COMPUTATIONAL VERIFICATION OF PUBLISHED HUMAN MUTATIONS

FREDRICK KINYUA KAMANU

Thesis submitted in fulfilment of the requirements for the degree of Magister Scientiae, at the South African National Bioinformatics Institute (SANBI), University of the Western Cape.

November 2008

Supervisors: Prof. Heikki Lehväslaiho & Prof. Vladimir Bajic

KEY WORDS

Bioinformatics

Human variation

Relational database

Database development

Mutation verification

Mutation checker

MutRes

ABSTRACT

Computational verification of published human mutations

F.K Kamanu

MSc Thesis, South African National Bioinformatics Institute, University of Western Cape

The completion of the Human Genome Project, a remarkable feat by any measure, has provided over three billion bases of reference nucleotides for comparative studies. The next, and perhaps more challenging step is to analyse sequence variation and relate this information to important phenotypes. Most human sequence variations are characterized by structural complexity and, are hence, associated with abnormal functional dynamics. This thesis covers the assembly of a computational platform for verifying these variations, based on accurate, published, experimental data. Before the actual design could be implemented, the methodology adopted for each of the software components constituting the mutation verification platform covers the requirement analysis, conceptualization of the design, prototyping and validation. Once the software has been put in place, the final design pattern looks at its ease of evolution and the subsequent maintenance issues. Following the implementation, each software component has been discussed under three functional levels namely; structure, navigation and presentation.

The web facilitates cross-border collaborative research, hence, the ability to easily develop sophisticated web-based applications, without compromising safety and performance, is of critical importance. The website hosting the mutation verification platform has been designed using Cascading Style Sheets (CSS), Perl/CGI scripting and a TWiki framework. The project runs on a FreeBSD UNIX server and the website can be accessed at http://mutation.sanbi.ac.za. The

knowledge gained through accurate analysis of sequence variation data could aid in drawing conclusive inferences regarding the medical state of an entire physiological system.

DECLARATION

I declare that *"Computational Verification of Published Human Mutations"* is my own work, that it has not been submitted for degree or examination at any other university, and that all the sources I have used or quoted, and all work which was the result of joint effort, have been indicated and acknowledged by complete references.

Fredrick Kinyua Kamanu **November 2008**

Signed: _____

ACKNOWLEDGEMENT

I would like to express my deep and sincere gratitude to my supervisors, Professor. Heikki Lehväslaiho and Professor. Vladimir Bajic, for all the support and guidance they have given me. I would also wish to thank Marius Albertyn, Dale Gibbs and Peter Van Heusden for the technical assistance they offered me during the setting up of the mutation verification platform. Special thanks to Dr. Adam Dawe for reliable tips especially with regard to the write-up.

Finally I want to express my deepest gratitude towards my family for their love and support during my study.

TABLE OF CONTENTS

KEY WORDS....II
ABSTRACT....III
DECLARATION....V
ACKNOWLEDGEMENT....VI
TABLE OF CONTENTS....VII
LIST OF FIGURES....IX
LIST OF TABLES....X
LIST OF ILLUSTRATIONS....XI
ABBREVIATIONS....XII
PREFACE....XV
Chapter 1: Introduction....1
1.1. Motivation and rationale....1
Chapter 2: Theoretical Framework....4
2.1. Human sequence variation: Brief overview....4
2.2. Mutation detection and quantification....8
2.3. Mutation description and nomenclature....15
2.4. The Human Variome Project (HVP)....17
2.5. Need for human mutation databases....18
Chapter 3: Technology Applied....22
3.1. Database Theory....22
3.1.1. Relational databases....22
3.1.2. Database normalization....22
3.1.3. Structured Query Language (SQL)....23
3.1.4. The Perl Database Independent Interface (DBI)....23
3.1.5. Federated databases....24
3.2. Software Development....25
3.2.1. UNIX....25
3.2.2. Practical Extraction and Report Language (PERL)....26
3.2.3. BioPerl....26

3.3. Web Development..... 28
3.3.1. Overview.....28
3.3.2. Apache web server.....30
3.3.3. TWiki.....31
3.3.4. Common Gateway Interface (CGI)..... 32
3.3.5. Cascading Style Sheets (CSS)..... 33
Chapter 4: Research Design and Methodology.....34
4.1. The mutation server.....34
4.2. Mutation Checker.....35
4.3. Database of Mutation Databases and Related Resources (MutRes).....40
4.4. Federated Mutation Database (FMD).....45
4.5. Project website.....46
4.6. Other tools..... 47
Chapter 5: Results: Presentation and Discussion..... 48
5.1. The mutation server..... 48
5.2. Mutation Checker..... 49
5.3. Database of Mutation Databases and Related Resources (MutRes).....54
5.4. Federated Mutation Database (FMD).....60
5.5. Project website.....62
Chapter 6: Conclusion and Recommendations..... 64
References..... 65
Appendices..... 71
Appendix I: Multi-line remover script..... 71
Appendix II: MutRes statistics gathering script..... 73
Appendix III: MutRes database validity script.....75
Appendix IV: Sample status output file.....77
Appendix V: Sample text output from MutRes database..... 78
Appendix VI: Mutation server sample output..... 79
Appendix VII : Table of optional Perl modules required by TWiki..... 80
Appendix VIII : Table summarizing TWiki features 81
Appendix IX : TWiki directory structure..... 83

LIST OF FIGURES

Figure 1: Structural variation within the human genome.....5
Figure 2: Genetic approach to disease diagnosis and treatment.....7
Figure 3: Microarray-based Genomic Selection (MGS).....9
Figure 4: 454-Sequencing of individual genome.....11
Figure 5: Percentage of nsSNPs predicted.....13
Figure 6: Size distribution of coding indels.....14
Figure 7: Description of sequence variants. 16
Figure 8: Mutation data in HGMD.....19
Figure 9: Cross-annotation of the dbSNP.....21
Figure 10: Database Independent Interface for Perl.....23
Figure 11: Federated database architecture.....24
Figure 12: UNIX file system.....25
Figure 13: BioPerl object model.....27
Figure 14: Sequence retrieval.....27
Figure 15: Representation of a web-based application.....28
Figure 16: Life cycle of web application.....29
Figure 17: The HTTP Request life cycle.....30
Figure 18: Perl CGI Session.....32
Figure 19: Cascading Style Sheets (CSS) application.....33
Figure 20: Mutation Checker web form.....51
Figure 21: MutRes main search page.....54
Figure 22: An ERD highlighting MutRes database schema.....55
Figure 23: MutRes results web display.....56
Figure 24: MutRes page for an entry edit.....57
Figure 25: Mutres addition web page.....57
Figure 26: Sample listing of verified mutation databases.....61
Figure 27: Project web page.....63

LIST OF TABLES

Table 1: Sequence variant description nomenclature.....15
Table 2: Mutation Checker parameters..... 37
Table 3: Critical Perl modules for running TWiki 47
Table 4: Command-line options for the Mutation Checker.....50
Table 5: URL table information for the MutRes database..... 58
Table 6: Distribution of Hugo approved genes in MutRes and the LSDB listing..58
Table 7: Federate Mutation Database.....60

LIST OF ILLUSTRATIONS

Illustration 1: Loading of name spaces 36
Illustration 2: Implementation of the command-line options.....38
Illustration 3: Automatic sequence download.....39
Illustration 4: Creating a log file for usage statistics.....39
Illustration 5: SQL code snippet highlighting the resource table design.....41
Illustration 6: SQL code snippet highlighting the gene table design.....41
Illustration 7: SQL code snippet highlighting the contact table design.....42
Illustration 8: Code snippet highlighting the link table design..... 43
Illustration 9: Code snippet highlighting the URL table design.....43
Illustration 10: Code snippet highlighting the reference table design.....44
Illustration 11: Use of MySQL Join Query..... 45
Illustration 12: Login into the mutation server..... 48
Illustration 13: File transfer.....48
Illustration 14: The mutation server terminal.....49
Illustration 15: Server web access.....50
Illustration 16: Command-line options for the Mutation Checker.....50
Illustration 17: Mutation Checker flat format results.....52
Illustration 18: Mutation Checker XML format results..... 53

