

A VISUAL SPREADSHEET USING HTML5 FOR WHOLE
GENOME DISPLAY

NADA ALHIRABI

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

MARCH 2015
© NADA ALHIRABI, 2015

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Nada Alhirabi**

Entitled: **A Visual Spreadsheet using HTML5 for Whole Genome Display**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr.Rajagopalan Jayakumar

_____ Examiner
Dr.Juergen Rilling,

_____ Examiner
Dr.Hovhannes Harutyunyan

_____ Supervisor
Dr.Gregory Butler

Approved by _____
Chair of Department or Graduate Program Director

Dean of Faculty

Date _____

Abstract

A Visual Spreadsheet using HTML5 for Whole Genome Display

Nada Alhirabi

Modern sequencing technology has enabled the cheap, rapid production of whole genomes. There is a need for visualization tools to show the data collected about a whole genome such as genes, proteins, annotations, and expression data. Many common approaches are developed such as the genome browser where sequence features are displayed as visual elements in tracks and features are aligned with their genome coordinates, visual networks where the data elements represented as nodes and relationship as edges, and traditional spreadsheet where each row captures the information about a gene/genome where the information is textual in nature, such as identifiers, descriptions, or sequences. Our study is focusing in the last approach with introducing some advanced features.

To build the system, the common used similar systems are reviewed, and during the implementation some software artifacts are reused such as reusing some JavaScript libraries to reduce the complexity of software development. Generally, an incremental method is used to develop the web-page starting from collecting the data from AspGD database, analyzing them, coding then testing them once at time. Our research group studies fungal genomes, so the spreadsheets are tested by displaying each of the *Aspergilli* genomes in the AspGD database (www.aspgd.org).

We have developed CGene and CGenome, pronounced See-Gene and See-Genome respectively, as a HTML5 web-based spreadsheets that can incorporate visual displays, as well as text, within the spreadsheet cells. Current displays use Scalable Vector Graphics (SVG) to present these spreadsheets which are generated from standard GFF3 files, standard output files from InterProScan, aspgd files from AspGD Gene Ontology Annotations File, and Chromosomal Feature File. All these files are analyzed to present them in a visual way that requires less effort to understand.

The main aim of our study is to take the advantages of the ability of humans to recognize patterns. The user can see the gene/genomes of interest as row-by-row of visualization. This can play powerful roll to ease the understanding of quantitative data by replacing them by graphical figures that make the comparison easier.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Dr.Greg Butler for his support, patience, knowledge and guidance throughout my period of my research.

I am particularly grateful to my beloved husband Majed for being supportive during the hardest moments, my daughter Jood for giving me the warmest smile every day, and to my baby Lujain for sharing the most stressful moments with me. Many thanks to them for sharing with me this experience by its good and ugly parts.

I owe my deepest gratitude to my mother Fawziah and my father Abdullah, whom I miss a lot, for their unconditional love, support and encouragement for every step in my life. Thanks to my family: Majed, Muhammed, Hessah, Munerah and Abdulaziz for their nice words that encouraged me to follow my dreams. Thanks to my grandmother for her warm words and tears every time she called me.

Many thanks go to the friendly lab mates at Concordia University: Asma Mistadi, Christine Kehyayan, Faizah Aplop, Lin Cheng, Patricia Hanney and Stuart Thiel for their help during my research.

Contents

Abstract	iii
Acknowledgments	iv
List of Figures	viii
List of Tables	x
Abbreviations	xi
1 Introduction	1
1.1 Visual Spreadsheet for Whole Genome	1
1.1.1 What is Data Visualization, and Why Do It?	1
1.1.2 Genomics Sequencing and Annotation	3
1.1.3 Whole Genome Visualization Tools	3
1.2 Motivation	5
1.3 Research Challenges	5
1.4 Research Contributions	6
1.5 Organization of Thesis	6
2 Background and Basic Concepts	7
2.1 Basic Concepts of Biology	7
2.1.1 Cell, Nucleic Acids and Amino Acids	7
2.1.2 Central Dogma of Molecular Biology	8
2.2 Gene and Gene Model	10
2.2.1 Genes, Exons, Introns and Coding Segments	10
2.2.2 Gene Model	10
2.3 Gene Annotation and Gene Ontology	11
2.3.1 Gene Annotation	11
2.3.2 Gene Ontology	13
2.3.3 Go Slims	16
2.4 Protein Structure, Protein Domain and its Architecture	16
2.4.1 Protein Structures	16

2.4.2	Protein Domains	17
2.4.3	Protein Domains Architecture	19
2.5	Principles of Biological Data Visualization	19
2.5.1	Visual Representation Understanding	20
2.5.2	Multiple Views	21
2.6	Whole Genome Visualization Tools	26
2.6.1	Genome Browsers	26
2.6.2	Textual Spreadsheet	27
2.6.3	Network Visualization Tools	28
2.6.4	Tree Layout Visualization Tools	30
2.6.5	Circular Layout Visualization Tools	31
2.6.6	Related Works	32
3	CGene and CGenome: System Analysis and Design	35
3.1	Project Scope and Objectives	35
3.2	System Activity Diagram	35
3.3	General Requirements	36
3.3.1	Functional Requirements	36
3.4	Domain Model	38
3.4.1	Gene and Gene-Model Conceptual Domain Model	38
3.4.2	Protein and Protein-domain Architecture Conceptual Domain Model	39
3.5	Data Sources and Formats	43
3.5.1	Data Selection	43
3.5.2	Data Formats	54
3.6	Data File Preprocessing	55
3.7	Why HTML5 and SVG?	62
3.7.1	HTML5	62
3.7.2	SVG	63
3.7.3	JavaScript	67
3.7.4	Why HTML 5, SVG and JavaScript?	68
3.8	Graphical User Interface	69
4	Conclusion and Future Work	74
4.1	Summary of Works	74
4.2	Work Contributions	75
4.3	Research Limitations and Future Work	75
4.4	Resource of the system	79
	Bibliography	79
	A Gene Model Catalog	85

B Protein Domain Catalog	98
C Comparative Genomics	114
D Glossary	115

List of Figures

1	Tabular view for fungus <i>A_acidus.CBS_106.47</i> features.	2
2	Genome sequencing growth and its cost.	3
3	Illustration of chromosome that have DNA double helix stranded molecule.	8
4	A summary of Central Dogma of Molecular Biology	9
5	Removal of introns during the splicing process to produce a mature mRNA in eukaryotic gene	9
6	Gene model in general eukaryotic gene and <i>A acidus CBS 106.47</i> fungi in CGene system.	10
7	Graphs illustrate the increased amount of Genome Sequencing Projects and annotation.	11
8	Generalized flow chart illustrates Genome Annotation procedure.	12
9	Graph shows <i>regulates</i> and <i>part_of</i> relationships between Molecular Function and Biological Process.	14
10	Gene Ontology terms and annotations from <i>QuickGO</i> web browser.	15
11	3D crystal structure of <i>Endophilin BAR</i> domain with the domain architecture representation.	17
12	Domain architecture representation for DNA repair helicase UVH6 in different DBs.	19
13	Illustration of A Multi-Level Typology of Abstract Visualization Tasks	20
14	A summary illustration of dataset types and their types.	21
15	MulteeSum: an example of Multiple-View system.	22
16	Model-View-Controller (MVC) architecture.	25
17	Rat genome on UCSC Genome Browser	27
18	ISA-TAB representation tool	28
19	Visualization of protein interaction networks in Cytoscape.	29
20	Representing phylogenetic tree in iTOL	30
21	Using Circos Circular layout to express genomic annotation data.	31
22	Screenshot of FancyGene website	33
23	FeatureStack showing RFX gene family members over a diverse set of species	34
24	Activity diagram for visual HTML5 spreadsheets system: CGene and CGenome.	37
25	Gene-Model domain conceptual class diagram.	41
26	Protein and Protein-domain conceptual class diagram.	42

27	Pareto chart shows InterPro entries with their frequencies in the five most common filamentous fungi of the genus <i>Aspergillus</i>	44
28	Pareto chart shows the top 25 selected InterPro domains in the five most common filamentous fungi of the genus <i>Aspergillus</i>	46
29	Chart shows Molecular Function Go Slims <i>Aspergillus</i> occurrence percentage in the specified genome.	49
30	Chart shows Cellular Component Go Slims <i>Aspergillus</i> occurrence percentage in the specified genome.	51
31	Chart shows Biological Process Go Slims <i>Aspergillus</i> occurrence percentage in the specified genome.	53
32	GFFParser.java code UML diagram.	56
33	ProteinDomainParser.java code UML diagram.	57
34	GeneProteinLinking.java code UML diagram.	58
35	GeneModelSVG.java code UML diagram.	59
36	DomainArchitectureSVG.java code UML diagram.	60
37	ChromosomeLocationParser.java code UML diagram.	61
38	CreatHTMLtable.java code UML diagram.	61
39	Illustration of the effect of magnification on vector graphics versus raster graphics. .	64
40	Vector graphics concept illustration.	64
41	Basic vector shapes.	65
42	SVG image for gene locations in chromosomes as part of CGenome spreadsheet. . . .	67
43	Master page of the visual spreadsheet system using HTML5 for the whole genome. .	70
44	CGene spreadsheet with normalized SVG images	71
45	CGene spreadsheet with unnormalized SVG images	71
46	Snapshot of CGene spreadsheet.	72
47	CGenome spreadsheet for different genomes comparative analysis.	73

List of Tables

1	Proteins' 20 amino acids	8
2	The status of Gene Ontology as September 2007 and September 2009.	15
3	The top 25 selected InterPro domains information	45
4	Table shows Molecular Function Go Slims Aspergillus occurrence in specified genomes: A. niger, A. nidulans, A. fumigatus and A. oryzae.	48
5	Table shows Cellular Component Go Slims Aspergillus occurrence in specified genomes: A. niger, A. nidulans, A. fumigatus and A. oryzae.	50
6	Table shows Biological Process Go Slims Aspergillus occurrence in specified genomes: A. niger, A. nidulans, A. fumigatus and A. oryzae.	52
7	Catalog of visualization tools for Gene Models	86
8	Catalog of tabular view for Gene Annotations	90
9	Different glyph's visualization for Gene Models	91
10	Catalog of visualization tools for Protein Domain Architecture.	99
11	Different glyph's for Protein Domain Architecture	103
12	Pfam Domain Architecture graphics.	107
13	DOG: Protein Domain Structure Visualization	111
14	Comparative Genomics catalog	114

Abbreviations

BED	Browser Extensible Data format
CMV	Coordinated and Multiple Views
EV	Exploratory visualization
GFF	General Feature Format
GO	Gene Ontology
HTML	HyperText Markup Language
HMM	Hidden Markov Models
ISA-Tab	Investigation-Study-Assay Tab-delimited
MSA	multiple sequence alignments
SVG	Scalable Vector Graphics
Vis	Visualization
W3C	World Wide Web Consortium

Chapter 1

Introduction

1.1 Visual Spreadsheet for Whole Genome

1.1.1 What is Data Visualization, and Why Do It?

Information overload (infobesity) is considered as one of the major human-computer communication problems. Many efforts were presented to solve this problem, and one of them is data visualization. The main benefit of using this solution is enabling the users to explore and explain data better by taking the advantage of the human ability to recognize patterns [O'Donoghue et al., 2010]. Data visualization, as defined by [Latham and Latham, 1995], is the process of “representing data as visual image”. The created images are combination of graphics primitives such as lines and triangles accompanied by attributes such as shades and colours that used to represent different quantities measurements [Tufte and Graves-Morris, 1983].

When we are given data, the first question is how to design a system to visualize this kind of data. However, we previously need to think deeply about what does this data mean in the real world and what is the type of this data. For example, if the data given is {Apple, Banana}, many questions about the meaning will be guessed such as do they mean fruit names or company names [Munzner, 2014]. The **semantic** is very crucial in this matter in order to see how to group or organize this data. In addition to the semantic, the structural or mathematical data **type** is important to perform visualization analysis. There are different categories for the data type of what can be visualized. One of the categorization divides the dataset types into four basic types that are **tables**, **networks**, **fields** and **geometries**. Each one of them has a combination of associated properties such as **attributes**, **links**, **items**, **positions** and **grid cells** [Munzner, 2014].

In our work, we are focusing on producing spreadsheets which is a tabular display method used frequently to organize and present a high-level summary of annotated genomic information [Dudley and Karczewski, 2013]. In general, a table as display is a combination of rows and columns and their intersection is a single cell. As a dataset, a simple table as seen in Figure 1, consists of items (rows), attributes (columns) and their intersection is a value (cell). Complex/multidimensional tables will have more complex cell indexing with multiple keys [Munzner, 2014].

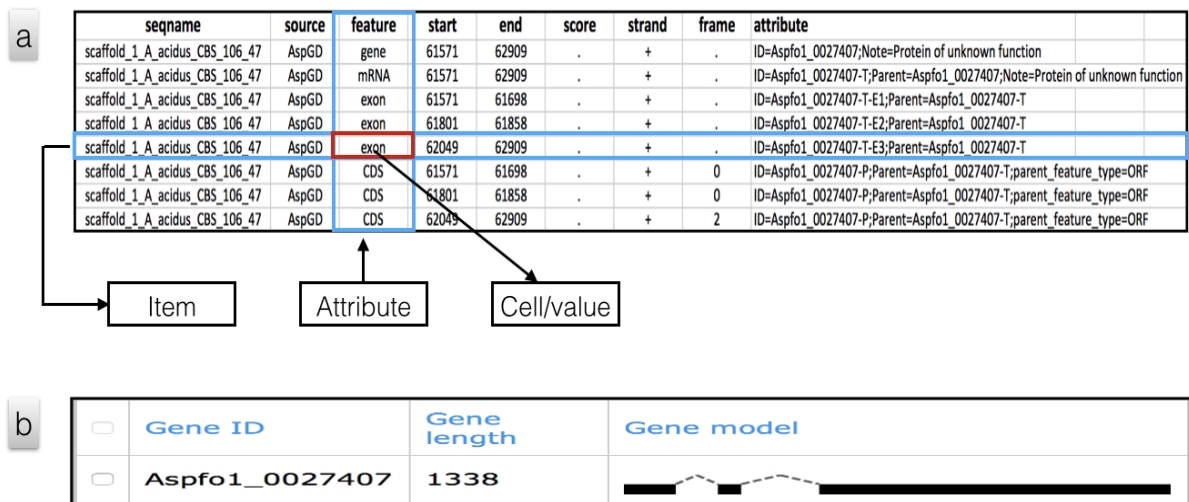


Figure 1: Tabular view for fungus *A.acidus_CBS.106.47* features; where items represented by rows, attributes represented by columns and their intersection is a value which represented by cell. Table (a) represents the features in simple tabular view with textual format from *A.acidus.CBS.106.47* GFF file. Table (b) represents the same information but in visual and textual format as produced by our system called *CGene*.

During the design of data visual representation, the scientists may spend minutes or hours to create the visual display manually. However, in some points their ability to draw by hand will be impossible especially with a massive data such as biological datasets. In this point, the computer-based system is required to save the human effort compared to the manual way [Munzner, 2014]. The computer-based data visualization systems have the ability to demonstrate the datasets in interactive visual representation. This representation is used to extend the understanding and help users to make decision effectively [Card et al., 1999].

According to [Munzner, 2014], data visualization is used in many situations. Visualization is mainly used in a situation when a human assessment and evaluation is needed to make decisions rather than depending on computational decision-making. When users do not know the exact questions to ask about the data and they need to explore things, visualization in this case is also required. In addition, visualization is needed when the experts know something and they want to explain it to others visually or to help them to assess the data easily. In some situations, the need for machine learning and computational techniques for making decisions is high. For example, if scientists know the questions about the dataset very well, but they want to make sure if the outcome from the system is as they expected or not. In that case, they use machine learning techniques to make decisions.

1.1.2 Genomics Sequencing and Annotation

Genomics is a scientific discipline that deals with sequencing, and analyzing genomes [Hrmova and Fincher, 2009]. The main goal of genomics is to identify an organism’s complete DNA sequence, including all of its genes. This organism can be an animal, a fungus, a plant, a virus etc.

Today, genome sequencing is growing fast and its cost is dropping at the same time as we see in Figure 2. Modern sequencing technology has enabled an inexpensive, rapid production of whole genomes. Millions of genomes such as human genome have been sequenced and over 2.7 billion bases are available in public databases [Kent et al., 2002].

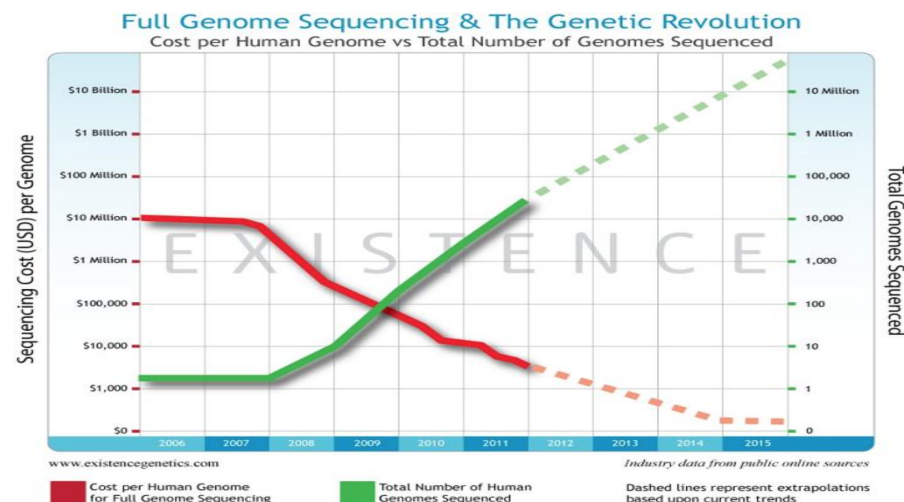


Figure 2: This chart illustrates that the number of genomes has speedily increased, while the costs of sequencing a whole human genome has decreased. Credit: How DNA sequencing works and how it became affordable (<http://www.extremetech.com/extreme/151133-the-quest-for-the-1000-genome/2>)

Once a genome is sequenced, there is a need to understand what these sequences mean since each genome has all the information required for building and maintaining that organism. The process of attaching identities and functions to sequences is called *genome annotation*. As a result of genome annotation, massive amounts of data are recorded in several databases; and understanding this huge amount of genomic data is quite a challenge [Nielsen et al., 2010]. To be focused, mostly scientists prefer genome assessment at the level of an exon, a gene, and a biochemical pathway according to [Kent et al., 2002]. The main purpose of this assessment is to find the key features of the genome, particularly the genes and their products [Stein, 2001].

1.1.3 Whole Genome Visualization Tools

With the exponential and rapid development of genomic data, the need for visualization tools for presenting genome annotation is becoming more pressing. As stated previously, scientists mostly prefer genome assessment at the level of an exon, a gene, and a biochemical pathway [Kent et al.,

2002]. Consequently, there is a need for tools for the visualization of the data collected about a whole genome such as genes, proteins, annotations, and expression data.

As we mention in the previous section that the dataset representation is divided into four basic types that are **tables**, **networks**, **fields** and **geometries**. There are some common approaches that used this categorization for the visualization purposes such as genome browsers, spreadsheets, trees and networks visualization tools.

Genome browsers are valuable tools for many purposes such as identifying genes, locating genes, and observing their detailed information in the genomic context. In these browsers, the sequence features are displayed as visual elements in tracks and features are aligned with their genome coordinates. For each view, users can display part of the genome and zoom in and out to control the amount of the displayed data such in UCSC browser (<http://genome.ucsc.edu>) [Kent et al., 2002]. Ensembl (<http://www.ensembl.org>), JBrowse (<http://jbrowse.org>), and GBrowse (<http://www.gbrowse.org>) are some of the most common browsers used to visualize genomic data.

Tree layout is commonly used in the applications that need an efficient way to present the phylogenetic context. By using the tree layout, a better understanding for organism evolution patterns in gene and genome can be achieved [Soltis and Soltis, 2003]. Interactive Tree Of Life (ITOL) (<http://itol.embl.de>) is one of the well-known systems adapted this kind of visualization where the phylogenetic tree is displayed as rooted, unrooted, or circular tree.

With massive datasets, the interpretation and analysis of relationships between biological molecules becomes a big challenge. One of the best ways to express these relations is using **networks** where nodes (circles) represents the elements and the edges represent the interaction between them [Pavlopoulos et al., 2008]. These relationships could be physical relationships such as protein interaction, gene-gene interaction, having a shared protein domain or family, regulatory transcription factors that regulate the genes, and functional interaction. Today, there are many 2D graph visualization tools that used to visualize biological interactions. These tools are varying on their user friendliness especially when thousands of circles with many connections have to be visualized and explored [Pavlopoulos et al., 2008]. One of the most common network visualization tools these days is *Cytoscape* (<http://www.cytoscape.org>) which is an open source network visualization and analysis tool for bioinformatics. *Cytoscape* enables users to display bioinformatic networks through the web with the ability to visualize a large-scale network that has thousands of nodes and edges with a variety of node shapes and colors [Pavlopoulos et al., 2008] [Gehlenborg et al., 2010].

Spreadsheets is another way to represent the data, in which the information is textual in nature. Each row captures the information about a gene/genome such as identifiers, descriptions, or sequences. For *tabular* (or spreadsheet) data representation in biology, there are some efforts that are done. Most of the efforts that are done was to find a universal format to exchange **omics** data and metadata; which is one of the challenges in genomic data exchange between heterogeneous systems [Sansone et al., 2008]. One of the most common tools is ISA-TAB which is stand for Investigation-Study-Assay (ISA) tab-delimited (TAB) (<http://www.isa-tools.org/>). According to [Jameson et al., 2011], ISA-TAB is initially developed as a standard in typical tabular format to describe data from diverse forms of **omics** experiments. This standard is mainly created to allow

easy data exchange of metadata in the future between public resources. These data particularly is "omics-based" experiments such as genomics, proteomics etc. In ISA-TAB, data are organized in tab delimited file which can be viewed as textual table representation using ISA-TAB web viewer for ISA-Tab files that found in (<https://github.com/ISA-tools/ISATab-Viewer>).

1.2 Motivation

Many organisms have been completely sequenced today. Consequently, there is high demand for visualization tools for the data collected about a whole genome such as genes, proteins, annotations, and expression data. Those tools are essential to understand what a gene and gene product do. Since the genes role can be noticed by watching some patterns, visualization in tabular (spreadsheet) format is one of the important methods to organize and identify these patterns. There are many special tools that created for visualizing the gene structure in non-tabular format such as FancyGene [Rambaldi and Ciccarelli, 2009] and GECA [Fawal et al., 2012]. However, they have limited visualizing options. For example in FancyGene there is rich annotation information for the gene model but it only shows a single gene. In contrast, GECA displays many genes but the features are limited. There is a need for visual representation for all the genes in the genome to be in a single place.

In our research, we put the annotated information about the gene and its encoded protein in a single place, and gather them in tabulate view to allow the user to explore patterns much easier. In our system, the protein feature that are associated to the gene is presented in single row with the gene that belongs to while this feature is highlighted on the top of gene models on FeatureStack [Frech et al., 2012]

1.3 Research Challenges

While data visualization is increasingly important, its design is challenging since it involves considering many aspects related to the problem domain and the end-users. According to [Krzywinski, 2013], the human ability to read visually is extremely sophisticated, so applying some principles will be required to have useful and usable visual system. Some of the important principles that considered in this thesis are stated in Chapter 2: Section 2.5.

In addition, usability is one of the big challenges in visualization since many powerful systems are available but the big question is: are they useable or not. Currently, researchers have started to act seriously and pay more attention to usability analysis. In [Wang Baldonado et al., 2000] paper, there are some guidelines that should be considered in designing usable visualization systems, which will be discussed in Chapter 2: Section 2.5.2. Since the targeted end-users are the most important thing while designing our system, we try to follow the common visual designs as there are metaphors familiar to users.

1.4 Research Contributions

The general goal of the thesis is to develop a web-based system that provides the visualization of genome annotation data in single spreadsheet. In order to achieve this, the contributions of this thesis are the following:

- Analyze different annotation files about different fungal genomes with different file formats from the AspGD database (www.aspgd.org) and InterPro (<http://www.ebi.ac.uk/interpro>). Then, extract the key features that required to display the following: gene model, protein domains, genes location in the chromosome and GO Slims availability in the genome.
- Implement a new approach of presenting genome analysis data, which is HTML5 visual spreadsheet to express genome annotation data. The system can be accessed from any device with a web browser.
 - In the CGene system, we lined up the gene structure with protein domain architecture in a single row of the spreadsheet. Each genome, that may have more than *10 000* genes, will be presented in a single spreadsheet which allows the scientist to explore genomic patterns easily. The visual image of a gene model for each genes in the genome is given in Scalable Vector Graphics (SVG) format. Any single image can be zoomed in/out without losing image details and can be downloaded as SVG file. In addition, the protein features are presented in the row with the gene. The protein domain architecture is also presented in SVG format.
 - In the CGenome system, we lined up different genomes in which each row in the spreadsheet represents a single genome. In each row of the spreadsheet we display the following in SVG format: Gene location, Go Slims bitmap and Go Slims histogram.

1.5 Organization of Thesis

The thesis document is organized the follows:

- In Chapter 2, we present some background basics that are needed to ease the understanding of the developed system. The background includes some biology, bioinformatic basics and principles for visualization. This chapter also includes some of the examples about the available visualization systems that used to display genomic context data.
- In Chapter 3, a detailed description of *CGene* and *CGenome* systems, pronounced See-Genes and See-Genome respectively, is presented. The details include: system requirements, system analysis, system construction and system user interface.
- Chapter 4 has the conclusion. This chapter summarizes the contributions, states limitations, and makes some suggestions for future work.

Chapter 2

Background and Basic Concepts

2.1 Basic Concepts of Biology

2.1.1 Cell, Nucleic Acids and Amino Acids

A cell is the basic building unit of creatures and living species. The human body is made of trillions of cells which come in many different varieties with many different functions. Each of the cells has the complete human genome on its nucleus, which composes of 23 pairs of chromosome. Each chromosome have a collection of DNA(Deoxyribonucleic acid) molecules that encode the genetic code which used in building and functioning of the human being and the different organisms [Majoros, 2007].

While DNA is found in the nucleus of *eukaryotic* cells, *prokaryotic* cells lack a membrane-bound nucleus. As shown in Figure 3, each chromosome has a collection of DNA double helix stranded molecule that determined in 1953 by Watson and Crick. This determination opened the way to understand most important biological process more deeply. DNA composes of sugar and phosphate backbone and four different bases {ATCG}: Adenine (A), Thymine (T), Cytosine (C) and Guanine (G). The pairing between these bases called *Watson-Crick complementarity* in which the bases are represented in the following rule: $C \Leftrightarrow G$, $A \Leftrightarrow T$ ($A \Leftrightarrow U$ in case of Ribonucleic acid (RNA)) [Watson et al., 1953].

Small parts of DNA strand are called *genes*. Genes control a hereditary characteristic and determine the physiology of the organism. The strand that contains the gene is called the *template* (or *sense strand*) for that gene while the complementary strand is referred as the *reverse strand* (or *antisense strand*).

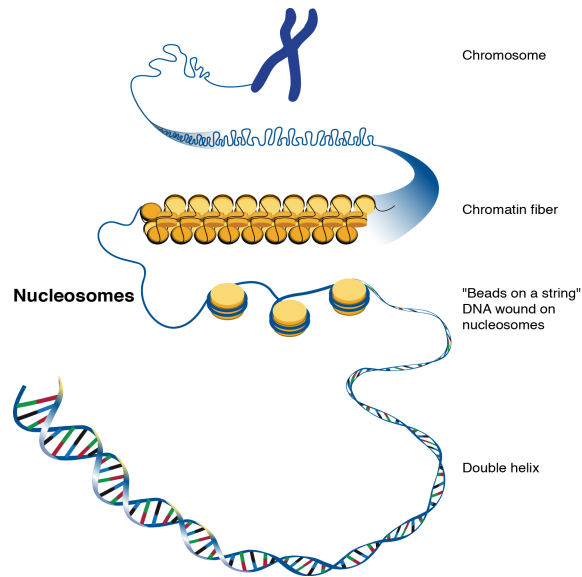


Figure 3: Illustration of chromosome that have DNA double helix stranded molecule. (Credit: Darryl Leja, The National Human Genome Research Institute (NHGRI))

A *protein* is a large biological molecule, which its primary structure is the sequence of one or more long amino-acid residues. A protein sequence is represented as a chain of 20 English alphabet letters {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y} where each letter represents an amino acid as shown in Table 1.

Amino Acid	Alphabet	Amino Acid	Alphabet
Alanine	A	Methionine	M
Cysteine	C	Asparagine	N
Aspartic acid	D	Proline	P
Glutamic acid	E	Glutamine	Q
Phenylalanine	F	Arginine	R
Glycine	G	Serine	S
Histidine	H	Threonine	T
Isoleucine	I	Valine	V
Lysine	K	Tryptophan	W
Leucine	L	Tyrosine	Y

Table 1: Proteins' 20 amino acids

2.1.2 Central Dogma of Molecular Biology

The central dogma of molecular biology deals with the detailed residue-by-residue transfer of sequential information. It states that such information cannot be transferred from protein to either protein or nucleic acid. [Crick et al., 1970]

In 1958, the central dogma of molecular biology was first stated by Francis Crick, and then it was

re-stated and published in a Nature paper 1970 [Crick et al., 1970]. It is a framework for describing the genetic information flow from DNA to messenger RNA (mRNA) molecules then to proteins. The transfer from DNA to mRNA is done through a process called *transcription*, in which a fragment of DNA sequence is copied into RNA by the enzyme RNA polymerase. Then protein is synthesized by using the information in mRNA as a template through an irreversible process called *translation* as illustrated in Figure 4 as a summary of *central dogma of molecular biology*.

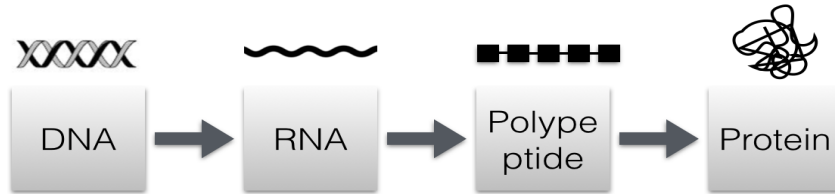


Figure 4: A summary of Central Dogma of Molecular Biology

During the **transcription** process, the enzyme *RNA polymerase* is used to copy a particular segment of DNA into RNA. Then, to produce *messenger RNA* (mRNA) that will act later as template for protein synthesis, the RNA nucleotides are joined by strong phosphodiester bonds. Before the translation process, the *transcript* (or *pre-mRNA*) is spliced by large and complex molecular machine called *spliceosome*. Its job is to remove portions of RNA known as *introns* from the sequence and keeps the *exons* as shown in Figure 5.

In **translation**, *mRNA* is decoded by a *ribosome complex* which moves from codon to codon along the mRNA to produce a chain of amino acids. After terminating translation, a complete polypeptide is released from the ribosome which will later fold into an active protein [Majoros, 2007].

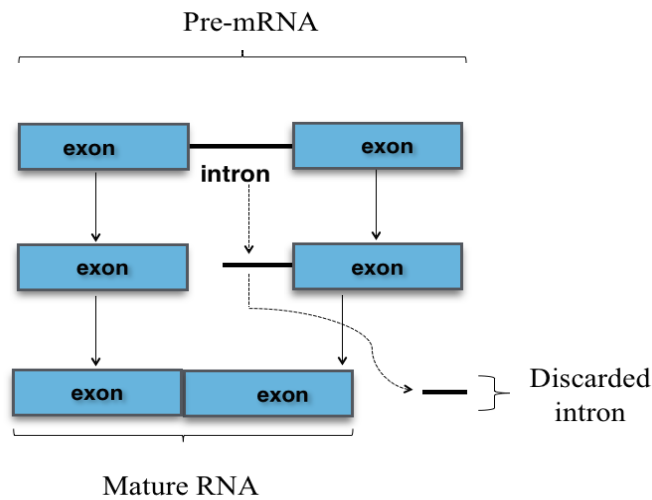


Figure 5: Removal of introns during the splicing process to produce a mature mRNA in eukaryotic gene

2.2 Gene and Gene Model

2.2.1 Genes, Exons, Introns and Coding Segments

As mentioned before in translation process at Section 2.1.2, the continuous sequence of bases in RNA transform later to protein. However, not all these sequences convert to protein. Specifically, this means not all *exons* that found in the gene are translated to protein, only the specific part called the *coding region* (or *coding segment CDS*) that encode the actual recipe for gene's product. The coding segment usually starts with a *start codon* (or *translation start site*) and terminates with a *stop codon* (or *translation stop site*). Lying in between, there are exons that only translated to protein which located apart from the *five prime* region and *three prime* region of the start and stop codons.

2.2.2 Gene Model

A gene model usually is drawn to reflect the location of introns and exons relative to the genomic sequence [Majoros, 2007]. In addition, it shows where the RNA polymerase started transcription and the location of the starts and stops of translation as seen in Figure 6. In this figure, a general *eukaryotic* gene model and *A acidus CBS 106.47* fungi gene model are illustrated. Both gene models in the figure have multiple exons and introns.

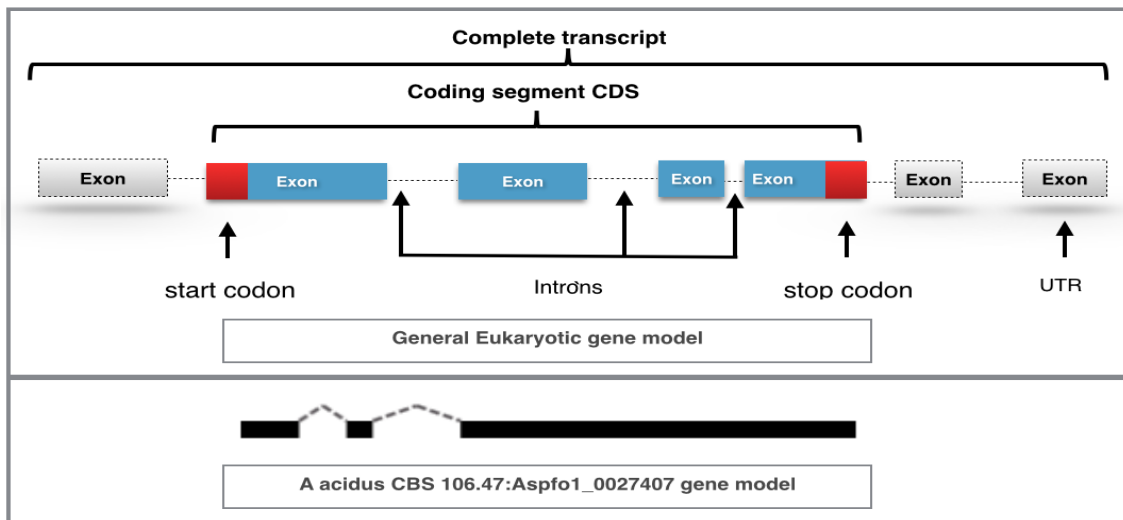


Figure 6: Gene Model in eukaryotic gene. It illustrates the coding segment (CDS) that contains the exons connected by introns. The grey rectangles are the untranslated regions (UTR). The gene model below is for one of the genes in *A acidus* CBS 106.47 fungi.

It is worth to mention that in *eukaryotic* gene, some parts of the original gene, called *introns*, are removed during the splicing process in order to complete the translating process. In case of *prokaryotic* gene, it is simpler since it does not have introns. Each intron begins with donor splice site and end with acceptor splice site. Codons work as *signals* in DNA sequence and they have

specific codes. ATG is the code for *start codon* while TAG, TGA, or TAA are the codes for *stop codons*. GT and AG are the codes for *splice donor site* and *acceptor donor site* respectively. However, the identification for these codons in DNA sequence is not distinct which means not every presence of ATG is start codon and not each AG is acceptor donor site [Majoros, 2007].

2.3 Gene Annotation and Gene Ontology

2.3.1 Gene Annotation

Over the years, the amount of the genome sequencing data has expanded significantly. As we see in Figure 7, the number of Genome Sequencing Projects in Genome Online Database (GOLD) has been increased generally from 2007 to 2014. The figure also shows the growth of sequence databases and annotation since 1982. Sequence discovery helps for identifying new genes, observing chromosome organization and structure, and for performing comparative genomics. These data may lead to advance in fields such as medicine, agriculture etc.

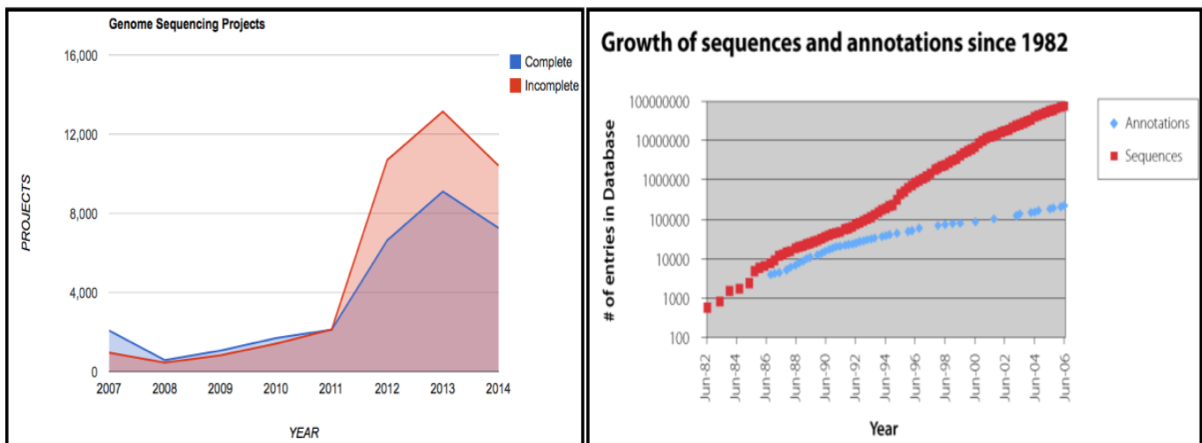


Figure 7: The graph in the left illustrates the increased amount of Genome Sequencing Projects in Genome Online Database (GOLD) from year 2007 to 2014. The graph in the right represents the growth of sequence databases and annotation since 1982. The numbers are taken from of the databases Genbank(NCBI) and Swissprot(EBI). Credit: (Genome Online Database (GOLD) and (Folker Meyer, Argonne National Laboratory).

As much as this data are discovered, it does not become useful for biologist until its components are identified and analyzed [Rouzé et al., 1999]. The process of attaching identities and functions to sequences is called *genome annotation*. *Genome annotation* is sub-field of genome analysis, which deals with computational processes in genome sequences.

The main purpose of genome annotation is to find the key features of the genome, particularly the genes and their products [Stein, 2001]. After the sequences are discovered, some steps are done on the sequences to achieve the extracted results. This is done by following schematic data flow, as shown

in Figure 8 from [Koonin EV, 2003], to produce these key features. The steps start with identifying genome elements that encode genes, then predicting some functional elements. These steps are integrated with statistical gene prediction, structural features prediction, and database searching. The database searching for sequence similarity can be done in general databases such as NCBI non-redundant database or specialized databases such as Pfam and SMART.

