JAR file which contains Java packages that can be used for pulling out genetic information from the MetNet
database. In our web application, we use the MetNet API primarily for retrieving pathway information. The MetNet database currently contains pathway information for Arabidopsis and Saccharomyces cerevisiae (yeast).

The code in Figure 3.14 gives a brief idea as to how pathway information is retrieved for Arabidopsis.

The identify() method in the above code is used for retrieving a entity object corresponding to a given locus id from the MetNet database. The entity object thus retrieved uses the getPaths() method for returning a collection of pathway objects. The PathwayVector Class in the MetNet API package is used for managing this collection of pathway objects. The PathwayVector class contains ‘id’ and ‘name’ attributes which provides a detailed description of the pathways retrieved. In order to use the MetNet API, the MySQL driver needs to be imported, but we don’t have to reference it in our source code.

### 3.3.7 Getting KEGG pathways using KEGG API

The KEGG(11)(12) API (application programming interface) consists of the SOAP/WSDL interface and the REST interface to the KEGG system. It allows searching biochemical pathways in processes. In order to use the KEGG API, the Apache Axis web services library has been imported into our project. This is because the class developed in KEGG API extend classes and implements interfaces developed in the Apache Axis library. Apache Axis is essentially a SOAP engine, a framework for constructing SOAP processors such as clients, servers.
and gateways. The API is open source and the current version of Axis is written in Java ((11) and ((12)).

The code in Figure 3.15 first creates an instance of KEGGLocator class and uses the getKEGGPort() method for getting the KEGG port. This method returns an instance of KEGGPortType class. The get_pathways_by_genes() method of this instance is used for retrieving pathway information for the entered gene.

3.3.8 Creating views using Struts tag

The application also uses Struts tag libraries in some pages for creating input boxes for getting input from the user. The below section uses the example of ‘View Correlation Matrix’ for Arabidopsis thaliana for explaining how these tags work. Three main tag libraries are available with Struts.

- html - This taglib contains tags used to create Struts input forms, as well as other tags generally useful in the creation of HTML-based user interfaces. Some of the important Struts tags available are html:text, html:radio, html:form, html:select, html:submit etc.

A Java Bean is used for storing the above form data so that it can be used by the controller layer. These Beans are automatically made visible to the controller components. The client submits the data from the web browser using POST/GET methods, and the Struts system updates that data in the Bean before calling the controller component. The Java Bean extends the Struts ActionForm class.
Figure 3.16 Example of a view component created using struts tags

Figure 3.17 shows the Java Bean for the view component in Figure 3.16:

- **bean** - This tag library contains tags that help in accessing beans and their properties.

- **logic** - This tag library contains tags that are useful in implementing conditional processing of response from the controller and iterating over object collections for generation of output.

Once the request is sent to the servlet, the ActionForm (Java Bean) is populated, the business logic is processed by the Action class and the response is sent back to a JSP which is determined by the struts-config.xml file.
public class correlationinfoform extends ActionForm{

    String textarea1;
    String searchcriteria;
    String dataset;

    public String getDataset() {
        return dataset;
    }

    public void setDataset(String dataset) {
        this.dataset = dataset;
    }

    public String getSearchcriteria() {
        return searchcriteria;
    }

    public void setSearchcriteria(String searchcriteria) {
        this.searchcriteria = searchcriteria;
    }

    public String getTextarea1() {
        return textarea1;
    }

    public void setTextarea1(String textarea1) {
        this.textarea1 = textarea1;
    }

}

Figure 3.17  Example of an ActionForm
CHAPTER 4. GENERAL CONCLUSIONS

The RegulonIT framework thus provides an user friendly way for viewing regulon, gene and co-expression data for three species - Arabidopsis thaliana, Saccharomyces cerevisiae, and Homo sapiens using a web interface. It provides interactions with external websites and web services like TAIR, yeast genome, MetNet, KEGG, etc., for providing more genetic information. The tool also provides integration with MetaOmGraph for viewing accumulation levels of genes for different samples.

The tool has been developed using the Struts framework as it provides a clean separation between the model, view and controller components and hence allows easy maintenance. AJAX has been used so that pages are not refreshed each time a request is made. The web tool allows easy addition of new datasets by privileged users and hence makes it possible to maintain live data.
BIBLIOGRAPHY


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