ABBREVIATIONS

API	Application Programming Interface
CDS	Coding Sequences
CNV	Copy Number Variation
CPAN	Comprehensive Perl Archive Network
CSS	Cascading Style Sheets
DBMS	Database Management System
DBS	Database System
DBI	Database Independent Interface
DDL	Database Definition Language
DML	Database Manipulation Language
DNA	Deoxyribonucleic Acid
DSCA	Double Strand Conformation Analysis
EMBL	European Molecular Biology Laboratory
EMD	Enzymatic Mutation Detection
ERD	Entity Relationship Diagram
FMD	Federated Mutation Database
GDB	Genome Database
GNU	GNU's Not Unix
GPL	General Public Licence
HGMD	Human Gene Mutation Database
HGP	Human Genome Project
HGVS	Human Genome Variation Society

HTML	Hypertext Mark-up Language
HTTP	Hypertext Transfer Protocol
HUGO	Human Genome Organization
HUGO-MDI	Human Genome Organization Mutation Database Initiative
HVP	Human Variome Project
IP	Internet Protocol
IBM	International Business Machines
LSDB	Locus Specific Database
MIM	Mendelian Inheritance in Man
NHGRI	National Human Genome Research Institute
NCBI	National Center for Biotechnology Information
OMIM	On-line Mendelian Inheritance in Man
OS	Operating System
PCR	Polymerase Chain Reaction
PERL	Practical Extraction and Report Language
RNA	Ribonucleic Acid
SANBI	South African National Bioinformatics Institute
SCP	Secure Copy Protocol
SNP	Single Nucleotide Polymorphism
SSCP	Single Strand Conformation Polymorphism
SSH	Secure Shell
SSL	Secure Sockets Layer
SQL	Structured Query Language
URL	Uniform Resource Locator

WWW World Wide Web

XML Extensible Marker Language

PREFACE

The work undertaken during the writing of this thesis forms part of the Human Sequence Variation Project. The project has a decade long history in defining human gene variation nomenclature and uniting clinically oriented geneticists maintaining Locus Specific Mutation Databases, towards an agreement on a common endeavour.

UNIVERSITY of the
WESTERN CAPE

Chapter 1: Introduction

1.1. Motivation and rationale

With a wide array of applications, the study of human sequence variations plays a fundamental role in understanding structural genomic complexities and how they affect an organism's phenotype. Some of the areas in which these findings are applied include genomics, gene finding and forensics. The study of human sequence variations also aims at shedding light into the mechanisms underlying critical processes such as evolution, phenotypic adaptations, inherited syndromes, and also in human development and tissue compatibility (Lehväslaiho, **2000**).

Recently, substantial progress has been made in disease genetics and genome-related medicine, which has led to a vast amount of data being generated. Unfortunately, this progress has not been matched by adequate database projects aimed at gathering and organizing these datasets, in order to enable their useful exploitation (Patrinos & Brooks, **2005**).

Computational analysis plays a critical role in modern biological research, with numerous application software, geared towards data analysis, being developed. However, it has proved quite difficult to automatically combine and analyse data from disparate sources (Stajich *et al.,* **2002**). Rapid growth of the Internet and the cost-effective proliferation of its key supporting technologies are revolutionizing information technology, and hence, the field of computational biology. This has opened up new opportunities for developing large-scale distributed applications (Joshi *et al.,* **2001**).

With the Internet facilitating cross-border collaborative research, the ability to easily develop sophisticated web-based applications, without compromising safety and performance, is of critical importance in the post-genomic era. Lack of proper disease-genetics data organization has been addressed in this thesis by designing a database of published human mutations, dubbed MutRes. Using Structured Query Language (SQL), we have put in place a mechanism that facilitates easy retrieval of accurate information regarding human sequence variations. The problem encountered while analysing datasets originating from disparate sources has been circumvented through the development of a Federated Mutation Database (FMD), whose logic lies squarely at achieving data integration.

A central component of this thesis is the Mutation Checker. This is a computational tool which aids the study of sequence variations, by facilitating the verification of transcription and translation effects of these variations, at the molecular level.

This thesis has been written in six chapters. Chapter one provides the motivation and the rationale underlying the work. Chapter two offers the theoretical framework upon which the work carried out in this thesis has been reviewed. The review includes a brief description of human sequence variations, how they are detected and the nomenclature used to catalogue them. The Human Variome Project (HVP) is also reviewed and the need for human mutation databases is established. Chapter three reviews the technologies used in the development of a mutation verification platform, the key component of this work. This chapter has been reviewed under three broad categories, namely the database theory, the software development as well as the web development aspect.

Chapter four lays out the methods used in the development of the various tools constituting the mutation verification platform. Under each component, four central themes are looked at. These include the requirement analysis,

conceptualization, prototyping and validation as well as design and implementation. In chapter five, the results obtained in the development of the key components of the mutation verification platform are presented and discussed by looking at the structure, navigation and presentation. Chapter six, which is the final chapter, gives a conclusion and articulates the recommendations which could be proposed following the successful completion and implementation of all the development tasks.

Chapter 2: Theoretical Framework

2.1. Human sequence variation: Brief overview

Discovering the genetic basis of human phenotypic differences calls for an in-depth understanding of all forms of genetic variation (Eichler, **2006**). An estimated ninety percent (90%) of sequence variants in humans are differences in single bases of Deoxyribonucleic Acid (DNA), referred to as single nucleotide polymorphisms (SNPs). It has also been found that SNPs in the coding regions of genes (cSNPs) or in regulatory regions are more likely to cause phenotypic alterations than SNPs elsewhere (Collins, **1998**). Despite enormous advances in uncovering the pattern and nature of single-nucleotide differences, a similar effort in documenting larger and more complex forms of genetic variation still lags behind (Eichler, **2006**).

The completion of the Human Genome Project (Collins & McKusick, **2001**), which was a remarkable feat, provided three billion bases of reference nucleotides for comparative studies. The development and application of modern technologies to detect the extent and position of genomic alterations within this dataset have made it evident that large fragments of the human genome have been deleted or duplicated (Lupski, **2007**). Although assembling a working draft of the human sequence marks a turning point in molecular genetics, vast amount of additional work remains to be done to comprehend its functionality (Collins & McKusick, **2001**). The identification of possible genes that confer susceptibility or resistance to common human ailments should be possible, with improved molecular techniques aimed at finding DNA sequence variants on a genome-wide scale (Collins *et al.,* **1998**).

A number of molecular genetic and cytogenetic analytical protocols have been developed and subsequently applied in curating numerous sequence variations in the human genome. These variations (Figure 1) take different forms within the human genome which include single-nucleotide polymorphisms, small insertion-deletion polymorphisms, variable numbers of repetitive sequences, as well as genomic structural modifications (Iafrate *et al.,* **2004**).

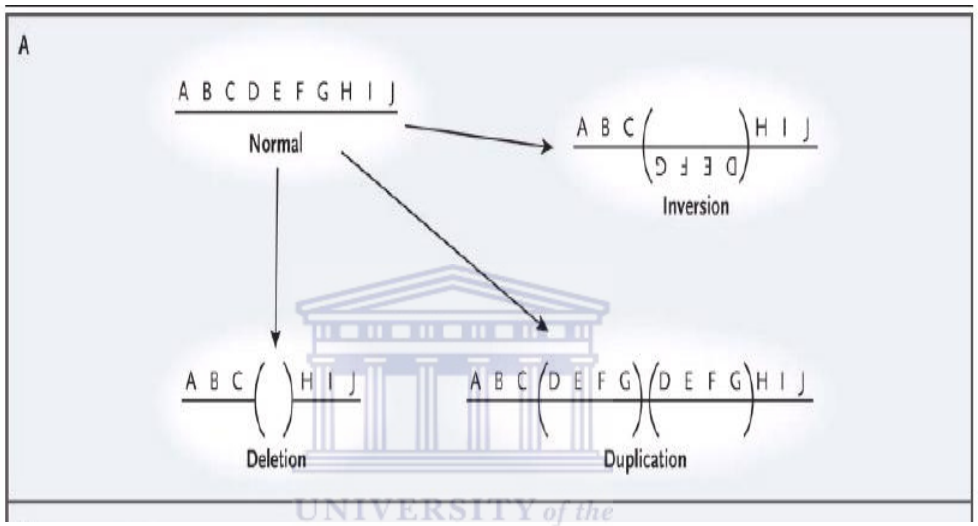

Figure 1: Structural variation within the human genome.