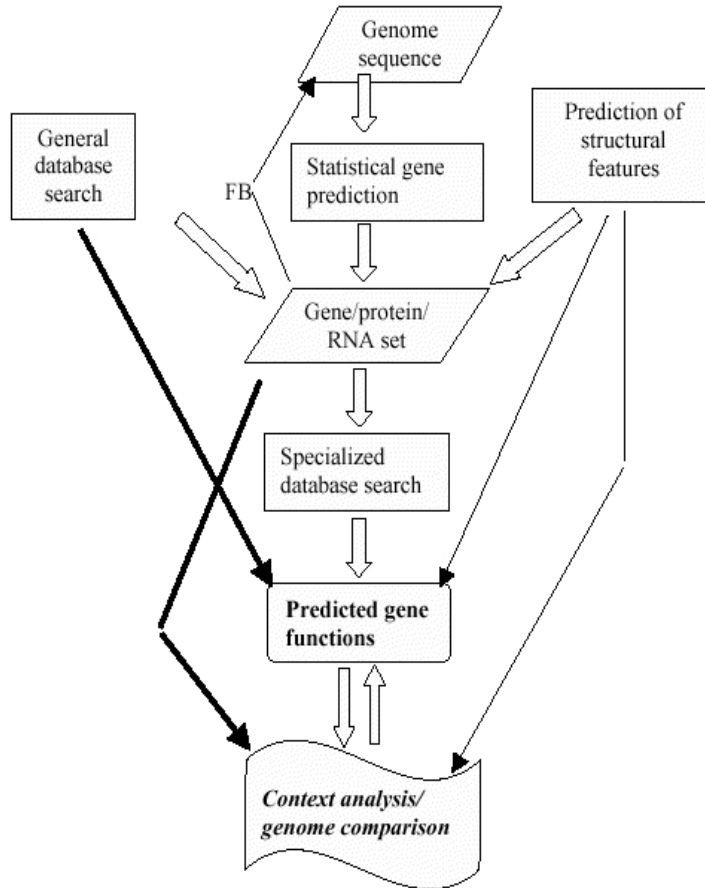


Figure 8: Generalized flow chart illustrates Genome Annotation procedure. The steps are integrated with statistical gene prediction, structural features prediction, and database searching to produce genome key features. Credit: ([Koonin EV, 2003] paper. For further reading about the steps in the chart please visit: (<http://www.ncbi.nlm.nih.gov/books/NBK20253/>))

Many computational tools have been developed in order to help in generating genome annotation. One of the most basic and common tools is Basic Local Alignment Search Tool (BLAST)(<http://www.ncbi.nlm.nih.gov/BLAST/>), which allows the user to query protein or nucleotide sequence and find its similarities against a large number of sequences. The genome is annotated based on that similarity. However, doing the whole process in BLAST then waiting the results is time consuming especially if the number of sequences is huge [Koonin EV, 2003]. Consequently, it is essential to use some level of automation in genome annotation by using some softwares that do the query

process then organize the results. After that, the expert is responsible to assign gene functions manually or automatically such as GeneQuiz (<http://www.sander.ebi.ac.uk/genequiz/>) project, which is general public automatic system for genome analysis. In GeneQuiz, results evaluation and function annotation initiation are done automatically by an expert system after similarity search is finished [Koonin EV, 2003].

2.3.2 Gene Ontology

As we discussed in the previous section, many genes are identified. Consequently, we need to assign names and functions for them. It is common in biology to have more than one word to describe the same phenomenon such as *citric acid cycle*, which also called *tricarboxylic acid cycle* (TCA cycle) and the *Krebs cycle*. The need for uniform vocabulary and naming structure for genes, its products and their relationships became essential. For this reason, Gene Ontology (GO) project has been developed as cooperative effort between three databases: FlyBase (<http://flybase.org>), the Mouse Genome Database (MGD) (<http://www.informatics.jax.org>), and the Saccharomyces Genome Database (SGD) (<http://www.yeastgenome.org>) [GeneOntology, 2014].

The main role of this GO consortium (<http://www.geneontology.org>) is to create a systematic language that has consistent vocabulary and description for genes and its products across different databases [Gene Ontology Consortium and others, 2008]. Each GO term is given a name and unique number that consists of seven digits in the form of GO:nnnnnnn such as **GO:0005102** for **receptor binding**.

The GO terms are divided into three aspects based on the biological domains, which are Molecular Function, Biological Process, and Cellular Component. Molecular Function explains activities that happen at the molecular level such as catalysis or binding activities. Biological Process explains the higher-level activity that the molecular function provides such as DNA metabolic process. Cellular Component describes the cell and its environment such as chromosome, ribosome, or cytoplasm.

GO terms can be assigned to a UniProtKB entry in electronic or manual way. Each term is associated with a specific reference to describe what type of analysis done to identify the association between the GO term and the gene product. This reference called evidence code which each annotation must has to indicate how the annotation to a specific term is supported [GeneOntology, 2014]. For example, IEA is a code for Inferred from Electronic Annotation which is used when annotation transferred from database or derived from computation.

These terms are not organized in a tree structure, but instead they organized in a directed acyclic graph, in which one term can have more than one parent term. In Figure 9, we see an example of *regulates* and *part_of* relationships between Molecular Function and Biological Process [Gene Ontology Consortium and others, 2010].

GO terms are increasing and the relationship between the three branches has been expanded too. In Table 2, a summary of the GO content in the years 2007 and 2010 is presented. This table shows that terms are increased in biological process from 13 916 to 17069 terms, in molecular function from 7878 to 8637 terms and in cellular component from 2007 to 2432 terms.

There are browsers for GO terms such as QuickGO, which is a open-source web-based browser

for Gene Ontology terms and annotations searching. This browser is developed by the *UniProt-Gene Ontology Annotation (UniProt-GOA)* project (<http://www.ebi.ac.uk/QuickGO/>). In QuickGO, as shown in Figure 10, a GO term can be viewed alongside with its definition, synonyms, child terms, etc.

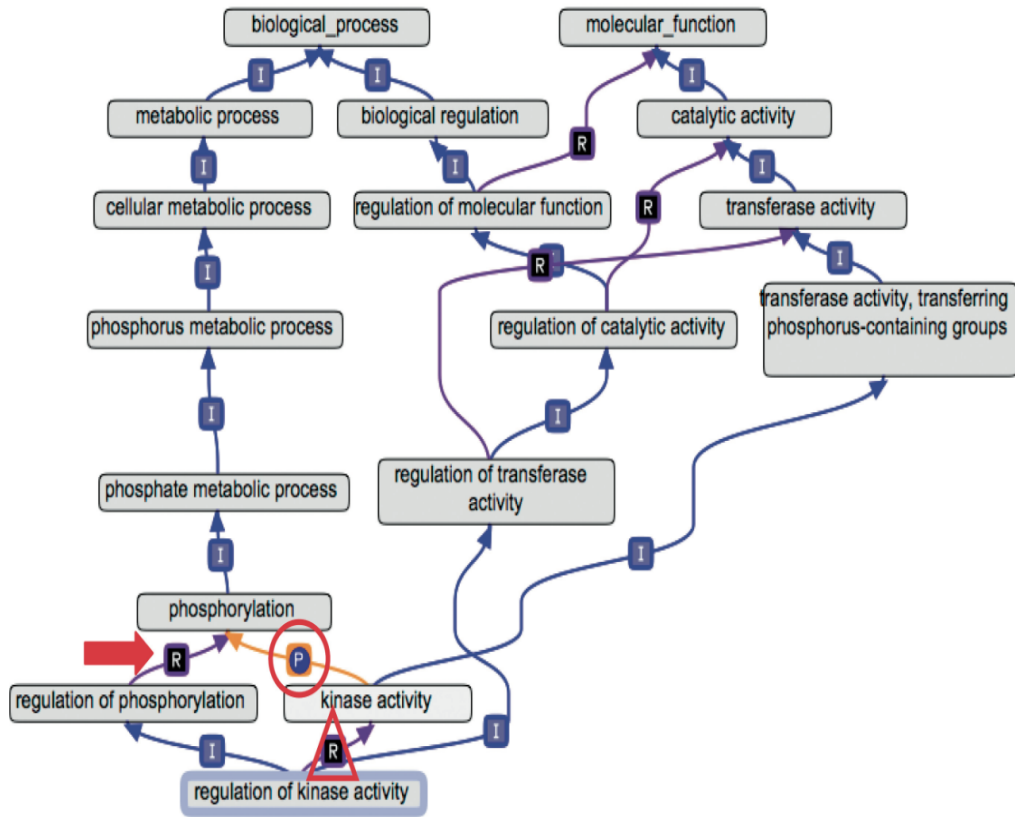


Figure 9: Graph shows *regulates* and *part_of* relationships between Molecular Function and Biological Process. Triangle indicates a *regulates* relationship and circle indicates *part_of* relationship between them. Credit: ([Gene Ontology Consortium and others, 2010])

Table 2: The status of Gene Ontology as September 2007 and September 2009, which shows increase in Go terms between 2007 and 2009. It shows that terms are increased in biological process from 13916 to 17069 terms, in molecular function from 7878 to 8637 terms and in cellular component from 2007 to 2432 terms. Sources: [Gene Ontology Consortium and others, 2008] and [Gene Ontology Consortium and others, 2010].

Gene Ontology status		
	Gene Ontology status in 2007	Gene Ontology status in 2009
Biological process terms	13 916	17069
Molecular function terms	7878	8637
Cellular component terms	2007	2432
Sequence ontology terms	1305	1603
Annotation datasets ^a	35	52
Annotated gene products		
Total	3 34 7495	44 545 253
Electronic ^b	3 128 309	43 655 159
Manual	2 19 186	890 094

Note: [Gene Ontology Consortium and others, 2010] ^a: Most datasets represent single species.

^b: Annotations using the IEA (inferred from electronic annotation) evidence code.

The screenshot shows the QuickGO web browser interface for the GO term GO:0003774. The top navigation bar includes 'QuickGO', a search box with 'GO:0003774', and buttons for 'Search!', 'Web Services', 'Dataset', and 'Term Basket: 0'. Below the navigation bar are tabs for 'Term Information', 'Ancestor Chart', 'Child Terms', 'Protein Annotation', 'Co-occurring Terms', and 'Change Log'. The 'Term Information' tab is active, displaying the following details:

- ID:** GO:0003774
- Name:** motor activity
- Ontology:** Molecular Function
- Definition:** Catalysis of movement along a polymeric molecule such as a microfilament or microtubule, coupled to the hydrolysis of a nucleoside triphosphate.
- GONUTS:** [GO:0003774 Wiki Page](#)

Below the term information, there are tabs for 'Annotation Guidance', 'GO Slims', and 'Cross-references'. The 'GO Slims' tab is active, showing a table of GO slim names and their counts:

GO slim name	Count of all terms
goslim_metagenomics	119
goslim_candida	89
goslim_pir	462
goslim_plant	100
goslim_aspergillus	85

Figure 10: Gene Ontology terms and annotations from *QuickGO* web browser, where GO term can be viewed alongside with its definition, synonyms, child terms, etc.

2.3.3 Go Slims

It is difficult to handle all terms in the gene ontology, especially for biologists who focus in specific species or who do not need a lot of details about the terms. Consequently, the need for cut-down versions of the GO ontologies became necessary. *Go Slims* terms are a sub-list of whole GO terms that give general outline of the ontology content without many details. GO Slims are useful for having a general classification of gene products for a genome, microarray, or complementary DNA (cDNA) collection [GeneOntology, 2014]. There are generic GO slim, which are suitable for general purposes, since they are not species-specific. In the other hand, there are species-specific slim that are developed for specific organisms or groups of organisms. For example, there is Yeast slim which developed by Saccharomyces Genome Database (<http://www.yeastgenome.org>) and Candida albicans which developed by Candida Genome Database (<http://www.candidagenome.org>). In addition, biologists can create their own slim by contacting Gene Ontology Consortium GO helpdesk. All these subsets can be downloaded from Gene Ontology Consortium website (<http://geneontology.org/page/download-ontology>).

Below in the box is an example for GO term named *motor activity* from [GeneOntology, 2014] for Aspergillus slim that developed by Aspergillus Genome Data (<http://www.aspgd.org>), which our research group is focus on, and is used in this research in CGenome visual spreadsheet.

```
[Term]
id: GO:0003774
name: motor activity
namespace: molecular_function
def: "Catalysis of movement along a polymeric molecule such as a microfilament or microtubule, coupled to the hydrolysis of a nucleoside triphosphate." [GOC:mah, ISBN:0815316194]
subset: goslim_aspergillus
subset: goslim_candida
subset: goslim_metagenomics
subset: goslim_pir
subset: goslim_plant
subset: gosubset_prok
is_a: GO:0016787 ! hydrolase activity
```

2.4 Protein Structure, Protein Domain and its Architecture

2.4.1 Protein Structures

Generally, protein structure is divided into three or four levels structure based on the references that categorized the protein structure. According to [Rangwala and Karypis, 2010], protein structure is divided into four levels which are primary structure, secondary structure, tertiary structure, and

quaternary structure. **Primary structure** is the main level structure of the protein which is the Amino Acids Sequence. There are mainly 20 different amino acids that form the building protein sequence blocks which stated previously in Section 2.1.1:Table 1. These amino acids is linked by peptide bonds to form the sequence chain. The **secondary structure** composes the second level of protein structure. It consists of some regions, which form themselves independently from the rest of the protein into repeated patterns that are occurring in the protein. The most common forms of the secondary structure are α -helices which has a coil-like structure, and β -sheets that contains parallel strands of residues. These regular structures can have critical role in biological function. **Tertiary structure** is the global 3D shape that represents the protein in three dimensional crystal coordinate. This shape contains of a variety arranged secondary structure elements that bonds together. **Quaternary structure** represents the interaction between various polypeptide chains which happens because of the "non-covalent interactions between the atoms of the different chains" [Rangwala and Karypis, 2010].

2.4.2 Protein Domains

The word domain denotes a certain region in the protein such as domain family. According to [Morrison, 2013], protein domains are high-level representation for protein parts. These domains can fold independently from the rest of the protein sequence into a tertiary (three-dimensional) structure. A protein sequence can have many domains, and most of them have a specific function or participate with other domains in particular function on their protein.

An example of protein domain is BAR (Bin-Amphiphysin-Rvs) domains, which shaped as banana shape and binds to membrane through its concave face, are extremely conserved protein dimerization domains. These domains occur in several proteins involved in membrane dynamics in a cell [Peter et al., 2004]. A single protein domain can be structured into a three-dimensional structure. To represent protein sequence that has multiple or single domain, we can use protein domain architecture as represented in Figure 11. In this figure, a 3D crystal structure of *Endophilin BAR* domain and the domain architecture representation for the same domain are illustrated.

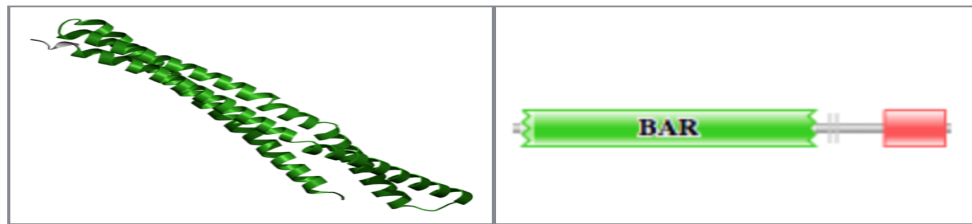


Figure 11: In the left image is the 3D crystal structure of *Endophilin BAR* domain, while the right image is the domain architecture representation for *BAR Domain* in *Endophilin-B1* from **Pfam DB**. Credit: both images are taken from **Pfam DB**

2.4.2.1 Protein Domain Databases

There are many domain databases (DBs) that store protein annotation resources. These DBs differ on the way that they are constructed by [Morrison, 2013]. Below is a list of the most common protein DBs each associated with URL for further details.

- **SCOP:** it is stand for Structural Classification of Proteins. It uses the information that gathered and experimentally determined from protein 3D structure. For more details: <http://scop.mrc-lmb.cam.ac.uk/scop/>.
- **CATH:** it is a classification DB of protein structures which are downloaded from the Protein Data Bank. For more details: <http://www.cathdb.info>.
- **SUPERFAMILY:** it is a structural and functional annotation DB for all proteins and genomes. It uses SCOP domains and assigns them to protein sequences by matching them using Hidden Markov Models (HMMs). For more details: <http://supfam.cs.bris.ac.uk/SUPERFAMILY/>.
- **Gene3D:** it is constructed in a way similar to SUPERFAMILY, but using CATH domains instead of SCOP domains. For more details: <http://gene3d.biochem.ucl.ac.uk/Gen3D/>.
- **Pfam:** it is a large DB that has many protein families which represented by multiple sequence alignments (MSA) and hidden Markov models (HMMs). For more details: <http://pfam.xfam.org>.
- **NCBI CDD:** it is stand for Conserved Domain Database that created by the National Center for Biotechnology Information. It consists of a group of sequence alignments and profiles that represent conserved protein domains in molecular evolution. For more details: <http://www.ncbi.nlm.nih.gov/cdd/>.
- **SMART:** it is stand for Simple Modular Architecture Research Tool. It is manually curated DBs that mainly focus on signaling and extracellular domains. For more details: <http://smart.embl-heidelberg.de>.
- **INTERPRO:** it is meta-database that combined its data from several DBs such as Pfam. For more details: <http://www.ebi.ac.uk/interpro/>.
- **UniPort:** it is stand for Universal Protein Resource, which has collection of protein sequence and functional information. For more details: <http://www.uniprot.org>.
- **ProDom:** it is automatically created DB that gathered from Uniport For more details: <http://prodom.prabi.fr/prodom/current/html/form.php>.
- **ADDA:** it is stand for Automatic Domain Decomposition Algorithm. Its concept is similar to ProDom since its data are gathered and clustered form several Dbs. For more details: url <http://ekhidna.biocenter.helsinki.fi/sqgraph>.

2.4.3 Protein Domains Architecture

Domain architecture, or domain arrangement, refers to the domains and their arrangements in individual protein. The direction of these domains goes along the amino acid chain from N- to C-terminal [Morrison, 2013]. The design of the shape and the colour of each domain differ depending on the system that built the architecture as seen in Figure 12. In this figure, protein domain architecture from SMART, ProDom, InterPro and Pfam is illustrated. Most of the domains in those systems use the rectangular shape with or without rounded or jagged edges. Generally, the rectangular shape is the most common shape that used to represent the domain, however some systems use different shapes for different domain families such as SMART system DB. Other systems such as Pfam use rectangle for the domain with different colour for different domain families. The colour is also varying between the systems and some of them use patterns such as ProDom. In Appendix B: Table 12, there is a descriptive catalog for Pfam domain architecture graphics design. For more examples about protein domain architecture in different systems see Appendix B.

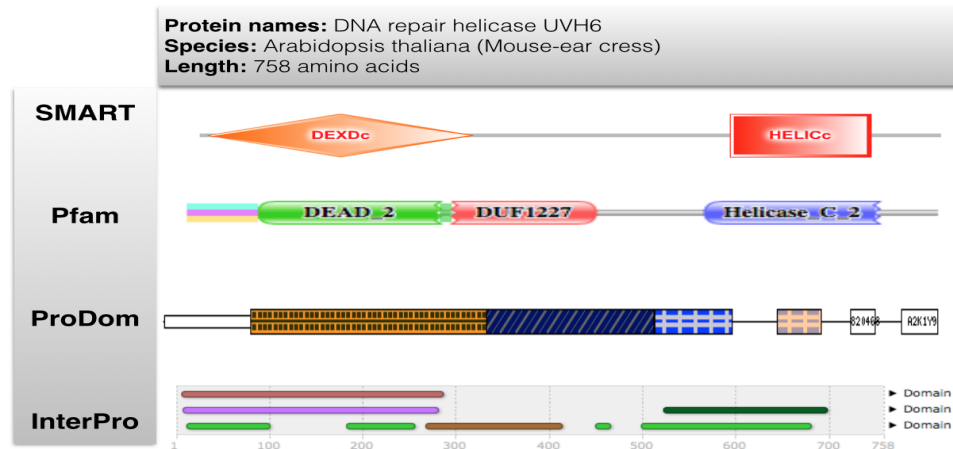


Figure 12: Domain architecture representation for *DNA repair helicase UVH6* protein which belong to Arabidopsis thaliana (Mouse-ear cress) Organism. It is gathered from four different databases: SMART, ProDom, InterPro and Pfam. The rectangular shape is the shape that used to represent the domains in ProDom, InterPro and Pfam while SMART uses different shapes. The color is varying between the systems while ProDom uses colors and patterns for domain architecture displaying.

2.5 Principles of Biological Data Visualization

In some situations the ability to display data in visual format is difficult especially with a massive data such as biological datasets. Consequently, the computer-based system is required to save the human efforts. For computer-based visualization to be effective tool, some visualization principles should be considered during the visualization design such as visual representation understanding and multiple views system building principles. In the next two Sections 2.5.1 and 2.5.2 these principles

will be discussed.

2.5.1 Visual Representation Understanding

In order to understand the visual representation, we need to analyze three important questions first, which are: **what** is the data type that we are willing to show, **why** are users looking at this data, and **how** to show the data [Munzner, 2014]. To answer these questions clearly, we need to divide the analysis to four levels which are domain situation, abstraction, idiom and algorithm. These three questions are found inside the four levels as seen in Figure 13 [Munzner, 2014].

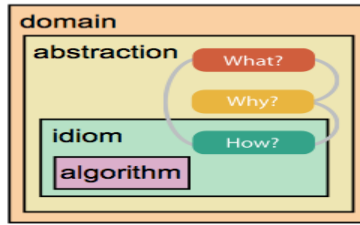


Figure 13: Illustration of A Multi-Level Typology of Abstract Visualization Tasks. These levels are needed to answer three important question in visualization, which are: **what** is the data type that we are willing to show, **why** are users looking at this data, and **how** to show the data. (Credit: [Brehmer and Munzner, 2013])

Each system has its own targeted users, and a visualization tool has to be more than just graphical demonstration of information when its users are scientific researchers [Wong, 2012]. The objectives and researchers’ expectations need to be considered while designing the system. After knowing the domain that the system is targeting, there is a need to think deeply about what does the data that need to be displayed means in the real world and what is the type of this data [Munzner, 2014].

The **semantic**, which care about what does the data mean, is very crucial in visualization in order to see how to group or organize this data. In addition to the semantic, the structural or mathematical data **type** is important to perform visualization analysis. According to [Munzner, 2014], there are different categories for the dataset type which can be **tables**, **networks**, **fields** and **geometries**. **Table** data type where the data is represented as a combination of rows and columns and their intersections is a value in a single cell. **Network** where there is a relationship between two or more items. The data is represented as nodes (or vertex) and multiple links (or edges) connecting these nodes based on the relationship between them. In **field** dataset type there are values in cells, but these cells have measurements from continues domain where a new measurement can be taken between two available cells. These measurements could be pressure, temperature, speed, density etc. For example, if a scan of human body is made and spread through a volume of 3D space. The density of the tissue at the sample could be taken apart for a coarser grid, or closer with a higher resolution grid of cells [Munzner, 2014]. In table and network data types the value is discrete, and showing values in between them is not a meaningful concept. The **geometry** dataset type has information about “the shape of items with explicit spatial positions” [Munzner, 2014]. These

items could be points, one-dimensional lines, 2D surfaces, or 3D volumes. Each one of these dataset types may have a combination of types of data which are **attributes**, **links**, **items**, **positions** and **grid cells**. These data types describe the values and their relationships and positions. A summary of these dataset types and their types is illustrated in Figure 14. To encode the data, we need to categorize the data such by their colour, shape, specific pattern or all of them. We also need to group them even by one or combination of connection, similarity, or positions. Then the data need to be ordered by ordinal or quantitative order by saying the data size is small or its measure is 10 inches. At data abstracting stage, it is essential to translate the targeted data from a specific domain to visualization vocabulary in a process called the *visual encoding of data* in order to display it later on. After analyzing the data, an efficient computation algorithm has to be applied based on the system objectives. In the next Section:2.5.2, there will be a suggested principles of how to build a successful visualization system in biology.

Dataset Types

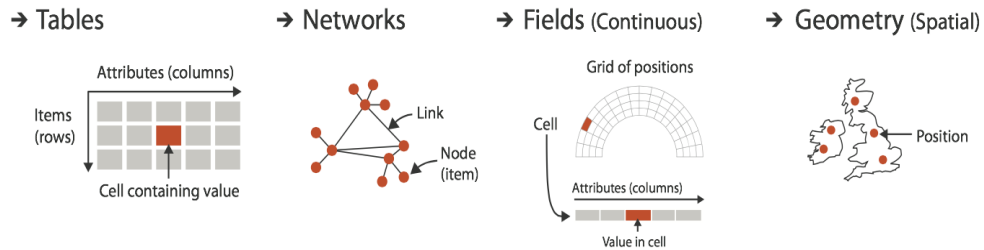


Figure 14: A summary illustration of dataset types and their types. Table and network have discrete values while fields has continuous values. Source: (<http://www.cs.ubc.ca/~tmm/talks/minicourse14/halfdaycourse14.pdf>).

2.5.2 Multiple Views

The Coordinated and Multiple Views part of study is a specific exploratory visualization technique in which the data is represented in different ways that enables the users to understand and interact with the data better. According to [Roberts, 2007], there are several reasons make the Coordinated and Multiple Views study more attractive. One of the reasons is that scientific experts are seeking to investigate complex data. These data may come from different datasets, which requires data aggregation and more analysis to generate comprehensive and useful information. The other reason is that human beings are always looking for things that they are familiar with. As a result, they will follow the system that they used to deal with previously even if it lacks the richness that other systems may offer. Therefore, it is favourable having a multiple viewing environment that shows the data in different displays to meet users requests. An example of a multiple-view system is *MulteeSum*, which is a tool for “*comparative spatial and temporal gene expression data*” as seen in Figure 15 [Meyer et al., 2010]. The figure shows that *MulteeSum* tool supports inspection and curation of datasets presenting gene expression over time.

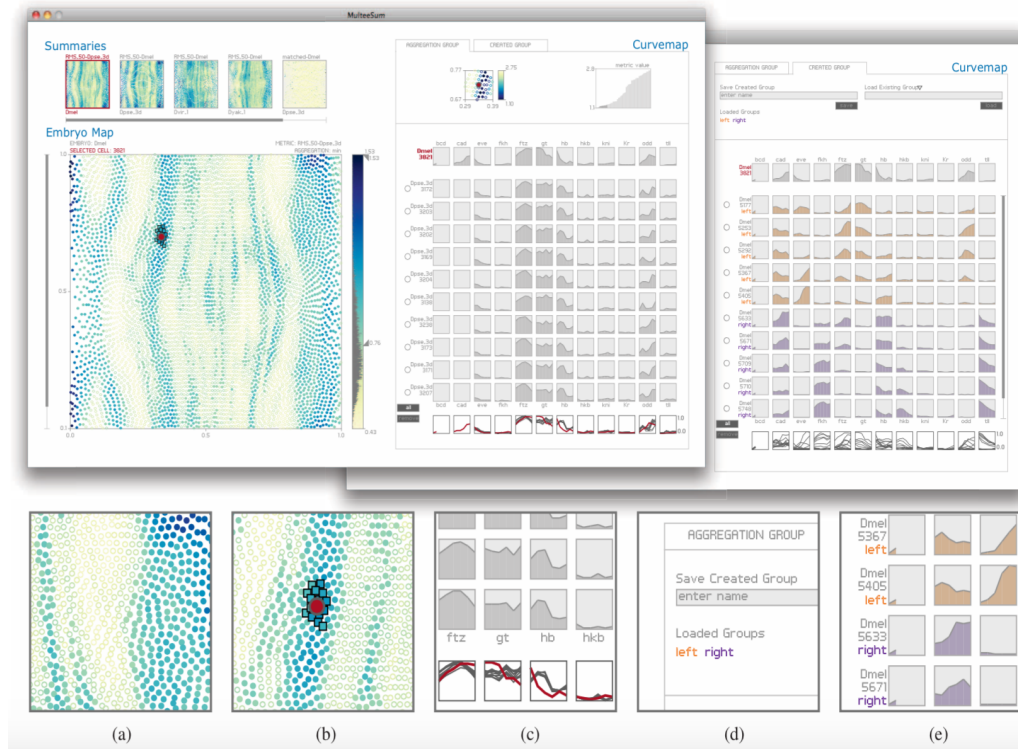


Figure 15: MulteeSum is a tool for Comparative Spatial and Temporal gene expression data. This tool supports inspection and curation of datasets presenting gene expression over time. This screenshot contains data for four species of *Drosophila*. (a) An embryo map where the summary value of a cell is encoded with colour. Circles are the filtered cells while the filled disks are the remaining cells. (b) The selected cell is coloured as red, while the other cells in the created group are represented as squares. (c) Curvemaps highlights the cell's related curves in the overlapped plots. (d) Current groups loaded into the curvemaps are given a colour. This is used when rendering a label for the groups. (e) The colour in the curvemaps is used to indicate the group members' expression profiles. Credit: ([Meyer et al., 2010]).

As any area of study, clear fundamentals should be stated as guideline or declaration for this area. Most of the demonstrations and the ideas for Coordinated and Multiple Views for Exploratory Visualization are taken from discussions and opinions that stated at Coordinated and Multiple Views Conferences. In [Roberts, 2007] paper, a seven fundamental areas are covered. These fundamentals will be declared briefly, mainly from the above mentioned paper, to explain the available suggestions about how to build an efficient *multiple-view system*.

Data Processing and Preparation: One of the core challenges for exploratory visualization is dealing with enormous amount of data. The reasons for this difficulty are that huge dataset: takes long time to process, has many relationships, are slow to explore and the data may come from different sources. Therefore, the huge dataset needs to be simplified first before displaying. In order to make to perform simplification, data should be aggregated and cleaned from any possible

errors. Most of the systems use some techniques to substitute the missing information by using average value; nevertheless the integration between these data fusion techniques and Coordinated and Multiple Views system still did not fully incorporated. For speeding up the running time for multiple views, many techniques are suggested such as parallel algorithms, data mining and grid based architectures. In addition, it is suggested that the liner algorithm should be avoided especially with large datasets. However, few researchers apply these recommendations to Coordinated and Multiple Views systems.

View Generation and Multiple Views: At the beginning of this part, view generation will be stated with mentioning its three common styles. Then the term of multiple-view and form will be declared with some examples.

One of the top questions that the Coordinated and Multiple Views system developer may ask is which style of visualization should be picked for viewing the system. Consequently, the developer will figure out how to map the information to this style. The other two questions that should be considered are: how to abstract this data to fit in the system and how the users will interact with the system.

When the users change one of the parameter, there are three reactions to this change. First possible reaction is the parameter modification will *replace* the information to new one. The second, a new window will be opened with the new information which called *replication*. The last scenario is the new information will be merged with the old one, which called *overlay*. Replication is one of CMV principle when the user can move from window to another.

Back to selecting a suitable style for interactive system, there are three categories for user-interface style. First, **menus** and **buttons** style in which the user pick the parameters from. Second, **modular approach** such as Module Visualization Environments. This style adapts the idea of giving users more flexibility by making them select the style from variety of offered modules. These modules will be linked together as multiple-view display after user selection. However, the coordination between these views is created using more command and modules. Third, using **automatic algorithms** from which the presentation is created by the users preferences. This style gives the user more flexibility for exploring the data but it increases the user task, which makes it not favourable in the developer's side.

After selecting the style, developers should think about how many forms are needed to be displayed in a single window. First we need to understand what is form and view meaning. *Form* means the way of representation such as maps, networks, charts, tables, matrix, symbols and glyphs. *Multi-form* in another hand describes the data that displayed in diverse forms. In another hand, *Multiple-Views* concept cares about the data that displayed in multiple windows beside each other in specific coordinates. These concepts can be merged in one system which called *dual view system* which may have variety of views such as: overview-detail, focus-context, difference views, master-slave and small-multiples. In *overview-detail* view, the data will be showed in one section and the details will be showed in another. If the dataset is huge, the overview should be abstracted and simplified first in order to represent the meaningful and useful information. *Focus-context* view is similar to overview-detail style in which the details will be showed in part while the context of the information

in another. *Difference views* style is used to evaluate algorithms or underline textual differences. In this style different view styles is combined to show the differences. In *master-slave* style, one of the dual view part control the other part representation. *Small-multiples* style is not strict dual since multiple and small views come beside each other. This is usually used with symbols and glyphs that shown in matrix.

Exploration techniques: Changing the data processing, filtering the displayed data, changing the display color, zooming through the data in/out, and placing the view windows in specific coordinate all these are kind of user interaction with the system. There is a need for techniques to build such kind of systems. There are two common techniques that allow data exploration which are *indirect* and *direct* manipulation.

For *indirect manipulation* technique the users deal with following: menus, buttons and sliders. These tools used to indicate the type of interaction, if it is filtering the displayed data or any other kind of interaction. In *direct manipulation* technique, users themselves can do any type of interactions. This technique has two approaches, which are *brushing* approach and *manipulators/widgets* approach. These two approaches are supported in Coordinated and Multiple Views system in diverge levels. In *brushing*, if an element is highlighted in one side, the related information in other views will be highlighted too. This method enables the user to explore the data deeply. In the second approach, each *manipulator/widget* is connected to particular elements, and the users can change element properties directly using the widget.

Coordination and Control: The way that the views are organized and structured is following some theories. These theories explain how data relationships determine the place of the view. First, if a parameter change another view this means they are related, so they can grouped together which called render groups. Second, the views are related based on the early stage such as during data preparation and clustering, during mapping and rendering, or during manipulation stage. Third, the views coordination may change based on user exploration sessions.

Tools and Infrastructure: Developing an effective Coordinated and Multiple Views system is not easy job. It required having models and tools that guide the users to create the desired system. In this section, some of the most common models and tools will be mentioned briefly to give an idea about them.

Constraint approach, *data centric approach* and *Model-View-Controller approach* are the main three architecture models that connected directly with Coordinated and Multiple Views system building. In *constraint approach*, the system building goes through many small constraints or at least one. Therefore, the constraints are identified at the beginning, then rest of the system restructure around it. For example, putting specific presentation constraints, then building the whole system based on it. *Data centralization* as concept is coming from relational database in which any change in one component is reflected immediately on the others. The last approach is *Model-View-Controller*, where the internal representation of information is separated from the ways that information is

presented to the user by splitting the software application into three communicated parts as seen in Figure 16 [Deacon, 2009].

While creating a tool for Coordinated and Multiple Views system is still challenging, there are some toolkits and languages that can help to create such system by allowing developers to insert graphical displays to their applications.

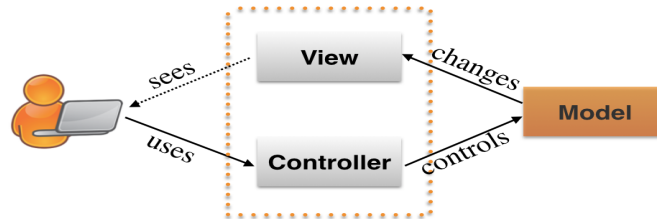


Figure 16: Model-View-Controller architecture where the software application divided to three communicated parts: model, view and controller. In this architecture, the internal representation of information is separated from the ways that information is presented to the user.

Human interface: Researchers started to pay attention to the importance of managing the users session and how they move from view to another. There are some tools, such as utilize data-flow modules, which enable storing users history session. However, these tools lack to the effective management. The screen size is an important factor that affects the user transferring between windows. The system is preferable to give the user the help by deleting unwanted windows or controlling where the windows is placed in different screens. Some systems use certain techniques to solve this issue such as zoomable user interfaces (ZUI's) and elastic windows. However, some of these techniques are not full joined with Coordinated and Multiple Views systems. Another solution to the large windows beside ZUI's is using big projected screens. Some companies propose a technology that operates multiple LCD panels to realize one large screen. However, these techniques not fully integrated with Coordinated and Multiple Views systems, and these screens still need a better navigation tools.

Usability and Perception: Usability is big issue in visualization area since many powerful systems are available but big question is: are they useable or not. Currently, researchers have started to act seriously and pay more attention to usability analysis. In [Wang Baldonado et al., 2000] paper, there are some guidelines should be considered in developing multiple-view visualization systems, which are summarized into eight rules which are: diversity, complementarity, decomposition, parsimony, space/time resource optimization, self-evidence, consistency and attention management. In diversity, multiple-views should be used when there are diversity of models, user profiles, attributes, abstractions or genres. In complementarity rule, multiple-views should be used when the views have connections or differences. Decomposition rule cares about dividing complex visualizations into smaller views for easier management while parsimony rule cares about using multiple views

slightly. In space/time resource optimization rule, developers should balance the space/time costs and benefits of presenting multiple views that displayed sequentially compared to side-by-side. Self-evidence cares about using perceptual cues that makes multiple-views relationships more obvious. Consistency rule states that developers should keep the multiple-views interfaces consistent. Attention management rule states that developers should use perceptual techniques that keep the user attentive on the right view.

2.6 Whole Genome Visualization Tools

As we mentioned previously, understanding huge amount of genomic data that resulted from sequencing and annotating genomes can be a challenge. Consequently, there is high demand for visualization tools for this data. There are some common available approaches that will be briefly discussed with some examples in this section, which are the genome browser, spreadsheets, network, circular and tree visualization tools.

2.6.1 Genome Browsers

Genome browser is a software that represents visually the enormous genomic data that gathered from specific databases for studying purpose. Since humans unable to understand the genomic data textually these browsers are designed to give users the ability to access and display sequence data in organized way. Most of the browsers are built in coordination system where the gene features aligned to the genome. For each view, users can display portion of the genome and scale by zooming in and out for controlling the amount of the displayed data. Ensembl (<http://www.ensembl.org>), JBrowse (<http://jbrowse.org>) and GBrowse (<http://www.gbrowse.org>) are some of the most common available browsers that used to visualize genomic data. For more examples about genome browsers, see the attached catalog in Appendix A about visualizing gene model in genome browsers. In this section, we will examine one of most common existing browsers which is UCSC Genome Browser.

University of California Santa Cruz (UCSC) is developed and supported by the Genome Bioinformatics Group at the University of California Santa Cruz (UCSC) <http://genome.ucsc.edu>. This browser is using the human genome assembly that is generated by NCBI. It has variety of living groups to pick from such as mammal, vertebrates, insects, nematodes, yeast etc. UCSC site has features that makes it useful browsing tool, such as easy navigating, ability of browser configuring, and data extracting. It allows the users to access the gene region and download its genomic sequence in easy-going steps. In UCSC browser, the users also can control the amount of detailed information that displayed in the browser. UCSC browser has many tracks option that enable displaying mRNA and expressed sequence tag alignments, multiple gene predictions, sites, transposon repeats and more [Kent et al., 2002].

To browse the genome, the user sets first the group, genome, assembly and position choices. Then, the user can submit it immediately, or configure the available tracks, or can add some custom

tracks by providing annotation data in some file formats such as GFF or BED format; then the user can submit the configuration to the site. Below in Figure 17 is an example that shows Rat genome on UCSC Genome Browser. In this figure, a small portion of the rat genome is displayed where the user can zoom in/out for more or less details from chromosome called chr1. There are only two chosen tracks to be displayed, which are *Other RefSeq* and *RefSeq genes*. *Other RefSeq* track shows the known protein and non-protein coding genes for organisms other than rat. *RefSeq genes* track shows known rat protein and non-protein coding genes taken from the NCBI RNA reference sequences collection (RefSeq).

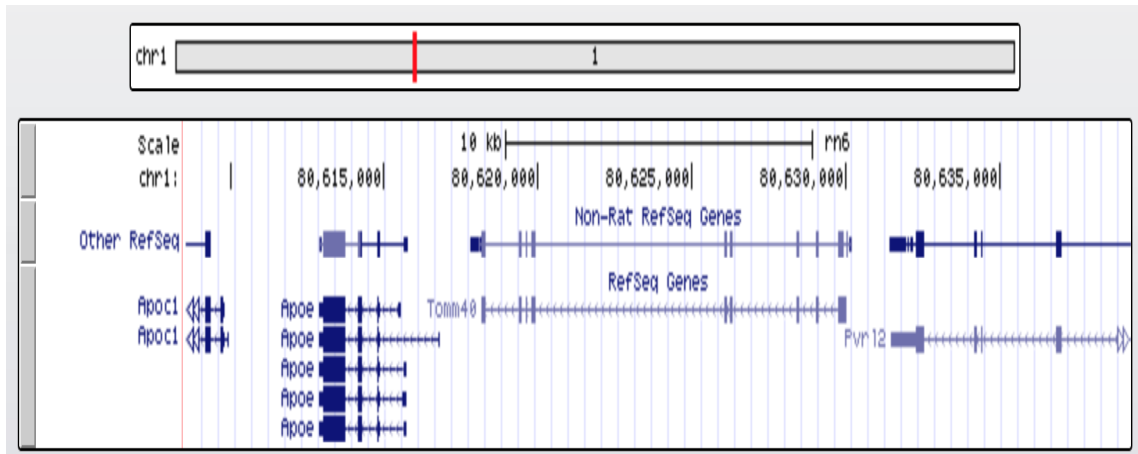


Figure 17: Rat genome on UCSC Genome Browser in Jul. 2014 (RGSC 6.0/rn6) Assembly. A small portion of the chromosome chr1 on rat genome is displayed here which is from the position 80608553 to 80639261. There are only two chosen tracks to be displayed from chromosome called chr1 which are *Other RefSeq* and *RefSeq genes*. *Other RefSeq* track shows the known protein and non-protein coding genes for organisms other than rat. *RefSeq genes* track shows known rat protein and non-protein coding genes taken from the NCBI RNA reference sequences collection (RefSeq). For each gene model on both tracks, the small rectangles are the exons.