Some of the most common structural variations within the human genome. The lines with letters above them depict the genome (Lupski, **2007**).

An inversion occurs when a segment of a genomic sequence is turned through one hundred and eighty degrees in geometric space. A deletion results from the severing and subsequent removal of a section within the genomic segment. Duplication arises from doubling of a given section leading to the elongation of that genomic sequence. Deletions and duplications result in variations in copy number in the sequences contained within the rearrangement. Although phenotypic manifestations of copy number variation may be benign, susceptibility to complex diseases such as Alzheimer's, Parkinson's disease and Crohn's disease may arise (Lupski, **2007**) .

Of critical importance within the field of molecular genetics is the establishment of the fraction of genetic variation that would constitute non-neutral allelic variants. This would enable in quantifying variants that affect phenotype and which are hence subject to the natural selection pressure (Sunyaev *et al.,* **2001**)

Currently, human geneticists are engaged in an ambitious task to identify and curate SNPs associated with various phenotypes. Special attention is given to the phenotypes associated with severe and complex human diseases. Progress in this endeavour has, for a long time, been hampered by the immense number of SNPs to be analysed as well as the complex nature of many phenotypes of interest (Sunyaev *et al.,* **2001**).

However, as pointed out by Bodmer and Bonilla (**2008**), the search for common variants with characterized correlation to the occurrence of a given disease can now be carried out without making any prior assumptions as to the type of the variant under investigation. This has been made possible through development of technologies and procedures that facilitate sufficient screening of a large number of well-spaced SNPs that offer the most complete genomic span. By scanning nearby genes for variants that possibly satisfy the requirement for an effect on the disease trait, it is technically possible to isolate the true disease-associated variants.

Given the fact that amino acid variants may impact folding, interaction sites, solubility or stability of a peptide, establishing a link between DNA variation and the variability observed at the level of a phenotype can be facilitated by studying the impact of DNA variation on the structure and function of proteins. The fundamental role of the functional analysis of amino acid allelic variants was discovered almost ten years ago which was well before the discovery of SNPs.

However only recent accumulation of data on human variations, curated in databases such as HGBASE, dbSNP and others, that has enabled large-scale studies aimed at linking genetic variations to their corresponding phenotypes as well as measuring key population genetic constants (Sunyaev *et al.,* **2001**).

Figure 2 outlines the key steps involved in a genetic approach to the diagnosis and treatment of a disease. When this approach is applied, the rate of progress differs from case to case and mainly depends on the research investment as well as the degree of biological complexity that underlies the disease condition.

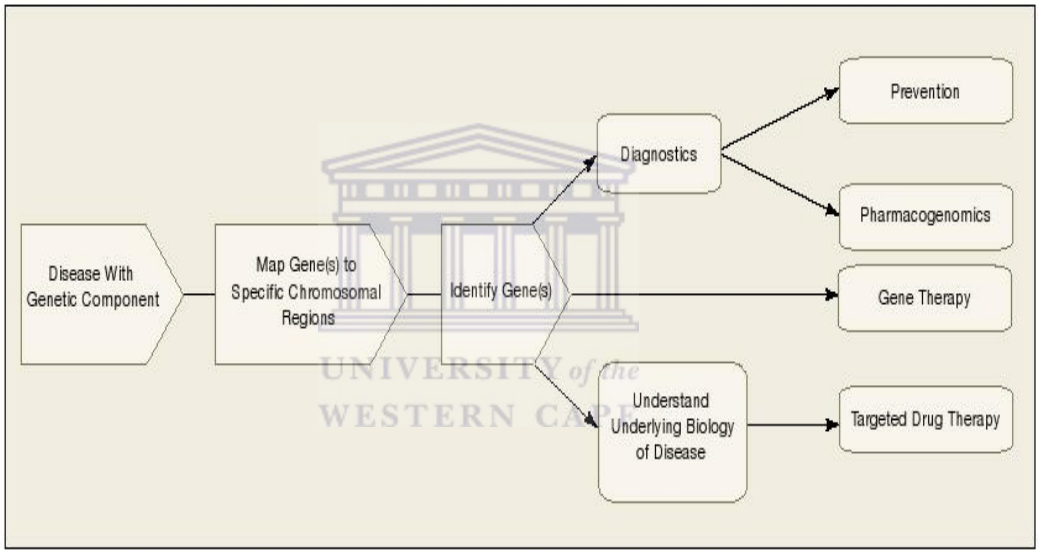

Figure 2: Genetic approach to disease diagnosis and treatment.

Steps involved in a genetic approach to the diagnosis and treatment of a disease (Collins & McKusick, **2001**)

2.2. Mutation detection and quantification

It has been observed that a large number of human genetic diseases are as a result of small genetic alterations which include single base substitutions as well as small additions or deletions. Such alterations which may be inherited, arise *de novo* in the germ-line, leading to sporadic diseases or may be acquired somatically, leading to cancerous phenotypes (Wagner *et al.*, **1995**). However, Mendelian (single gene) inherited human diseases constitute only a small part of the inherited-diseases spectrum. There are a number of cases where multiple genes have been known to play a role in the disease phenotype. Such cases include asthma and schizophrenia.

The development of diagnostic protocols for these DNA alterations will enhance both the preventive and therapeutic measures for a wide array of genetic and gene-related diseases. In addition, the development of technologies that enable the scanning of large DNA samples for mutations will enhance large scale studies of polymorphism in human and other species and also in identifying novel genes and their molecular role in disease conditions (Wagner *et al.,* **1995**). Advancement in DNA sequencing technology offers extensive, yet cost-effective, systematic ascertainment of sequence. However, there still lies a problem in isolating the target genomic segment to be sequenced. Huge eukaryotic genomes are complex in nature, hence there is a need to understand this complexity using sequence amplification techniques (Okou *et al.,* **2007**)

To address the challenges encountered in isolating any given user-defined genomic segment from the eukaryotic genomes, a Microarray-based Genomic Selection (MGS) procedure (Figure 3) has been developed. The protocol involves five main steps: (i) physical cutting of genomic DNA to create random fragments, each with an average size of three hundred base-pairs, (ii) addition of 3-prime adenine overhangs to repair the sheared ends. This is followed by ligating them to unique thymine overhangs which carry complementary base-pairs, (iii) use of a

custom high-density oligonucleotide microarray to hybridize and capture the fragments. The oligonucleotide microarray consists of complementary sequences derived from a reference genome sequence, (iv) elution of fragments bound to the probes, and (v) selected fragments are amplified through one round of Polymerase Chain Reaction (PCR) using the adaptors as a single set of primers.

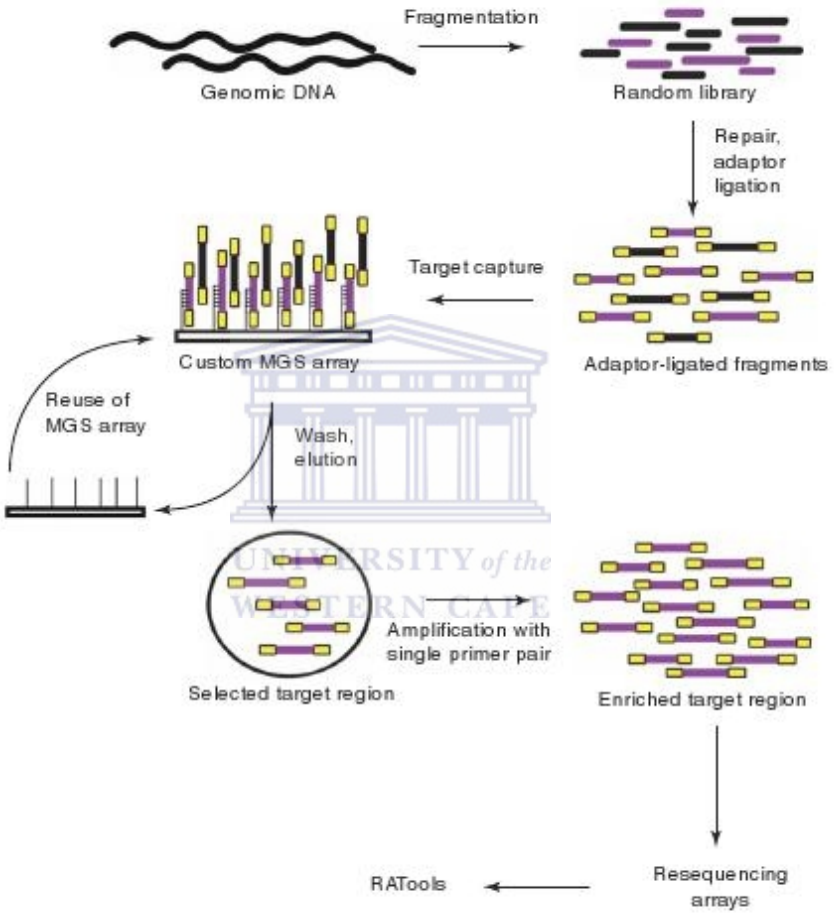

Figure 3: Microarray-based Genomic Selection (MGS).

The steps involved in a microarray-based genomic selection and resequencing of complex genomes (Okou *et al.,***2007**).

Using massively parallel sequencing, the DNA sequence of James D. Watson was sequenced to 7.4-fold redundancy (Figure 4a) in just two months at approximately one-hundredth of the cost of conventional capillary electrophoresis methods. To compile the genomic diversity, a total of 106.5 million high-quality reads were

generated by 454-sequencing, which represented an estimate 24.5 billion DNA bases. Stringent criteria was used to filter out reads that aligned to the genome hence ensuring mapping accuracy. The blue line in figure 4b depicts the overall coverage for all the SNPs, while the red line highlights only markers which exhibited a single allele by DNA sequencing. The alignments generated between the uniquely mapped reads and the reference genome were used to identify genetic variation in the subject's DNA. These sequence variations include single nucleotide polymorphisms (SNPs), small insertions and deletions (indels), and copy number variation (Wheeler *et al.,* **2008**).

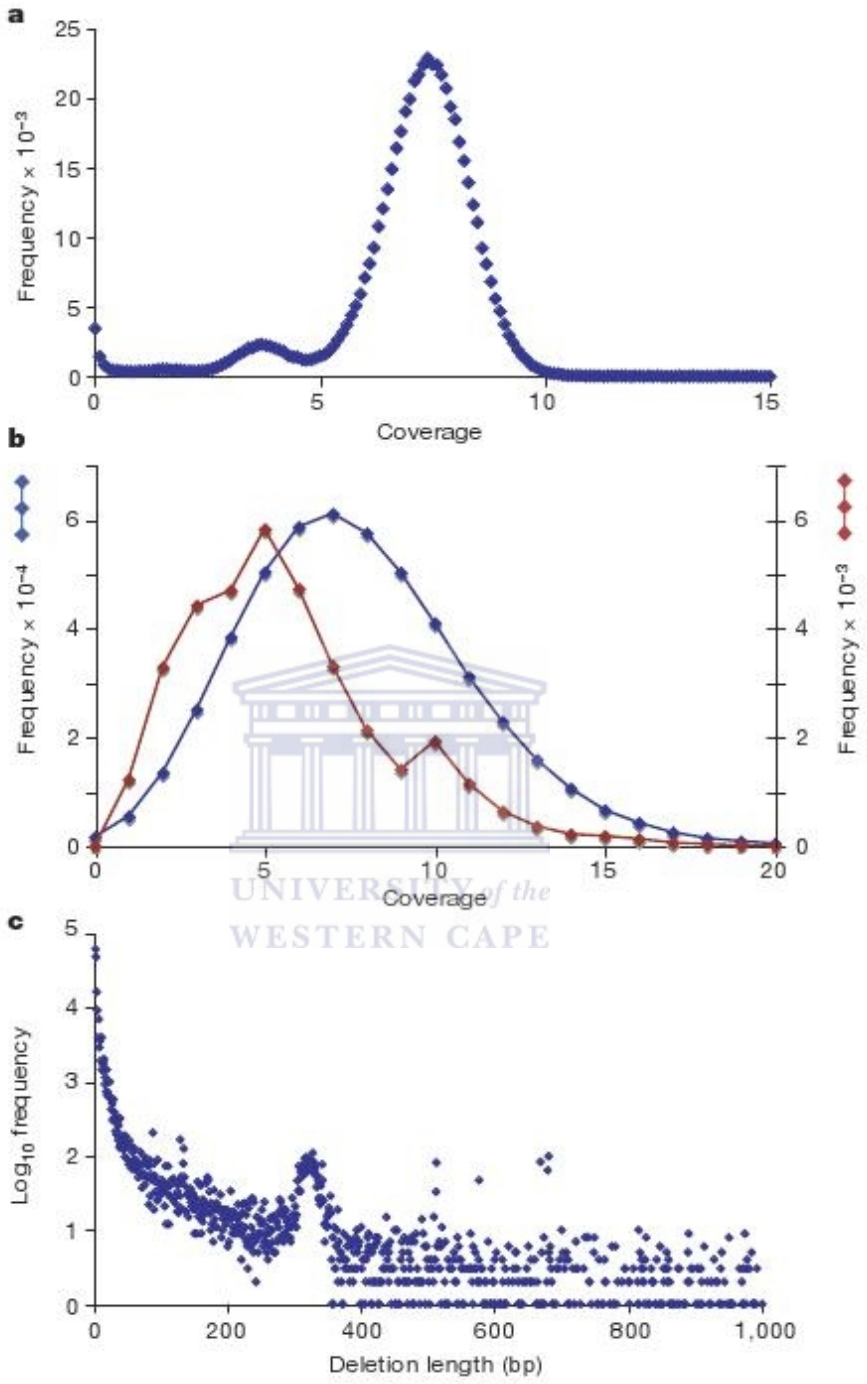

Figure 4: 454-Sequencing of individual genome.

a) Poisson distribution with a mean of 7.4-fold.
b) Poisson distribution of sequence coverage of reference genome.
c) Size distribution of deletions.
(Wheeler *et al,,* **2008**).

Comparison of that sequence to the reference genome led to the characterization of 3.3 million single nucleotide polymorphisms, of which 10,654 cause amino-acid substitution within the coding sequence. In addition, small-scale insertion and deletion polymorphism were accurately identified, as well as copy number variation that result in large-scale gain and loss of chromosomal segments, ranging from 26,000 to 1.5 million base-pairs. The results obtained concurred with the more recent results obtained following the sequencing of a second individual, Craig Venter, by traditional methods (Ng *et al.*, **2008**). However, in addition to being quicker and more cost effective, the 454-sequencing technology avoids the unnecessary loss of genomic sequences found in random shotgun sequencing by bacterial cloning. This could be attributed to the cell-free DNA amplification system.When putative SNPs catalogued with this technology in the subject's genome are compared with those in the dbSNP database (dbSNP: http://www.ncbi.nlm.nih.gov/projects/SNP/), 2.72 million are found to be common, given the fact that approximately 99% of SNPs in dbSNP are bi-allelic. A critical finding was that only in 10,425 positions did the subject's variant differ with the variant found in dbSNP (Wheeler *et al,,* **2008**).

Determining whether a sequence variant in a gene affects its protein product is of critical importance in human genetics. Ng *et al* (**2008**) has carried out a study focussing the analysis on the exome of an individual human, Craig Venter, and has identified 1,600 variants that potentially affect protein function and may be involved in phenotypic effects (Figure 5).