2.6.2 Textual Spreadsheet

For *tabular* (or spreadsheet) data representation in biology, there are some efforts that are done. Some of them are developed for the purpose presenting annotation data textually such as UCSC Table Browser (<http://genome.ucsc.edu>) and MaizeGDB (<http://www.maizegdb.org>), while others developed for of data exchange such as ISA-TAB.

UCSC Table Browser is mainly used for retrieving and manipulating data that associated with a track in genome browser in textual format. Having a table that has DNA sequence or annotation data for the entire genome for specific coordinate and having some statistics about selected dataset are some of the benefit of using UCSC Table Browser. for visualization sample of these tables, see Appendix A.

The Investigation-Study-Assay (ISA) tab-delimited (TAB) format is developed initially to be a

universal purpose framework for collecting and exchanging complex experiments metadata such as protocol and sample characteristics. These data particularly is related to "omics-based" experiments such as genomics and proteomics, but it is not limited to them. In Figure 18 below, is a sample from ISA-TAB file for protein expression profiling data, which visualized after using ISA-TAB web viewer. In ISA-TAB, data are organized in tab delimited file which can be viewed as textual table representation using ISA-TAB web viewer for ISA-Tab files that found in (<https://github.com/ISA-tools/ISATab-Viewer>).



```

"S-0.1-aliquot11"      "protein extraction"  "S-0.1" "ITRAQ labeling"      "JC_S-0.1"  "iTRAQ
reagent
117"      ""      ""      "8761" "8761" "8761"  "spectrum.mzdata"      "norm1" "proteins.csv" "peptides.cs
v"      "ptms.csv"      "datatransformation1"  "PRIDE_Exp_Complete_Ac_8761.xml"      "sulphur"      ""
"      ""      "0.1"  "l/hr"  ""      ""
"C-0.1-aliquot11"      "protein extraction"  "C-0.1" "ITRAQ labeling"      "JC_C-0.1"  "iTRAQ
reagent
116"      ""      ""      "8761" "8761" "8761"  "spectrum.mzdata"      "norm1" "proteins.csv" "peptides.cs
v"      "ptms.csv"      "datatransformation1"  "PRIDE_Exp_Complete_Ac_8761.xml"      "carbon"      "EFO
"      "http://purl.obolibrary.org/obo/CHEBI_27594"  "0.1"  "l/hr"  ""      ""

```

Sample Name	Protocol REF	Extract Name	Protocol REF	Labeled Extract Name	Label	Term Source REF	Term Accession	MS Assay	Comment [PRIDE Accession]	PRIDE Comment [PRIDE Processed Data Accession]
S-0.1-aliquot11	protein extraction	S-0.1	ITRAQ labeling	JC_S-0.1	iTRAQ reagent 117			8761	8761	8761
C-0.1-aliquot11	protein extraction	C-0.1	ITRAQ labeling	JC_C-0.1	iTRAQ reagent 116			8761	8761	8761

Figure 18: Protein expression profiling mass spectrometry data that saved in ISA-TAB file format. In ISA-TAB, data are organized in tab delimited file as seen in the box above. This tab delimited file can be visualized as tabular format using ISA-TAB web viewer as seen in the box below.

2.6.3 Network Visualization Tools

In general, networks represent relationships and in biology these relationships could be functional or physical relationships. These include protein interaction, gene-gene interaction, regulatory transcription factors that regulate the genes interaction, or having a shared protein domain or family. The most benefit from using molecular interaction networks is to discover huge dataset, which helps later for understanding gene function in biology. It is more powerful method than tables to express many relationships among these massive datasets [Pavlopoulos et al., 2008]. Many network visualization tools are available to visualize biological interactions. These tools are varying on their public availability and user friendliness especially when thousands of circles with many connections have to be visualized and explored [Pavlopoulos et al., 2008].

One of the most common network visualization tools is *Cytoscape*. *Cytoscape* (<http://www.cytoscape.org>) is an open source network visualization and analysis tool for bioinformatics, which

enables users to display networks through the web. It is mainly used for biological research applications with many plugins that used to show specialized features [Shannon et al., 2003]. It is useful tool to visualize molecular interaction networks and biological pathways. In addition it integrates these networks with extra information such as gene expression profiles and annotations. *Cytoscape* works by loading network data then visualizes it as *nodes* (usually circles) and the *edges* as interaction between them. In Figure 19 below is an example of using Cytoscape in visualization of protein interaction networks [Gehlenborg et al., 2010]. In this figure, a network for *Mycoplasma pneumoniae* protein interaction data derived by mass spectrometry analysis. In the left network, there is an initial protein interaction network which has >400 proteins. Each node has a color that represents different data. In the right network, there is recomputed network remaining after removal of nodes not of interest. Nodes here are coloured according to functional annotation and their shapes represent different roles. In both networks, each node has relationship with other node if they are connected by link.

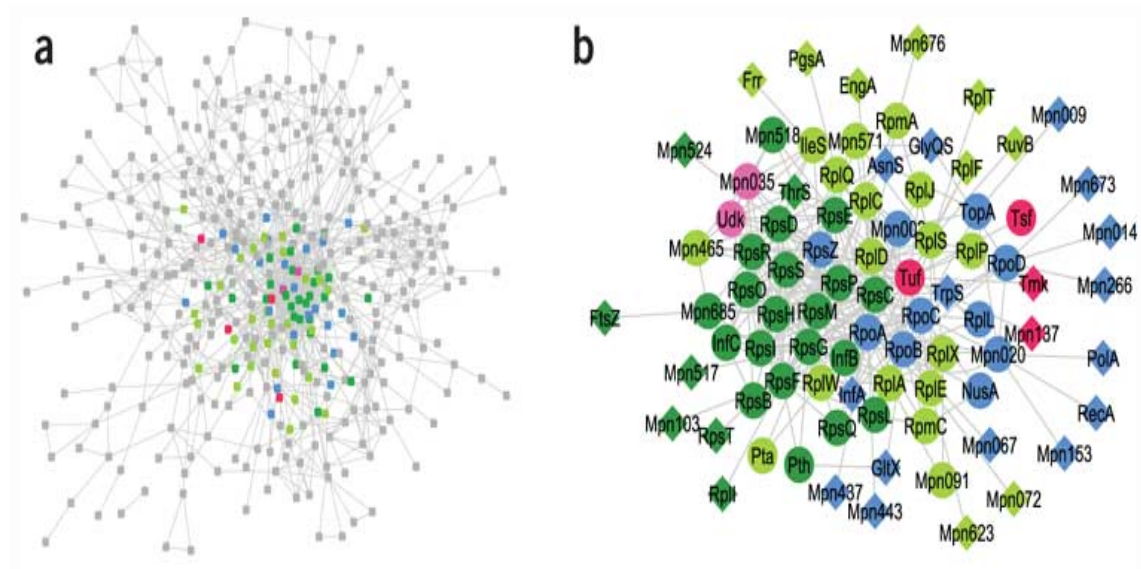


Figure 19: Cytoscape images for *Mycoplasma pneumoniae* protein interaction data derived by mass spectrometry analysis. (a) Initial protein interaction network (>400 proteins). Each node has a color that represents different data. For example, blue represents RNA polymerase; dark or light green represents small or large ribosomal subunits respectively; red represents elongation factor. (b) Recomputed network remaining after removal of nodes not of interest. Nodes are coloured according to functional annotation and their shapes represent different roles. Circle shape represents core protein of complex while diamond represents protein attached to complex but not part of the core. Credit: [Gehlenborg et al., 2010]

2.6.4 Tree Layout Visualization Tools

Tree layout is commonly used in the application that needs an efficient arrangement for biodiversity information. It is the suitable way to present the phylogenetic context. By using the tree layout, a better understanding for organism evolution patterns in gene and genome can be achieved [Soltis and Soltis, 2003].

Interactive Tree Of Life (ITOL) (<http://itol.embl.de>) is one of the well-known systems adapted this kind of visualization. It is web-based tool for displaying phylogenetic trees with adding customization features such as pruning and collapsing branches, downloading the tree in different formats, configuring and uploading dataset from text file to the tree [Letunic and Bork, 2007] [Letunic and Bork, 2011]. The created tree can be represented in circular, normal and unrooted (radial) modes as shown in Figure 20. In this figure a tree in three different modes are showed to represent 191 species from Archaea, Bacteria and Eukaryota in green, purple and pink respectively. Each mode has different settings such as rotation and arc options in circular and unrooted mods. The circular made is useful to display mid-sized trees that have several thousand leaves.

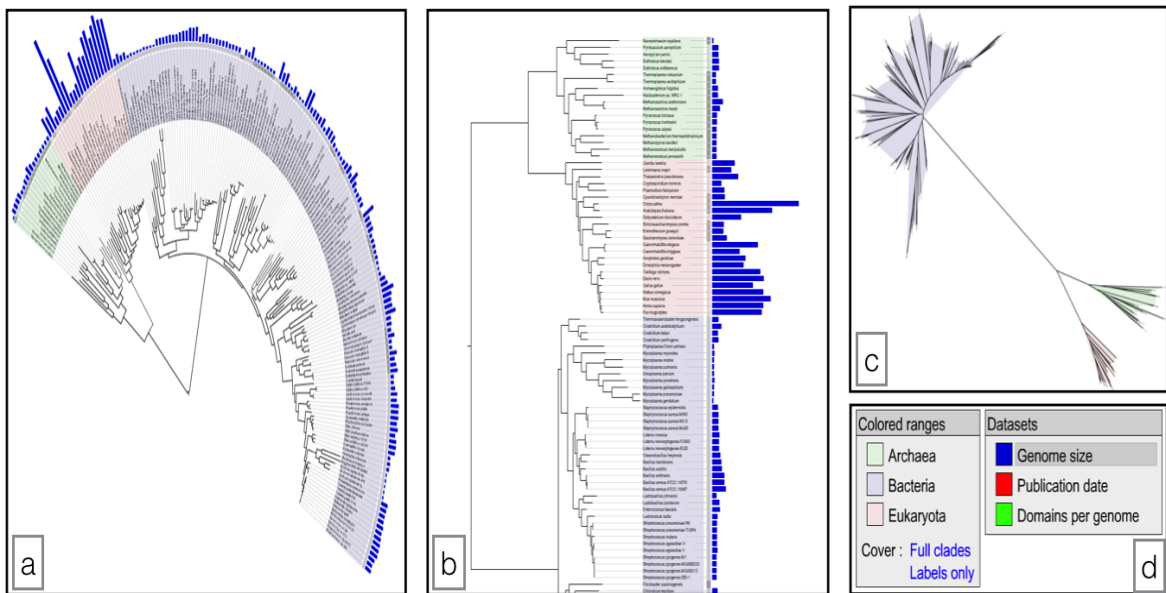


Figure 20: Representing phylogenetic tree in iTOL in different modes: circular, normal and unrooted respectively. The colours represent different type of data as seen in the small box (d) in the bottom-right. a) A tree has 191 species from Archaea, Bacteria and Eukaryota in green, purple and pink respectively. The tree here is presented in a 180° arc (350° is the default). The blue bars represent the genome size dataset. b) The same tree represented in the normal mode where labels are aligned to the leaves. c) The tree in unrooted (radial) display mode with 360° arc by default.

2.6.5 Circular Layout Visualization Tools

Circular Layout is practical for exploring relationships between objects especially when the tabular layout fails to express them in single view. One of the common interactive systems that uses circular layout visualization is Circos (http://circos.ca/intro/circular_approach/). It is software that creates composite circular display of genomic data and multi-layered annotations such as visualizing alignments, conservation and chromosomal relationships as seen in Figure 21. In this figure, whole-genome RNA-seq data is showed. The data is represented in six tracks which are cytoband, genomic location of top 100 genes, genome-wide RPKM, gene location related to placenta in the OMIM, enriched genes in placenta and functional clustering of unique genes. Each one of them represents different data. Circos is commonly used in genomics and cancer biology, although it can illustrate several types of data. There is a comparison between tabular and circular layout that can be found in (Circos and tabular visualization).

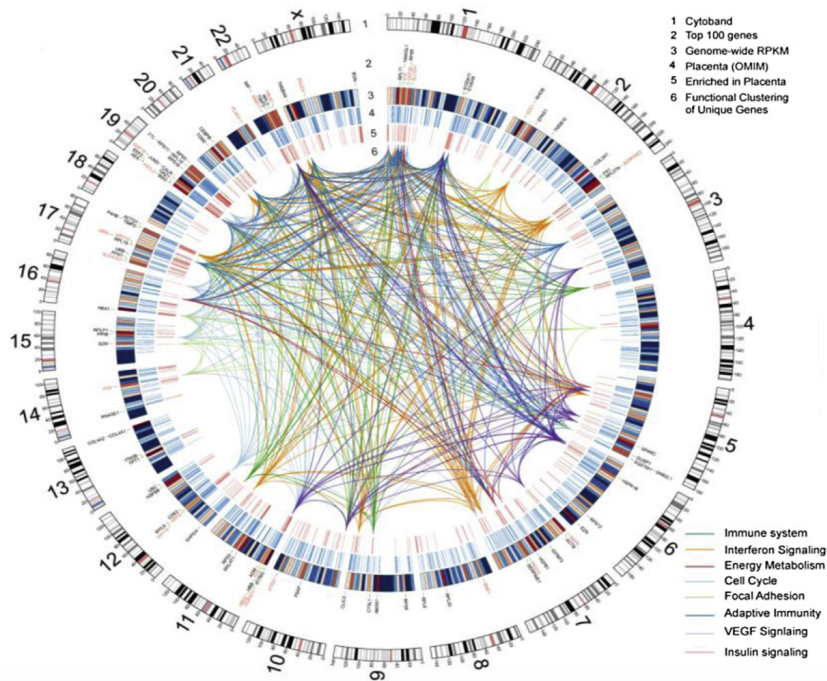


Figure 21: Using Circos Circular layout to express genomic annotation data for comprehensive analysis of the human placenta transcriptome. The data is represented in six different tracks. Track 1 represents cytoband where chromosomes are represented qter to pter. Track 2 represents genomic location of top 100 highly expressed genes in placenta based on average RPKM values. (Gene names in red represent enriched genes in placenta). Track 3 represents average RPKM values summarized over 6 MB regions showing regions of high gene expression. Track 4 represents locations of genes related to placenta in the OMIM database. Track 5 represents enriched genes in placenta (3-fold over 7 other tissues); Track 6 represents biological functions enriched among placenta-enriched genes. Credit: [Saben et al., 2014]

2.6.6 Related Works

FancyGene

FancyGene is a web-based software that represents the gene structure in graphical way based on genomic coordinates of one or more genes. Users can dynamically add some features after the standard representation of the gene structure is shown. These features may include adding the architecture of protein domains and changing the appearance of each object using a simple web interface.

The software can generate two output formats. First, high resolution PNG format that can be used in two ways: screen and poster presentations. Second, portable document format (PDF) that can be edited to allow the user to generate high-quality images suitable for publications [Rambaldi and Ciccarelli, 2009]. In Figure 22, gene representation for locus on human chromosome 17 taken from FancyGene. The figure also shows the option field where the user can set and change some of displaying options such as changing the colour of the exon box.

FancyGene has several features. First of all, it accepts data in a many input formats such as GFF (General Feature Format) and GTF (Gene Transfer Format) files. FancyGene is able to accept a simpler format where each line contains gene exons, the start and end points. The introns will be computed automatically. Labels to define the direction of transcription and to add UTRs information also can be added. Once input file is uploaded, it will be converted into the FancyGene format where each line contains four mandatory fields: the gene name label, the object tag, the start and stop of the object. The object tag can be intron, exon, UTR, marker or domain. The user can save the input file and the configuration file to re-use them later to create the same layout for the locus of interest. The second feature of FancyGene is having the ability to represent all parts of the gene structure that labeled in the input file such as coding exons, introns, UTRs and several types of labels. Third feature of the software is the ability to project protein domains information on the gene structure by associating exons to the encoded domains. FancyGene automatically alters the coordinates of the protein domain amino acidic into nucleotide coordinates. Then, it covers the coding exon by the domain architecture. Final feature is that the users have the ability to add features and new genes. They can also modify the look of any object and change their order such as changing gene labels positions. The coordinates of the locus also can be improved and the background can be changed [Rambaldi and Ciccarelli, 2009].

FeatureStack

FeatureStack is one of the Perl modules that used for automatic generation of multi-gene images. This software can be used to take BioPerl-compliant gene or transcript features as input and provide them on top of each other using a user-defined BioPerl glyph. The user has the option to generate SVG or PNG format for the output images. In addition, FeatureStack comes with a new BioPerl glyph and decorated_gene, which is able to highlight protein features on top of gene models [Frech et al., 2012]. According to [Frech et al., 2012], the goal of the FeatureStack is to provide maximum flexibility in image generation. This is done by providing some options and enabling some parameters

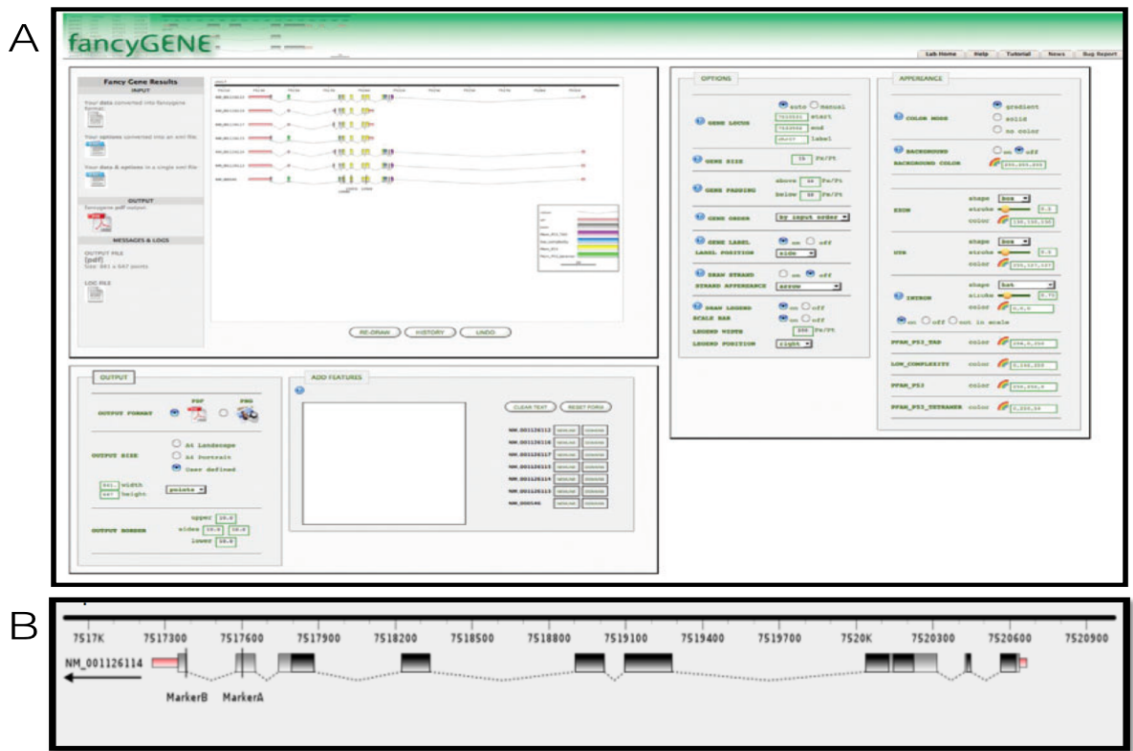


Figure 22: A) Screenshot of FancyGene website which has a gene representation for locus on human chromosome. The locus on human chromosome 17 corresponding to the p53 gene (entrez id: 7157) is shown. Credit: [Rambaldi and Ciccarelli, 2009]. B) Gene structure elements where intron represented as dash, exons represented as grey rectangles and encoded protein domain represented as black and grey rectangles.

settings to have better control for all aspects of the rendering process. One of FeatureStack advantages is that the user can use it with any BioPerl glyph that is compatible with the input features' structure. These glyphs will be more powerful when combined with FeatureStack decorated_gene glyph. The combination between them can generate rapid and automated large and annotation-rich images of stacked gene models. The comparison of gene structures will be much easier, because decorated_gene allows the highlighting and labeling of protein motifs like signal peptides, transmembrane domains, or protein domains on top of gene models. In Figure 23, RFX transcription factor gene family are represented by FeatureStack where the genes are ordered by their phylogenetic distance. They are also horizontally aligned by their most conserved feature which is the DNA-binding domain that drawn in black. By default, both exons and introns are drawn to scale, but exons in this figure is scaled while the introns has fixed size of 50 bp.

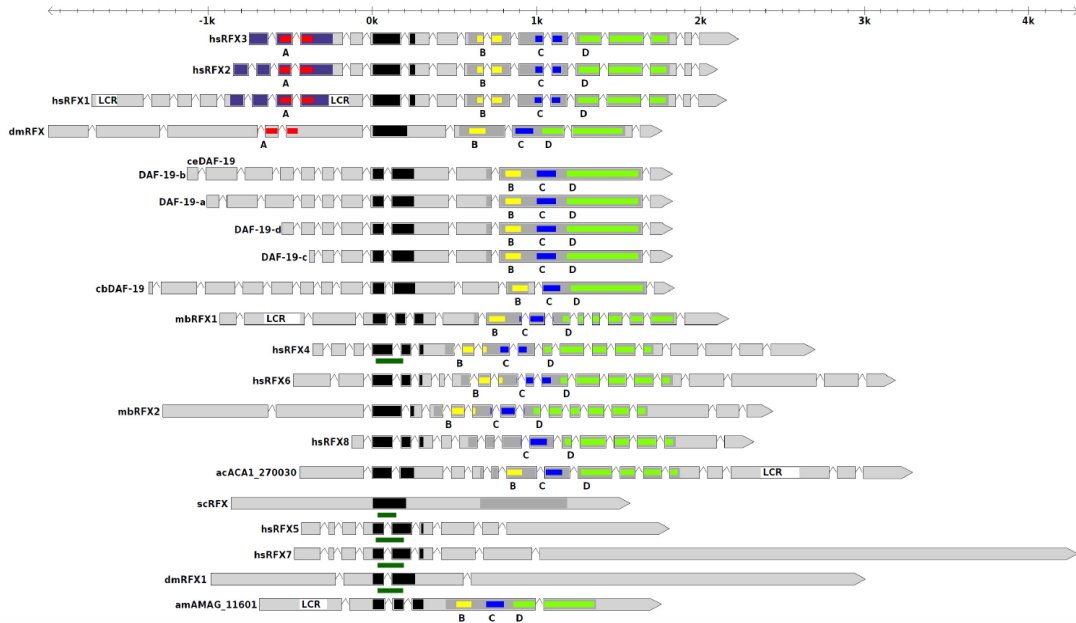


Figure 23: FeatureStack output which represents RFX gene family members over a different species: human (hs), fly (dm), *C. elegans* (ce, 4 isoforms), *C. briggsae* (cb), *M. brevicollis* (mb), *A. castellanii* (ac), *S. cerevisiae* (sc), and *A. macrogynus* (am). The genes are ordered by their phylogenetic distance. They are also horizontally aligned by their most conserved feature which is the DNA-binding domain. Domains are drawn in different colors: DNA-binding domain (black); N-terminal activation domain (dark slate blue); A, B, C, and D domains (red, yellow, blue, and green, respectively); combined BCD domain (dark grey); low complexity regions (LCR) in white; dark green bars below DBD indicate regions of similarity with viral Pox.D5 domain. Credit: [Frech et al., 2012].

Chapter 3

CGene and CGenome: System Analysis and Design

In this chapter we will see what services the system offers, what requirements our system should have, how the system is built, and what data format and technology are used to construct the system.

3.1 Project Scope and Objectives

In our research, we are concerned with the development of a visual system offering the possibility to produce a visual HTML5 spreadsheet for whole genome, in which the visual images are Scalable Vector Graphics (SVG) images for gene annotation features. The main services of the system are:

- Creating HTML5 spreadsheets.
- Browsing the spreadsheets.
- Presenting SVG images for both spreadsheets. For CGene spreadsheet, Gene model and Protein domain architecture images are presented. For CGenome spreadsheet, Gene location, Go Slims bitmap and Go Slims histogram images are presented.
- Downloading the SVG images.
- Selecting a dataset from the spreadsheet.

3.2 System Activity Diagram

Generally, an *Incremental Method* is used to develop the webpage starting from collecting the data from AspGD database, analyzing them, coding then testing them one at a time to make sure that the system objectives are achieved. Figure 24 shows the system activity diagram used to accomplish the objectives in Section 3.1. As shown in this figure, the first step is collecting the files from the AspGD

database (www.aspgd.org) about *Aspergillus* fungal genomes with different file formats. Those files include gene features, Protein Domain Predictions information, Gene Ontology (GO) annotations, AspGD chromosomal features and Go Slims description. Then, the system analyzes these files to extract the key features that required for displaying the following features: gene model, protein domains, genes location in the chromosome and GO Slims availability in the genome. After that, a scalable vector graphics (SVG) images are created for these features. Then, multiple HTML5 pages are created which include the master page, CGene pages and CGenome page. When the master page is displayed to users, they can select the link to any genome spreadsheet in CGene or select the link to CGenome page. In both spreadsheets, CGene and CGenome, the users can brows the spreadsheet, select row dataset and download the SVG images from the spreadsheet.

3.3 General Requirements

In general, the system should reflect the biology concepts. The system components such as a gene model should harmonious with the existing visualizations. Consequently, it does not require much time and effort from the user to understand the system.

3.3.1 Functional Requirements

3.3.1.1 Functionality of CGene

CGene, pronounced as See-Genie, is an HTML5 web-based spreadsheet, the data from a single genome will be explored and analyzed interactively as a visual spreadsheet. Each row in the visual spreadsheet captures the information about the gene and protein as text, images, and numerical data. The requirements for the visual spreadsheet using HTML5 for whole genome display are:

1. The system shall analyze different file formats that collected from the AspGD database (www.aspgd.org) about *Aspergillus* fungal genomes such as standard GFF3 files, standard output files from InterProScan. Those files include data about gene features and Protein Domain Predictions information that used for displaying the following features: gene model, protein domains.
2. The system shall produce a spreadsheet.
3. Each row on the spreadsheet shall hold the information about one gene.
4. For each gene, the following columns: gene ID, gene length, gene-model, protein-length and protein domain architecture shall be available.
5. If the gene is not associated with a protein, the system shall put blank in the domain architecture cell and in protein length cell.
6. The system shall present the gene ID as text.
7. The system shall present the gene length and protein-length as numbers.

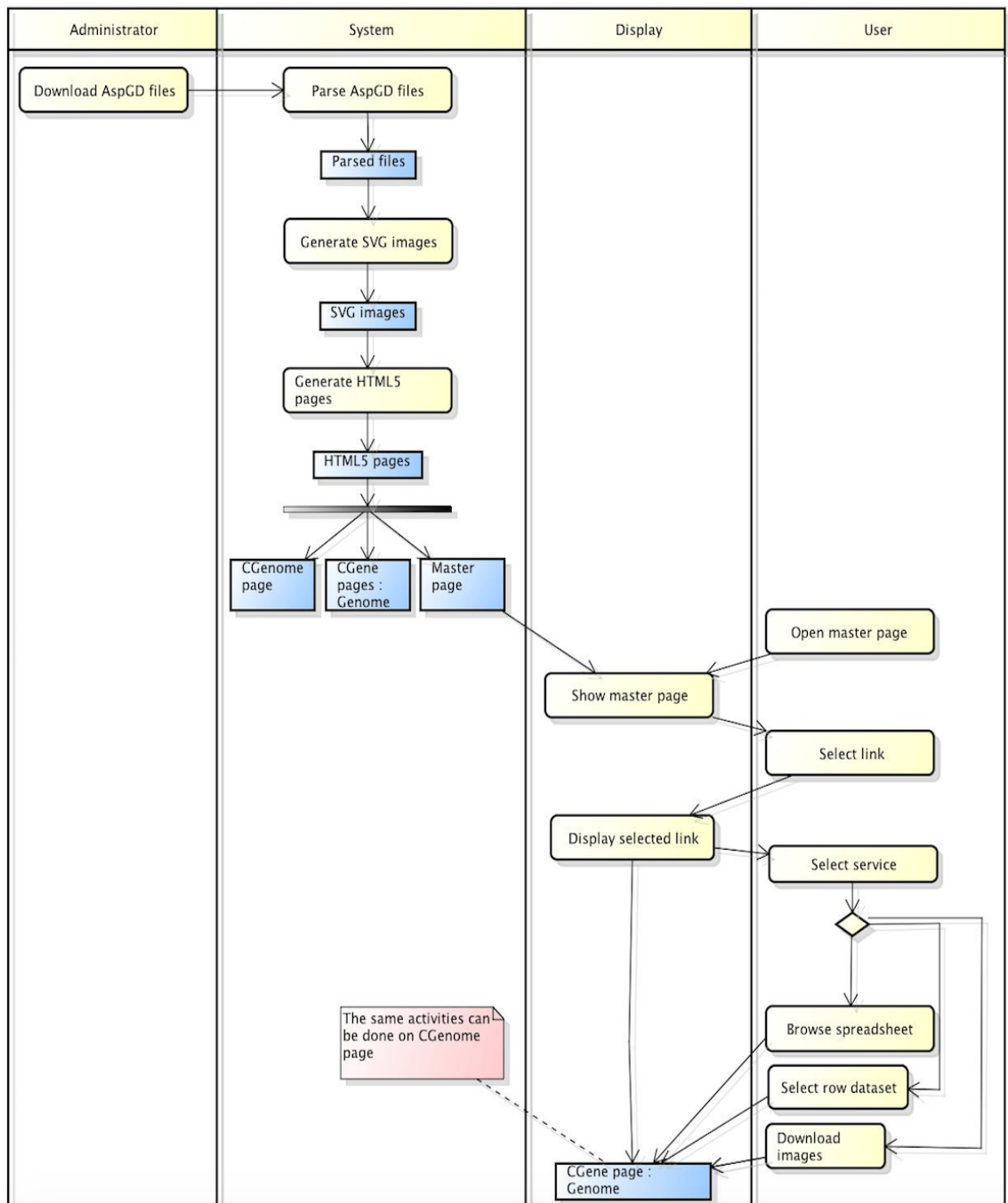


Figure 24: Activity diagram for visual HTML5 spreadsheets system: CGene and CGenome. The first step is collecting the files from AspGD database (www.aspgd.org) about Aspergillus fungal genomes. After that, the system extracts some key features that required for displaying gene model, protein domains, genes location in the chromosome and GO Slims occurrence in the genome. Then, a scalable vector graphics (SVG) images are created for these features. Then, multiple HTML5 pages are created which are the master page, CGene pages and CGenome page. The users can select a link from the master page to any CGene genome spreadsheet or select the link to CGenome page. For both spreadsheets, users can brows the spreadsheet, select row dataset and download the SVG images from the spreadsheet.

8. The system shall present the gene model and protein domain architecture as Scalable Vector Graphics (SVG) format.
9. The system shall enable the user to select a dataset from spreadsheet rows.

3.3.1.2 Functionality of CGenome

CGenome, pronounced See-Genome, produces a visual spreadsheet where the data comes from multiple genomes. Each row in the visual spreadsheet captures the information about a single genome.

The requirements for CGenome are:

1. The system shall analyze the annotation statistics extracted from AspGD DB. The available data are for four genomes, which are *A. nidulans* genome, *A. fumigatus* genome, *A. niger* genome and *A. oryzae* genome.
2. Each row on the spreadsheet shall represent the information about one genome.
3. For each genome, the following columns: Genome ID, Gene Location, GO Slims Bitmap and GO Slims Histogram shall be available.
4. The system shall present Gene Location, GO Slims Bitmap and GO Slims Histogram as Scalable Vector Graphics (SVG) format.
5. In the Gene location cell, the system should show the genes location on each chromosome.
6. In the Go Slims bitmap cell, the system should represent the GO slim occurrence in the specified genome.
7. In GO Slims histogram cell, the system should represent the number of occurrence of each GO Slim term.
8. The system shall enable the user to delete rows.

3.4 Domain Model

Before describing our system in details, we need to explain a domain model for system components since this step is *"the most important artifact"* to generate during object-oriented analysis [Larman, 2012]. This step will illustrate the meaningful conceptual classes in our problem domain. In this section, the most common conceptual classes in the system that reflects the real world will be defined. Then, the domain model diagram will be drawn.

3.4.1 Gene and Gene-Model Conceptual Domain Model

In order to build the Gene and Gene-Model conceptual domain model, we need to make a list of the important components that reflects the real-world conceptual classes.

As mention previously in Chapter 2: Section 2.1.1, a **cell** is the basic building unit of creatures and living species. Each of the cells has the complete human genome on its nucleus, which composes of 23 pairs of **chromosome**. Each chromosome have a collection of **DNA**(Deoxyribonucleic acid) molecules that encode the genetic code which used in building and functioning of the human being and the different organisms [Majoros, 2007]. DNA contains many **Genes**. **Gene** is the fundamental unit of inheritance, comprising a segment of DNA (or RNA in some viruses) that codes one or several related functions and occupies a fixed position (locus) on a chromosome. Generally, genes are made of three kinds of nucleotide sequence which are: Series of **exons** (coding regions), **introns** (non-coding regions) and **Regulatory regions** (Promoter). Since gene is represented by gene model, there is a need to define some of the components that most well known genomic browsers use to represent gene-model (see Appendix A for genome browsers examples). **Exons** are segments of gene that are represented in the mature RNA product while **introns** are segments of DNA in a gene that connect the exons, but are removed from the transcript by splicing. **Coding region** is DNA sequence region of a gene that will be translated later into a protein. This region is started with start codon and end with stop codon. **Start codon** is a place that spots the start site at which translation into protein sequence begins while **stop codon** is a place that spots the end site at which translation into protein ends [Majoros, 2007]. The gene also has untranslated regions (UTRs), which are three-prime untranslated region (3' UTR) and five-prime untranslated region (5' UTR). The **3' UTR** comes after the stop (termination) codon and located at the 3' end of the transcript. The **5' UTR** comes before the start codon and located at the 5' end of a transcript.

After listing the important components that most well known genomic browsers use to represent gene-model, we create the domain conceptual model for visualizing these concepts as shown in Figure 25. In this figure, the components that reflect the real-world conceptual classes and their relationships are illustrated. This diagram has the basic building unit of creatures that called cell, which consists of DNA that found in chromosome. The DNA is transcribed into pre-mRNA. The pre-mRNA is spliced into mRNA, which is translated later to protein. The chromosome consists of multiple genes from specific genome. There are many components that the gene may include such as exons, introns, coding regions and UTRs. Gene model is the visual representation for the gene and its components.

3.4.2 Protein and Protein-domain Architecture Conceptual Domain Model

To build the Protein and Protein-domain architecture domain model, we need to make a list of the important components that reflects the real-world conceptual classes.

Protein, as described previously in Chapter 2: Section 2.1.1, is a large biological molecule, which its **primary structure** is the sequence of one or more long amino-acid residues. **Amino Acids Sequence** is the unique chain of amino acids that characterizes a given protein. They put together into a polypeptide chain on the ribosome during protein synthesis. A protein sequence is represented as a chain of 20 English alphabet letters (A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y) where each letter represents an amino acid. This Amino Acids Sequence may have many patterns and repeats. **Patterns** are regions that consist of a few amino acids. When these regions are

identified in multiple sequence alignment process, it helps to recognize some sequence features such as the active sites of enzymes and binding sites [EMBL-EBI, 2014]. These alignments can be used to define either evolutionary or functional relationships [Hertz and Stormo, 1999]. Many proteins have runs of single amino acids that called **repeat** [Albà et al., 2007]. These repeats have specific roles in protein function and evolution. For example, repeats with complex patterns such as *leucine-rich* and *WD* mostly refer to the functional domain repeats, which are commonly participate in protein-protein interaction [Luo and Nijveen, 2013].

As stated previously in Chapter 2: Section 2.4.1, protein structure is divided into three or four levels structure based on the references that categorized the protein structure. These structures are **primary structure**, **secondary structure** and **Tertiary Structure**. The next level of protein structure after amino acid sequence is **Protein Structural Motif**. It called super-secondary structure, which is defined by the connectivity between secondary structure elements. *Helices* and *sheets* are the most common secondary structures motif, but there are irregular secondary structure motifs that can have critical role in biological function [Rangwala and Karypis, 2010]. The most common structural motifs are *Alpha Helix*, *Beta Sheets*, *Coil(or Loops)* and *Turns*. **Tertiary structure** is the global 3D shape that represents the protein in three dimensional crystal coordinate. This shape contains of a variety arranged secondary structure elements that bonds together. There is a stable form of the tertiary structure which called **protein folds**. Protein folding is the process by which a protein structure assumes its functional shape or conformation. It is the physical process by which a polypeptide folds into its characteristic and functional three-dimensional structure from random coil [Wikipedia, 2014].

Protein Domain Architecture, or domain arrangement, refers to the domains and their liner arrangements in individual protein. The direction of these domains goes a long the amino acid chain from N- to C- terminal [Morrison, 2013]. Domain architecture has many elements such as sites, motifs and domains. **Site** consists of the residues at specific position while **motifs** simply define the structural characteristics in a protein [Biologyreference., 2014]. Sequence motifs share a certain sequence of amino acids where a specific arrangement of amino acids that found in one protein can be found in other proteins. Motif can be part of domains such as the *leucine zipper* motif that frequently found as segment of a dimerization domain in several transcription factors. **Domains** consider as the basic building block of a protein structure. Usually they are responsible for a particular function or interaction, contributing to the overall role of a protein.

After listing the important components, we create the visualizing concepts in Figure 26. In this figure, the components that reflect the real-world conceptual classes and their relationships are illustrated. This diagram shows the protein, its structures, its domain representation and their relationships. at the beginning, the protein is structured into primary structure which is the amino Acids Sequence. This sequence has many patterns and repeats. The primary structure encodes protein secondary structure that encodes tertiary structure. Protein Structural Motif which called super-secondary structure consist of the common structural motifs which are *Alpha Helix*, *Beta Sheets*, *Coil(or Loops)* and *Turns*. Domain architecture represents many protein elements such as sites, motifs and domains.