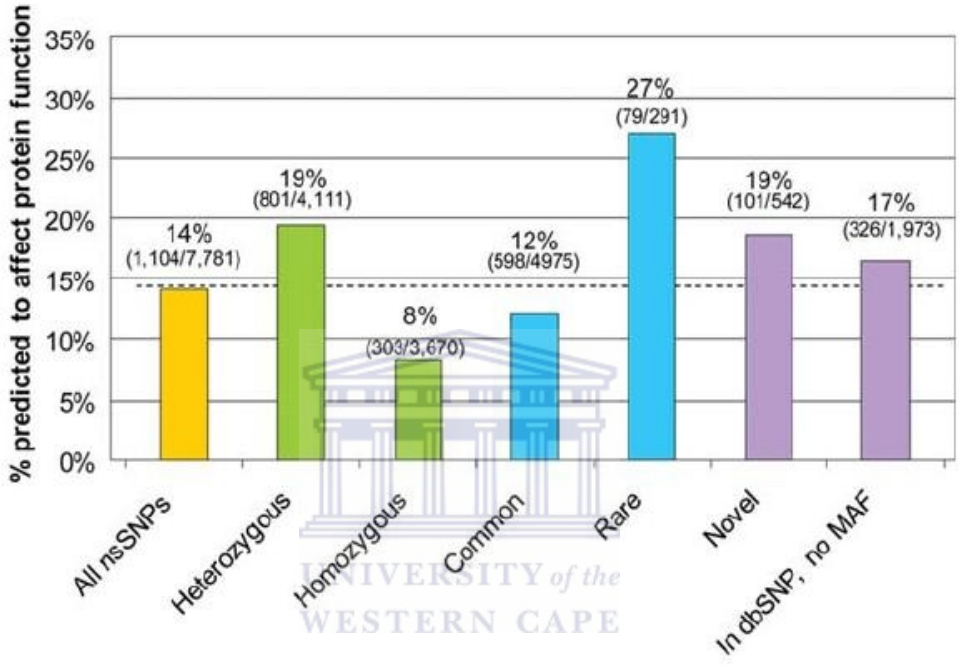

Figure 5: Percentage of nsSNPs predicted.

A bar chart illustrating the percentage of non-synonymous single nucleotide polymorphisms predicted to affect protein function (Ng *et al.*, **2008**).

The Venter genome was found to have a total of 739 coding insertion/deletions (indels), of which 281 were heterozygous and 458 homozygous (Figure 6). Kidd *et al* (**2008)** has constructed clone-based maps of eight human genomes which highlights how a significant portion of CNVs are still uncovered.

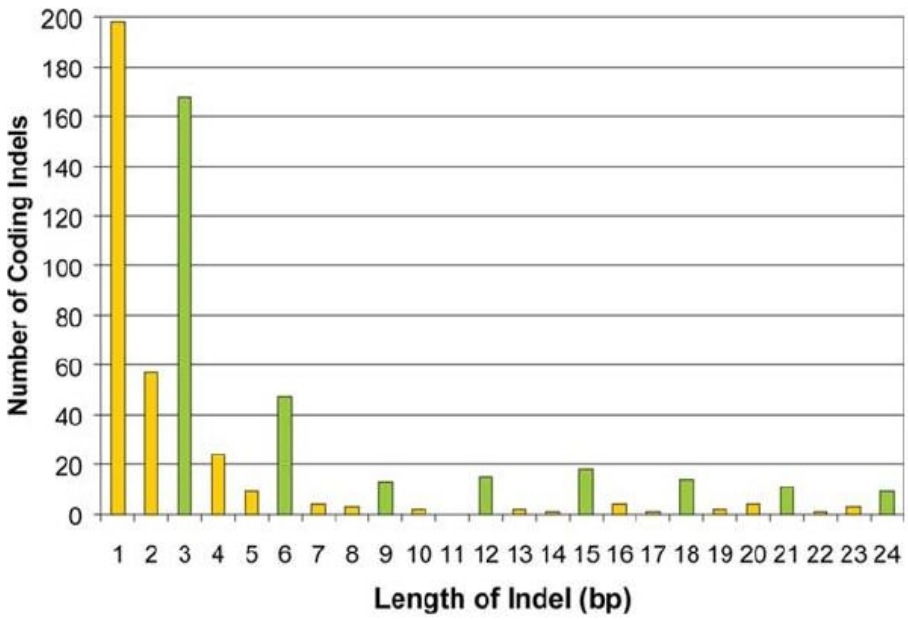

Figure 6: Size distribution of coding indels.

Mathematically, coding indels are predominantly the size of 3n, where n is an integer. Multiples of 3 in indels are so common because they lead to inframe mutations that are less likely to disrupt protein function than frameshift mutations. (Ng *et al.*, **2008**)

2.3. Mutation description and nomenclature

Current scientific usage applies the terms mutation and polymorphism synonymously in an effort to define genetic variations. However, there is no agreement about the definitions of these terms, and they do not fully define all the variants (Condit *et al.,* **2002**). Guidelines for naming alleles have now been implemented and strong recommendations have been made for their use as shorthand measures for mutation description (Scriver *et al.,* **1999**).

The Human Genome Variation Society (HGVS) (Human Genome Variation Society, **2008**) recommends that all variants should be described at the most basic level, which is the DNA level. These descriptions should however always be in relation to a reference sequence, which could either be genomic or a coding DNA. Despite a genomic reference sequence been the best, practically a coding DNA is employed as it is more convenient.

In order to avoid confusion in the description of a variant, the variant should be preceded by a letter that indicates the type of reference sequence used (Table 1). It is possible to use several different reference sequences (Figure 7).

Table 1: Sequence variant description nomenclature.

Nomenclature for describing human sequence variation, as proposed by the HGVS (Human Genome Variation Society, **2008**)

Letter	variant	example
c	Coding DNA	c.76A>T
g	Genomic sequence	g.476A>T
m	Mitochondrial sequence	m.8993T>C
r	RNA sequence	r.76a>u
p	Protein sequence	r.76a>u

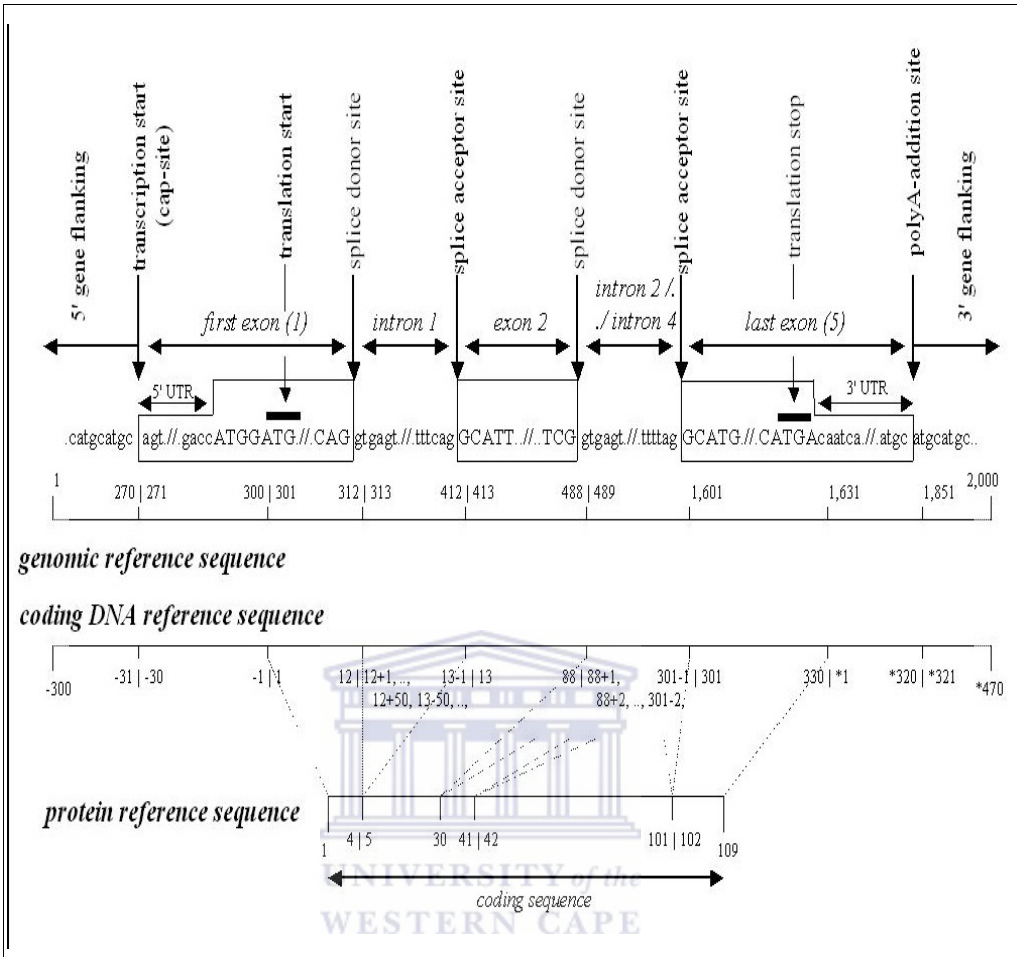

Figure 7: Description of sequence variants.