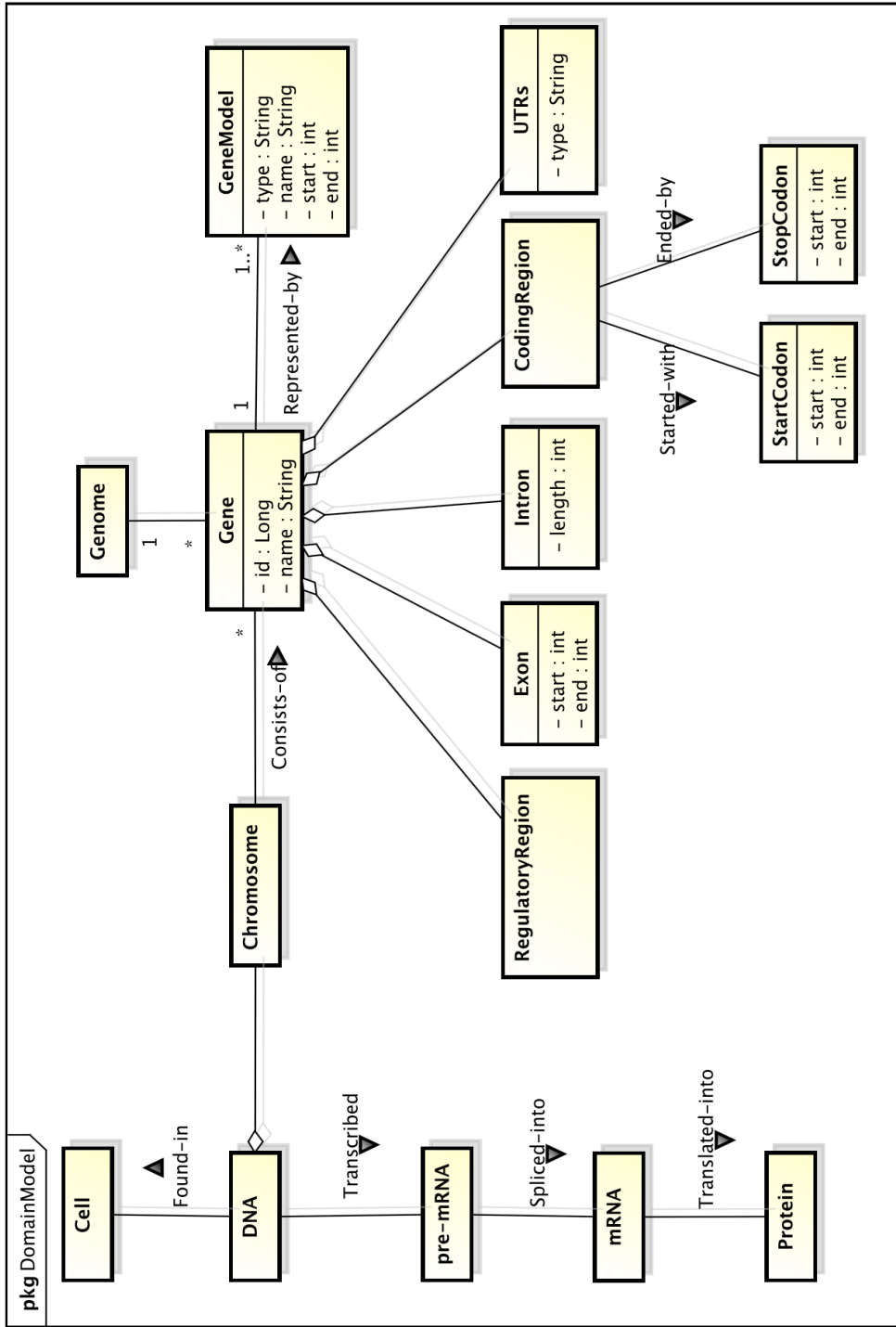


Figure 25: Gene-Model domain conceptual class diagram showing classes and their relationships. As seen, the genome has multiple genes, which comes from the cell's chromosomes. These genes may have some components such as exons, introns, coding regions and UTRs. The DNA that found in the cell is transcribed into pre-mRNA. The pre-mRNA is spliced into mRNA, which is translated later to protein.

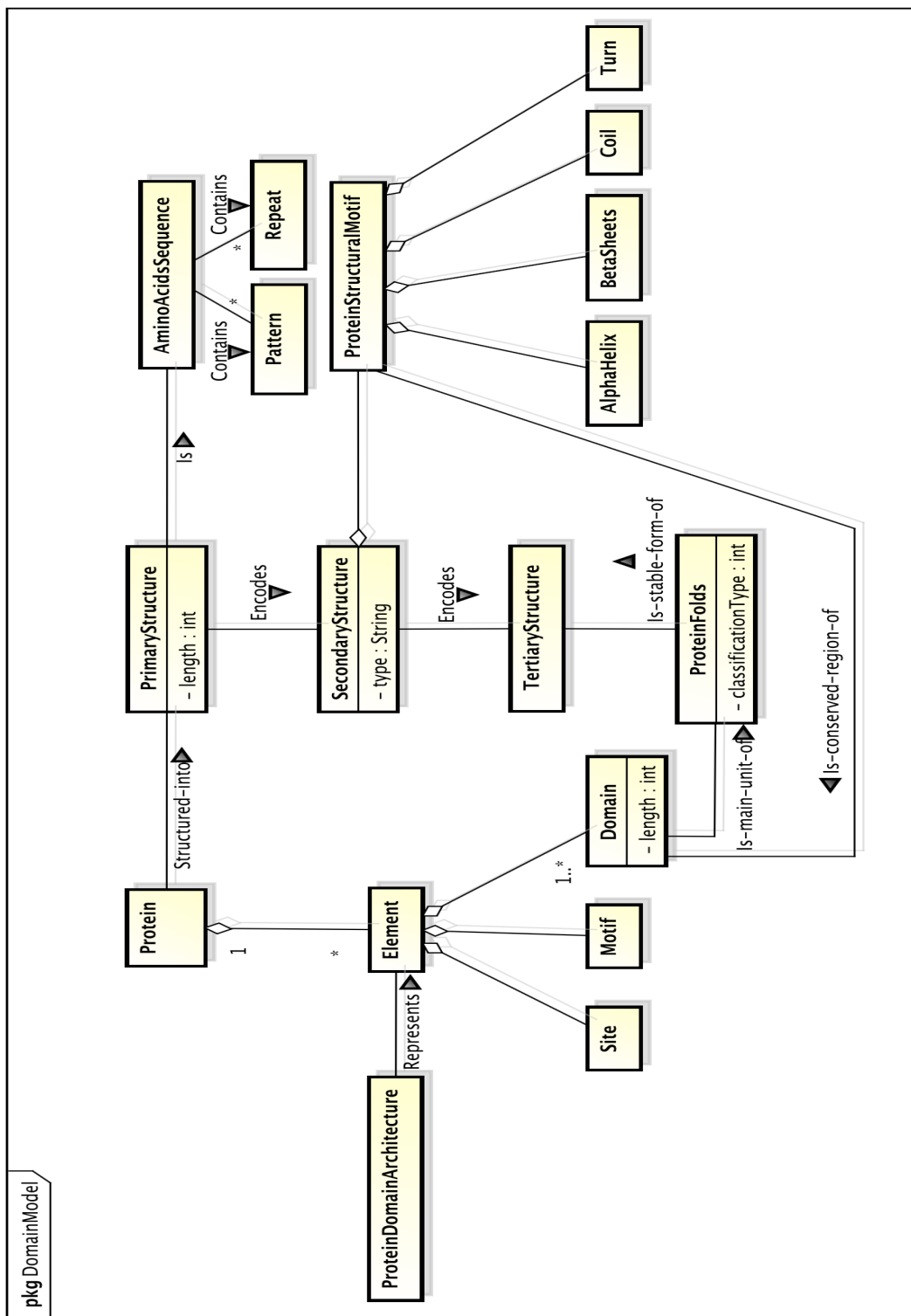


Figure 26: Protein and Protein-domain conceptual class diagram showing classes and their relationships. The protein is structured into primary structure, which is the amino acids sequence. This sequence has many patterns and repeats. The primary structure encodes protein secondary structure that encodes tertiary structure. Protein structural motifs which are called super-secondary structures consist of common structural motifs which are *Alpha Helix*, *Beta Sheets*, *Coil(or Loops)* and *Turns*. Domain architecture represents many protein elements such as sites, motifs and domains.

3.5 Data Sources and Formats

In this research, we focus on fungal genomes, so the genomic data used to produce the spreadsheets are stored in the AspGD database (www.aspgd.org). The generated data is come from standard GFF3 files, standard output files from InterProScan, aspgd file from AspGD Gene Ontology annotations file, and chromosomal feature file. For example, for *A. acidus*_CBS_106_47 genome the following files are processed: *A_acidus_CBS_106_47_features.gff.txt* created at 16-12-2013, *A_acidus_CBS_106_47_iprscan.out.txt* created at 19-11-2013. The other processed genomes have the same file format as *A_acidus_CBS_106_47*. *Gene.association.aspgd.txt* created at 29-6-2014 is the aspgd file that used for AspGD Gene Ontology annotations. *A_fumigatus_A1163_chromosomal_feature.tab.txt* created at 7-6-2014, *A_nidulans_FGSC_A4_current_chromosomal_feature.tab.txt* created at 7-6-2014 and *A_niger_CBS_513_88_current_chromosomal_feature.tab.txt* created at 6-6-2014 are the processed chromosomal feature files. All these file formats will be described later in Section 3.5.2.

3.5.1 Data Selection

Data selection for some of protein domain architecture

For the protein domain architecture, we did an analysis to understand what are the most repeated domains. Since there is no standard to visualize the protein domain architecture as mentioned in Chapter 2: Section 2.4.3 and CGene spreadsheets are implemented to be fixed, this study will help in case if we want to present some domains in different colors/shapes to help the user notice them. This analysis is done on five different *standard output* files which have *protein domain predictions* information that was predicted using *IprScan* software from the *InterPro* DB. The files are collected for five interested fungi according to the contents of AspGD DB [AspGD, 2014], covering the following genomes: *A. nidulans*, as an excellent model organism; *A. fumigatus* which is an important pathogen of the immunocompromised; *A. flavus* which is an important agriculture toxin producer; *A. niger* and *A. oryzae* which are two well known species that used in industrial processes. The analysis also includes testing 11547 distinct *InterPro* entries from *interpro2go.txt* file that found in [InterPro, 2014] website. For each *InterPro* entry, the frequencies is counted as the sum of the entry occurrence in the five fungi. We find out that 3472 entries out of the 11547 occur at least once in the five fungi. Then, Pareto chart is applied for all the *InterPro* entries. First, we applied Pareto principle 80:20 rule, which states that 80% of the effects come from 20% of the causes [Koch, 2011]. This chart used to see what are the most important domains (or most repeated) as seen in Figure 27. However, in this figure we see that large number of entries is included in the 20%. Around 666 out of 3472 entries will be selected by applying this chart. These entries are large dataset to present in different shapes/colors and they are difficult to distinguish from users, especially some of them are repeated only less than ten times. Consequently, for the sake of clarity we zoomed out the chart, and only the top 25 domains were selected as seen in Figure 28 and Table 3. In this table we picked up the top 25 repeated domain entries. This analysis helps to see what are the important domains, which is the top 25 domains from the domains that included in the 20% rule. After applying this study, these domains can be visualized in CGene spreadsheet in different shapes/colors

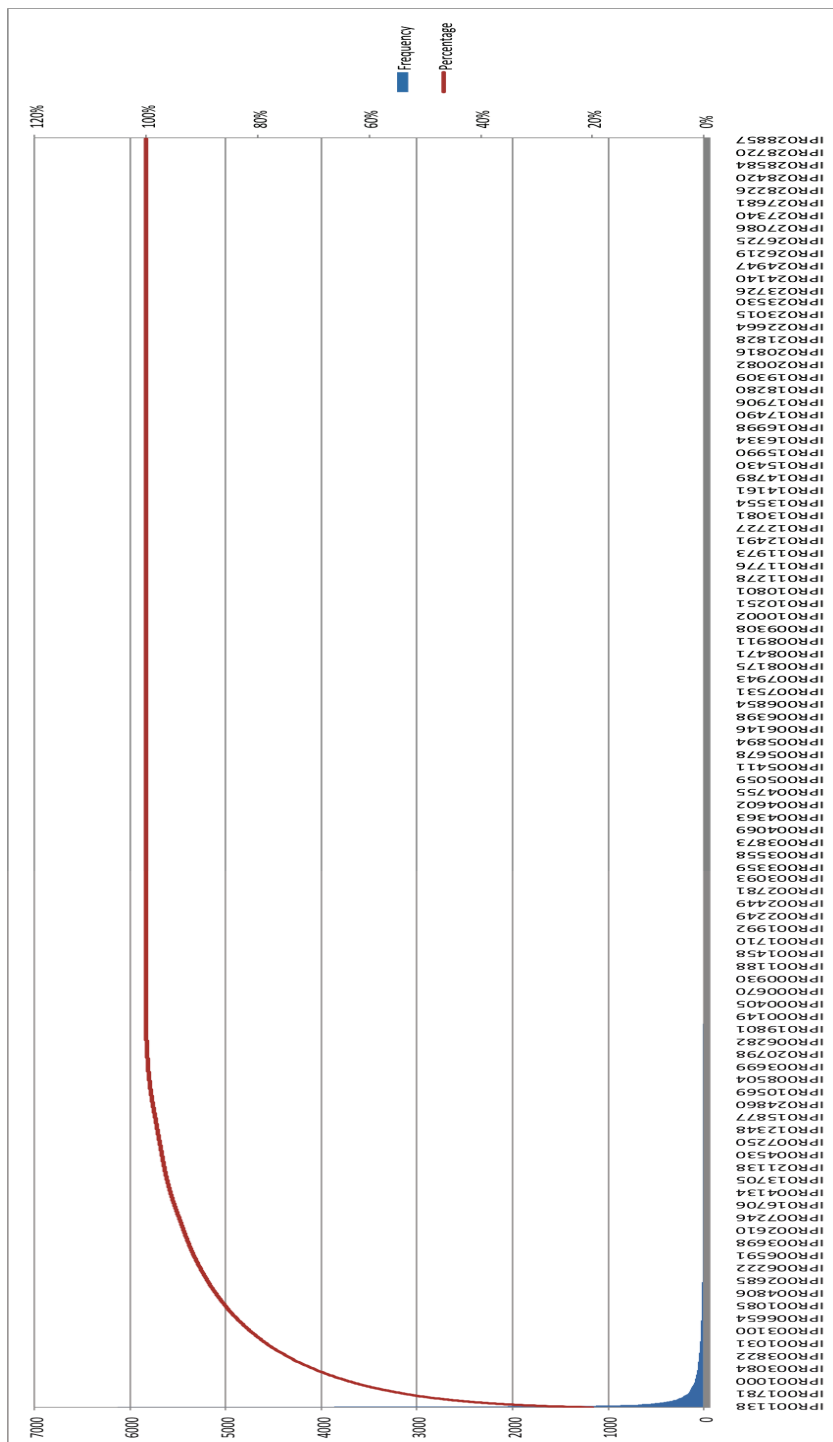


Figure 27: Pareto chart shows InterPro entries with their frequencies in the five most common filamentous fungi of the genus *Aspergillus*: *A. nidulans*, *A. fumigatus*, *A. flavus*, *A. niger* and *A. oryzae*. The analysis includes testing 11547 distinct *InterPro* entries. We find out that 3472 entries out of the 11547 occur at least once in the five fungi. Around 666 out of 3472 entries will be selected by applying Pareto principle on this chart. Since CGene spreadsheet contents, including protein domain architecture, are fixed and not configurable by the user, this analysis is done to see what are the important domains that included in the 20% rule. These domains can be visualized in CGene in different shapes/colors to help the user distinguish them.

InterPro Entry	Frequency	Cumulative Frequency	Cumulative Percentage
IPR001138	6130	6130	4%
IPR001680	5931	12061	9%
IPR002110	3867	15928	11%
IPR001128	3786	19714	14%
IPR002401	2329	22043	16%
IPR003663	2064	24107	17%
IPR007219	2056	26163	19%
IPR002198	1704	27867	20%
IPR011701	1581	29448	21%
IPR003042	1344	30792	22%
IPR003593	1301	32093	23%
IPR019734	1134	33227	24%
IPR000504	1046	34273	25%
IPR015943	992	35265	25%
IPR011009	911	36176	26%
IPR001757	874	37050	27%
IPR001650	852	37902	27%
IPR000719	832	38734	28%
IPR017986	794	39528	28%
IPR011990	763	40291	29%
IPR001199	687	40978	29%
IPR001806	682	41660	30%
IPR002403	654	42314	30%
IPR003439	650	42964	31%
IPR016024	647	43611	31%

Table 3: The top 25 selected InterPro domains data from the five most common filamentous fungi of the genus *Aspergillus*: *A. nidulans*, *A. fumigatus*, *A. flavus*, *A. niger* and *A. oryzae*. These domains are selected from the 11547 entries that showed in Figure 27. Since CGene spreadsheet contents, including protein domain architecture, are fixed and not configurable by the user, this analysis is done to see what are the top repeated domains. These domains can be visualized in CGene in different shapes/colors to help the user distinguish them.

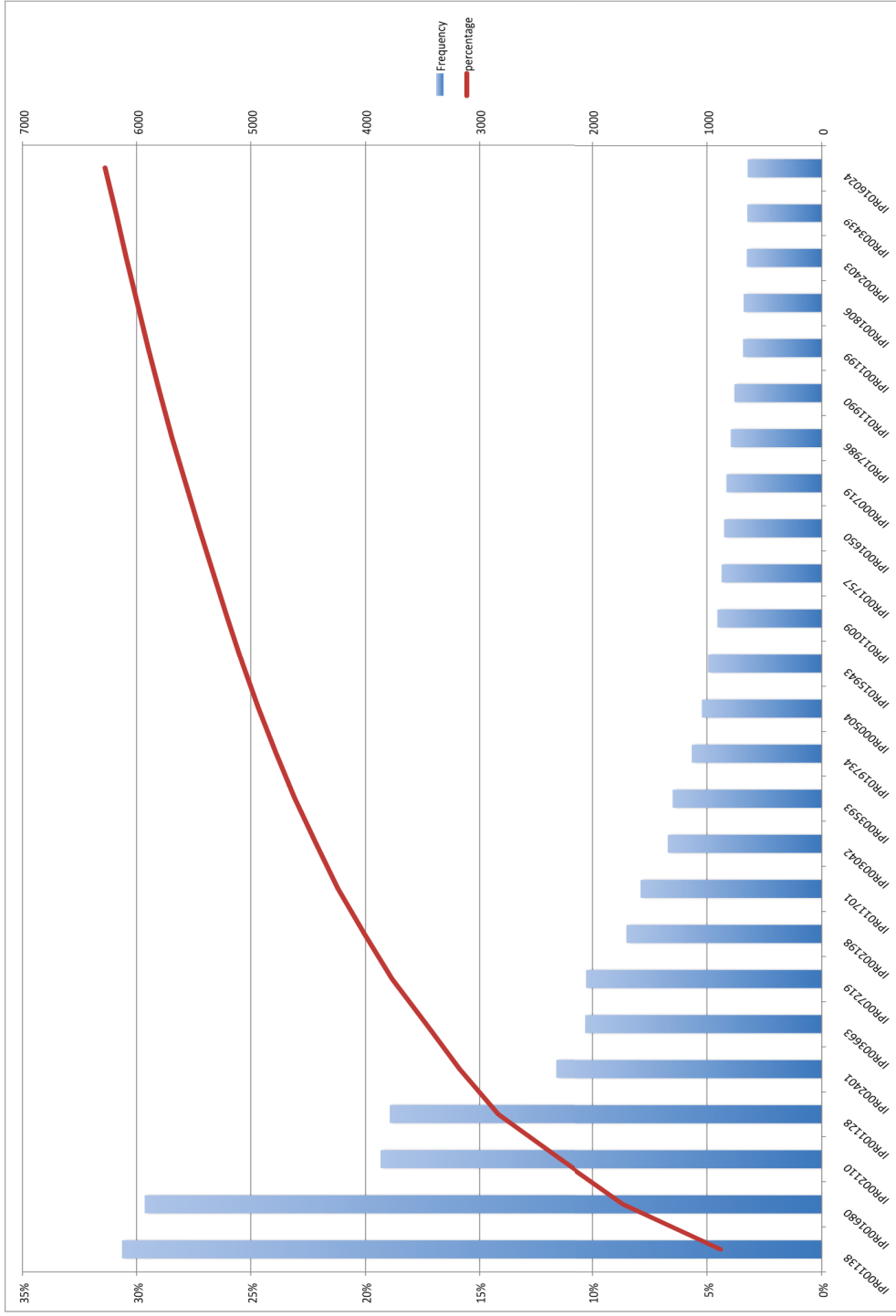


Figure 28: Pareto chart shows the top 25 selected InterPro domains in the five most common filamentous fungi of the genus *Aspergillus*: *A. nidulans*, *A. fumigatus*, *A. flavus*, *A. niger* and *A. oryzae*. These 25 domains are selected from the 11547 entries that showed in Figure 27.

Go Slims data selection for CGenome visual spreadsheet

Go Slims, discussed previously in Chapter 2: Section 2.3.3, that used for Go Slims bitmap and Go Slims histogram in CGenome spreadsheet are *Go Slims Aspergillus* which selected from Gene Ontology [GeneOntology, 2014] website (<http://geneontology.org/page/go-slim-and-subset-guide>). In [GeneOntology, 2014], slims are named to be in agreement with the application that uses these slims. For example, Go Slims Aspergillus file includes terms that mainly valuable when annotating Aspergillus. There is also a generic Go Slims file that used for most applications without species preference. Before creating CGenome spreadsheet, we did comparative analysis to have an idea about the GO Slims and to see what are the most repeated *Go Slims Aspergillus* in four Aspergillus fungi which are: A.nidulans_FGSC_A4, A.fumigatus_Af293, A.niger_CBS_513_88 and A.oryzae_RIB40. This analysis is done before implementing Go Slims bitmap and Go Slims histogram. By applying the analysis, we found out that *protein binding* GO Slim (ID: GO:0005515) has the highest number of occurrence among other molecular function Go Slims Aspergillus for all the four genomes with 4958, 3963, 3519 and 4024 for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively as shown in Table 4. Figure 29 shows the same information but as percentage of occurrence instead of number of occurrence. In this figure, *protein binding* has the highest percentage of occurrence for all the four genomes as 46.3%, 43.8%, 44.9% and 43.0%. In Table 5, *nucleus* GO Slim (ID: GO:0005634) has the highest number of occurrence among cellular component Go Slims Aspergillus for all the four genomes with 2609, 2333, 1696 and 1832 for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively. In Figure 30, *nucleus* GO Slim has the highest percentage of occurrence for all the four genomes as 59.0%, 54.9%, 49.4% and 48.2%. For biological process Go Slims Aspergillus, *carbohydrate metabolic proces* GO Slim (ID: GO:0005975) has the highest number of occurrence for all the four genomes with 875, 907, 941 and 943 as shown in Table 6; it has also the highest percentage of occurrence as 27.9%, 30.2%, 32.7% and 29.9% for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively as shown in Figure 31. Some Slims did not occur at all in the four genomes such as *lipase activity* and *site of polarized growth* molecular function slims, *membrane fraction* cellular component slim and *cellular respiration* biological process slim.

After creating CGenome spreadsheet, we compare the results from the analysis and the images that showed in the spreadsheet for Go Slims bitmap and Go Slims histogram. The drawn images should have the same results that obtained from the analysis.

Molecular function	Go Slims Aspergillus	ID	Organism				
			Number of Existence				
			A_niger_CBS_513_88	A_nidulans_FGSC_A4	A_fumigatus_Af293	A_oryzae_RIB40	
protein binding transcription factor activity		GO:0000988	0	0	0	0	
molecular function		GO:0003674	0	0	0	0	
DNA binding		GO:0003677	1223	1235	1182	1145	
RNA binding		GO:0003723	386	385	361	343	
motor activity		GO:0003774	29	29	30	29	
helicase activity		GO:0004386	218	178	186	185	
protein kinase activity		GO:0004672	344	248	236	266	
signal transducer activity		GO:0004871	124	99	91	106	
structural molecule activity		GO:0005198	15	85	67	77	
transporter activity		GO:0005215	202	125	86	163	
protein binding		GO:0005515	4958	3963	3519	4024	
peptidase activity		GO:0008233	31	27	24	36	
lipase activity		GO:0016298	0	0	0	0	
oxidoreductase activity		GO:0016491	2401	2032	1495	2309	
transferase activity		GO:0016740	142	65	37	65	
nucleotidyltransferase activity		GO:0016779	31	34	34	39	
hydrolase activity		GO:0016787	455	405	377	443	
phosphatase activity		GO:0016791	42	34	30	39	
lyase activity		GO:0016829	51	50	40	40	
isomerase activity		GO:0016853	19	15	13	16	
ligase activity		GO:0016874	35	32	27	30	
enzyme regulator activity		GO:0030234	3	3	2	2	
site of polarized growth		GO:0030427	0	0	0	0	
triplet codon-amino acid adaptor activity		GO:0030533	0	0	0	0	
translation regulator activity		GO:0045182	0	0	0	0	

Table 4: Table shows Molecular Function Go Slims Aspergillus occurrence in specified genomes: A. niger, A. nidulans, A. fumigatus and A. oryzae. As Shown, *protein binding* GO Slim (ID: GO:0005515) has the highest number of occurrence for all the four genomes with 4958, 3963, 3519 and 4024 for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively. The second highest number of occurrence for all the four genomes is *oxidoreductase activity* (ID: GO:0016491) with 2401, 2032, 1495 and 2309 for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively. Some of the GO Slims does not occur for all of four genomes such as *lipase activity* and *site of polarized growth*.

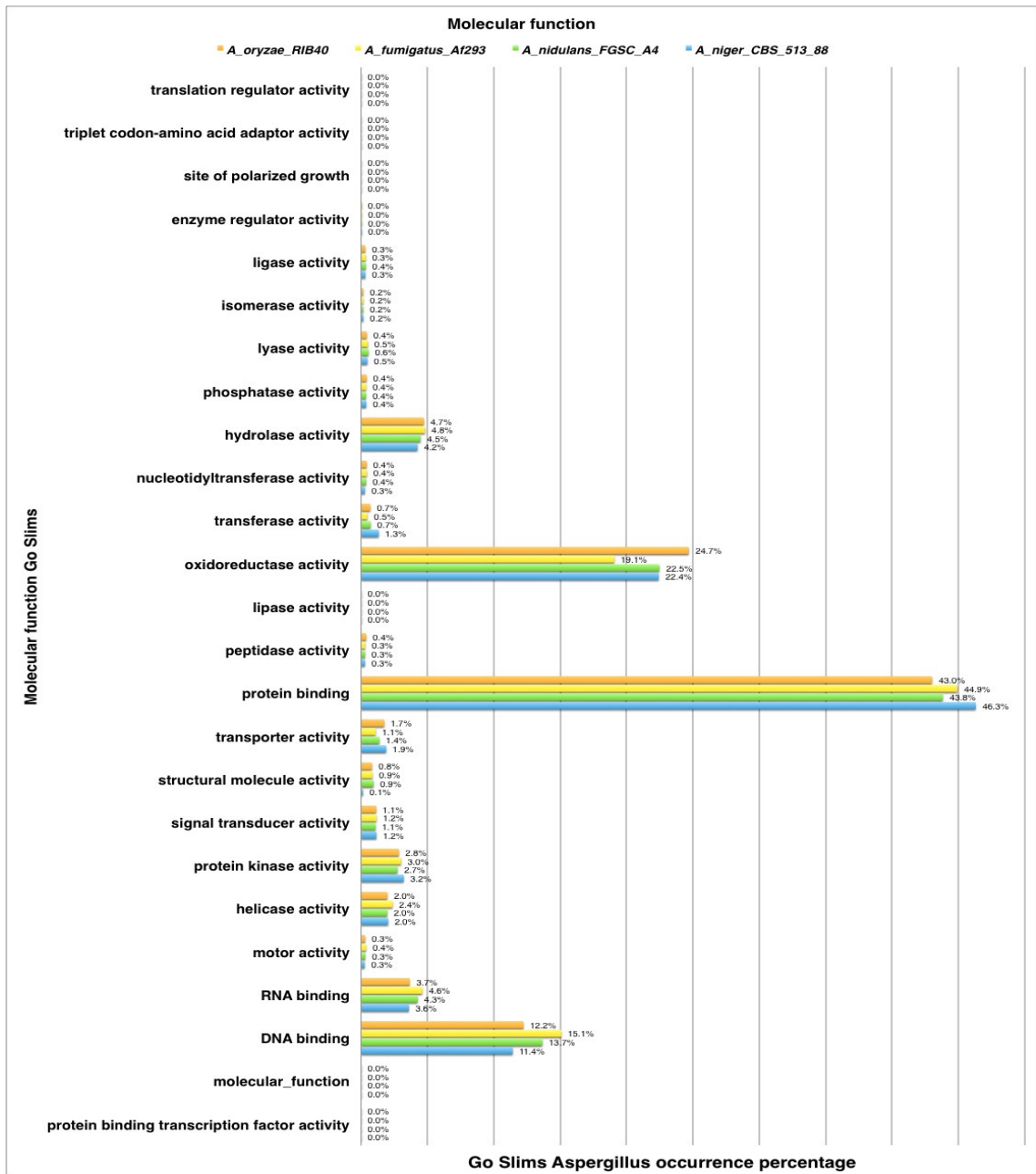


Figure 29: Chart shows Molecular Function Go Slims Aspergillus occurrence percentage in the specified genome. As Shown, *protein binding* GO Slim (ID: GO:0005515) has the highest percentage of occurrence for all the four genomes as 46.3%, 43.8%, 44.9% and 43.0% for *A. niger*, *A. nidulans*, *A. fumigatus* and *A. oryzae* respectively. The second highest percentage of occurrence for all the four genomes is *oxidoreductase activity* (ID: GO:0016491) with 22.4%, 22.5%, 19.1% and 24.7% for *A. niger*, *A. nidulans*, *A. fumigatus* and *A. oryzae* respectively. Some of the GO Slims has zero occurrence percentage for all of four genomes such as *lipase activity* and *site of polarized growth*.

Cellular component	Go Slims Aspergillus	ID	Organism			
			Number of Existence			
			A_niger_CBS_513_88	A_nidulans_FGSC_A4	A_fumigatus_Af293	A_oryzae_RIB40
cellular component	GO:0005575		0	0	0	0
extracellular region	GO:0005576		74	57	94	50
cell wall	GO:0005618		5	6	7	6
membrane fraction	GO:0005624		0	0	0	0
nucleus	GO:0005634		2609	2333	1696	1832
chromosome	GO:0005694		59	72	72	75
nucleolus	GO:0005730		9	8	8	8
mitochondrion	GO:0005739		25	25	22	15
vacuole	GO:0005773		1	1	1	1
peroxisome	GO:0005777		5	9	8	11
endoplasmic reticulum	GO:0005783		33	25	23	23
GOlg1 apparatus	GO:0005794		1	1	1	1
cytosol	GO:0005829		1	1	4	0
ribosome	GO:0005840		340	338	319	254
cytoskeleton	GO:0005856		19	18	16	17
plasma membrane	GO:0005886		23	17	12	22
cell cortex	GO:0005938		2	2	2	2
endomembrane system	GO:0012505		0	0	0	0
actin cytoskeleton	GO:0015629		3	8	8	4
microtubule cytoskeleton	GO:0015630		0	0	0	0
cytoplasmic membrane-bounded vesicle	GO:0016023		0	0	0	0
membrane	GO:0016020		1216	1332	1142	1477

Table 5: Table shows Cellular Component Go Slims Aspergillus occurrence in specified genomes: A. niger, A. nidulans, A. fumigatus and A. oryzae. As Shown in the table, *nucleus* GO Slim (ID: GO:0005634) has the highest number of occurrence for all the four genomes with 2609, 2333, 1696 and 1832 for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively. The second highest number of occurrence for all the four genomes is *membrane* (ID: GO:0016020) with 1216, 1332, 1142 and 1477 for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively. Some of the GO Slims does not occur for all of four genomes such as *endomembrane system* and *membrane fraction*.

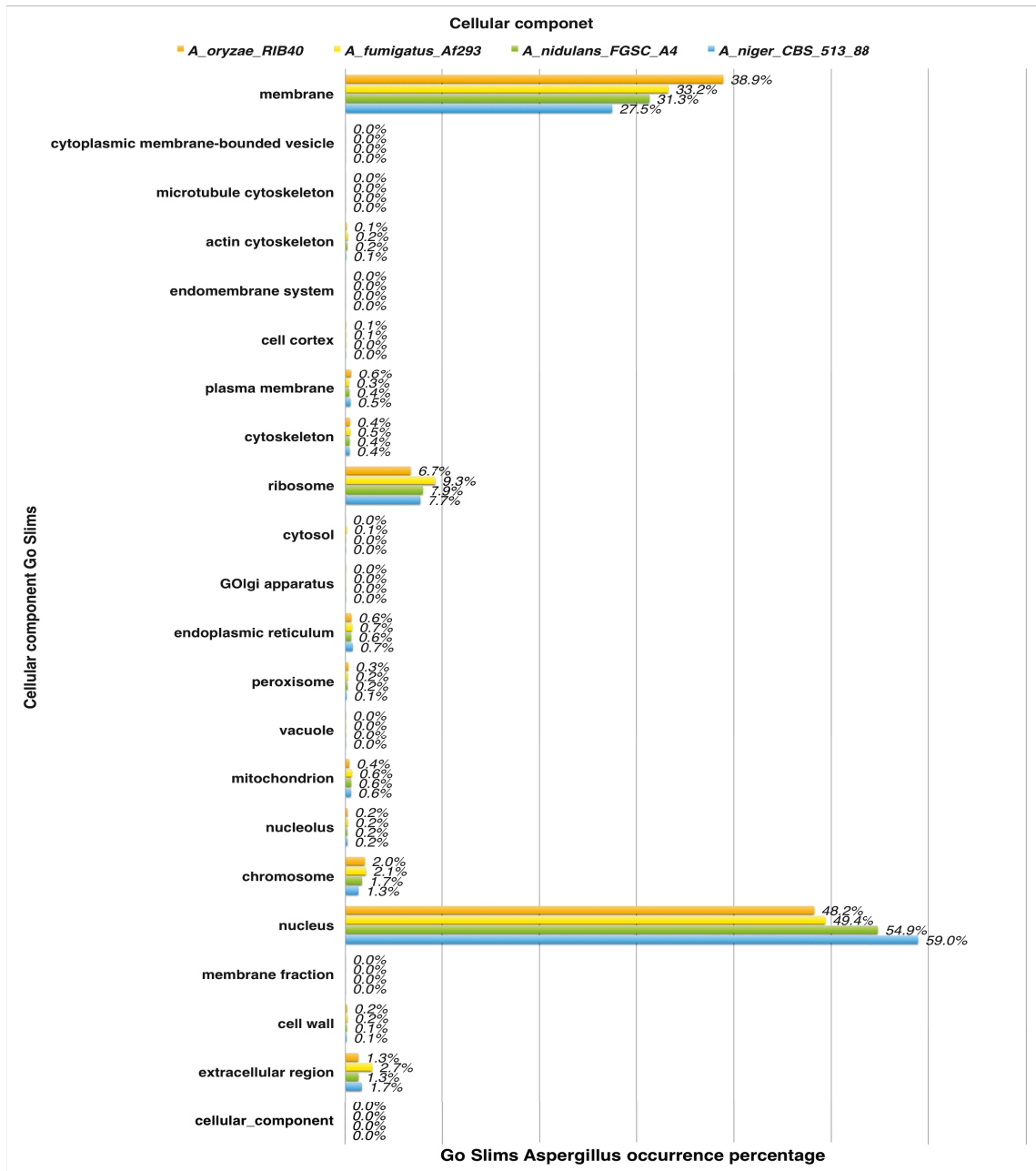


Figure 30: Chart shows Cellular Component Go Slims Aspergillus occurrence percentage in the specified genome. As Shown, *nucleus* GO Slim (ID: GO:0005634) has the highest percentage of occurrence for all the four genomes as 59.0%, 54.9%, 49.4% and 48.2% for *A. niger*, *A. nidulans*, *A. fumigatus* and *A. oryzae* respectively. The second highest percentage of occurrence for all the four genomes is *membrane* (ID: GO:0016020) with 27.5%, 31.3%, 33.2% and 38.9% for *A. niger*, *A. nidulans*, *A. fumigatus* and *A. oryzae* respectively. Some of the GO Slims has zero occurrence percentage for all of four genomes such as *endomembrane system* and *membrane fraction*.

Biological process	Go Slims Aspergillus	ID	Organism				
			A_niger_CBS_513_88	A_nidulans_FGSC_A4	A_fumigatus_Af293	A_oryzae_RIB40	
carbohydrate metabolic process		GO:0005975	875	907	941	943	
DNA metabolic process		GO:0006259	25	20	18	21	
transcription, DNA-templated		GO:0006351	598	523	441	553	
translation		GO:0006412	382	382	359	293	
protein folding		GO:0006457	172	236	248	221	
cellular protein modification process		GO:0006464	48	47	47	47	
cellular amino acid metabolic process		GO:0006520	96	89	81	102	
lipid metabolic process		GO:0006629	89	83	76	89	
vitamin metabolic process		GO:0006766	0	0	0	0	
transport		GO:0006810	383	285	241	364	
response to stress		GO:0006950	33	26	28	30	
organelle organization		GO:0006996	0	0	0	0	
nucleus organization		GO:0006997	0	0	0	0	
cytoskeleton organization		GO:0007010	7	13	13	8	
cell cycle		GO:0007049	31	29	30	29	
cell adhesion		GO:0007155	2	0	2	7	
signal transduction		GO:0007165	220	210	207	299	
biological process		GO:0008150	0	0	0	0	
toxin metabolic process		GO:0009404	0	0	0	0	
pathogenesis		GO:0009405	0	0	0	0	
RNA metabolic process		GO:0016070	16	14	15	15	
vesicle-mediated transport		GO:0016192	117	112	108	104	
cellular homeostasis		GO:0019725	2	1	1	1	
secondary metabolic process		GO:0019748	0	0	0	0	
protein catabolic process		GO:0030163	10	9	8	8	
asexual sporulation		GO:0030436	0	0	0	0	
filamentous growth		GO:0030447	0	0	0	0	
transposition		GO:0032196	0	0	0	0	
developmental process		GO:0032502	0	0	0	0	
sexual sporulation		GO:0034293	0	0	0	0	
establishment of nucleus localization		GO:0040023	0	0	0	0	
response to chemical		GO:0042221	0	0	0	0	
ribosome biogenesis		GO:0042754	18	17	16	17	
cellular respiration		GO:0045333	0	0	0	0	
regulation of biological process		GO:0050789	0	0	0	0	
establishment of vesicle localization		GO:0051650	0	0	0	0	

Table 6: Table shows Biological Process Go Slims Aspergillus occurrence in specified genomes. As Shown in the table, *carbohydrate metabolic process* GO Slim (ID: GO:0005975) has the highest number of occurrence for all the four genomes with 875, 907, 941 and 943 for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively. The second highest number of occurrence for all the four genomes is *transcription, DNA-templated* (ID: GO:0006351) with 598, 523, 441 and 553 for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively. Some of the GO Slims does not occur for all of four genomes such as *cellular respiration* while others occur in one of the genomes only such *pathogenesis* which occur twice in A. oryzae and does not occur in the others.

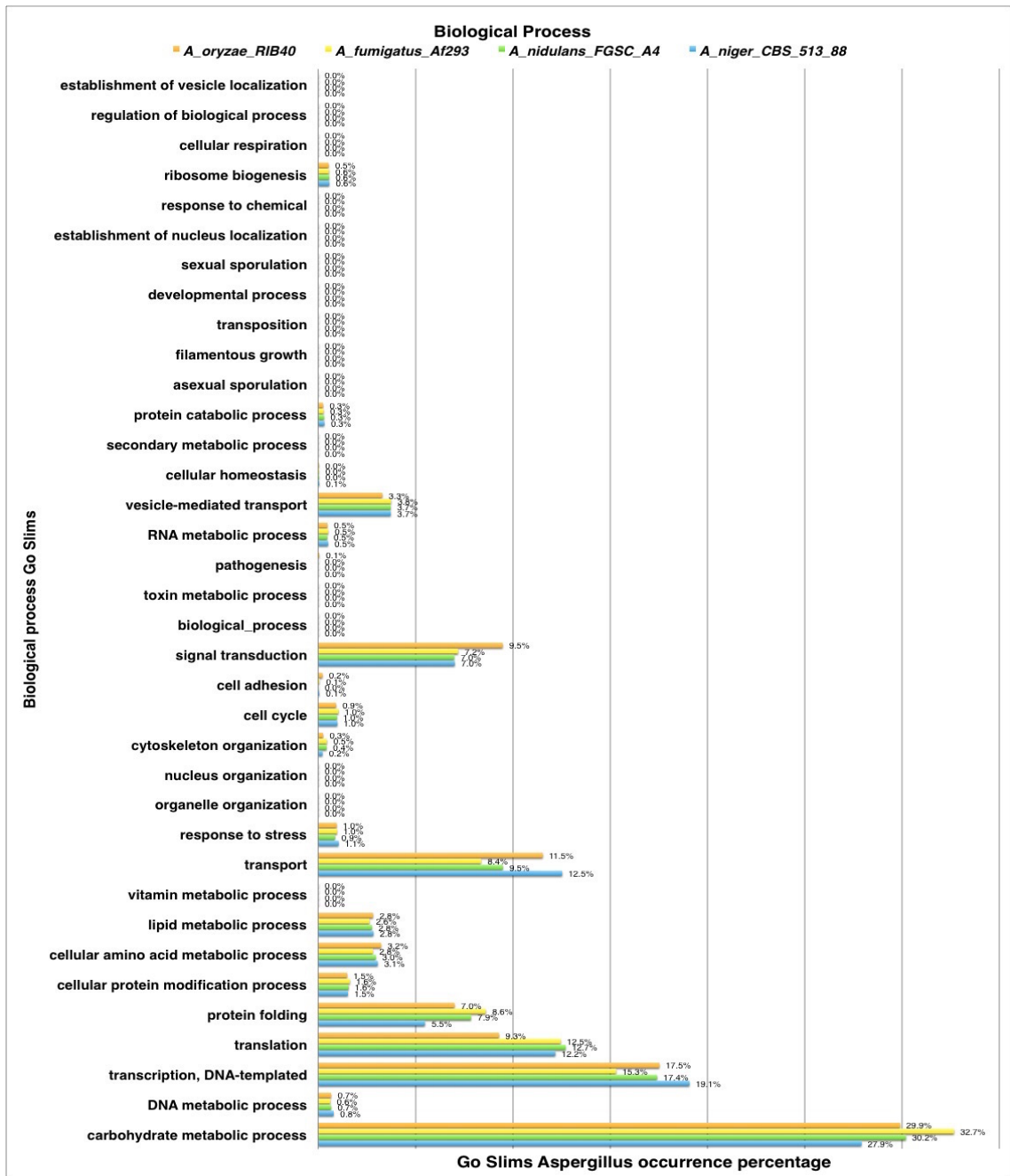


Figure 31: Chart shows Biological Process Go Slims Aspergillus occurrence percentage in the specified genome. As Shown, *carbohydrate metabolic proces* GO Slim (ID: GO:0005975) has the highest percentage of occurrence for all the four genomes as 27.9%, 30.2%, 32.7% and 29.9% for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively. The second highest percentage of occurrence for all the four genomes is *transcription, DNA-templated* (ID: GO:0006351) with 19.1%, 17.4%, 15.3% and 17.5% for A. niger, A. nidulans, A. fumigatus and A. oryzae respectively. Some of the GO Slims has zero occurrence percentage for all of four genomes such as *cellular respiration*.