A graphical illustration of the nomenclature for describing sequence variants (Human Genome Variation Society, **2008**)

2.4. The Human Variome Project (HVP)

Through a series of meetings held over the last few years, the Human Genome organization (HUGO) has recognized that coordinating research between groups curating human mutations sequence data is not only desirable but quite necessary (Lehväslaiho *et al.,* **1998**). The HVP is an ambitious project that has been set up to facilitate this group effort. By establishing that all human genetics and genomics contribute to a single aim, the HVP welcomes contributions from all sources hence reduces duplication of effort while increasing credit for participation (Nature Genetics Editorial, **2007**).

Lehväslaiho (**2000**) had proposed that the potential for mutation data could be maximised through integration of these data into a conceptually single database. This would not only be essential in the analysis of sequence variations, but would be crucial in quantification and mapping of normal genomic variations within and among populations. This knowledge could be used by a wide array of professions, especially within the medical fraternity.

The HVP aims to get all contributors to use compatible nomenclature and phenotype reporting systems and to index variant and phenotype data to gene models in the coordinate system generated by the Human Genome Project. However, proper reporting mechanisms ought to be put in place so as to avoid creating unnecessary new databases, complicated procedures and bureaucracy (Nature Genetics Editorial, **2007**).

2.5. Need for human mutation databases

Despite recent progress in disease genetics and genome-related medicine generating vast amounts of data, comprehensive coverage of these data by any central database has not yet been achieved. This could be attributed to computational challenges encountered in handling mutation data. Such challenges include gathering, exchange, integration and interpretation (Lehväslaiho *et al.,* **1998;** Patrinos & Brookes, **2005**). Independent locus-specific mutation databases are growing at a rapid rate as a response to pressing research and clinical needs, unfortunately, their usefulness is curtailed by their lack of uniformity in design, content as well as nomenclature (Lehväslaiho *et al.,* **1998**).

Mutation databases can be categorized into two main groups namely; the core or central databases and the locus specific databases (LSDBs). The principle behind the establishment of core databases is an effort to map all described mutations in all genes, but with each mutation being represented in limited detail. On the contrary, LSDBs curate one or a few specific genes, usually related to a specified disease condition. LSDBs which hold rich and informative data are highly curated and the information contained within them may be used to complement that in the core databases (Patrinos & Brookes, **2005**).

Although there are no official statistics on the Internet regarding the usage of the information stored in general mutation databases, their relevance cannot be underscored. Among many other applications, they are used in functional genomics to identify essential base or amino acid changes. Other than detailing the actual mutation information, such as position and source, general databases highlight other properties of the mutation as well as the patient concerned. Links and mutation maps are usually included as well (George *et al.,* **2008**)..

The Human Gene Mutation Database (HGMD) contains a comprehensive catalogue of data on germ-line mutations in nuclear genes underlying or associated with human inherited diseases (Stenson *et al.,* **2003**). Within the HGMD, information about each mutations in a given gene is allocated its own web page. Bi-directional links between the HGMD (Figure 8) and other databases such as OMIM and Genome Database (GDB) enhance meaningful integration with phenotypic, structural and mapping information.

TABLE 1. Summary of Mutation Data in HGMD, March 2003

Mutation type	Number of entries
Single base-pair substitutions	
Missense/nonsense	22,682
Splicing	3,783
Regulatory	370
Other lesions	
Micro-deletions (≤20 bp)	6,587
Micro-insertions (≤20 bp)	2,573
Indels (≤20 bp)	377
Gross (>20 bp) deletions	2,172
Gross (>20 bp) insertions and duplications	348
Complex rearrangements (including inversions)	452
Repeat variations	71
Total	39,415

Figure 8: Mutation data in HGMD.

Mutation data contained in the Human Gene Mutation Database as of the year 2003 (Stenson *et al.,* **2003**)

Locus-specific mutation databases, which are curated by experts in a given gene or disease, play a fundamental role in the publication and access of mutation data and they mostly include gene or disease-specific information, phenotypes and population frequency data (Stenson *et al.,* **2003**). Similar to the general mutation databases, a lack of uniformity in the design makes searching for mutations in closely-related genes quite frustrating.

The National Center for Biotechnology Information (NCBI) has established the dbSNP database in response to a need for a general catalogue of genome variation geared toward addressing the large-scale sampling designs needed by association studies, gene mapping and evolutionary biology (Sherry *et al.,* **2001**). The quality of data within the dbSNP is enhanced through cross-annotation with other internal information resources such as the PubMed and GenBank entries (Figure 9).

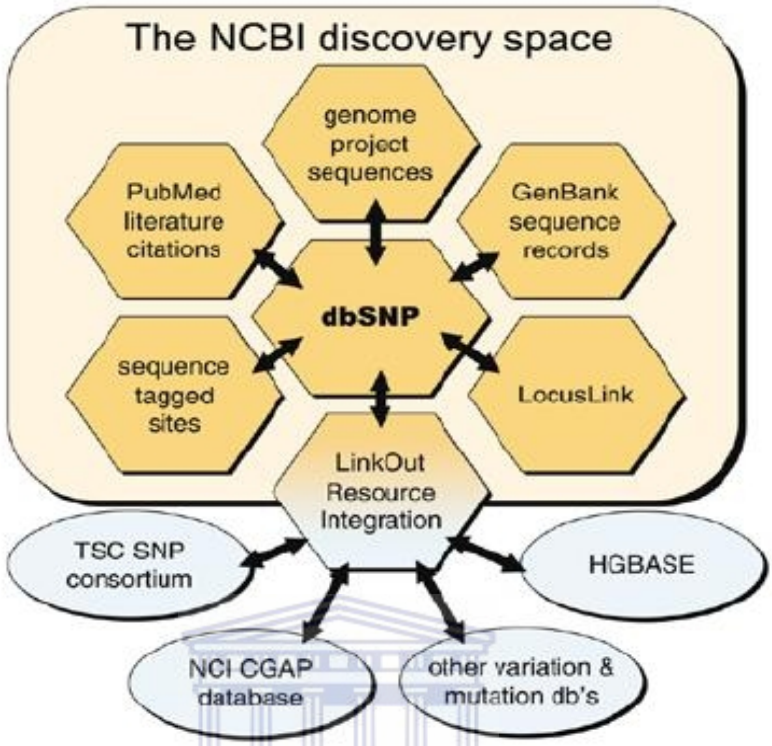

Figure 9: Cross-annotation of the dbSNP.

Cross-annotation of the dbSNP within other internal information sources at the National Center for Biotechnology Research (NCBI) (Sherry et al., **2001**)

Chapter 3: Technology Applied

3.1. Database Theory

3.1.1. Relational databases

A database is a collection of one or more related tables, and the software that governs the storage, retrieval, deletion, security, and integrity of data within a database is called a Database Management System (DBMS) (Moorehouse & Barry, **2004**; Ambler, **2008**). The most common way of manipulating data contained within Relational Databases is through the use of Structured Query Language (SQL) (Moorehouse & Barry, **2004**). Buch (**2002**) has made an observation, that the manageability of a Relational Database Management System is proportional to the number of entities it defines and maintains.

3.1.2. Database normalization

Normalization is the process of efficiently organizing data in a database which is aimed at eliminating redundant data, such as duplication, and also ensuring that sensible data dependency mechanisms are implemented (Chappel, **2008**). Other than enhancing proper disc-space usage, database normalization also guarantees data logic. Norm forms are a series of guidelines developed by database communities for ensuring that databases are normalized. Norm forms are numbered from the lowest which is referred to as the first norm, to the highest which is referred to as the fifth norm (Chappel, **2008**).

3.1.3. Structured Query Language (SQL)

SQL is a unifying technology that is built into most modern database systems and typically provides two key functionalities; a database Definition Language (DDL) and a Data Manipulation Language (DML) (Moorehouse & Barry, **2004**). The data definition component of SQL enables creation of databases as well as tables within those databases. SQL allows all forms of table alterations including structural modifications as well as renaming and deletions. The data manipulation component facilitates data modification, that is, addition and removal. Of critical importance, is the SQL search-and-extract mechanism which is commonly used in establishing data relationships using tables in a database (Moorehouse & Barry, **2004**).

3.1.4. The Perl Database Independent Interface (DBI)

The DBI is a database access module written in the Perl programming language. DBI constitutes a set of methods, variables, and conventions aimed at providing a consistent database interface, regardless of the actual database being used (Bunce, **2008**). The combination of SQL, Perl and DBI is fundamental in the design of custom programs that interact with any database system. This leads to development of optimal applications with relatively little effort (Moorehouse & Barry, **2004**). Figure 10 highlights the architecture of a DBI application.

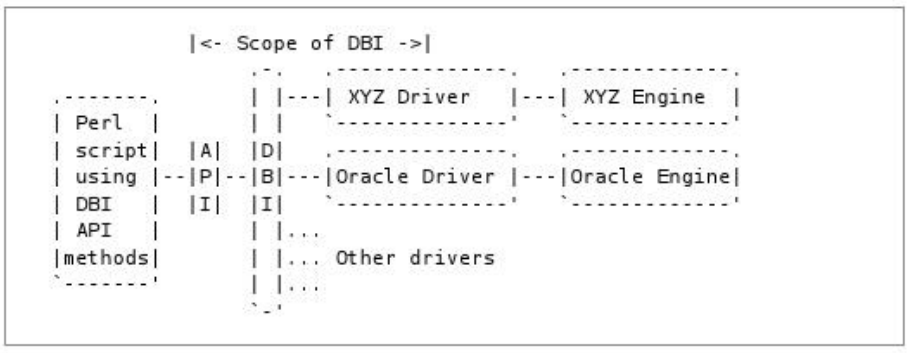

Figure 10: Database Independent Interface for Perl.

Architecture of a DBI Application (Bunce, **2008**).

3.1.5. Federated databases

A logical unification of distinct databases running on independent servers constitute a Federated Database. These databases share no resources and are connected by a Local Area Network (LAN) (Buch, **2002**). Figure 11 illustrates the architecture of a Federated Database System.

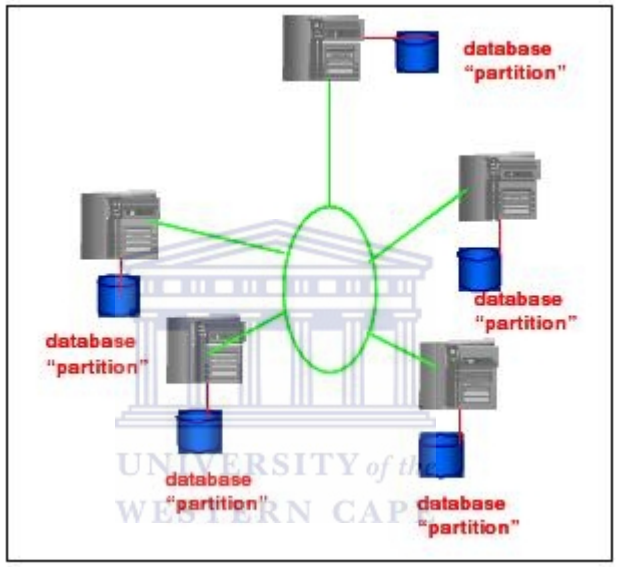

Figure 11: Federated database architecture.

Graphical illustration of the federated database architecture (Buch, **2002**).

3.2. Software Development

3.2.1. UNIX

Unix is an Operating System (OS) commonly used both in servers and in work stations. There are a number of Unix variants which include, AIX (IBM), Solaris (SUN), Xenix, Linux and the FreeBSD. They all conform to the Posix standard and often comes with a variety of command shells, such as the Bourne, Korn and C shell (Haden, **2008**). The FreeBSD has a strong bias towards web, mail, file, and support services and is hence the most famous for its strengths as an Internet server. In most instances, it is applied as the OS of choice and as an underlying platform for any network service (Lucas, **2008**). Figure 12 illustrates the typical Unix file system structure.

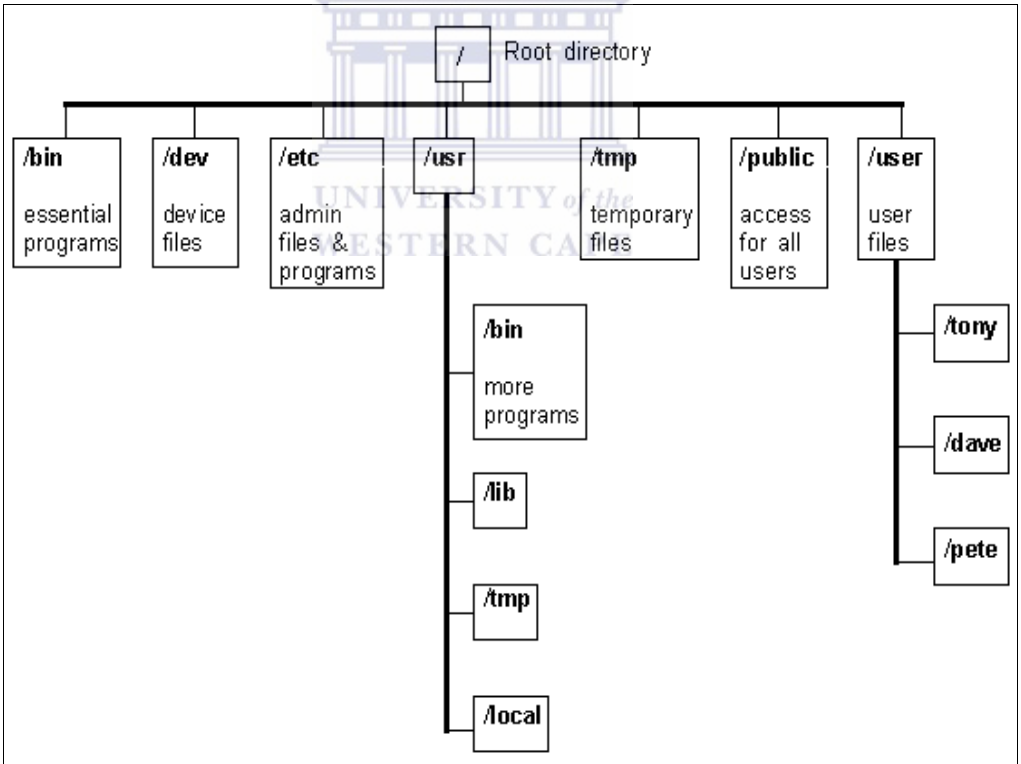

Figure 12: UNIX file system.

The UNIX file system structure (Haden, **2008**).

3.2.2. Practical Extraction and Report Language (PERL)

Perl is a stable, cross platform programming language that is applied in critical projects both in the public and private sectors. The popularity of Perl has been attested in programming of web applications for all needs (The Perl Foundation, **2008**). Designed for handling text-based manipulations, Perl has a wide rage of applications within the bioinformatics field. It has proved extremely successful for linking software applications together into sequence analysis pipelines, file format conversions, and extracting information from the output of analysis programs and other text-based files (Stajich *et al*., **2002**).