3.5.2 Data Formats

As mentioned in Section 3.5, there are several data files that used in CGene and CGenome HTML5 visual spreadsheet system. These files are: Generic Feature Format (GFF) files, standard output files from InterProScan, AspGD file for AspGD Gene Ontology Annotations file, chromosomal feature file, Go Slims file and Interpro2go flat file. Some of these files are downloaded from Aspergillus website while others from different websites such as InterPro website.

In Aspergillus genome website (www.aspgd.org), there is download directory which has many sub-directories, where the data can be downloaded. The used sub-directories are: brows download/[gff/](#), brows download/[domains/](#), brows download/[chromosomal_feature_files/](#) and brows download/[go/](#).

Generic Feature Format (GFF) file

“GFF is a format for describing genes and other features associated with DNA, RNA and Protein sequences [Sanger-Institute, 2014].”

Gff files are found in download/[brows download/gff](#) sub-directory. This directory contains files that store data about several Aspergillus species in the *Generic Feature Format (GFF)*. Each GFF3 file has features description, which includes chromosomes, ORFs, CDSs, introns, sequence gaps, intergenic regions, etc. These files are parsed to extract the information about each gene and its features such as axons, gene ID, and gene name. The introns and the gene length are calculated from the available data. For more details about the Generic Feature Format (GFF) specification, please visit the URLs: (<http://www.sequenceontology.org/gff3.shtml>) and (<http://www.sanger.ac.uk/resources/software/gff/spec.html>).

Standard output files from InterProScan

These files are found in download/[brows download/domains/](#) directory. This directory has *Protein Domain Predictions* information that was predicted using *IprScan* software from the *InterPro* DB. Each file has InterPro domains information that related to *Aspergillus* genome such as systematic identifier of the input sequence, length of the sequence, analysis method, etc. All these informations are organized in tab-delimited file which each column represent feature. For more details about the InterPro database and IprScan output file, please visit the URL: (<ftp://ftp.ebi.ac.uk/pub/software/unix/iprscan/README.html>).

AspGD file for AspGD Gene Ontology Annotations

This file is found in download/[brows download/go/](#) directory. This directory consists of file that has all Gene Ontology (GO) annotations for AspGD genes. The file is tab-delimited file and has GO annotations related to *Aspergillus* genome. This file follows the standard file format of GO Consortium for gene_association files [GeneOntology, 2014]. For more details, please visit the URLs: (<http://geneontology.org/>) and (http://www.aspergillusgenome.org/download/go/gene_association_README.txt).

Chromosomal feature file

These files are found in brows download/chromosomal_feature_files/ directory. This directory consists of data for several *Aspergillus* species in chromosomal_feature.tab file format. Each file about one *Aspergillus* species, and has information about chromosomal features in AspGD such as feature name, aliases (synonyms), descriptions, type, chromosomal coordinates, etc. All this information is organized in tab-delimited file which each column represent feature. For more details, please visit the URL: (http://www.aspergillusgenome.org/download/chromosomal_feature_files/).

Go Slims file

Go Slims that used for Go Slims bitmap and Go Slims histogram in **CGenome** spreadsheet are *Aspergillus Slims* which selected from [GeneOntology, 2014] website (<http://geneontology.org/page/go-slim-and-subset-guide>). *Aspergillus slim* includes terms that mainly beneficial when annotating *Aspergillus* while the generic slim is used for most applications without species preference.

Interpro2go flat file

This file has a list of Gene Ontology (GO) terms, which are mapped to InterPro entries from InterPro website. *Interpro2go* file is used in this research to count the InterPro domain appearance in the *Protein Domain Predictions* file based on the GO terms. To downloads, see: (<http://www.ebi.ac.uk/interpro/download.html>).

3.6 Data File Preprocessing

In order to achieve the wanted functionality from CGene and CGenome, the gathered files from AspGD DB (<http://www.aspergillusgenome.org>) should be analyzed. Below is general description of the analysis that done on the files.

- **Code:** *GFFParser.java*

Input: .gff file that contains features of the genome assembly in Generic Feature Format (GFF) from AspGD DB (<http://www.aspergillusgenome.org> for a specific organism).

Output: .txt file for the entered organism called OrganismNameGFFExtracted.txt.

Functionality: extract some of the features from the GFF file, then create a new file which has the desired features for the system from the GFF file.

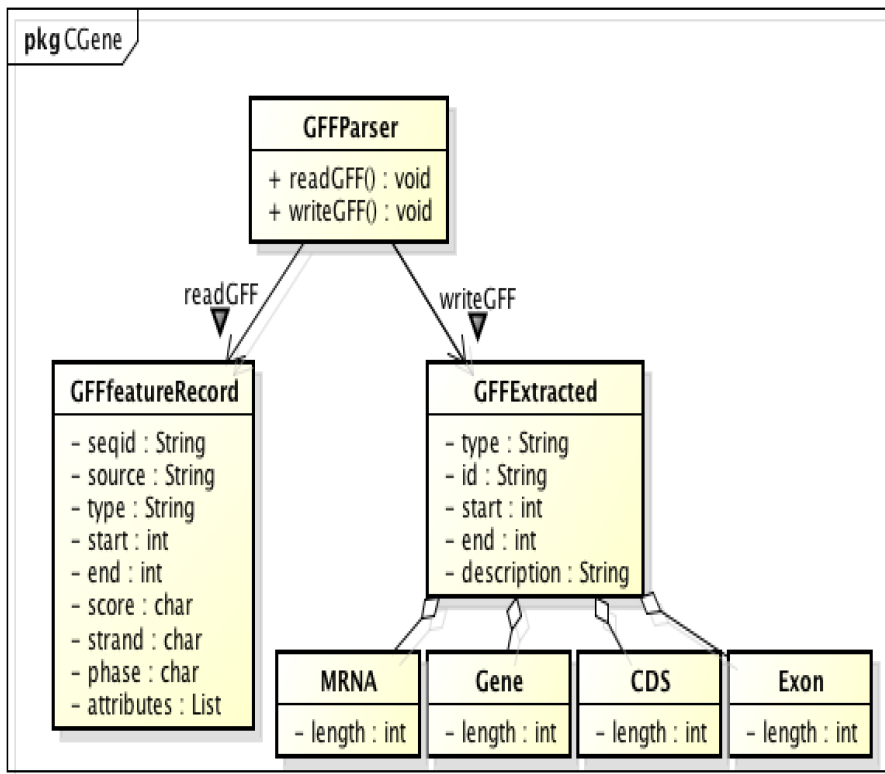


Figure 32: GFFParser.java code UML diagram.

- **Code:** *ProteinDomainParser.java*

Input:

- .out file from domains directory in **AspGD** DB (<http://www.aspergillusgenome.org>) for a specific organism.

Output: .txt file for the entered organism.

- OrganismNameExtractedIprScanOutputFile.txt

Functionality: extract the desired features from IprScan Output File and create new .txt file which has the extracted features from the domain file.

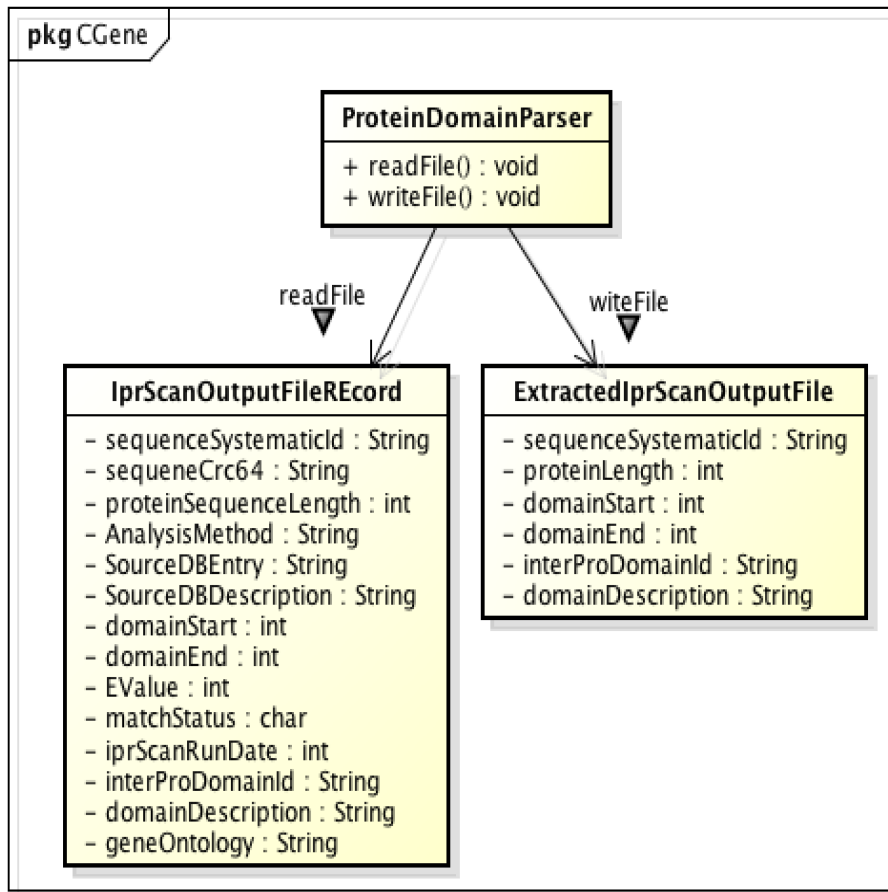


Figure 33: ProteinDomainParser.java code UML diagram.

- **Code:** *GeneProteinLinking.java*

Input: the two resulted .txt files from *ProteinDomainParser.java* and *GFFParser.java* for a specific organism.

1. OrganismNameExtractedIprScanOutputFile.txt
2. OrganismNameGFFExtracted.txt

Output: FeaturesLinkedFile.txt

Functionality: join the tow files by linking each gene in the (*GFFExtracted.txt*) file with its associated protein domains in the (*ExtractedIprScanOutputFile.txt*) domain file if the linked protein is available. Then place them in a single (*FeaturesLinkedFile.txt*) file.

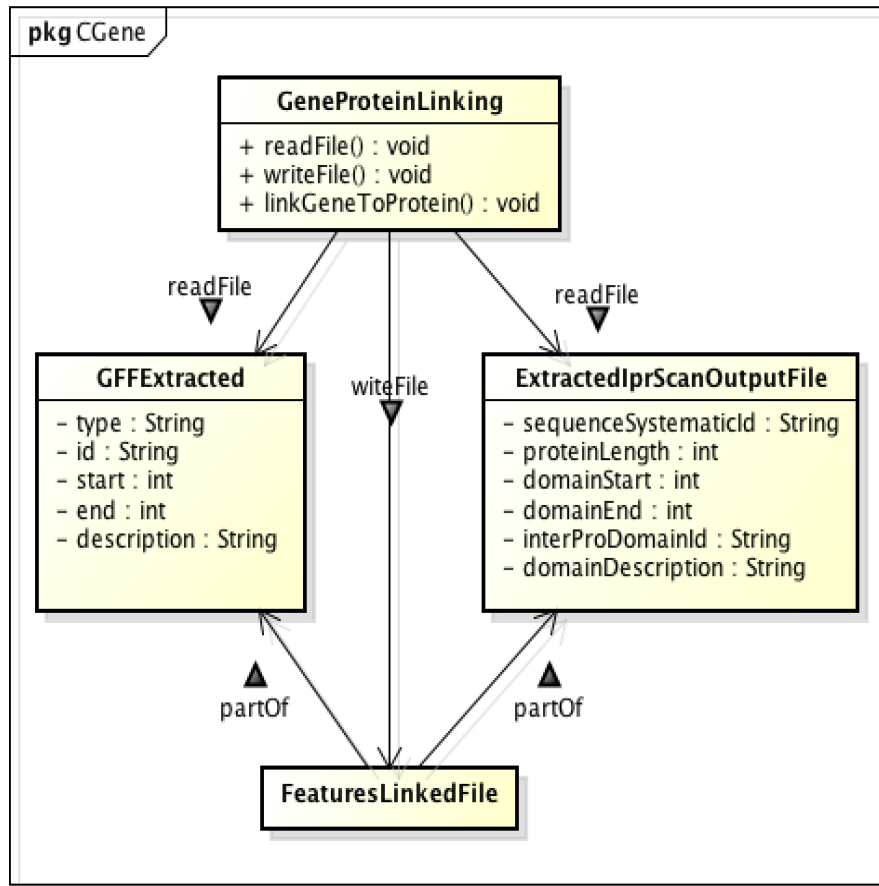


Figure 34: GeneProteinLinking.java code UML diagram.

- **Code:** *GeneModelSVG.java*

Input: (OrganismNameGFFExtracted.txt) file that created by *GFFParser.java* for a specific organism.

Output: multiple *gene model* SVG files.

Functionality: gene model SVG file is created for each gene in the entered organism.

The naming schema is:

- For gene model SVG: OrganismName_GeneID_SVGFile.svg

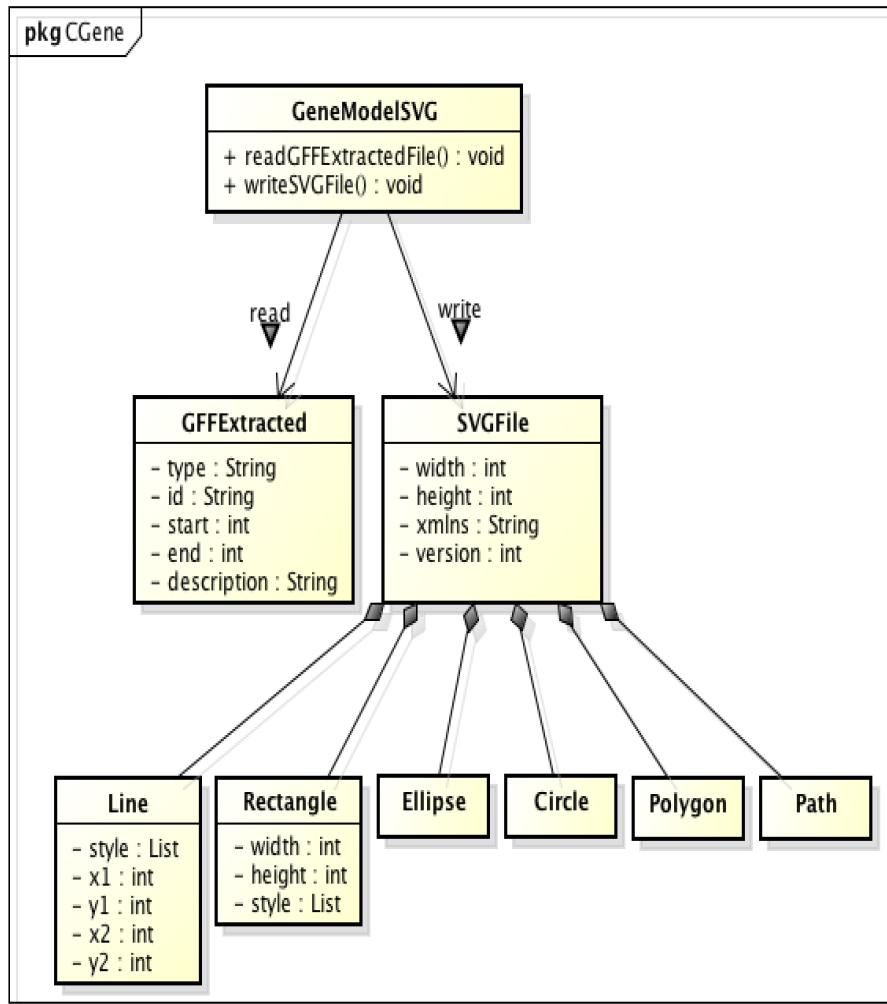


Figure 35: GeneModelSVG.java code UML diagram.

- **Code:** *DomainArchitectureSVG.java*

Input: ExtractedIprScanOutputFile.txt file that created by *ProteinDomainParser.java* for a specific organism.

Output: multiple *protein domains* SVG files.

Functionality: protein domain architecture SVG file is created for each protein.

The naming schema is:

- For protein SVG: OrganismName_GeneID_Dom_SVGFile.svg

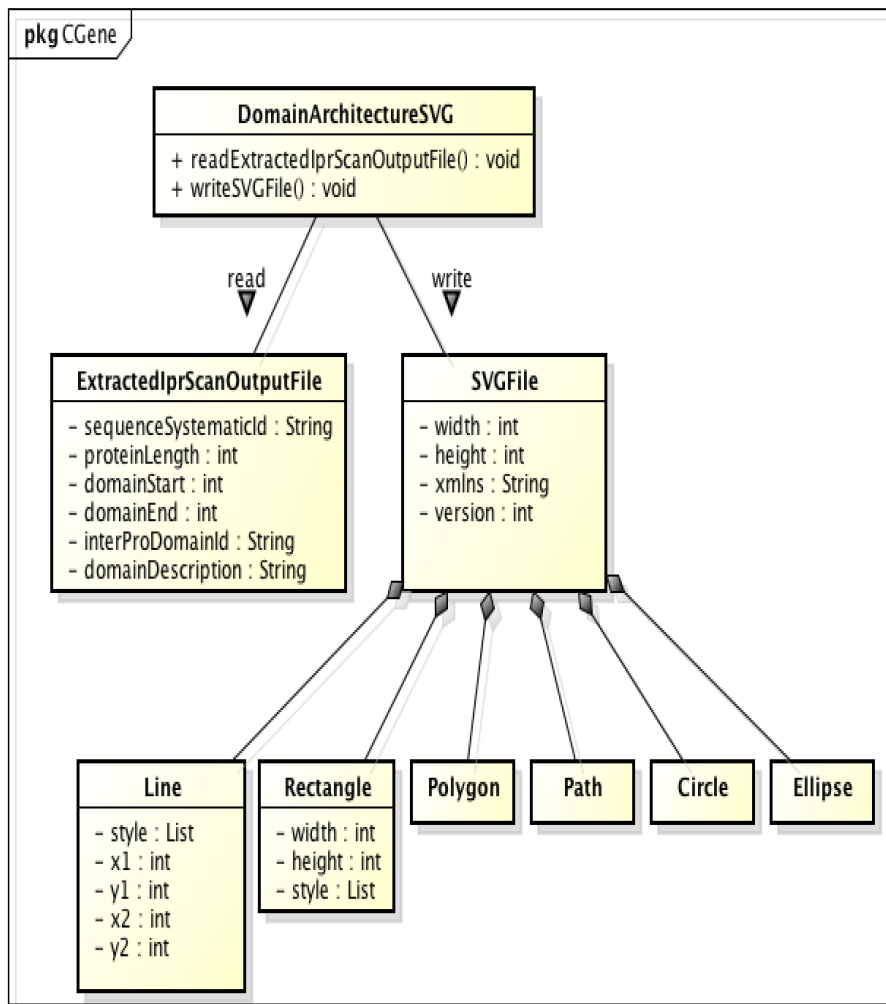


Figure 36: DomainArchitectureSVG.java code UML diagram.

- **Code:** *ChromosomeLocationParser.java*

Input: .txt file that contains chromosome features for a specific organism from AspGD DB (<http://www.aspergillusgenome.org>). **Output:** .txt file for the entered organism.

– OrganismNameChromosomeLocationExtracted.txt

Functionality: extract some of the features from the file, then create a new file which has the desired features for Genome system.

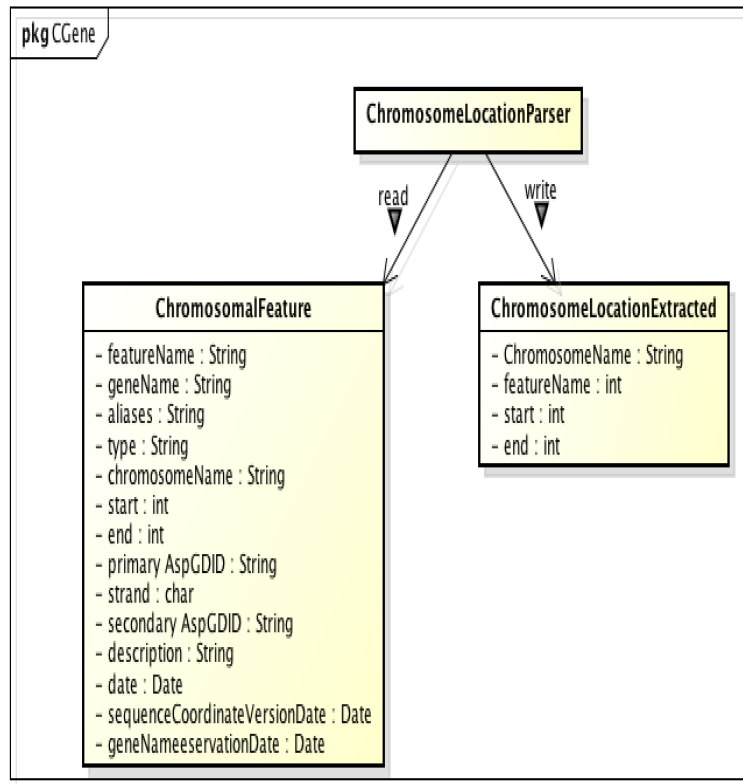


Figure 37: ChromosomeLocationParser.java code UML diagram.

- **Code:** *CreatHTMLtable.java*

Input: FeaturesLinkedFile.txt file that created by *GeneProteinLinking.java* for a specific organism.
Output: OrganismName_HTMLTable.txt
Functionality: create the whole HTML5 table syntax for the entered organism.

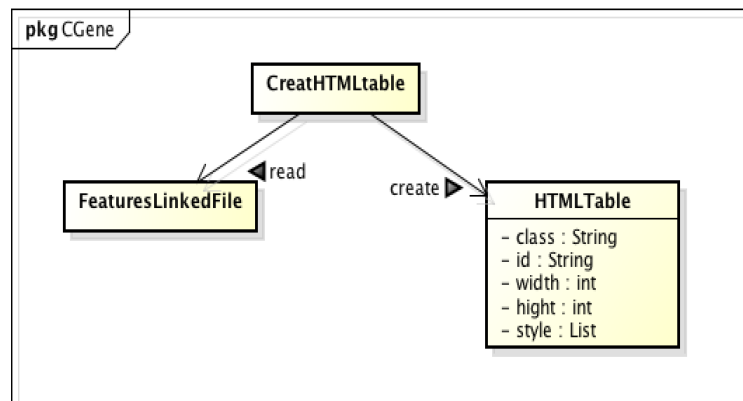


Figure 38: CreatHTMLtable.java code UML diagram.

3.7 Why HTML5 and SVG?

3.7.1 HTML5

HyperText Markup Language v.5 (HTML5) is a modern markup language that used to create web pages, but *HTML* roots go back to the early 90's. There are several versions appeared before HTML5, and the latest one was HTML 4.01. From v.4.01, the language has been improved significantly to support the latest multimedia. In 2004, Web Hypertext Application Technology Working Group (WHATWG) started to develop the specification for HTML5 and they began to add some extra features to support the web applications. Two years later, World Wide Web Consortium (W3C) cooperated with WHATWG, and their core aim was to propose a substitution for HTML 4, XHTML, and the HTML DOM Level 2. They published together the first working draft for the new version in 2008 [HTML5, 2014].

HTML5 developers aimed to create a strong language that satisfies four basic principles, which are: compatibility, utility, interoperability and universal access [Pilgrim, 2010]. The first thing that they care about while adding new features is keeping everything move smoothly and compatible with the available content. Even though new standards were added, they were developed in a way that close to which the user adapted with previously to make it easy to switch to the new HTML5 elements. For example, several of webpages analyzed by Google, and most of the web designers used to declare a DIV element with *id="header"* each time they want to create a header content. In HTML5, the redundancy issue is solved, and separate `<header>...</header>` element is added to make things much simpler.

In addition to compatibility, HTML5 developer put the user as their main priority over the other parties such as the browsers or the specifiers themselves. Their desire was to produce a simple and useful language as much as they could. Some of the previous features are supported such as using the CSS to spread the idea of separation between the code content and the web presentation. Using feature like CSS will increase the simplicity and reduce the redundancy, which may cause slower loading pages and hardness in code reading. Moreover, HTML5 designed to be accessed universally. Consequently, its functionality should be device and platform independent. HTML5 offers the users the ability to create dynamic, rich and interactive experiences for any platform and device with a web browser. That's why many famous companies use HTML5 such as Netflix in for its User Experiences on Devices. It also supports most of the world languages. For this matter, Ruby annotations that used in East Asian typography is supported in a new `<ruby>` element [Pilgrim, 2010].

In addition to the previous features, reducing the need for multimedia external plugins is one of the strongest features that have been supported strongly in HTML5. External plugins like *Flash* have many problems since many of them cannot be installed, may be blocked or disabled by some devices and browsers. Plugins also are difficult to integrate due to their boundaries, clipping and transparency problems.

To minimize the need for the plugins, new elements were added to perform new functionalities that not supported before HTML5. These elements such as Scalable Vector Graphics(SVG) element can be styled using CSS and scripted using JavaScript. Although SVG was available previously

as a web plugin, its first introduction as HTML5 element was in W3C recommendation in 2001. Another example for reducing the need for multimedia plugins is watching a video on the webpage. In HTML5, a video frame can be added in a `<canvas>` element, and the user just click in this frame and play it. This functionality can be done in the native code without using the plugins to play the video as supposed previously [Pilgrim, 2010].

Why HTML5? In addition to: simplicity, compatibility, the universal access and the other stated features, HTML5 supports many JavaScript libraries perfectly. These libraries help biological visualization in different ways such as *D3.js* which displays dynamic graphical forms in the web browsers. Moreover, it supports Scalable Vector Graphics (SVG) natively which is started to be used widely in biological visualization. More details about JavaScript and SVG will be stated in Section 3.7.2 and in Section 3.7.3.

Before concluding this section, there is a some of key features that are supported in HTML5 (<http://www.w3schools.com>). This features include Supporting graphics and media such as Scalable Vector Graphics(SVG), canvas (2D and 3D), audio and video. it also supports WebSocket API and protocol, indexed database and web storage, forms, micro-data, MathML, drag and drop.

3.7.2 SVG

Scalable Vector Graphics (SVG) is a vector image format that used to describe two-dimensional graphics, particularly it is used for displaying images in the Web browsers. It is based on Extensible Markup Language (XML) rules for encoding documents. The first full specification as HTML5 element was initiated in W3C recommendation in 2001. Before HTML5, SVG can be embedded in `` element or as a link to *.SVG* document in order to show the SVG image in the webpage. Now, it is part of HTML5 elements with `<svg>...</svg>` tags at the beginning and the end respectively.

As the name of SVG indicates, it is a vector graphic, which is distinctive from other formats such as JPEG, PNG and GIF that are raster graphics. In order to understand the importance of the Scalable Vector Graphics in web graphics, we need to differentiate between the *vector graphics* and one of most common image graphics format, which is *raster images*.

Raster vs. Vector A *raster image*, or bitmap, is made up of rectangular pixels that represented in a grid of x and y coordinates. All these coordinates have the same size with the same or different colors. Pixels colors are represented in the screen with a numerical value such as *RGB*, which consists of three numbers to calculate its color. The amount of the numbers that needed to calculate the colour is different depending on the type of image such as eight bits for *gif* image. The *Canvas 2D API*, one of HTML5 elements, is an example of raster graphics. Since the raster images consist of fixed number of pixels, it cannot be scaled up. Otherwise, the edges will look pixelated and not clear as seen in Figure 39, which lost some of its details. In order to minimize this problem, the resolution should be increased which increase the size of the file consequently.

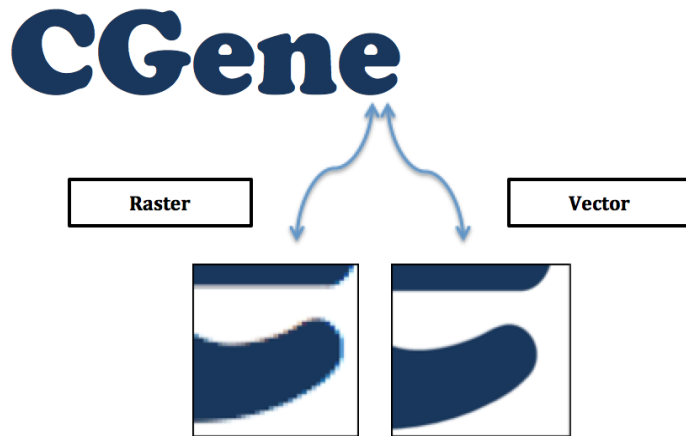


Figure 39: Illustration of the effect of magnification on vector graphics versus raster graphics. The vector-based is at the right and the raster-based at the left. (Credit: the conversion from bitmap to SVG is done in <http://vectormagic.com/home> website)

On the other hand, *vector graphics* are represented with mathematical descriptions of geometry. The images basically consist of sufficient vector points that connected by lines then filled with different colours to create diversity of closed shapes. Each point has information that guide the computer how to connect this point with other points in a straight or curved lines as seen in Figure 40. This feature makes the editing of the vector images much easier than raster one. To resize the vector images, the vector points distribute to fit the new size, and the computer just reconnect the image lines and fill the colour again.

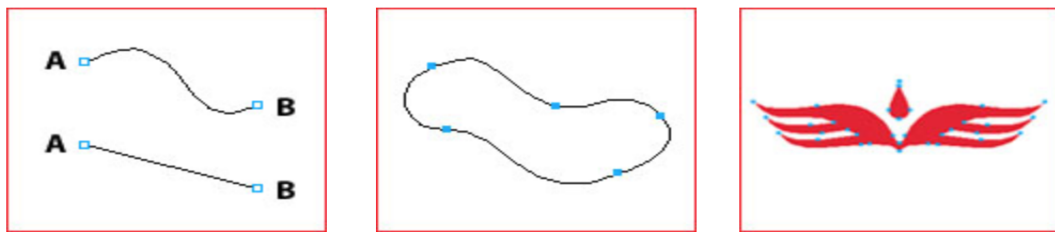


Figure 40: Vector graphics concept illustration. The left image shows two points A and B that are linked in different ways depending on the descriptive information in the SVG file. The middle image shows five points connected together to create the shape. The right one shows enough points to make a specific image. (Credit: these images are taken from http://freerangestock.com/understanding/vector_bitmap/Part2_Vector.html website)

3.7.2.1 Basic Vector Shapes

SVG has six basic shape elements: `<line>`, `<circle>`, `<ellipse>`, `<rect>`, `<polyline>`, and `<polygon>`. Each one of these shapes has its values and attributes that enable controlling them [Dailey et al., 2012]. All the shapes that stated in the examples below is shown in Figure 41.

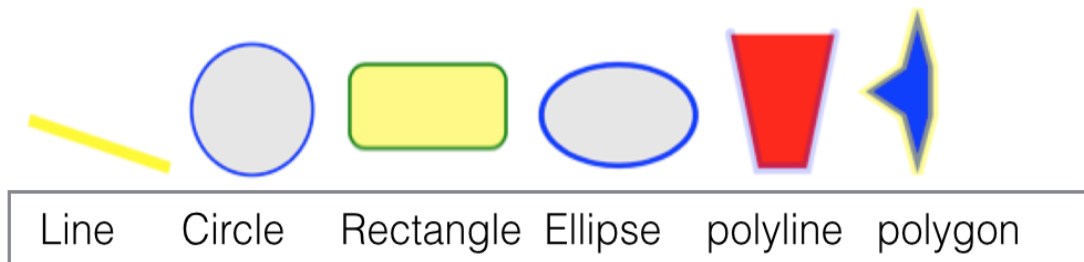


Figure 41: Basic vector shapes for SVG.

Line: has $(x1,y1)$ values as starting coordinate from the upper-left corner of the screen which is $(0,0)$, and $(x2,y2)$ values as ending coordinates. For instance:

`<line x1="10" y1="10" x2="150" y2="50" stroke="yellow" stroke-width="10" >`. This means the line starting coordinate from the upper-left corner of the screen will be at $(10,10)$ point while the ending coordinate point will be at $(150,50)$. The colour of the line is yellow with 10 pixels width.

Circle: has (cx,cy) values for the centre and r for the radius. For instance:

`<circle cx="150" cy="150" r="100" stroke="blue" stroke-width="5" fill="grey" fill-opacity="0.2">`. This means the circle centre will be at $(150,150)$ point from the upper-left corner of the screen, with 100 pixels radius. The circle outside stroke colour is blue with 5 pixels width. The inside filling is grey with .2 filling opacity.

Rectangle: has (x,y) values as top-left corner coordinate and $width$, $height$ values for the rectangle width and height. In addition, there are rx and ry to declare the corners rounding degree. For instance:

`<rect x="50" y="100" width="100" height="50" rx="10" ry="10" stroke="green" stroke-width="2" fill="yellow" fill-opacity="0.6" >`. This means $(50,100)$ point is the top-left corner coordinate of the rectangle, and 100, 50 are width and height values for the rectangle width and height respectively. In addition, 10 pixels are the corners rounding degree. The rectangle outside stroke colour is green with 2 pixel width. The inside filling is yellow with .6 filling opacity.

Ellipse: has (cx,cy) values for the centre and (rx,ry) for (x,y) radius values. For instance:

`<ellipse cx="150" cy="150" rx="50" ry="70" stroke="blue" stroke-width="5" fill="grey" fill-opacity="0.2">`. This means the ellipse centre will be at $(150,150)$ point with 50 and 70 as x,y radius respectively. The ellipse outside stroke colour is blue with 5 pixels width. The inside filling is grey with .2 filling opacity.

Polyline: it is similar to the line shape but with chain of the x,y coordinates which will be drawn with open end unless the user add the closing line. For instance:

`<polyline points="200,60 240,230 310,230 350,60" fill="red" stroke="blue" stroke-width="15" stroke-linecap="round" stroke-opacity="0.2" >`. This means the polyline starting coordinate

from the upper-left corner of the screen will be at (200,60) point. This line will be connected with other line with the following coordinates respectively: (240,230), (310,230) and (350,60) at the end as ending coordinate point. The colour of the polyline stroke is blue with 15 pixels width. The stroke has round linecap with .2 opacity. The filling color between the starting and ending points is red.

Polygon: has $(x1,y1)$ values as starting coordinate and $(x2,y2)$ values as ending coordinates which will be drawn with closed ends automatically. For instance:

```
<polygon points="100,50 115,120 115,180 100,250 85,180 50,150 85,120" fill="blue" stroke="yellow" stroke-width="10" stroke-opacity="0.5" stroke-linejoin="miter">
```

This means the polygon starting coordinate from the upper-left corner of the screen will be at (100,50) point. This line will be connected with other line with the following coordinates respectively: (115,120), (115,180), (100,250), (85,180), (50,150), (85,120) and (350,60) at the end as ending coordinate point. The colour of the polygon stroke is yellow with 10 pixels width. The stroke has a miter shape to joint between two line segments with .5 opacity. The filling color is blue.

3.7.2.2 Why SVG?

SVG over the other image formats has many advantages. One of them comes from being a vector based, which is having a smaller file size. Therefore, SVG can be loaded and transferred across the Internet more quickly than other formats which makes SVG suitable for the web application that has massive images representation such as complex biological networks or trees. In addition to speed of transfer, SVG text is searchable. While text inside the raster file is joined with the image and considered as apart of the image itself, the text information inside the SVG images is searchable. For this matter, the text in Google indexes is saved in SVG format in the web [Pilgrim, 2010]. This feature is beneficial in a genome studies. For example, searching in the web about specific protein domain that drawn as a text inside SVG image can show the protein domain architecture images for this domain.

In web graphics design, the choice of the type of multimedia images is depending a lot on the application itself and the main purpose of this image. In the biological images, it is important to resize and zoom-in/out some figures to see how much details that have as seen in Figure 42. When raster images such as canvas 2D is used, the image should has high resolution with a large file size to prevent the pixelated edges. As a result, page-loading time is affected negatively. For this matter, SVG images will fit perfectly on the applications that give the resize and re-colour features high priority.

To conclude, SVG is a better choice for the biological applications that required high resolution than any other raster images for many reasons. One of the most important reasons is the easiness of scaling the images in SVG format at any monitor, small as cell phones or big as TV monitors, without fixed pixel size limitation. In contrast, modifying raster images are problematic, and if it happened the images will loose a lot of details after scaling [Peng and Zhang, 2004]. The other reason is that SVG file is based on XML file, which supports portability. That's mean the SVG

image can be displayed and edited at any platform and environment regardless of any web browser or operating system.

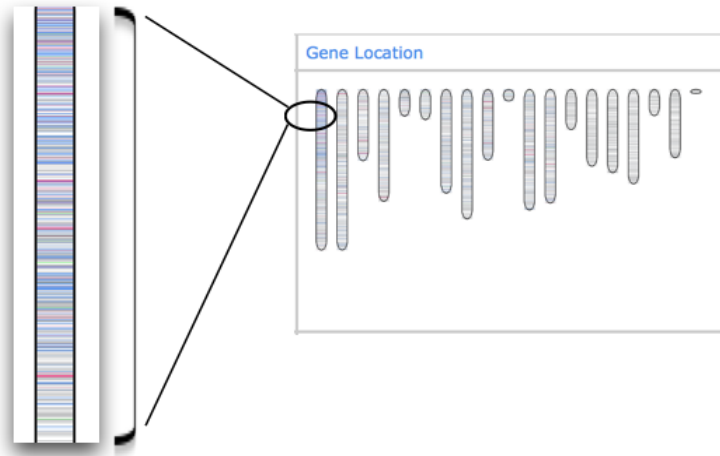


Figure 42: The image draws gene locations that locate in chromosome using scalable vector format as part of CGenome spreadsheet. It shows a small portion of gene location after scaling which is scaled in high quality without affecting the image clarity.

3.7.3 JavaScript

Unlike old-style static websites, modern websites should offer the user a full interaction features with the page contents. For that purpose, an interactive and strong programming language that has more flexible characteristics is becoming something desirable more than before.

Today, JavaScript (JS) is one of the most popular dynamic programming languages that mainly used with web browsers. Many computer-programming languages, *JavaScript* in our case, support the client-side scripts for directing the browser, enabling user interaction, displaying/altering dynamic document content. In addition to the client-side support, JavaScript supports the following: server-side programming using *node.js* application, mobile application creation, and game development [Flanagan, 2011].

Back to the history, JavaScript was created by Netscape, which is now called Mozilla Foundation. It is one of the multi-paradigm languages that influenced by: C, Java, Python and scheme languages. It supports imperative, functional, scripting and object-oriented (OO) programming styles. JavaScript has the basic API for working with the following regular terminologies: text, arrays and dates. Since JavaScript does not include any input/output functionality, the I/O function is the responsibility of the host that embed the JavaScript codes, which particularly is a web browser such as HTML5. Using JavaScript makes the browser's responding to the user action much faster than other languages because the code runs in the client's browser not the server-side.

JavaScript is commonly used with HTML as embedded script with the element `< script >` at the beginning of the script and ended by `< /script >` element such as the below example.

```

1 <script> //----To delete selected rows ----
2 onLoad : function deleteSelectesRow(tableID) {
3     try {
4         var table = document.getElementById(tableID);
5         var rowCount = table.rows.length;
6         for(var i=0; i<rowCount; i++) {
7             var row = table.rows[i];
8             var chkbox = row.getElementsByTagName('input')[0];
9             if(null != chkbox && true == chkbox.checked) {
10                table.deleteRow(i);
11                rowCount--;
12                i--;
13            } }
14        } catch(e) {
15            alert(e);
16        } }
17 <\script>

```

This embedded code used to supplement the client-side behaviour of HTML pages by interacting with the HTML Document Object Model (DOM).

Why JavaScript? There are many scripting languages that can be used but JavaScript is selected for having some important features. First feature, all modern browsers support and implement JavaScript natively such as embedding JS on HTML, described in the previous paragraph. Second, using JavaScript open the gates to take the advantages of using many popular and powerful JavaScript libraries that support visualization, such as *jQuery* and *D3.js*. Finally, JavaScript enhances the speed of the web browsers because its implementation does not require an internet connection since it uses its own Graphical User Interface (GUI). The others require plugins which needs online connectivity in order to generate the tree server-side.

3.7.4 Why HTML 5, SVG and JavaScript?

As known, SVG has many features, and the most important features in this research are having the ability to zoom-in/out without losing the resolution, having small file size, having interactive feature, and having the support from almost all the web browsers except Internet Explorer v8 and older. Moreover, SVG elements can be controlled by CSS and JavaScript code in HTML5 environment [Smits and Ouverney, 2010].

In biological web-application, the usage of JavaScript with some markup languages offers the web-designers a complete and stable platform for building interactive webpages. For instance, **HTML5** can be used as a template to identify the content of web pages, CSS to indicate the style that contents can be presented in the web pages, **SVG** to display and edit the genome resizable images, and **JavaScript** to control how the webpages contents will act [Flanagan, 2011].

3.8 Graphical User Interface


In our system, the master page has links for two kinds of spreadsheets. The links for *CGene* are divided into two kinds, which are genome spreadsheet with normalized images while the other is for the images that reflect the real coordinates of the exons, introns and protein domains. For both normalize and unnormalized links, there is a visual spreadsheet to present all genes in a specific genome. For each single gene, many properties are displayed such as gene ID, gene length, gene model, protein length and the protein domain architecture. For *CGenome*, there is only one link that directs to the comparative spreadsheet. In this spreadsheet, a comparative analysis for four different genomes is done by using three different visualizations, which shown in separate columns. The Gene location column is used to show the gene location on each chromosome. In the Go Slims bitmap column, the GO Slims occurrence in the specified genome is declared. Lastly, the GO Slims histogram column represents the number of occurrence of each GO Slim term.

The next Figures 43, 44, 45 and 47 are illustration of the outcome of our visual spreadsheet system using HTML5 for the whole genome.

In Figure 43 is screenshot of the master page of the visual spreadsheet system using HTML5 for the whole genome. As we see there are multiple links for two different kind of spreadsheets: *CGene* and *CGenome*. In the upper box there is spreadsheet that has multiple links for *CGene* pages. In the column called “genome spreadsheet with normalized images” there is multiple links for different genomes. When these links are pressed, they will show spreadsheets that have normalized lengths images. On the other hand, in the column called “genome spreadsheet with real length images” there is multiple links for different genomes. When these links are pressed, they will show spreadsheets that have real lengths images. In the lower box there is a single link for *CGenome* page.

In Figures 44 and 45 are the two kind spreadsheets of *CGene*. In Figure 44 is spreadsheet with normalized images of the gene model and protein domains. On the other hand, in Figure 45 is the other kind of *CGene* spreadsheet, where all the images reflect the real coordinates of the exons, introns and protein domains. For both spreadsheets gene length and protein length are presented as number reflect the real length. The gene ID is represented textually. Any row can be selected using the check box in order to enable the user to have limited dataset. Figure 46 is a snapshot of *CGene* spreadsheet where it possible now to display more than > 10000 genes in single spreadsheet. This spreadsheet has normalized SVG images gene model and normalized protein domain architecture images.

Figure 47 shows a screenshot of single row of the comparative spreadsheet, which is for *Aspergillus niger* CBS 513.8 genome. In this spreadsheet, there is a comparative analysis for four different genomes which are *Aspergillus fumigatus* Af293, *Aspergillus nidulans* FGSC A4, *Aspergillus niger* CBS 513.8 and *Aspergillus oryzae* RIB40. This analysis is done using three different visualizations, which shown in separate columns. The Gene location column is used to show the gene location on each chromosome. In the Go Slims bitmap column, the GO Slims occurrence in the specified genome is declared. Lastly, the GO Slims histogram column represents the number of occurrence of each GO Slim term.



Home About CGene Help Contact Us

Genome

	Genome spreadsheet with normalized images	Genome spreadsheet with real length images
A acidus CBS 106.47	Normalized	Not normalized
A aculeatus ATCC16872	Normalized	Not normalized
A brasiliensis CBS 101740	Normalized	Not normalized
A carbonarius ITEM 5010	Normalized	Not normalized
A clavatus NRRL 1	Normalized	Not normalized
A flavus NRRL 3357	Normalized	Not normalized
A fumigatus Af293	Normalized	Not normalized
A nidulans FGSC A4	Normalized	Not normalized
A niger CBS 513.88	Normalized	Not normalized
A acidus CBS 106.47	Normalized	Not normalized
A oryzae RIB40	Normalized	Not normalized

CGenome

Press the links to go to [CGenome](#) page.

Figure 43: Master page of the visual spreadsheet system using HTML5 for the whole genome. As we see there are multiple links for two different kind of spreadsheets: CGene and CGenome. In the upper box there is spreadsheet that has multiple links for CGene pages. In the lower box there is a single link for CGenome page.





<input type="checkbox"/>	Gene ID	Gene length	Gene model	Protein length	Protein Domain architecture
<input type="checkbox"/>	Aspfo1_0027407	1338			
<input type="checkbox"/>	Aspfo1_0027447	701			
<input type="checkbox"/>	Aspfo1_0027472	1667		535	

Figure 44: *CGene* spreadsheet with normalized SVG images where gene model and protein domain architecture images are all normalized.





<input type="checkbox"/>	Gene ID	Gene length	Gene model	Protein length	Protein Domain architecture
<input type="checkbox"/>	Aspfo1_0027407	1338			
<input type="checkbox"/>	Aspfo1_0027447	701			
<input type="checkbox"/>	Aspfo1_0027472	1667		535	

Figure 45: *CGene* spreadsheet with unnormalized SVG images where gene model and protein domain architecture images are all reflect their real lengths.

Gene ID	Gene length	Gene model	Protein length	Protein Domain architecture
Aspfo1_0027407	1338			
Aspfo1_0027447	701			
Aspfo1_0027472	1667		535	
Aspfo1_0027483	3241		972	
Aspfo1_0027486	4250		1400	
Aspfo1_0027487	3564		907	
Aspfo1_0027501	1984		439	
Aspfo1_0027504	2408			
Aspfo1_0027563	560		186	
Aspfo1_0027564	5428		1761	
Aspfo1_0027571	1064			
Aspfo1_0027594	766		186	
Aspfo1_0027617	2069			
Aspfo1_0027638	1418		444	
Aspfo1_0027657	1098		346	
Aspfo1_0027659	1348		384	
Aspfo1_0027663	1008		306	
Aspfo1_0027665	1734		521	
Aspfo1_0027676	3398		1113	
Aspfo1_0027677	1777		548	
Aspfo1_0027681	644			
Aspfo1_0027697	422		140	
Aspfo1_0027701	1713		505	
Aspfo1_0027714	1254		326	
Aspfo1_0027752	1640		546	
Aspfo1_0027764	2045			
Aspfo1_0027792	1349			
Aspfo1_0027860	2086		673	
Aspfo1_0027870	904		225	
Aspfo1_0027874	3969		1252	
Aspfo1_0027887	1320		396	
Aspfo1_0027888	461			
Aspfo1_0027900	1296		410	
Aspfo1_0027906	2123		707	
Aspfo1_0027924	1047			
Aspfo1_0027949	2354		749	
Aspfo1_0027953	2273		694	
Aspfo1_0027955	1299		357	
Aspfo1_0027956	1493		419	
Aspfo1_0027976	1850		538	
Aspfo1_0027988	1665			

Figure 46: Snapshot of *CGene* spreadsheet where it possible now to display more than > 10000 genes in single spreadsheet. This spreadsheet has normalized SVG images gene model and normalized protein domain architecture images.

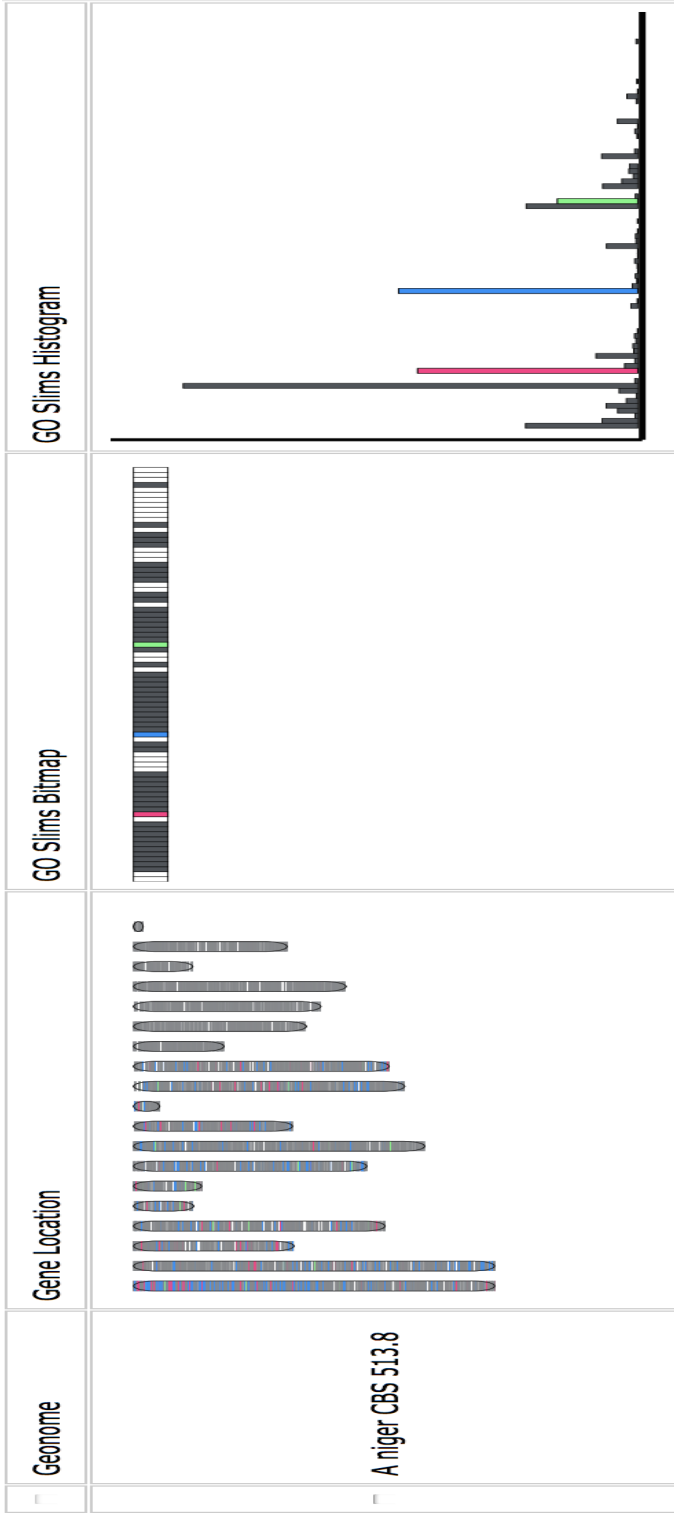


Figure 47: *CGenome* spreadsheet where a comparative analysis for different genomes is done by using three different visualizations which are Gene location column, Go Slims bitmap column and GO Slims histogram column.

Chapter 4

Conclusion and Future Work

Visualization tool for presenting the whole genome annotation data is becoming more pressing, especially with the exponential and rapid development of genomic data. Analyzing the visual display of the gathered annotations enables better understanding for the genome and organism development. For the assessment, many scientists prefer observing the genome at high level, which is the level of an exon, a gene, and a biochemical pathway. Consequently, there are many special tools that created for visualizing the gene structure and its encoded regions, but many of them have limited visualizing options. For example, some of the tools are presenting single gene with its products in single view, while others are presenting limited number of genes in specific location on the genome such as genome browsers, which makes the whole genome assessment difficult. Consequently, there is a need to present all the genes in the genome associating with its encoded protein domains if found.

4.1 Summary of Works

In our work, we have implemented CGene and CGenome, pronounced See-Gene and See-Genome respectively, as a HTML5 web-based spreadsheets system. That makes the visualization of the whole genome annotation data in integrated visual displays, as well as text, within the spreadsheet cells possible.

We first analyzed different annotation files from the AspGD database (www.aspgd.org) and InterPro (<http://www.ebi.ac.uk/interpro>) about *Aspergillus* fungal genomes with different file formats. These files include gene features, Protein Domain Predictions information, Gene Ontology (GO) annotations, AspGD chromosomal features and Go Slims description. Then, we extracted the key features that required for displaying the following: gene model, protein domains, genes location in the chromosome and GO Slims availability in the genome. We also did a comprehensive study about the protein domains on five of the common used *Aspergillus* fungal genome, which are *A. nidulans*, *A. fumigatus*, *A. flavus*, *A. niger* and *A. oryzae*. This study was to observe which are the top twenty-five repeated domains.

After that, we implemented two types of HTML5 spreadsheets using Java and JavaScript languages. The first spreadsheet, called CGene, is including the visual display for all the genes in single

genome lined up with its related protein domain. This spreadsheet has some textual and visual SVG images about the genomes' genes and proteins in each row. The second spreadsheet, called CGenome, has different genomes lined up, in which each row in the spreadsheet represent single genome. In each row of the spreadsheet we display the following in SVG format: Gene location, Go Slims bitmap and Go Slims histogram. In both spreadsheets, some rows can be selected to be the only viewed dataset, and all the SVG images can be downloaded as SVG format for user personal use.

4.2 Work Contributions

As mentioned previously, there is a high demand for visual representation for all the genes in the genome to be in a single place. In our research, we put the annotated information about the gene and its encoded protein in a single place, and gather them in tabulate view to allow users to explore patterns much easier. Most of the efforts that are done previously in the tabular/spreadsheet layout were to find a universal format to exchange “omics” data/metadata and experiment results. In our system we implemented two visual spreadsheets. One called CGene that used to represent single organism while the other, which called CGenome, used to represent multiple genomes.

In CGene, we created a spreadsheet in which users are able to visualize more than 10 000 genes of single organism in single spreadsheet. In addition of summarizing all these genes in a one place, they are presented visually in SVG format. Some of the genes can be selected as dataset or/and the images can be downloaded for users further study. There is no such bioinformatics system that organizes data of thousands of genes in a single spreadsheet visually.

In CGenome, we created a spreadsheet in which users are able to compare different organisms by using three different visualizations. These visualizations are Gene location, Go Slims bitmap and GO Slims histogram. Gene location column is used to show the gene location on each chromosome where the genes are colored in pink, green and blue based on the three gene ontology aspects; One color for each aspect. This will allow the user to see the distribution of the GO Slims over the chromosomes. In Go Slims bitmap column, GO slim availability in the specified genome is shown. GO Slims histogram column that used to illustrate the distribution of the dataset by representing the number of occurrence of each GO Slim term. Any of those images can be selected and downloaded for users further study.

By presenting that information visually the scientists can use this spreadsheet to gain some understanding about the data collected about a whole genome such as genes, proteins, annotations, and expression data.

4.3 Research Limitations and Future Work

Although we implemented an HTML5 spreadsheet for the whole genome, the created spreadsheet is fixed, in which the user can browse, select dataset, and download specific SVG images. There is no flexible customization while creating the spreadsheet. The other limitation is about protein domain

architecture, since there is no clear standard about how to visualize protein domains. Different systems are using different colours and shapes to represent protein domains architecture. As a result of having hundreds of domains without having a standard about visualizing them, we decided to display them in rectangle shape with black colour and the top 25 repeated domains could be displayed in different colours.

One of the limitations that need to be stated is the lack of system evaluation. We did not evaluate the system since our goal at the beginning was to create a stable visualization system as a starting point.

Despite the fact that many information visualization evaluation techniques became widespread, the process of evaluating information visualization interfaces still complicated process [Lam et al., 2011]. When a new technology is presented, some early adaptors, who like to try any new features, will use this technology straightway. The challenge is how this technology reaches the early majority users who care about systems that are reliable, useful and solve problems [Plaisant, 2004]. Information visualization is one of the technologies having utility evaluation as major challenge [Plaisant, 2004]. Moreover, the current information visualization evaluation techniques have many weaknesses such as buggy implementation of new software, testing the wrong users and users familiarity with the traditional interfaces [Andrews, 2006]. As a result, it is unfair to compare new-implemented interfaces with tested and familiar interfaces since the users will prefer the familiar one. In addition, the wrong users such as computer science users test many visualization systems, which leads to inaccurate results. The users are better to be real users such as intelligence analysts or pharmaceutical.

According to [Andrews, 2006], to evaluate the information visualization systems, there are three common testing methods which are summative test, formative tests, and usage tests. *Formative testing* requires generally observing a small group of test users such as three to five. After the observation, there will be an idea about what and why problems happen. One of the formative testing techniques is thinking aloud where the users are asked to give their opinions while they are using the system. This kind of test is useful during visualization system development in order to fix the design and implementation bugs. The result of this test cannot be generalized to other visualization systems because the occurred problems are related to specific tasks and the number of the test group is small. In thinking aloud test, some usability attributes are not measured such as task completion time and effectiveness. Both of them require formal experiment to be measured. In contrast, *Summative testing* requires data measurements such as task completion time, number of clicks and many statistical analysis. One of summative testing techniques is formal experiment. To compare between multiple interfaces, these experiments can be designed into two ways, which are between-groups and within-groups experiment. In between-groups experiment, each group is responsible on testing one interface. In contrast, one group tests multiple interfaces in within-groups experiment. Within-groups are commonly used because it requires less test users and reduces the individual differences while testing different interfaces. The number of the test users is usually between 50-100. The third method that used for evaluating the visualization system is *usage studies*. This method requires observing a group of test users while working on the interface for long time. Most usage studies depend on self-reporting where users creating a log about what they did, what

is happened, and how long time it takes. This way is time consuming but it is useful for studying part of the software or for comparing between two or more interfaces.

In addition to the previous methods, one of the ways that can determine the success/failure of a website is using user-centric method while designing the software, which can be done by using personas. Persona as Cooper defined is “a fictitious, specific and concrete representation of target users” [Cooper, Alan and others, 1999]. Personas will act as stand-ins for real users, which can help to guide decisions about software functionality and design. By defining system personas, developer can reference the users’ needs and ensure a more useful and successful software. The designer also can evaluate the new site feature ideas based on users’ goals and needs. We have created three personas to represent the users that may use our system, which are: Carolyn Smith, Salem Hadi and Kumar Amit. We can determine the usability of the system by knowing the users motivation to use our system and implementing their needs.

Carolyn Smith is a Biologist in BioVis Lab. She is 30 years old and single. She has a master’s degree in Bioinformatics Engineering. Carolyn is goal-oriented and focused person with good team management skills. One of her concerns is data analysis and visualization. She spent her work time on collecting data, visualizing them and performing experimental tests for data visualization especially on eukaryote organisms. She focused in genes functionality and GO Slims information. Her workplace is connected with wifi. She works in the laboratory and performs various experiments in *Aspergillus* genome research. She works 9 hours a day and keeps log for every activity and task that is done. She has project about Aspergillosis, which is the group of diseases caused by *Aspergillus* organisms. In this project, she has to identify some of unknown genes functions. She focused on an experiment that states the gene function can be identified by knowing the gene location and the Gene Ontology Slim information from neighbouring genes [Amthauer and Tsatsoulis, 2010]. Since getting information from images is less time-consuming than looking through numbers and texts, she needs software that helps her to visualize the genes location in the chromosome and shows the GO Slims distribution in the chromosome. This software can help her to discover the interested gene and to identify its function by knowing the gene location and GO Slim information of a gene’s two-nearest neighbouring genes. She needs to download this image with high resolution on her own device in case she wants to send the result to her lab mates by email. The high-resolution scalable image for the chromosome will help her to see the gene location among hundred of genes in the chromosome .

Salem Hadi is a bioinformatics researcher at King Khaled University. He is 25 years old and single. He has a bachelor’s degree in Computer Science. Hamad is focused person with good communication skills. One of his concerns is genomic data analysis and visualization. He spent his work time on collecting data, analyzing them and finding software to visualize them. His group research focused on studying fungal genomes especially the *Aspergilli* genomes. He focused in genes annotation. His workplace is connected with wifi. His work is mainly on visualizing the annotated data. He works 9 hours a day and keeps log for every activity and task that is done. He needs a tool to visualize the extracted gene features alongside with its coding protein. He wants to study specific genes on selected *Aspergillus* genome. He needs a tool help him to compare between different genes. When

he finishes comparing between genes, he wants to download the gene model and the protein domain architecture images in high resolution so he can post it on his paper research.

Kumar Amit is bioinformatics PhD full time student at Concordia University. He is 30 years old and married. He has a master's degree in Bioinformatics Engineering. He has strong communication skills and had an experience on teaching. One of his concerns is genomic comparative analysis. His research requires performing a comparative analysis between different genomes and one of them is *Aspergillus* genomes. He works on a poster about comparative analysis between genomes in order to present it on a conference. He needs an example to represent on his poster with scalable images that does not lose its resolution when scaling. This example should have multiple genomes compared in different aspects and one of them is showing gene location on the chromosomes.

Future Work As the system has some limitations as noted previously, we consider some suggestions for a further improvement to the system.

First, apply some evaluation techniques on the system to fix the bugs and to improve the system utility. To evaluate CGene and CGenome, we need to include a biologist on the testing group since they are one of the main targeted users. As a result, we can have an idea about the system usage and easiness. We will be able to measure how much biologist familiar with our system comparing to the available system. This can be done by applying some of evaluation methods on information visualization stated previously. Since CGene and CGenome are both HTML5 webpages, the website usage can be studied where a user's usage data are recorded for period of time such as 30 days. After recording this log, we can have an idea about what are users looking for and how much time they spend on each interface. We will count how many users used the website and for how long. We can have an idea about their satisfaction by offering online survey or by observing if they return back or not. We also can apply within-groups experiment to test every interface. From this experiment we can have an idea about how many clicks the user needs to browse the spreadsheet, select dataset and to download SVG images. We will also have an idea about how much time is required for browsing and downloading dataset. The testing group should log the bugs that may happen during browsing, selecting some rows and downloading images.

Second, support customization by engaging the user during the creation of the spreadsheet. By applying this way the settings done by specific user will not affect another user's setting since the changes will be applied to the user outcome on its local device. Consequently, more than one user can use the same system but the outcome may differ based on individual settings.

Third, support configuration files since it will offer more power and flexibility to the user. By using configuration files during the creation of the spreadsheet, the user can set its own preference such as the colours or the shape of the protein domains, which makes noticing patterns in the spreadsheet easier. There are some suggested user-configuration options which can be applied in the future. For example, making SVG images for both *gene model* and *protein domain architecture* configurable to be normalized images or to be images that reflect the real length. The SVG images also can be coloured based on user preferences. The configuration could include putting specific colour or shape for specific protein domains, and configuring the spreadsheet schema for the requested organism such as setting the width/height of the table and setting spreadsheet columns' names.

In addition to customization and configuration file support, there are some of spreadsheet operations that can be added. These operations include keyword search, pattern search, filtering, sorting, and clustering of columns containing visualizations. By adding those operations the user can search for specific keyword or pattern where the pattern could be a regular expression or a simple string. For example users can search for a gene whose name includes a particular string. On filtering the user can select certain rows based on certain criteria such as selecting all genes whose length value smaller or greater than a specified value. By using sorting operation the user can sort the spreadsheet rows based on the selected columns such sorting the rows based on gene length from low to high. On clustering, the rows can be grouped based on columns containing visualizations such as grouping the genes based on having specific domains.

4.4 Resource of the system

The codes that used to create CGene and CGenome have uploaded on Google subversion server that powered by Google Project Hosting. The code can be accessed on the address: <https://a-visual-spreadsheet-using-html5-for-whole-genome-display.googlecode.com/svn/trunk/>.

Bibliography

- [Albà et al., 2007] Albà, M., Tompa, P., and Veitia, R. (2007). Amino acid repeats and the structure and evolution of proteins.
- [Amthauer and Tsatsoulis, 2010] Amthauer, H. A. and Tsatsoulis, C. (2010). Classifying genes to the correct Gene Ontology Slim term in *Saccharomyces cerevisiae* using neighbouring genes with classification learning. *BMC Genomics*, 11(1):340.
- [Andrews, 2006] Andrews, K. (2006). Evaluating Information Visualizations. In *Proceedings of the 2006 AVI workshop on Beyond time and errors: novel evaluation methods for information visualization*, pages 1–5. ACM.
- [AspGD, 2014] AspGD (2014). Available at: www.aspgd.org.
- [Biologyreference., 2014] Biologyreference. (2014). Available at: <http://www.biologyreference.com/Po-Re/Protein-Structure.html>.
- [Brehmer and Munzner, 2013] Brehmer, M. and Munzner, T. (2013). A multi-level typology of abstract visualization tasks. *IEEE Trans. Visualization and Computer Graphics (TVCG) (Proc. InfoVis)*, 19(12):2376–2385.
- [Card et al., 1999] Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). *Readings in information visualization: using vision to think*. Morgan Kaufmann.
- [Cooper, Alan and others, 1999] Cooper, Alan and others (1999). *Inmates Are Running the Asylum, The: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity*, volume 261. Sams Indianapolis.
- [Crick et al., 1970] Crick, F. et al. (1970). Central dogma of molecular biology. *Nature*, 227(5258):561–563.
- [Dailey et al., 2012] Dailey, D., Frost, J., and Strazzullo, D. (2012). *Building web applications with SVG*. O’Reilly Media, Inc.
- [Deacon, 2009] Deacon, J. (2009). Model-View-Controller (MVC) architecture. *Online: <http://www.jdl.co.uk/briefings/MVC.pdf>*.

- [Dudley and Karczewski, 2013] Dudley, J. T. and Karczewski, K. J. (2013). *Exploring personal genomics*. Oxford University Press.
- [EMBL-EBI, 2014] EMBL-EBI (2014). Available at: <http://www.ebi.ac.uk/training/online/course/introduction-protein-classification-ebi/what-are-protein-signatures/signature-types/what-ar-2>.
- [Fawal et al., 2012] Fawal, N., Savelli, B., Dunand, C., and Mathé, C. (2012). GECA: a fast tool for gene evolution and conservation analysis in eukaryotic protein families. *Bioinformatics*, 28(10):1398–1399.
- [Flanagan, 2011] Flanagan, D. (2011). *JavaScript: The definitive guide: Activate your web pages*. O’Reilly Media, Inc.
- [Frech et al., 2012] Frech, C., Choo, C., and Chen, N. (2012). Featurestack: Perl module for comparative visualization of gene features. *Bioinformatics*, 28(23):3137–3138.
- [Gehlenborg et al., 2010] Gehlenborg, N., O’Donoghue, S. I., Baliga, N. S., Goesmann, A., Hibbs, M. A., Kitano, H., Kohlbacher, O., Neuweger, H., Schneider, R., Tenenbaum, D., et al. (2010). Visualization of omics data for systems biology. *Nature Methods*, 7:S56–S68.
- [Gene Ontology Consortium and others, 2008] Gene Ontology Consortium and others (2008). The Gene Ontology project in 2008. *Nucleic Acids Research*, 36(suppl 1):D440–D444.
- [Gene Ontology Consortium and others, 2010] Gene Ontology Consortium and others (2010). The Gene Ontology in 2010: extensions and refinements. *Nucleic Acids Research*, 38(suppl 1):D331–D335.
- [GeneOntology, 2014] GeneOntology (2014). Gene Ontology Consortium. Available at: <http://geneontology.org/page/download-ontology>.
- [Hertz and Stormo, 1999] Hertz, G. Z. and Stormo, G. D. (1999). Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7):563–577.
- [Hrmova and Fincher, 2009] Hrmova, M. and Fincher, G. B. (2009). Functional genomics and structural biology in the definition of gene function. In *Plant Genomics*, pages 199–227. Springer.
- [HTML5, 2014] HTML5 (2014). Available at: http://www.w3schools.com/html/html5_intro.asp.
- [InterPro, 2014] InterPro (2014). InterPro DB. Available at: <http://www.ebi.ac.uk/interpro>.
- [Jameson et al., 2011] Jameson, D., Westerhoff, H. V., and Verma, M. (2011). *Methods in systems biology*, volume 500. Academic Press.

- [Kent et al., 2002] Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., and Haussler, D. (2002). The human genome browser at UCSC. *Genome Research*, 12(6):996–1006.
- [Koch, 2011] Koch, R. (2011). *The 80/20 principle: the secret to achieving more with less*. Random House LLC.
- [Koonin EV, 2003] Koonin EV, G. M. (2003). *Sequence-Evolution-Function: Computational Approaches in Comparative Genomics*. Kluwer Academic, Boston.
- [Krzywinski, 2013] Krzywinski, M. (2013). Points of view: Elements of visual style. *Nature Methods*, 10(5):371–371.
- [Lam et al., 2011] Lam, H., Bertini, E., Isenberg, P., Plaisant, C., Carpendale, S., et al. (2011). Seven Guiding Scenarios for Information Visualization Evaluation. Technical Report 2011-992-04, University of Calgary.
- [Larman, 2012] Larman, C. (2012). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3/e*. Pearson Education India.
- [Latham and Latham, 1995] Latham, R. and Latham, R. (1995). *The dictionary of computer graphics and virtual reality*, volume 2. Springer.
- [Letunic and Bork, 2007] Letunic, I. and Bork, P. (2007). Interactive tree of life (iTOL): an online tool for phylogenetic tree display and annotation. *Bioinformatics*, 23(1):127–128.
- [Letunic and Bork, 2011] Letunic, I. and Bork, P. (2011). Interactive tree of life v2: online annotation and display of phylogenetic trees made easy. *Nucleic Acids Research*, 39:W475–W478.
- [Luo and Nijveen, 2013] Luo, H. and Nijveen, H. (2013). Understanding and identifying amino acid repeats. *Briefings in Bioinformatics*, page bbt003.
- [Majoros, 2007] Majoros, W. H. (2007). *Methods for computational gene prediction*, volume 1. Cambridge University Press Cambridge.
- [Meyer et al., 2010] Meyer, M., Munzner, T., DePace, A., and Pfister, H. (2010). MulteeSum: A tool for comparative spatial and temporal gene expression data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):908–917.
- [Morrison, 2013] Morrison, D. A. (2013). Evolutionary Genomics: Statistical and Computational Methods. Volumes 1 and 2. *Systematic Biology*, 62(2):348–350.
- [Munzner, 2014] Munzner, T. (November 17, 2014). *Visualization Analysis and Design*. AK Peters Visualization Series. A K Peters/CRC Press, 1 edition.
- [Nielsen et al., 2010] Nielsen, C. B., Cantor, M., Dubchak, I., Gordon, D., and Wang, T. (2010). Visualizing genomes: techniques and challenges. *Nature Methods*, 7:S5–S15.

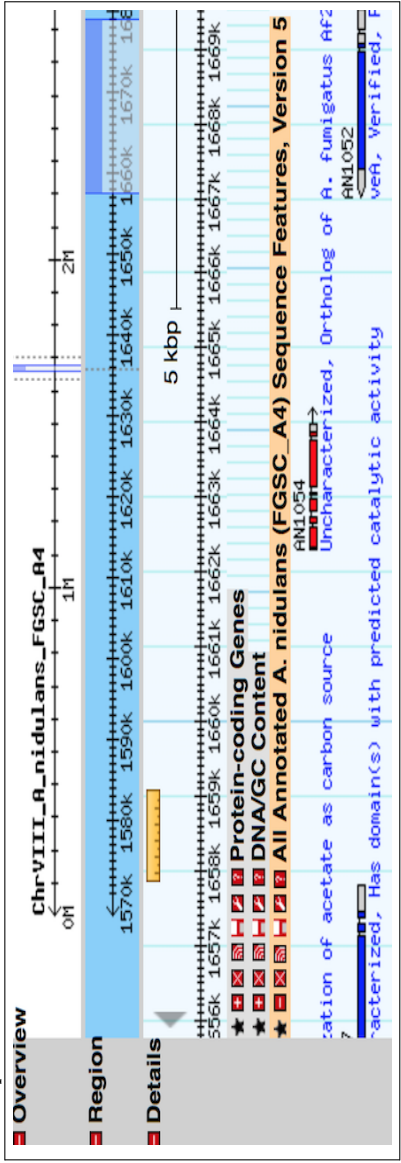
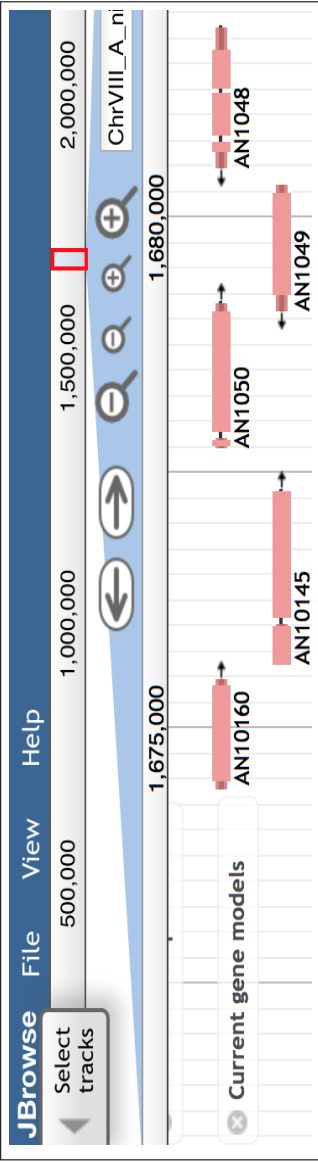
- [O'Donoghue et al., 2010] O'Donoghue, S. I., Gavin, A.-C., Gehlenborg, N., Goodsell, D. S., Hériché, J.-K., Nielsen, C. B., North, C., Olson, A. J., Procter, J. B., Shattuck, D. W., et al. (2010). Visualizing biological data—now and in the future. *Nature Methods*, 7:S2–S4.
- [Pavlopoulos et al., 2008] Pavlopoulos, G. A., Wegener, A.-L., and Schneider, R. (2008). A survey of visualization tools for biological network analysis. *Biodata Mining*, 1(1):1–11.
- [Peng and Zhang, 2004] Peng, Z.-R. and Zhang, C. (2004). The roles of geography markup language (GML), scalable vector graphics (SVG), and web feature service (WFS) specifications in the development of internet geographic information systems (GIS). *Journal of Geographical Systems*, 6(2):95–116.
- [Peter et al., 2004] Peter, B. J., Kent, H. M., Mills, I. G., Vallis, Y., Butler, P. J. G., Evans, P. R., and McMahon, H. T. (2004). Bar domains as sensors of membrane curvature: the amphiphysin bar structure. *Science*, 303(5657):495–499.
- [Pilgrim, 2010] Pilgrim, M. (2010). *HTML5: Up and Running*. O'Reilly Media, Inc.
- [Plaisant, 2004] Plaisant, C. (2004). The Challenge of Information Visualization Evaluation. In *Proceedings of the Working conference on Advanced Visual Interfaces*, pages 109–116. ACM.
- [Rambaldi and Ciccarelli, 2009] Rambaldi, D. and Ciccarelli, F. D. (2009). FancyGene: dynamic visualization of gene structures and protein domain architectures on genomic loci. *Bioinformatics*, 25(17):2281–2282.
- [Rangwala and Karypis, 2010] Rangwala, H. and Karypis, G. (2010). *Introduction to Protein Structure Prediction: Methods and Algorithms*, volume 14. John Wiley & Sons.
- [Ren et al., 2009] Ren, J., Wen, L., Gao, X., Jin, C., Xue, Y., and Yao, X. (2009). DOG 1.0: illustrator of protein domain structures. *Cell Research*, 19(2):271–273.
- [Roberts, 2007] Roberts, J. C. (2007). State of the art: Coordinated & multiple views in exploratory visualization. In *fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization, 2007.*, pages 61–71. IEEE.
- [Rouzé et al., 1999] Rouzé, P., Pavy, N., and Rombauts, S. (1999). Genome annotation: which tools do we have for it? *Current Opinion in Plant Biology*, 2(2):90–95.
- [Saben et al., 2014] Saben, J., Zhong, Y., McKelvey, S., Dajani, N., Andres, A., Badger, T., Gomez-Acevedo, H., and Shankar, K. (2014). A comprehensive analysis of the human placenta transcriptome. *Placenta*, 35(2):125–131.
- [Sanger-Institute, 2014] Sanger-Institute (2014). Gff: an exchange format for feature description gff: an exchange format for feature description. Available at: <http://www.sanger.ac.uk/resources/software/gff/>.

- [Sansone et al., 2008] Sansone, S.-A., Rocca-Serra, P., Brandizi, M., Brazma, A., Field, D., Fostel, J., Garrow, A. G., Gilbert, J., Goodsaid, F., Hardy, N., et al. (2008). The first RSBI (ISA-TAB) workshop: “can a simple format work for complex studies?”. *OMICS A Journal of Integrative Biology*, 12(2):143–149.
- [Schultz et al., 2000] Schultz, J., Copley, R. R., Doerks, T., Ponting, C. P., and Bork, P. (2000). SMART: a web-based tool for the study of genetically mobile domains. *Nucleic Acids Research*, 28(1):231–234.
- [Shannon et al., 2003] Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504.
- [Smits and Ouverney, 2010] Smits, S. A. and Ouverney, C. C. (2010). jsPhyloSVG: a javascript library for visualizing interactive and vector-based phylogenetic trees on the web. *PLoS One*, 5(8):e12267.
- [Soltis and Soltis, 2003] Soltis, D. E. and Soltis, P. S. (2003). The role of phylogenetics in comparative genetics. *Plant Physiology*, 132(4):1790–1800.
- [Stein, 2001] Stein, L. (2001). Genome annotation: from sequence to biology. *Nature Reviews Genetics*, 2(7):493–503.
- [Tufte and Graves-Morris, 1983] Tufte, E. R. and Graves-Morris, P. (1983). *The visual display of quantitative information*, volume 2. Graphics Press, Cheshire, CT.
- [Wang Baldonado et al., 2000] Wang Baldonado, M. Q., Woodruff, A., and Kuchinsky, A. (2000). Guidelines for using multiple views in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 110–119. ACM.
- [Watson et al., 1953] Watson, J. D., Crick, F. H., et al. (1953). Molecular structure of nucleic acids. *Nature*, 171(4356):737–738.
- [Wikipedia, 2014] Wikipedia (2014). Available at: http://en.wikipedia.org/wiki/Protein_folding.
- [Wong, 2012] Wong, B. (2012). Points of view: Visualizing biological data. *Nature Methods*, 9(12):1131–1131.

Appendix A

Gene Model Catalog

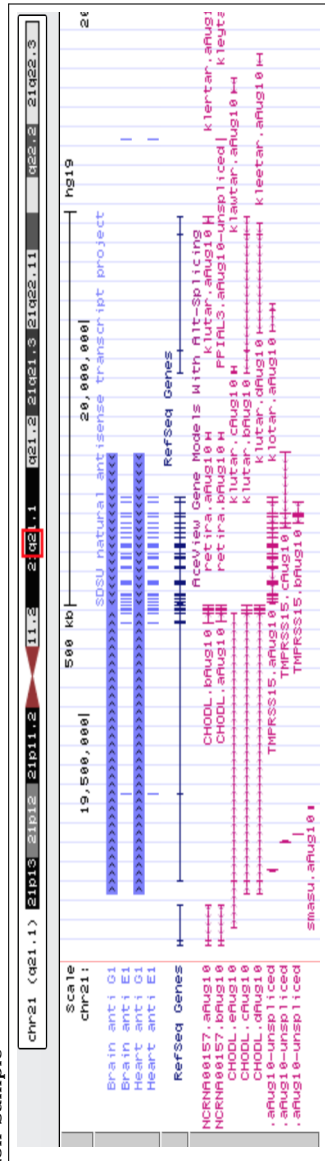
Table 7: Catalog of visualization tools for Gene Models

Vis. Num	Vis. Name	Source	Description
7.1	GBrowse	(http://www.gbrowse.org/index.html)	GBrowse is “an interactive web system for manipulating and displaying annotations on genomes” where sequence features are displayed as visual elements in tracks and features are aligned with their genome coordinates.
<p>Visualization Sample</p> 			
7.2	JBrowse	(http://jbrowse.org)	JBrowse is “a genome browser with a fully dynamic AJAX interface, being developed as the eventual successor to GBrowse” where sequence features are displayed as visual elements in tracks and features are aligned with their genome coordinates.
<p>Visualization Sample</p> 			

“The UCSC Genome Browser is a graphical viewer for genomic data. Since the early days of the Human Genome Project, it has presented as an integrated view of genomic data of many kinds. Now it is a home to assemblies for 58 organisms, the Browser presents visualization of annotations mapped to genomic coordinates.”

7.3 . UCSC Genome Browser (<http://genome.ucsc.edu>)

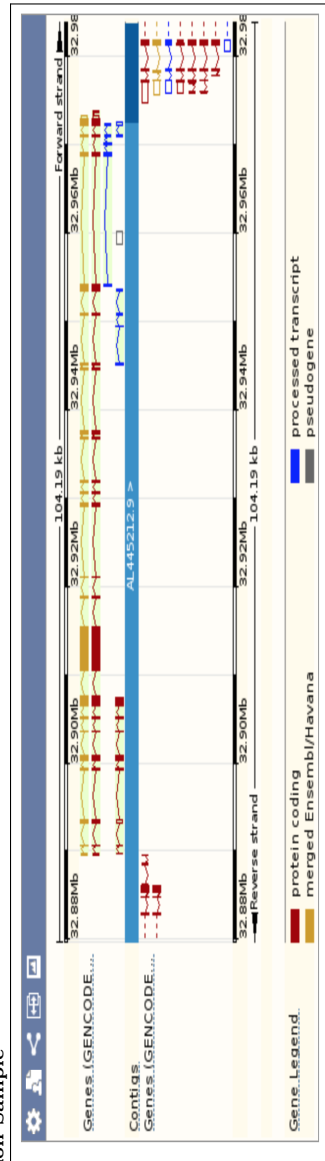
Visualization Sample



“The Ensembl project provides high-quality genome annotation across chordate species through a comprehensive set of methods, which result in unique data sets including Ensembl’s gene annotations, multiple alignments, gene homology relationships and regulatory annotation.”

7.4 . Ensembl (<http://www.ensembl.org>)

Visualization Sample

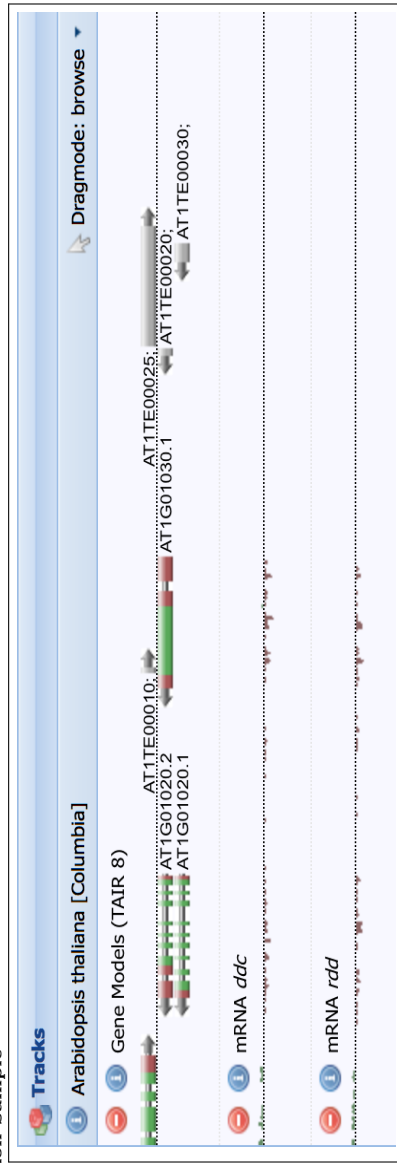


“Anno-J is a Web 9.0 application designed for visualizing deep sequencing data and other genome annotation data. It is intended to run in modern W3C compliant browsers, and allows flexible configuration of plugins and data streams from providers located anywhere on the internet.”

Anno-J (<http://www.anno.j.org>)

7.5

Visualization Sample

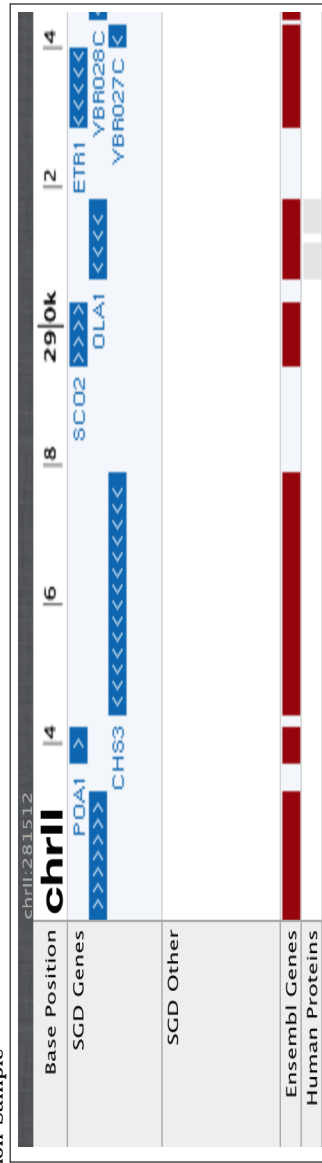


“ChromoZoom is a flexible, fluid, web-based genome browser which pre-renders and caches general-use tracks into tiled images on the server and serves them in an interactive web interface with inertial scrolling and precise, fluent zooming via the mouse wheel or trackpad.”

ChromoZoom (<http://chromozoom.org>)

7.6

Visualization Sample



CoGe's genome visualization library, GeLo, is very flexible in the types of graphics it can produce. GeLo is CoGe's Genomic Visualization Library and stands for Genome Location Visualization. GeLo is similar to other genomic visualization libraries in the sense that it allows you to visualize a genomic region and paint genomic features on it using tracks. However, GeLo differs substantially in how much freedom it provides to allow you to create many types of genomic visualization schemes. In essence, GeLo lets you create a virtual chromosome, paint its background any way you wish, add genomic features to it based on nucleotide coordinates, and then will render your image for you based on the size you wish it to be. All the details of scaling the genomic features and determining their placement is handled for you behind the scenes. If the genomic feature is on the top strand, it is drawn in the top half of the chromosome automatically.

(Note: Some of the information in GeLo is out of date. However, I put this system here because I found clear examples for Gene Models.)

(https://genomeevolution.org/wiki/index.php/GenomeView_examples)

GeLo: Genome View

7.7

Visualization Sample

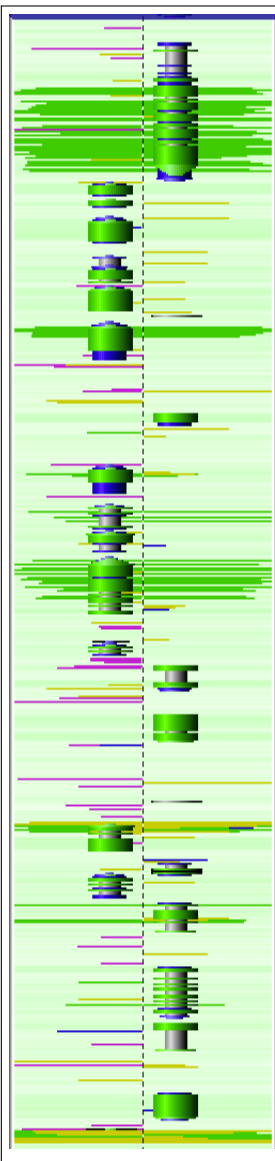


Table 8: Catalog of tabular view for Gene Annotations

Vis. Num	Vis. Name	Source	Description																																																																																
8.1	Table Browser	(http://genome.ucsc.edu)	The Table Browser from (http://genome.ucsc.edu) provides a powerful and flexible graphical interface for querying and manipulating the Genome Browser annotation tables. This program is used "to retrieve the data associated with a track in text format, to calculate intersections between tracks, and to retrieve DNA sequence covered by a track."																																																																																
Visualization Sample																																																																																			
<table border="1"> <thead> <tr> <th>#bin</th> <th>name</th> <th>chrom</th> <th>strand</th> <th>txStart</th> <th>txEnd</th> <th>cdsStart</th> <th>cdsEnd</th> <th>exonCount</th> <th>exonStarts</th> <th>exonEnds</th> <th>score</th> <th>name2</th> <th>cdsStartStat</th> <th>cdsEndStat</th> <th>exonFrames</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>NW_001167506</td> <td>chr1</td> <td>-</td> <td>66774604</td> <td>67134611</td> <td>66774604</td> <td>67134611</td> <td>2</td> <td>66774604,67134410,</td> <td>66774691,67134611,</td> <td>0</td> <td>mondDomV1R1250</td> <td>incmpl</td> <td>incmpl</td> <td>0,0,</td> </tr> <tr> <td>0</td> <td>NW_001253553</td> <td>chr1</td> <td>-</td> <td>66774622</td> <td>67175373</td> <td>66774622</td> <td>67175373</td> <td>2</td> <td>66774622,67175274,</td> <td>66774790,67175373,</td> <td>0</td> <td>ORNMANV1R3161</td> <td>incmpl</td> <td>incmpl</td> <td>0,0,</td> </tr> <tr> <td>0</td> <td>NW_001166810</td> <td>chr1</td> <td>-</td> <td>66774625</td> <td>67134497</td> <td>66774625</td> <td>67134497</td> <td>2</td> <td>66774625,67134398,</td> <td>66774820,67134497,</td> <td>0</td> <td>mondDomV1R1232</td> <td>incmpl</td> <td>incmpl</td> <td>0,0,</td> </tr> <tr> <td>0</td> <td>NW_001166814</td> <td>chr1</td> <td>-</td> <td>66774625</td> <td>67134497</td> <td>66774625</td> <td>67134497</td> <td>2</td> <td>66774625,67134398,</td> <td>66774820,67134497,</td> <td>0</td> <td>mondDomV1R1228</td> <td>incmpl</td> <td>incmpl</td> <td>0,0,</td> </tr> </tbody> </table>				#bin	name	chrom	strand	txStart	txEnd	cdsStart	cdsEnd	exonCount	exonStarts	exonEnds	score	name2	cdsStartStat	cdsEndStat	exonFrames	0	NW_001167506	chr1	-	66774604	67134611	66774604	67134611	2	66774604,67134410,	66774691,67134611,	0	mondDomV1R1250	incmpl	incmpl	0,0,	0	NW_001253553	chr1	-	66774622	67175373	66774622	67175373	2	66774622,67175274,	66774790,67175373,	0	ORNMANV1R3161	incmpl	incmpl	0,0,	0	NW_001166810	chr1	-	66774625	67134497	66774625	67134497	2	66774625,67134398,	66774820,67134497,	0	mondDomV1R1232	incmpl	incmpl	0,0,	0	NW_001166814	chr1	-	66774625	67134497	66774625	67134497	2	66774625,67134398,	66774820,67134497,	0	mondDomV1R1228	incmpl	incmpl	0,0,
#bin	name	chrom	strand	txStart	txEnd	cdsStart	cdsEnd	exonCount	exonStarts	exonEnds	score	name2	cdsStartStat	cdsEndStat	exonFrames																																																																				
0	NW_001167506	chr1	-	66774604	67134611	66774604	67134611	2	66774604,67134410,	66774691,67134611,	0	mondDomV1R1250	incmpl	incmpl	0,0,																																																																				
0	NW_001253553	chr1	-	66774622	67175373	66774622	67175373	2	66774622,67175274,	66774790,67175373,	0	ORNMANV1R3161	incmpl	incmpl	0,0,																																																																				
0	NW_001166810	chr1	-	66774625	67134497	66774625	67134497	2	66774625,67134398,	66774820,67134497,	0	mondDomV1R1232	incmpl	incmpl	0,0,																																																																				
0	NW_001166814	chr1	-	66774625	67134497	66774625	67134497	2	66774625,67134398,	66774820,67134497,	0	mondDomV1R1228	incmpl	incmpl	0,0,																																																																				
Gene Model Data In																																																																																			
8.2	MaizeGDB: Gene Models with Associated Genes (B73 RefGen.v2)	(http://www.maizegdb.org)	MaizeGDB is a community-oriented, long-term, federally funded informatics service to researchers focused on the crop plant and model organism Zea mays.																																																																																
Visualization Sample																																																																																			
<table border="1"> <thead> <tr> <th>Gene Model ID</th> <th>Gene Symbol</th> <th>Source</th> <th>Chromosome</th> <th>Start</th> <th>End</th> </tr> </thead> <tbody> <tr> <td>GRMZM2G164696</td> <td>tub1</td> <td>Classical Gene</td> <td>chr1</td> <td>2,037,790</td> <td>2,039,758</td> </tr> <tr> <td>GRMZM2G090568</td> <td>cat2</td> <td>Classical Gene</td> <td>chr1</td> <td>7,160,314</td> <td>7,162,590</td> </tr> <tr> <td>GRMZM2G339563</td> <td>lls1</td> <td>Classical Gene</td> <td>chr1</td> <td>10,092,334</td> <td>10,096,672</td> </tr> <tr> <td>GRMZM2G092542</td> <td>rtcs1</td> <td>Classical Gene</td> <td>chr1</td> <td>10,826,297</td> <td>10,827,541</td> </tr> <tr> <td>GRMZM2G172322</td> <td>gsr1</td> <td>Classical Gene</td> <td>chr1</td> <td>12,987,102</td> <td>12,993,471</td> </tr> <tr> <td>GRMZM2G091822</td> <td>ms26</td> <td>Classical Gene</td> <td>chr1</td> <td>14,506,335</td> <td>14,508,661</td> </tr> <tr> <td>GRMZM2G410515</td> <td>vp5</td> <td>Classical Gene</td> <td>chr1</td> <td>17,660,941</td> <td>17,667,054</td> </tr> <tr> <td>GRMZM2G155242</td> <td>lpa1</td> <td>Classical Gene</td> <td>chr1</td> <td>25,284,139</td> <td>25,288,349</td> </tr> <tr> <td>GRMZM2G050514</td> <td>gln6</td> <td>Classical Gene</td> <td>chr1</td> <td>27,977,783</td> <td>27,980,656</td> </tr> <tr> <td>GRMZM2G110309</td> <td>ibp2</td> <td>Classical Gene</td> <td>chr1</td> <td>40,352,197</td> <td>40,357,695</td> </tr> <tr> <td>GRMZM2G087186</td> <td>pdcs3</td> <td>Classical Gene</td> <td>chr1</td> <td>45,512,802</td> <td>45,515,978</td> </tr> <tr> <td>GRMZM2G455809</td> <td>ts2</td> <td>Classical Gene</td> <td>chr1</td> <td>46,678,942</td> <td>46,680,363</td> </tr> </tbody> </table>				Gene Model ID	Gene Symbol	Source	Chromosome	Start	End	GRMZM2G164696	tub1	Classical Gene	chr1	2,037,790	2,039,758	GRMZM2G090568	cat2	Classical Gene	chr1	7,160,314	7,162,590	GRMZM2G339563	lls1	Classical Gene	chr1	10,092,334	10,096,672	GRMZM2G092542	rtcs1	Classical Gene	chr1	10,826,297	10,827,541	GRMZM2G172322	gsr1	Classical Gene	chr1	12,987,102	12,993,471	GRMZM2G091822	ms26	Classical Gene	chr1	14,506,335	14,508,661	GRMZM2G410515	vp5	Classical Gene	chr1	17,660,941	17,667,054	GRMZM2G155242	lpa1	Classical Gene	chr1	25,284,139	25,288,349	GRMZM2G050514	gln6	Classical Gene	chr1	27,977,783	27,980,656	GRMZM2G110309	ibp2	Classical Gene	chr1	40,352,197	40,357,695	GRMZM2G087186	pdcs3	Classical Gene	chr1	45,512,802	45,515,978	GRMZM2G455809	ts2	Classical Gene	chr1	46,678,942	46,680,363		
Gene Model ID	Gene Symbol	Source	Chromosome	Start	End																																																																														
GRMZM2G164696	tub1	Classical Gene	chr1	2,037,790	2,039,758																																																																														
GRMZM2G090568	cat2	Classical Gene	chr1	7,160,314	7,162,590																																																																														
GRMZM2G339563	lls1	Classical Gene	chr1	10,092,334	10,096,672																																																																														
GRMZM2G092542	rtcs1	Classical Gene	chr1	10,826,297	10,827,541																																																																														
GRMZM2G172322	gsr1	Classical Gene	chr1	12,987,102	12,993,471																																																																														
GRMZM2G091822	ms26	Classical Gene	chr1	14,506,335	14,508,661																																																																														
GRMZM2G410515	vp5	Classical Gene	chr1	17,660,941	17,667,054																																																																														
GRMZM2G155242	lpa1	Classical Gene	chr1	25,284,139	25,288,349																																																																														
GRMZM2G050514	gln6	Classical Gene	chr1	27,977,783	27,980,656																																																																														
GRMZM2G110309	ibp2	Classical Gene	chr1	40,352,197	40,357,695																																																																														
GRMZM2G087186	pdcs3	Classical Gene	chr1	45,512,802	45,515,978																																																																														
GRMZM2G455809	ts2	Classical Gene	chr1	46,678,942	46,680,363																																																																														

Table 9: Different glyph's visualization for Gene Models

Vis. Num	Vis Name	Source	Description
9.1	Gene Models only	(https://genomevolution.org/wiki/index.php/GenomeView_examples)	Gene models are drawn as composite coloured arrows such that grey is the extent of the gene. This is a part of GenomeView COGE platform that based on JBrowse. (<i>NOTE: This page is out of date and does not reflect recent migration to the JBrowse viewer.</i>)

Glyph

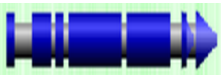


Glyphs on the track

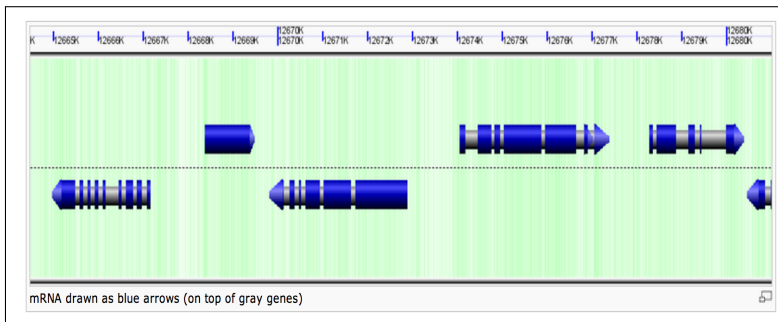


9.2	Gene Model with mRNA	(https://genomevolution.org/wiki/index.php/GenomeView_examples)	mRNA drawn as blue arrows (on top of grey genes).
-----	----------------------	---	---

Glyph

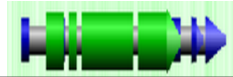


Glyphs on the track

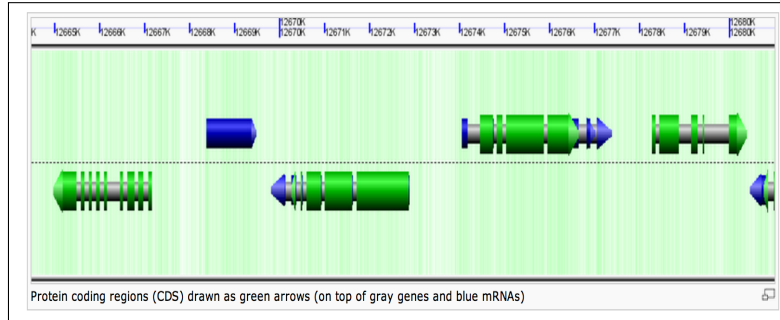


- 9.3 Gene Models with CDS (https://genomevolution.org/wiki/index.php/GenomeView_examples) Protein coding regions (CDS) drawn as green arrows (on top of grey genes and blue mRNAs).

Glyph



Glyphs on the track

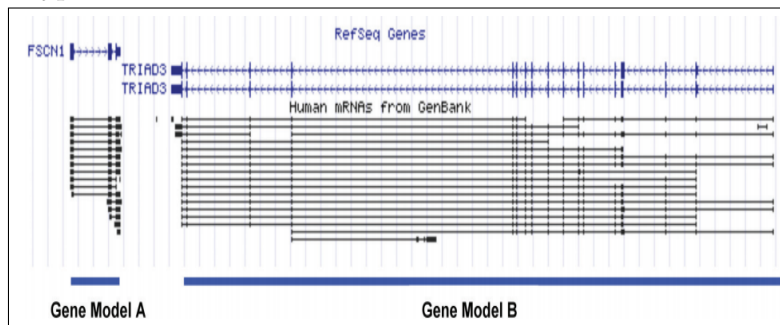


- 9.4 SwitchGear gene model (http://www.biocat.com/bc/pdf/LightSwitch_Genemodel_Technote.pdf) A SwitchGear gene model is made up of one or more **cDNAs** that align to the same region of the human genome. Specifically, **SwitchGear gene models** are defined as clusters of cDNA alignments that have overlapping exons on the same strand.

Glyph



Glyphs on the track

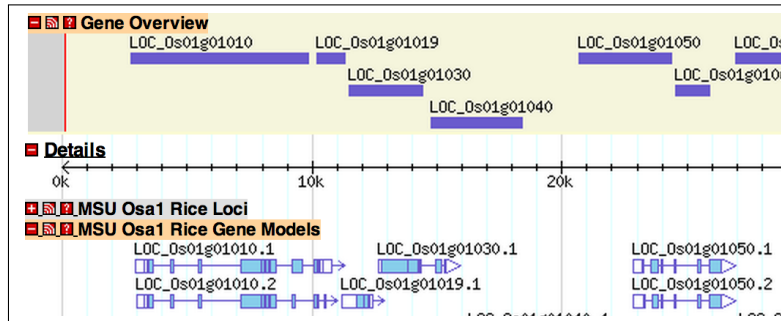


- 9.5 MSU Osa1 Rice Gene Models (<http://rice.plantbiology.msu.edu>) MSU Osa1 Rice Gene Models are gene models annotated by the Rice Genome Annotation Project. Gene models are drawn as coloured rectangles. The direction of the transcript is indicated by an arrow attached to the end of the glyph.

Glyph



Glyphs on the track



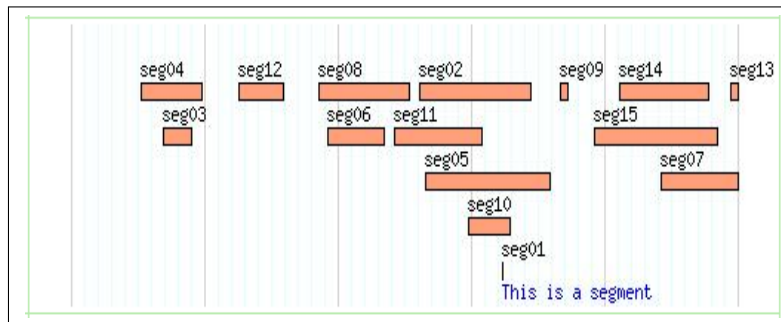
9.6 GBrowse: (http://webgbrowse
.cgb.indiana.edu/w
Box ebgbrowse/glyphdoc
.html)

Box glyph is the most basic glyph. It draws a filled box and optionally a label, but it does not draw subparts.

Glyph



Glyphs on the track



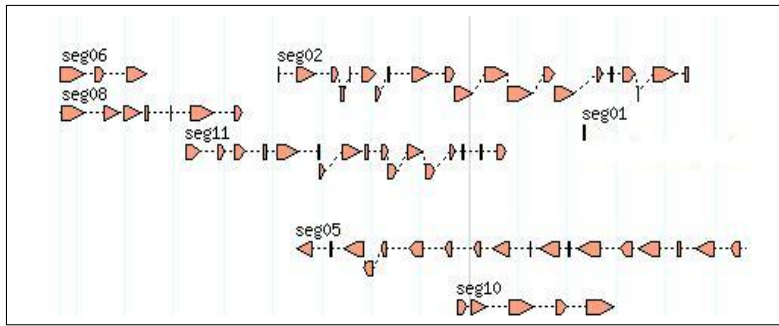
9.7 GBrowse: (http://webgbrowse
.cgb.indiana.edu/w
Generic ebgbrowse/glyphdoc
.html)

Generic is identical to the box glyph except that it will draw the subparts of features that contain subfeatures. Generic is the default glyph used in GBrowse.

Glyph



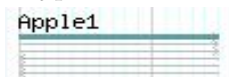
Glyphs on the track



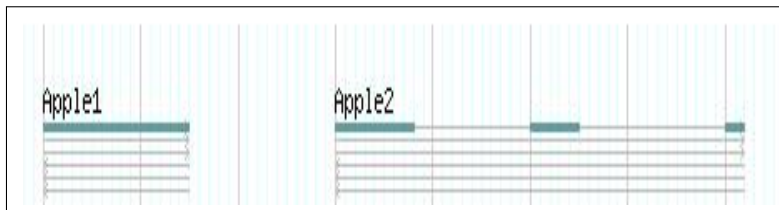
9.8 GBrowse: CDS (http://webgbrowse.cgb.indiana.edu/webgbrowse/glyphdoc.html)

Cds glyph draws features associated with a protein coding region. A series of color-coded boxes indicating the translation frame are drawn at high magnifications. But at low magnifications, the amino acid sequence of the resulting protein is drawn.

Glyph



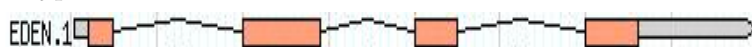
Glyphs on the track



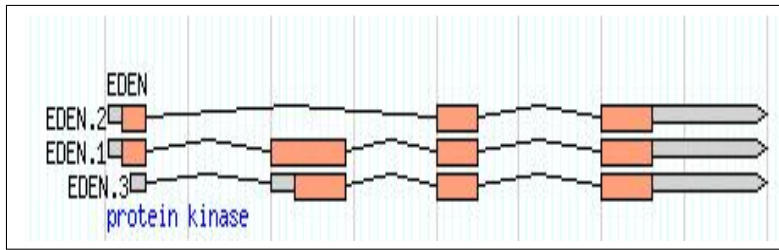
9.9 GBrowse: Gene (http://webgbrowse.cgb.indiana.edu/webgbrowse/glyphdoc.html)

The gene glyph is used for drawing genes that may have alternatively-spliced transcripts. The various isoforms are stacked on top of each other and given a single label and description that apply to the entire stack. The name of each individual transcript is optionally printed to the left of the transcript glyph.

Glyph



Glyphs on the track

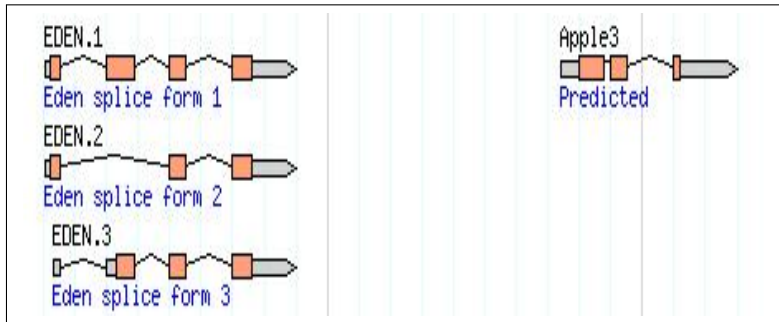


9.10 GBrowse: <http://webgbrowse.cgb.indiana.edu/webgbrowse/glyphdoc.html> The processed_transcript glyph is used for drawing processed transcripts that have both CDS and UTR segments.

Glyph



Glyphs on the track

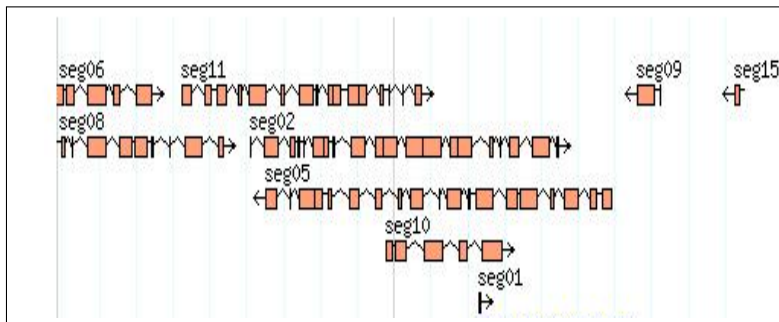


9.11 GBrowse: <http://webgbrowse.cgb.indiana.edu/webgbrowse/glyphdoc.html> The Transcript glyph is used for drawing transcripts. It is essentially a segments glyph in which the default connecting segments are hats. The direction of the transcript is indicated by an arrow attached to the end of the glyph.

Glyph



Glyphs on the track



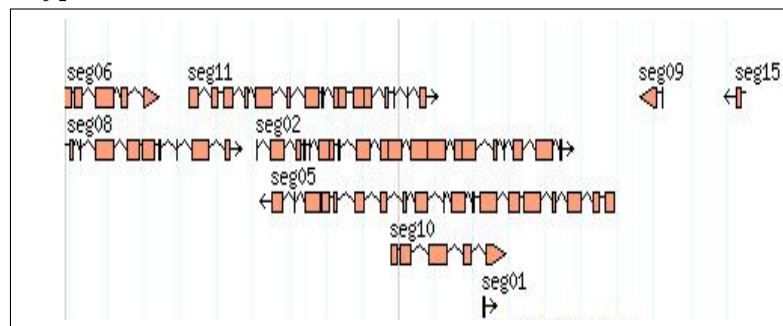
9.12 GBrowse: [.cgb.indiana.edu/webgbrowse/glyphdoc.html](http://webgbrowse.cgb.indiana.edu/webgbrowse/glyphdoc.html)
 Transcript2

The Transcript2 glyph is used for drawing transcripts. It is like transcript except that if there is sufficient room the terminal exon is shaped like an arrow in order to indicate the direction of transcription. If there is not enough room, a small arrow is drawn.

Glyph



Glyphs on the track



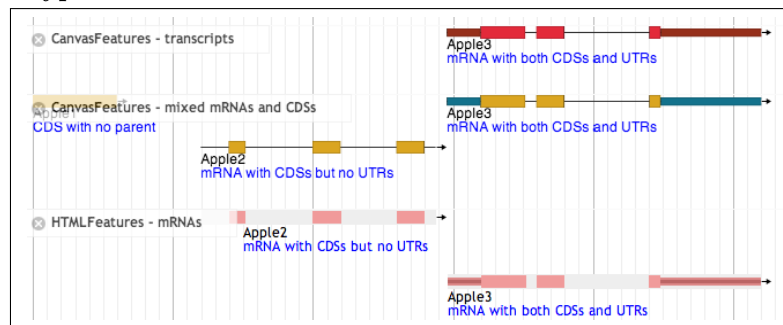
9.13 JBrowse: [\(http://jbrowse.org/\)](http://jbrowse.org/)
 exon

One of the gene feature subparts is drawn here in a box.

Glyph



Glyphs on the track



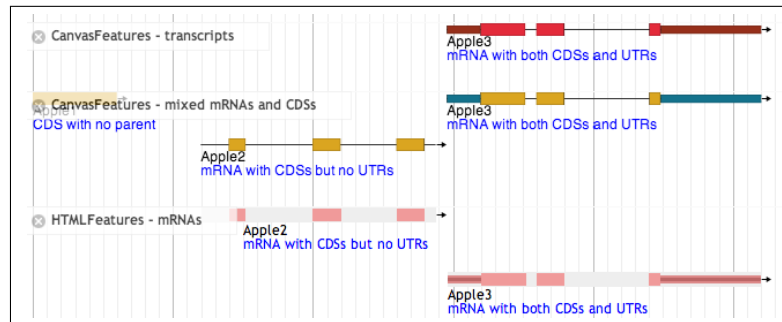
JBrowse:
9.14 mRNA with (http://jbrowse.or
CDs but no g/
UTRs

This glyph draws features that are associated with a protein coding region. At high magnifications, it draws a series of boxes that are color-coded to indicate the frame in which the translation occurs. At low magnifications, it draws the amino acid sequence of the resulting protein. Amino acids that are created by a splice are optionally shown in a distinctive color. Credit: <http://search.cpan.org/~lds/Bio-Graphics-2.39/lib/Bio/Graphics/Glyph/cds.pm>

Glyph



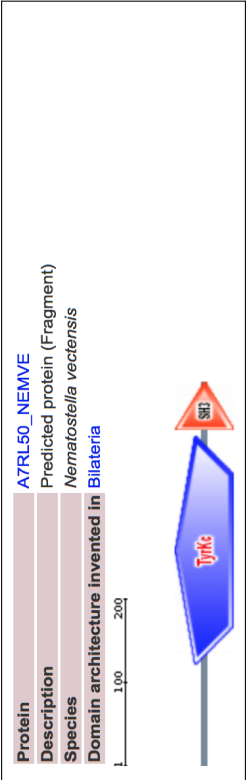
Glyphs on the track

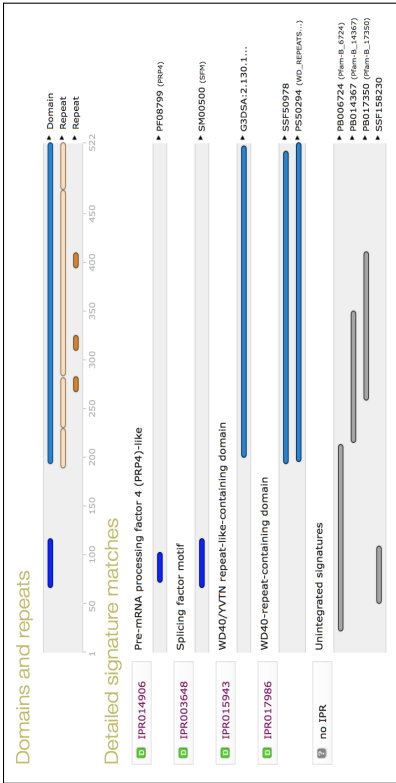


Appendix B

Protein Domain Catalog

Table 10: Catalog of visualization tools for Protein Domain Architecture.

Vis. Num	Vis. Name	Source	Description
10.1	SMART	(http://smart.embl-heidelberg.de/smart/show_many_proteins.pl)	SMART (Simple Modular Architecture Retrieval Tool) is a database of conserved domains that allows automatic identification and annotation of domains in user-supplied protein sequences. Th SMART data are used to create position-specific scoring matrix (PSSMs) used in the CD-Search.
Visualization Sample			
			
10.2	Interpro	(http://www.ebi.ac.uk/interpro/)	InterPro provides functional analysis of protein sequences by classifying them into families and predicting the presence of domains and important sites. To classify proteins in this way, InterPro uses predictive models, known as signatures, provided by several different databases (referred to as member databases) that make up the InterPro consortium.
Visualization Sample			



The Pfam database is a large collection of protein families, each represented by multiple sequence alignments and hidden Markov models (HMMs). Proteins are generally composed of one or more functional regions, commonly termed domains. Different combinations of domains give rise to the diverse range of proteins found in nature. The identification of domains that occur within proteins can therefore provide insights into their function...

Pfam domains (<http://pfam.xfam.org/browse>)

10.3

Visualization Sample

ZABA_CANTR (P53031)

Summary

This is the summary of UniProt entry ZABA_CANTR^{SP} (P53031^{SP}).

Descriptions: Protein phosphatase PPSA regulatory subunit B
Source organism: *Candida tropicalis* (Yeast)^{SP} (NCBI taxonomy ID 5482^{SP})
Length: 508 amino acids

Please note: when we start each new Pfam data release, we take a copy of the UniProt sequence database. This snapshot of UniProt forms the basis of the overview that you see here. It is important to note that, although some UniProt entries may be updated in subsequent releases, these entries will not be removed from Pfam until the next Pfam data release.

Pfam domains

This image shows the arrangement of the Pfam domains that we found on this sequence. Clicking on a domain will take you to the page describing that Pfam entry. The table below gives the domain boundaries for each of the domains. **More...**

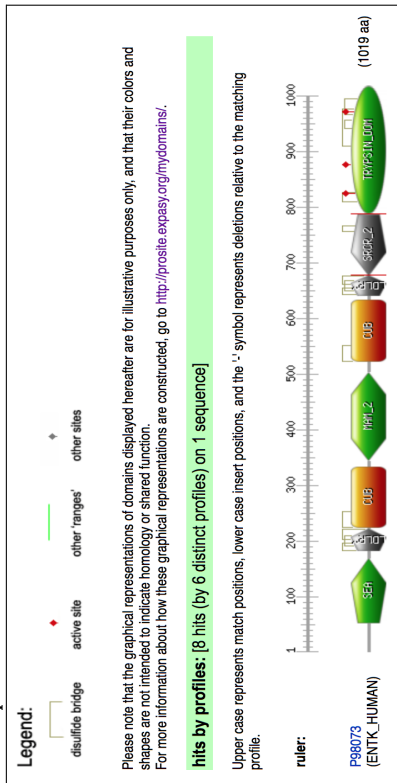
Source	Domain	Start	End
Pfam A	WD40	267	303
disorder		359	362
disorder		364	365
Pfam B	Pfam-B_334303	369	379
disorder		371	419
low_complexity		385	470
Pfam B	Pfam-B_3129	421	506
low_complexity		428	441

ScanProsite is a new and improved version of the web-based tool for detecting PROSITE signature matches in protein sequences. For a number of PROSITE profiles, the tool now makes use of ProRules context-dependent annotation templates to detect functional and structural intra-domain residues.

ScanProsite Results (<http://prosite.expasy.org/tool/scanprosite/>)

10.4

Visualization Sample

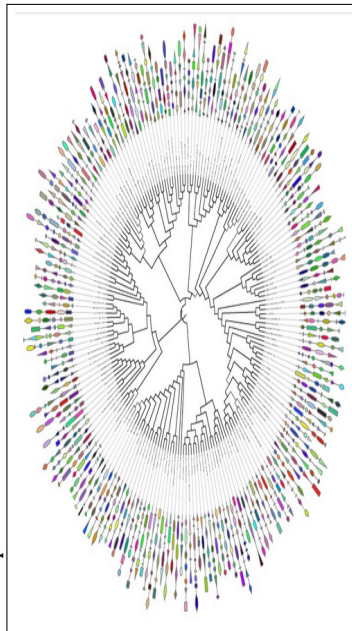


iTOL can easily display protein domain architectures directly on the tree. Each node in the tree can have a protein domain architecture associated with it.

iTOL: Protein domain architecture (<http://itol.embl.de/index.shtml>)

10.5

Visualization Sample



DOG (Domain Graph) is presented for experimentalists, to prepare publication-quality figures of protein domain structures. The scale of a protein domain and the position of a functional motif/site is precisely defined. DOG is allowing experimentalists to illustrate their own protein domain figures with ease. A four-step procedure makes DOG an easy-to-use software. The proportion of a functional domain and the position of a motif/site are precisely decided from given inputs. [Ren et al., 2009]

10.6 DOG: illustrator of protein domain structures (<http://dog.biocuckoo.org>)

Visualization Sample

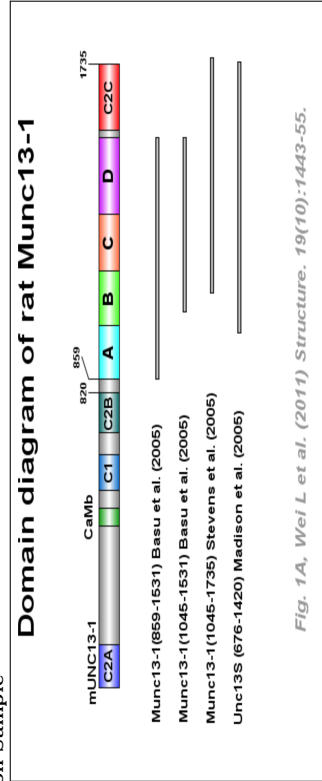


Fig. 1A. Wei L et al. (2011) Structure. 19(10):1443-55.

Conserved Domain Architecture Retrieval Tool (CDART) performs similarity searches of the Entrez Protein database based on domain architecture, defined as the sequential order of conserved domains in protein queries. CDART finds protein similarities across significant evolutionary distances using sensitive domain profiles rather than direct sequence similarity.

10.7 CDART: Domain Architecture (<http://www.ncbi.nlm.nih.gov/structure/lexington/lexington.cgi>)

Visualization Sample

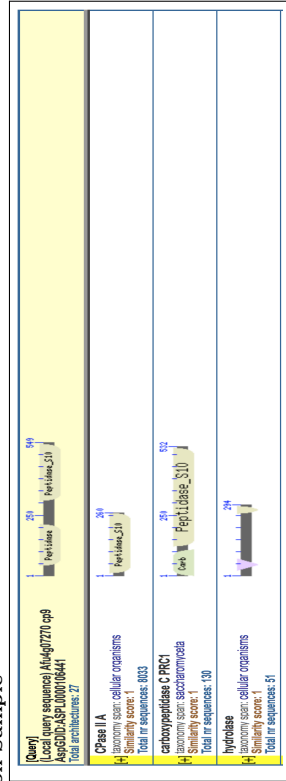
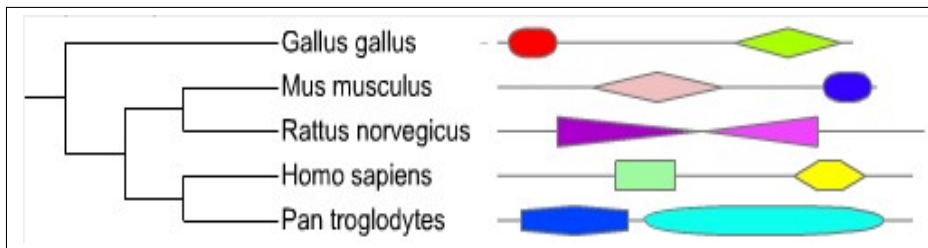


Table 11: Different glyph's for Protein Domain Architecture

Vis. Num	Vis Name	Source	Description
11.1	iTOL: Protein domain architecture	[Letunic and Bork, 2007]	<p>Each node in the tree can have a protein domain architecture associated with it. Even though its primary use is for the display of protein domains, it can be used for various other purposes. The format is as follows: Each line should have a node ID, total protein length, and the definitions of the domains. The domain definitions field contains one or more domains, separated using the same character which is used in the first two fields. Each domain definition consists of 5 parts, separated with vertical lines (). For example: (RE 100 150 #ff0000 SH2). The fields are:</p> <ol style="list-style-type: none"> 1. RE: 2 character code defining the domain shape (see below for supported shapes) 2. 100: domain start position 3. 150: domain end position 4. #ff0000: color definition (hexadecimal RGB notation) 5. SH2: domain label. <p>Exported trees with a protein domain architecture dataset will contain an additional legend listing all domains and their labels. The legend will not be shown in interactive mode, since you can hover your mouse cursor over any domain to show its label and other associated information. Domain architectures for internal nodes will only be displayed on collapsed clades. Source: http://itol.embl.de/help/help.shtml.</p>

Glyphs

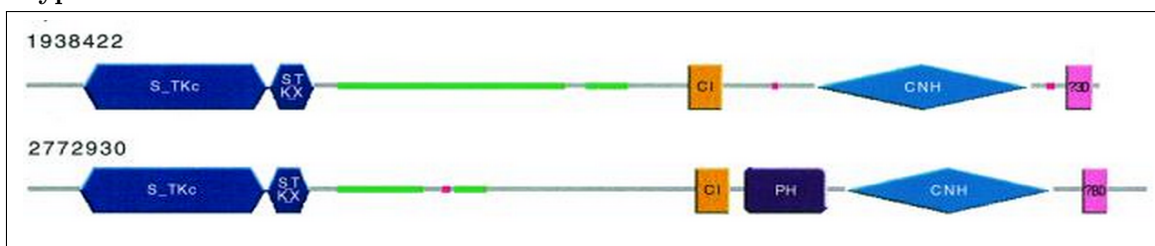


supported shapes

Code	Shape	Example
RE	rectangle	
HH	horizontal hexagon	
HV	vertical hexagon	
EL	ellipse	
DI	rhombus (diamond)	
TR	right pointing triangle	
TL	left pointing triangle	
PL	left pointing pentagram	
PR	right pointing pentagram	
PU	up pointing pentagram	
PD	down pointing pentagram	
OC	octagon	
GP	rectangle (gap)	

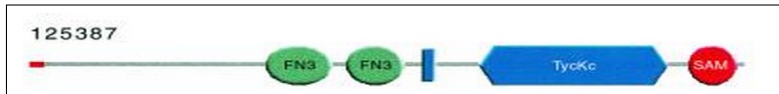
11.2	SMART: domain architecture	[Schultz et al., 2000]	“SMART (a Simple Modular Architecture Research Tool) allows the identification and annotation of genetically mobile domains and the analysis of domain architectures”. Green/grey lines indicate predicted coiled coil regions and pink lines show low complexity segments. Regions of proteins without any predicted features are marked with grey bars and can be subjected individually to gapped BLAST searches [Schultz et al., 2000].
------	----------------------------	------------------------	---

Glyphs



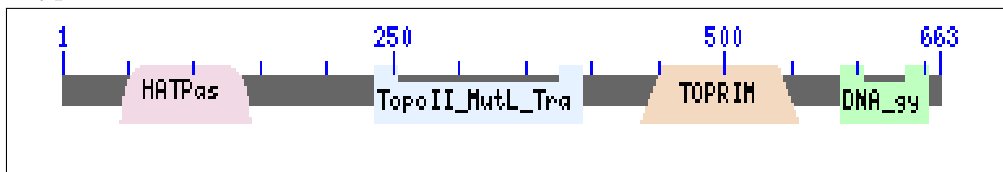
11.3	SMART: domain architecture C [Schultz et al., 2000]	Improved representation of results. SMART merges domain prediction and intrinsic features into a single line output. The red line indicates a signal sequence, the blue bar a transmembrane helix. [Schultz et al., 2000]
------	--	---

Glyphs

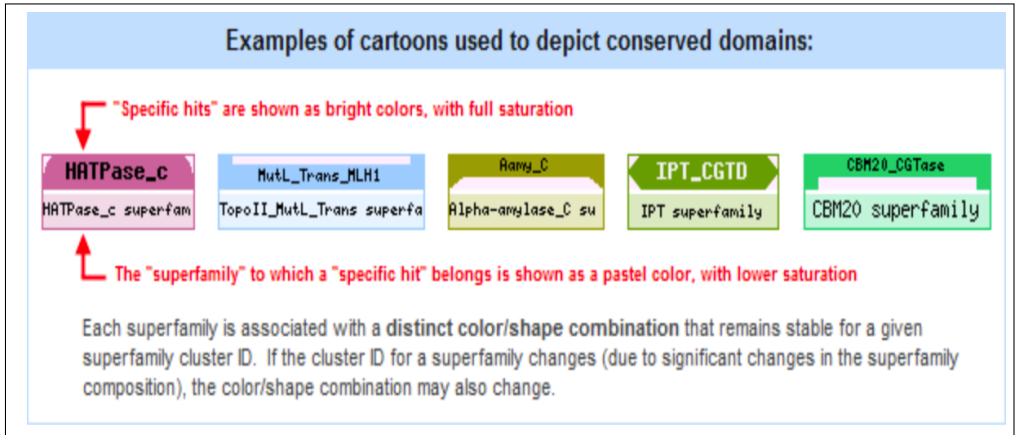


11.4	CDART: Conserved Domain Architecture Retrieval Tool (http://www.ncbi.nlm.nih.gov/Structure/lexington/lexington.cgi)	A domain architecture is defined as the sequential order of conserved domains in protein queries. Each domain architecture displayed by CDART therefore represents a unique set and order of conserved domain superfamilies found among sequences in the Entrez Protein database. “Architectures are displayed as graphs. Different superfamilies are displayed as different shape/colour combinations. Each superfamily is represented by a cartoon with a distinct color/shape combination. The color/shape combination remains stable for a given superfamily cluster ID, and is consistent across all NCBI tools that show conserved domain footprints on protein sequences. Specific hits are shown as bright colors, with full saturation. The superfamily to which a specific hit belongs is shown as a pastel color, with lower saturation. If there are no specific hits to a region of a protein query sequence, then the (Concise display) will show only the superfamily.” Source: http://www.ncbi.nlm.nih.gov/Structure/lexington/docs/cdart_help.html
------	--	---

Glyphs



Example of cartoons used to depict conserved domains



<p>11.5 DOG: Domain Graph</p>	<p>[Ren et al., 2009]</p>	<p>DOG is allowing experimentalists to illustrate their own protein domain figures with ease. The proportion of a functional domain and the position of a motif/site are precisely decided from given inputs [Ren et al., 2009]. The labels can be written in the top or in the bottom depending on the user choice.</p>
------------------------------------	---------------------------	--

Glyphs

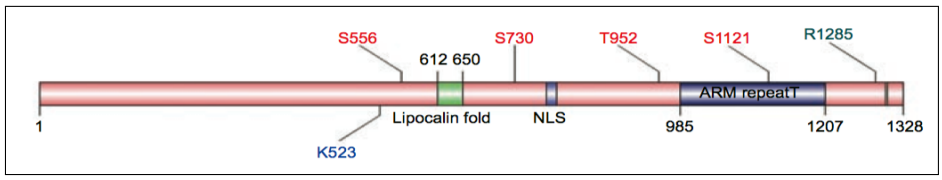









Table 12: Pfam Domain Architecture graphics. The description is taken from: <http://pfam.xfam.org/help#tabview=tab9>

Vis. Num	Vis. Name	Vis. Shape	Description
12.1	The sequence		<ul style="list-style-type: none"> • The base sequence, undecorated by any domains or features, is represented by a plain grey bar. The length of the domain graphic that is drawn is proportional to the length of the sequence itself. • The graphics in this page are drawn with a X-scale of 0.5 pixels per amino acid, so that a 400 residue sequence will result in a 200 pixel-wide image. Any domains or features which are drawn on the sequence are also scaled by the same factor.
12.2	Pfam-A: Family/domain (a full length match)		<p>The high quality, curated Pfam-A domains are classified into one of four different types: family, domain, repeat and motif.</p> <ul style="list-style-type: none"> • Both family and domain entries are rendered as rectangles with curved ends when the sequence is a <u>full length match</u>. (Sequence matches the full length of a Pfam HMM). <u>Different types of domain are displayed with different colours.</u> • When the domain image is long enough, the domain name is shown within the domain itself. From Pfam 24.0 onwards, Pfam has been generated using HMMER3, which introduces the concept of “envelope coordinates” for a match. Envelope regions are represented in domain graphics as lighter coloured regions. The graphic shows short envelope regions at the ends of both domains.

12.3	Pfam-A: Family/domain (a fragment match)		<p>Here sequence matches a portion of an HMM, so matching domain fragments are shown.</p> <ul style="list-style-type: none"> • When a sequence match does not pass through the <u>first</u> position in the HMM, the N-terminal side of the domain graphic is drawn with a jagged edge instead of a curved edge. • When a sequence match does not pass through the <u>last</u> position of the HMM, the C-terminal side of the domain graphic is drawn with a jagged edge. • In some rarer cases, the sequence match may not pass through either of the <u>first</u> or <u>last</u> positions of the HMM, in which case <u>both sides</u> are drawn with jagged edges.
12.4	Pfam-A: Discontinuous nested domains		<p>Some domains in Pfam are disrupted by the insertion of another domain (or domains) within them. A number of names have been given to this arrangement: discontinuous (referring to the outer domain), inserted or nested (both referring to the inner domain). For example, in many sequences containing an IMPDH domain, the IMPDH domain is continuous along the primary sequence. However, in some cases the linear sequence of the IMPDH domain is broken by the insertion of a CBS domain, as shown in the image.</p> <ul style="list-style-type: none"> • The discontinuous domain is represented in two parts (as shown in the image). These two parts are joined by a line bridging them.
12.5	Pfam-A: Repeat/motif		<ul style="list-style-type: none"> • Repeats and motifs are represented by rectangles with straight edges in order to distinguish them from domains of type family and domain. <u>Partial matches</u> are represented with jagged edges.

<p>12.6</p>	<p>Pfam-A: Context domains</p> 	<p>Context domains in Pfam are those that, despite not scoring above the family gathering threshold, are expected to be real, based on the presence of the surrounding domains found in the protein. In some cases it is possible for a protein without any matches to gain context domains. This happens when two or more <u>weak matches</u> support each other. This is most often seen with <u>multiple tandem repeats</u> such as WD40 and leucine rich repeats such as LRR_1.</p> <ul style="list-style-type: none"> • The context domains are represented by rectangles that are coloured from white to pink as shown in the image.
<p>12.7</p>	<p>Pfam other sequence motifs</p> 	<p>In addition to domains, smaller sequences motifs are represented by the domain graphics. Currently the following motifs are represented: signal peptides, low complexity regions, coiled-coils and transmembrane regions.</p> <ul style="list-style-type: none"> • Signal peptides: In Pfam, Phobius (A combined transmembrane topology and signal peptide predictor) is used for the prediction of signal peptides and it is represented graphically by a small orange box. • Low complexity regions: Within Pfam, SEG is used to calculate low complexity regions. The presence of a low complexity region is indicated by a cyan rectangle. • Coiled-coils: In Pfam, ncoils is used to identify these motifs. Coiled-coils are represented by a small lime-green rectangle. • Transmembrane regions: In Pfam, Phobius is used for the prediction of transmembrane regions, which are represented by a red rectangle.



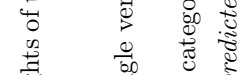

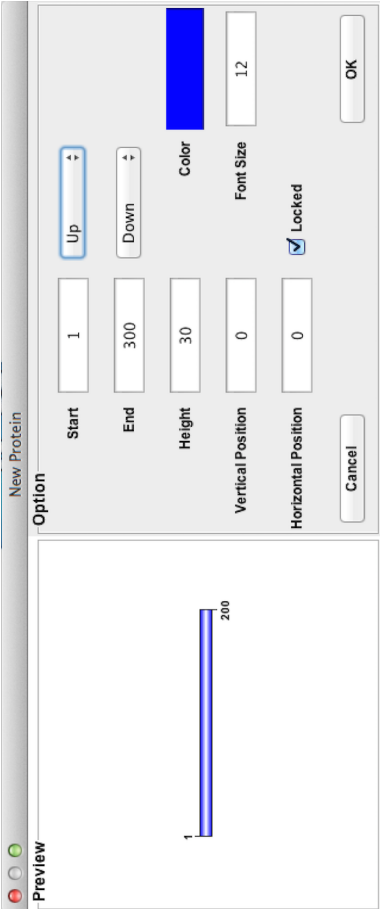

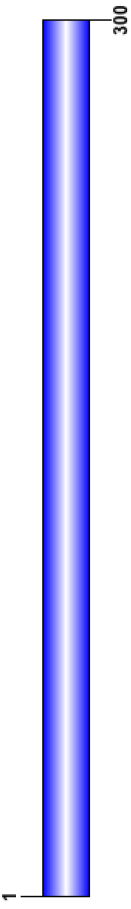
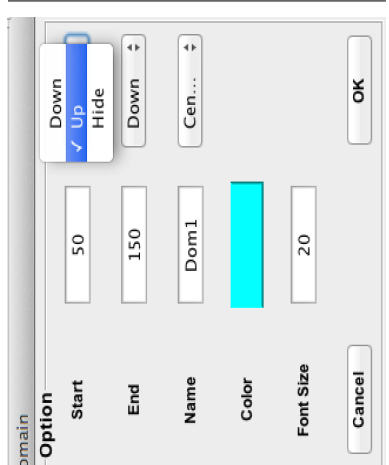
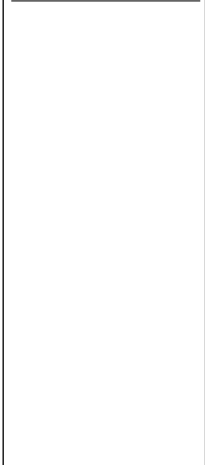
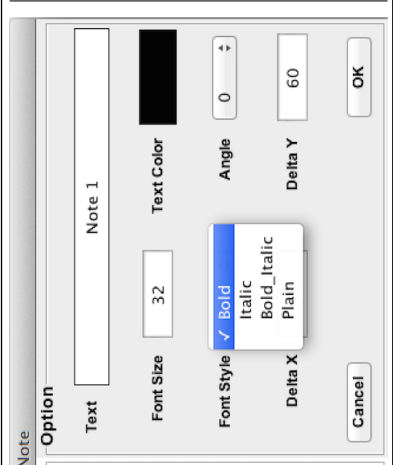

12.8	Pfam-B		<p>Pfam-B regions are automatically generated clusters that supplement the high quality Pfam-A regions.</p> <ul style="list-style-type: none"> • Pfam-B regions are represented by a small rectangle, coloured with three stripes.
12.9	Pfam: Lollipop		<p>A wide range of different lollipop styles can be create by combining different line and head colours with different drawing styles. The lollipop head can be drawn as a square, circle or diamond, as a simple coloured bar, or as an arrow (pointing away from the sequence) or a “pointer” (an arrow pointing towards the sequence).</p> <p>Each of these features can appear above or below the sequence, but in this image the disulphide bridges are shown above the sequence and the active site residues below the line.</p> <ul style="list-style-type: none"> • Disulphide bridges: The disulphide bridge annotations used in Pfam come from UniProt and are represented by a solid bridge-shaped line. <ul style="list-style-type: none"> – When multiple disulphide bonds occur, the heights of the bridges are adjusted to avoid overlaps between them. – Inter-protein disulphides are represented by single vertical lines. • Active site residues: Within Pfam there are three categories of active site: those that are <i>experimentally determined</i>, those that are <i>predicted by UniProt</i> and those predicted by Pfam. All three types are represented by a “lollipop” with a diamond head. The head is coloured red, pink and purple for each of the three types respectively.
12.10	Pfam: Other Sequence features		

Table 13: DOG: Protein Domain Structure Visualization

Vis. Num	Vis. Name	Vis.	Description
13.1	The DOG console		In this menu a new protein, domain and other features can be added.
13.2	Adding protein & its sub-features		In this menu a new protein with its sub-features can be added such as the start, end and colour.
13.3	The start-end labels position		In this menu the position of the start and end labels are chosen with three different choices: down, up or hide the label.
13.4	Protein		Protein illustration after adding the sub-features.

<p>13.5 Adding domain & its sub-features</p>		<p>In this menu a new domain with its sub-features can be added such as the start, end and colour.</p>
<p>13.6 The domain name label position</p>		<p>In this menu the position of the start and end labels are chosen with four different choices: center, down, up or hide the label.</p>
<p>13.7 Adding note</p>		<p>In this menu a note for this project can be added.</p>
<p>13.8 Note font style menu</p>		<p>In this menu the font style of the note is chosen with four different choices.</p>

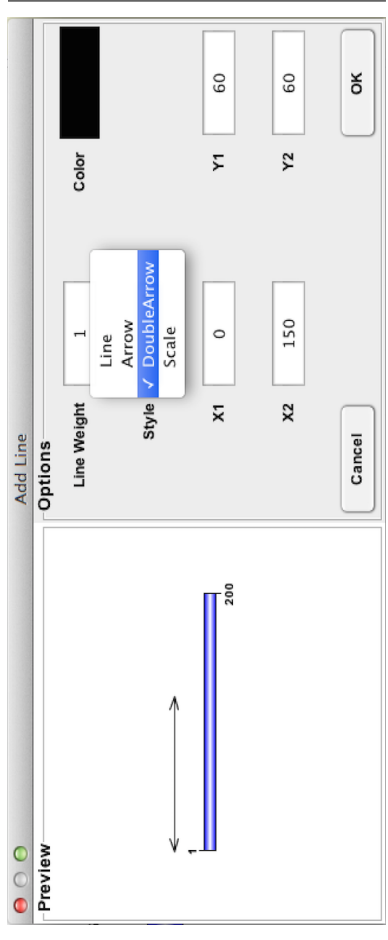

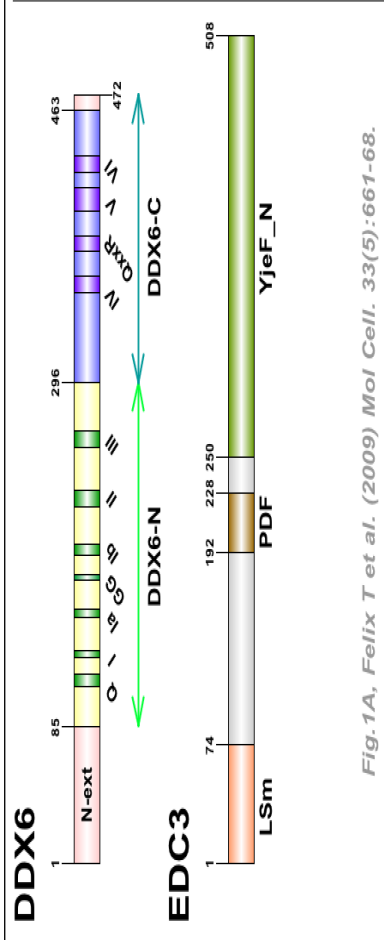
13.9 Adding line	 <p>In this menu a line for this project can be added.</p>
13.10 line style	 <p>In this menu the line style is chosen with four different choices.</p>
13.11 Example of DOG final result	 <p>An example of a protein with its domains and user defined features.</p>

Fig. 1A, Felix T et al. (2009) Mol Cell. 33(5):661-68.

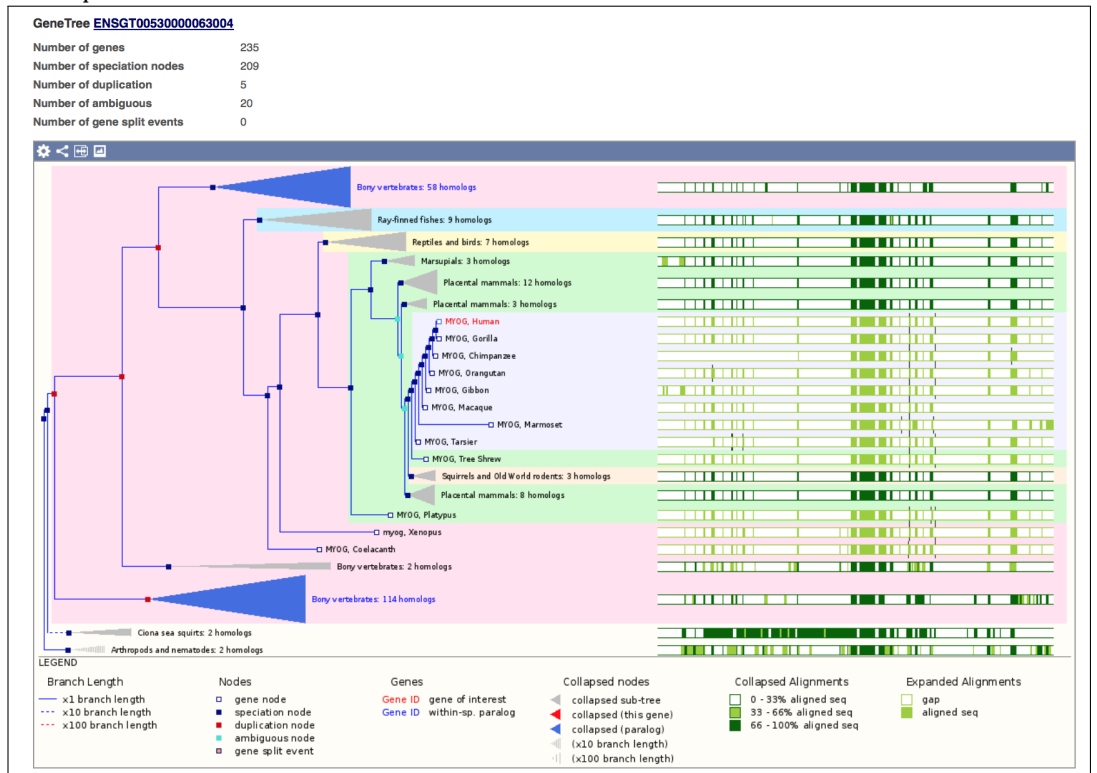
Appendix C

Comparative Genomics

Table 14: Comparative Genomics catalog

Vis. Num	Vis. Name	Source	Description
14.1	Gene tree	(http://useast.ensembl.org/info/website/tutorials/compara.html)	A representative protein for every gene in Ensembl across all species is used to determine gene trees and homologues.

Visualization Sample



Appendix D

Glossary

The glossary is combined from many web-based glossaries. They can be found at :

- <http://www.ncbi.nlm.nih.gov/Class/FieldGuide/glossary.html>
- <http://mousecyc.jax.org/glossary.shtml>
- <http://www.genome.gov/Glossary/>
- www.oxforddictionaries.com/definition/english/glossary
- http://www.biology-online.org/dictionary/Main_Page
- http://smart.embl-heidelberg.de/help/smart_glossary.shtml
- <http://www.ncbi.nlm.nih.gov/books/NBK21052/>
- <http://pfam.xfam.org/help#tabview=tab9>

Accession Number The accession number is the most general identifier used in the NCBI sequence databases. This is the identifier that should be used when citing a database record in a publication. The accession number points to a sequence record and does not change when the sequence is modified. In the Entrez system, using the accession number as a query will retrieve the most recent version of the record. The update history of a particular sequence record is tracked by the accession.version number. Changes in version numbers occur only when the actual sequence of a record has been modified and do not reflect any changes in the annotation. The specific version of a record is also tracked by another identifier that is mainly for internal NCBI use called the GI number.

Active site residues Within an enzyme, a small number of residues are directly involved in the catalysis of a reaction. These are termed active site residues.

Alias An alternative symbol or name for part of the sequence of a known gene that resembles

names for other anonymous DNA segments. For example, D6Mit236 is an alias for Cftr.

Amino acid An organic acid carrying an amino group. Proteins are linear polymers of the L-forms of 20 common amino acids linked by peptide bonds.

Annotation 1. n. Database entries that provide supplementary information about a biological entity, such as annotation of pathways or regulatory sites. 2. v. The analysis process used to create annotations such as **Sequence Annotation**.

Associated Genes Associated Genes are genes that have been linked to a gene model by hand curation.

Base pair One of the pairs of chemical bases joined by hydrogen bonds that connect the complementary strands of a DNA molecule or of an RNA molecule that has two strands; the base pairs are adenine with thymine and guanine with cytosine in DNA and adenine with uracil and guanine with cytosine in RNA.

Canonical transcript The canonical transcript is the best representative transcript for a given gene model. **Non-canonical:** All other transcripts for a gene model that are not the canonical transcript.

cDNA Complementary DNA. A DNA copy of an mRNA or complex sample of mRNAs, made using reverse transcriptase.

CDS Coding sequence; region of nucleotides that corresponds with the sequence of amino acids in a protein (location includes start and stop codons).

Central Dogma The principal statement of the molecular basis of inheritance. In its simplest form:

”DNA makes RNA makes protein.”

This means that (generally) genetic information is stored in and transmitted as DNA. Genes are expressed by being copied as RNA (transcription), which is processed into mRNA via splicing and polyadenylation. The information in mRNA is translated into a protein sequence using a genetic code to interpret three-base codons as instructions to add one of twenty amino acids or to stop translation.

Codon Three bases in a DNA or RNA sequence that specify an amino acid or a termination signal (stop codon).

Coiled-coils Coiled coils are motifs found in proteins that structurally form alpha-helices that wrap or wind around each other. Normally, two to three helices are involved, but cases of up to seven alpha-helices have been reported. Coiled-coil are found in a wide variety of proteins, many functionally very important.

Domain A domain is a discrete structural unit of a protein. In principle, **protein domains** are capable of folding independently from the rest of the protein. Domains can often be identified

by non-structural approaches based on conserved amino acid sequences. The NCBI's CDD-search uses information from curated multiple sequence alignments to identify domains in protein sequences.

Domain composition Proteins with the same domain composition have at least one copy of each of domains of the query.

Ensembl Ensembl is a joint project between EBI-EMBL and the Sanger Institute to provide automatic annotation of eukaryotic genomes.

Entrez Entrez is an integrated search and retrieval system that integrates information from various databases at NCBI, including nucleotide and protein sequences, 3D structures and structural domains, genomes, variation data (SNPs), gene expression data, genetic mapping data, population studies, OMIM, taxonomy, books online, and the biomedical literature.

Essential amino acid Essential amino acids cannot be made by the body. As a result, they must come from food. The nine essential amino acids are: histidine, isoleucine, leucine, lysine, methionine, phenylalanine, threonine, tryptophan, and valine.

Eukaryote A cell or organism with membrane-bound, structurally discrete nucleus and other well-developed sub-cellular compartments.

Evidence Type The source of evidence to support the gene model.

Exon Part of a gene whose sequence is present in a mature mRNA after splicing.

Expect Value (E-value) In BLAST statistics, the Expect value is the number of alignments with a particular score, or a better score, that are expected to occur by chance when comparing two random sequences.

Gene The fundamental unit of inheritance, comprising a segment of DNA (or RNA in some viruses) that codes for one or several related functions and occupies a fixed position (locus) on a chromosome. "proteins coded directly by genes". **(In technical use)** a distinct sequence of nucleotides forming part of a chromosome, the order of which determines the order of monomers in a polypeptide or nucleic acid molecule which a cell (or virus) may synthesize.

Gene expression The conversion of the information from the gene into mRNA via transcription and then to protein via translation resulting in the phenotypic manifestation of the gene.

Gene Model A representation of a gene that contains information about features of the transcript such as exon- intron boundaries, splice sites, UTRs, etc. A **gene model** is a mapping of gene features such as coding regions and exon intron boundaries onto the the genomic DNA of an organism. *Gene models* typically provide a predicted transcript and protein sequence. A simple kind of gene model can be made by aligning an expressed sequence (cDNA) to the genomic DNA sequence. More precise exon intron boundaries can be identified by constraining the aligned segments using consensus splicing signals.

Gene product A gene product is any protein or enzyme that occurs as a result of translation of a gene. A gene, then, refers to a particular piece of DNA that, in genetic code, contains the recipe for a gene product.

Gene products are the actual materials that carry on the daily functions of life, such as digesting foods, making new living tissue, and converting materials into energy; actions that virtually all living things undergo in life.

Intron Part of a gene whose sequence is transcribed but not present in a mature mRNA after splicing.

Locus Literally, “place”. The location of a gene or set of genes on a chromosome.

Low complexity regions Low complexity regions are regions of biased sequence composition, usually comprised of different types of repeats. These regions have been shown to be functionally important in some proteins, but they are generally not well understood and are masked out to focus on globular domains within the protein.

Motif A motif is a short, well-conserved nucleotide or amino acid sequence that represents a minimal functional domain. It is often a consensus for several aligned sequences. The PROSITE database is a popular collection of protein motifs, including motifs for enzyme catalytic sites, prosthetic group attachment sites (heme, biotin, etc), and regions involved in binding another protein. Examples of DNA motifs are transcription factor binding sites.

mRNA Messenger RNA. An RNA molecule that is the product of transcription of a gene. In eukaryotes, that molecule often has to be spliced and polyadenylated before it can be translated into a protein product.

Mutation

1. The process through which genes undergo a structural change.
2. Any permanent change in DNA, i.e., in its nucleotide sequence. Examples include chromosome rearrangements and point mutations.

Nonessential amino acid An amino acid that can be synthesized in the body and does not have to be obtained from the diet.

Open Reading Frame (ORF)

1. A length of nucleotide sequence that lacks termination codons in a given reading frame.
2. An open reading frame is a portion of a DNA molecule that, when translated into amino acids, contains no stop codons. The genetic code reads DNA sequences in groups of three base pairs, which means that a double-stranded DNA molecule can read in any of six possible reading frames—three in the forward direction and three in the reverse. A long open reading frame is likely part of a gene.

Orthologue Orthologues are genes derived from a common ancestor through vertical descent. This is often stated as the same gene in different species. In contrast, paralogs are genes within the same genome that have evolved by duplication.

Paralog Paralogs are usually described as genes within the same genome that have evolved by duplication.

PFAM Pfam is a database of protein domain families represented as (i) multiple alignments, and (ii) HMM-profiles.

Prokaryote A microscopic single-celled organism that has neither a distinct nucleus with a membrane nor other specialized organelles. Prokaryotes include the *bacteria* and *cyanobacteria*.

Protein A large complex molecule made up of one or more chains of amino acids joined by peptide bonds. Proteins are the principal constituents of cellular material and serve as enzymes, hormones, structural elements, and antibodies; they are synthesized in the body from their constituent amino acids.

Protein folding “Protein folding is the process by which a protein structure assumes its functional shape or conformation. It is the physical process by which a polypeptide folds into its characteristic and functional three-dimensional structure from random coil. Each protein exists as an unfolded polypeptide or random coil when translated from a sequence of mRNA to a linear chain of amino acids. This polypeptide lacks any stable (long-lasting) three-dimensional structure. Amino acids interact with each other to produce a well-defined three-dimensional structure, the folded protein (the right hand side of the figure), known as the native state. The resulting three-dimensional structure is determined by the amino acid sequence (Anfinsen’s dogma). Experiments beginning in the 1980s indicate the codon for an amino acid can also influence protein structure. Source: http://en.wikipedia.org/wiki/Protein_folding.

Sequence Alignment A sequence alignment is a residue by residue comparison of two or more sequences. In the alignment, the relative positions of the sequences are adjusted to optimize (usually maximize) the alignment score derived by reference to some scoring matrix. In some cases gaps with associated penalties may be inserted into one or more sequences to optimize the alignment score.

Sequence Annotation 1. n. Additional information added to genomic sequence to identify genes, delimit the intron and exon structures of those genes, identify regulatory elements, note the positions of allelic variation, etc. 2. v. The analysis process used to create sequence annotations. The process relies heavily on the homology principle, whereby similarity to known genes is used to help identify new genes and propose functions for them.

Sequence ID (SeqID) Sequence accession identifier. A unique alphanumeric character string that unambiguously identifies a sequence record in a database. Examples of genomic sequence providers are NCBI and Ensembl; examples of sequence IDs from these providers are 16590 and ENSMUSG00000053869, respectively.

Signal peptides Signal peptides are short regions (<60 residues long) found at the N-terminus of proteins, which direct the post-translational transport of a protein and are subsequently

removed by peptidases. More specifically, a signal peptide is characterized by a short hydrophobic helix (approximately 7-15 residues). This helix is preceded by a slight positively charged region of highly variable length (approximately 1-12 residues). Between the hydrophobic helix and the cleavage site is a somewhat polar and uncharged region, of between 3 and 8 amino-acids.

SMART SMART (Simple Modular Architecture Retrieval Tool) is a database of conserved domains that allows automatic identification and annotation of domains in user-supplied protein sequences. The SMART data are used to create one of the sets of PSSMs used in the CD-Search.

Splice In genetics: splice means Join or insert (a gene or gene fragment): “they have spliced a gene into tomatoes that improves flavour”

Splicing Part of the processing of an RNA transcript into mRNA, in which introns are removed enzymatically.

Strain Strain is a low-level taxonomic rank used in three related ways. In Microbiology, a strain is a genetic variant or subtype of a microorganism (e.g. virus or bacterium or fungus). In plants, a strain is a designated group of offspring that have descended from a modified plant, produced either by conventional breeding or by biotechnological means or result from genetic mutation. In rodents, a strain is a group of animals that is genetically uniform.

Stop Codon A stop codon is a trinucleotide sequence within a messenger RNA (mRNA) molecule that signals a halt to protein synthesis. The genetic code describes the relationship between the sequence of DNA bases (A, C, G, and T) in a gene and the corresponding protein sequence that it encodes. The cell reads the sequence of the gene in groups of three bases. Of the 64 possible combinations of three bases, 61 specify an amino acid, while the remaining three combinations are stop codons.

Structured Data Structured data are data that have been represented in a manner that allows computation with those data. Data become structured when they are carefully dissected and assigned to distinct fields of a database with clearly defined meanings, so that the data are independently queryable and computable. Therefore, we can ask questions across the data such as “find all enzymes that use magnesium as a cofactor” or “find all pathways in which pyruvate is an input substrate”.

Structural Gene A gene that encodes an enzyme or structural protein, in contrast to a regulatory gene.

Structural Protein A protein that functions as a structural element of cells rather than as an enzyme, for example, collagen.

STS Sequence Tagged Site. A short segment of unique sequence derived from genomic DNA. A large collection of STSs can be used to assemble a physical map of the genome from a collection of genomic clones (e.g., BACs or YACs) by testing each clone for the presence of each STS.

Two clones that contain one or more STSs in common must overlap. For examples, see the physical maps of the mouse genome at MGI.

Synthesize (chemistry, biochemistry) To produce substance by combining chemical precursors.

Tab-delimited file A text file that uses tabs to separate adjacent fields. It is a common format for downloading information into a spreadsheet.

Taxon A stable unique identification number for the taxon of the source organism. A taxonomy ID number is assigned to each taxon (species, genus, family, etc.) in the NCBI Taxonomy Database. See also the Organism field, above.

Transcription The enzymatic synthesis of an RNA molecule directed by information in a DNA molecule.

Translation The enzymatic synthesis of a protein molecule directed by the information in an mRNA molecule. The mRNA is read from the 5' end to the 3' end, with the protein being synthesized from the amino terminus to the carboxyl terminus.

Transmembrane domain Transmembrane domain usually denotes a single transmembrane alpha helix of a transmembrane protein. Also known as an integral protein.

Transmembrane protein Membrane protein that extends through the lipid bilayer, with part of its mass on either side of the membrane.

tRNA Transfer RNA. Small RNA molecules that bind to the codons of mRNA in the ribosome after being "charged" with amino acids.

Visualization idiom is a specific sequence of data enrichment and enhancement transformations, visualization mappings and rendering transformations that produce an abstract display of a scientific data set.

X Chromosome One of pair of chromosomes that is sexually dimorphic in mammals. Normal female mammals have two X chromosomes, while normal male mammals have an X chromosome and a Y chromosome.

Y Chromosome One of pair of chromosomes that is sexually dimorphic in mammals. Normal female mammals have two X chromosomes, while normal male mammals have an X chromosome and a Y chromosome.

3' (3-prime) A term that identifies one end of a single-stranded nucleic acid molecule. The 3' end is that end of the molecule which terminates in a 3' phosphate group. The 3' direction is the direction toward the 3' end. Nucleic acid sequences are written with the 5' end to the left and the 3' end to the right, in reference to the direction of DNA synthesis during replication (from 5' to 3'), RNA synthesis during transcription (from 5' to 3'), and the reading of mRNA sequence (from 5' to 3') during translation.

3' UTR 3' Untranslated Region. That portion of an mRNA from the 3' end to the position of the last codon used in translation.

5' (5-prime) A term that identifies one end of a single-stranded nucleic acid molecule. The 5' end is that end of the molecule which terminates in a 5' phosphate group. The 5' direction is the direction toward the 5' end. Nucleic acid sequences are written with the 5' end to the left and the 3' end to the right, in reference to the direction of DNA synthesis during replication (from 5' to 3'), RNA synthesis during transcription (from 5' to 3'), and the reading of mRNA sequence (from 5' to 3') during translation.

5' UTR 5' Untranslated Region. That portion of an mRNA from the 5' end to the position of the first codon used in translation.