3.2.3. BioPerl

Stajich *et al* (**2002**) defines the BioPerl project as "an international open-source collaboration of biologists, bioinformaticians, and computer scientists that has evolved over the past seven years into the most comprehensive library of Perl modules for managing and manipulating life-science information". The entire BioPerl source code is hosted by the Open Bioinformatics Foundation (OBF) so as to enable multiple developers to work in different time zones.

The BioPerl tool kit has been designed with the aim of developers focusing their energy on creating solutions which can be shared and save time that would have been otherwise wasted on duplicated effort (Stajich *et al.,* **2002**). Figure 13 highlights a portion of the BioPerl object model, while Figure 14 is a code snippet illustrating sequence retrieval from a remote database using the Bio::DB::EMBL BioPerl module.

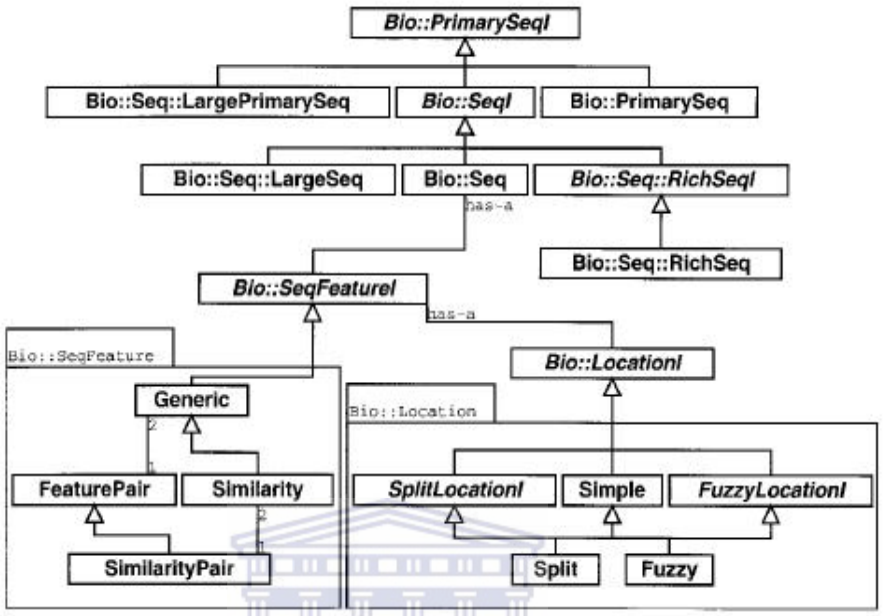

Figure 13: BioPerl object model.

A Portion of the BioPerl object model (Stajich et al., **2002**).


```
use Bio::DB::EMBL;
use Bio::SeqIO;

my $db = new Bio::DB::EMBL();
my $seq = $db->get_Seq_by_acc("U14680");
my $seqout = new Bio::SeqIO( -format => "genbank");
if (defined $seq) {
   $seqout->write_seq($seq);
}
```

Figure 14: Sequence retrieval.

Perl code illustrating sequence retrieval from a remote database with Bio::DB::EMBL. A sequence object is created and then an accession number is passed as a parameter to it's 'get_seq_by_acc' method (Stajich *et al*., **2002**).

3.3. Web Development

3.3.1. Overview

Based on multidimensionality and browsing, the World Wide Web (WWW) offers a powerful communication paradigm in modern times. Coupled with other factors such as the open architectural standards, that enable universal access, integration of different types of information systems have been enabled (Fraternali, **1999**). A web-based application is usually composed of a web client, network servers, and a back-end database system. This array of components enhances its representation as a three-tier architecture. Figure 15 illustrates this assembly.

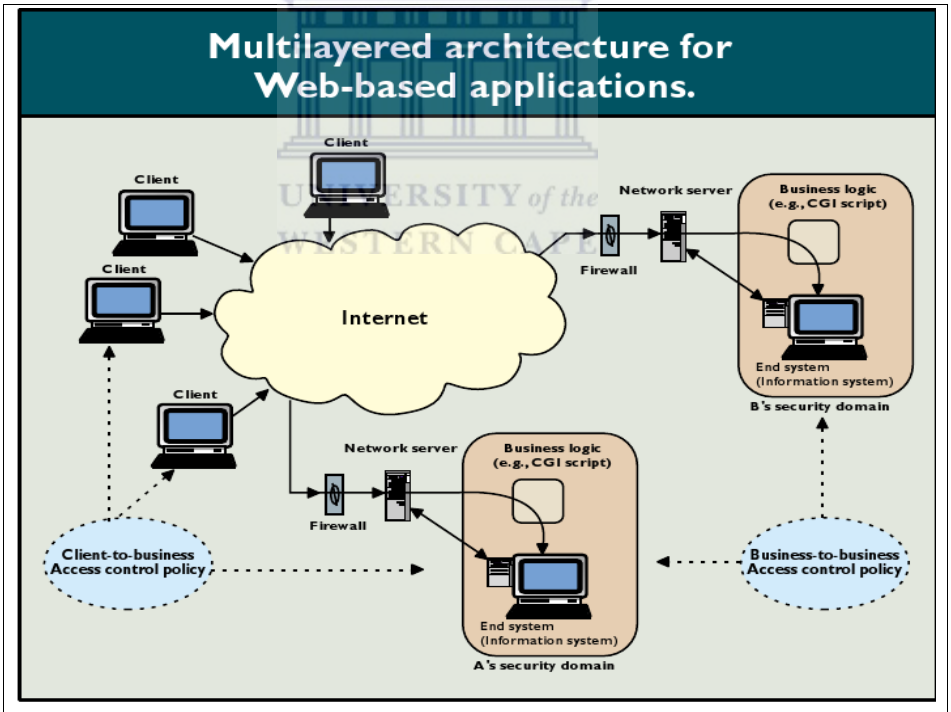

Figure 15: Representation of a web-based application.

Three-tier architectural representation of a web-based application (Joshi et al., **2001**).

In the design of a web application, both the user-oriented and system-oriented technical issues ought to be put into consideration for optimal performance of an application. Some of the technical issues to bear in mind include the necessity of handling both structured data, such as database records, and the non-structured data, such as the multimedia functionalities (Fraternali, **1999**). Other critical issues that need to be addressed include security, scalability, availability and the ease of evolution as well as maintenance. Figure 16 shows the life cycle of a web application.

Due to the fact that the Internet is accessed universally by individuals with limited or no skills in the use of computer applications, effective computer-human interfaces need to be implemented, through dynamic adaptation of content structure, navigation and presentation styles (Fraternali, **1999**).

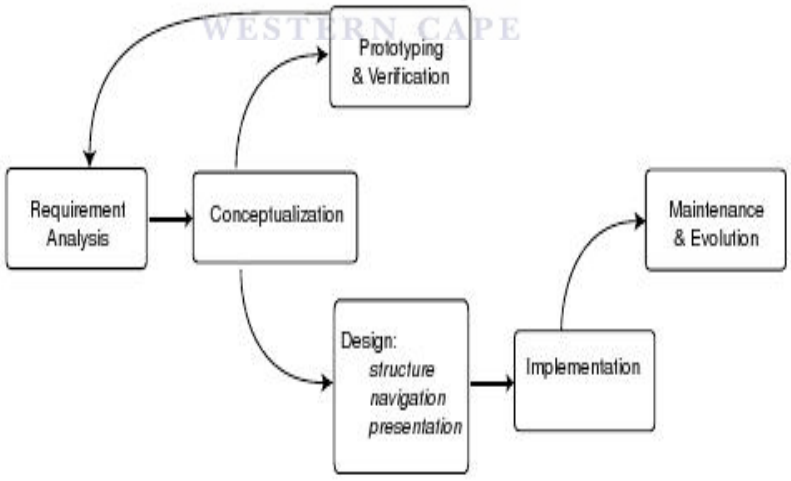

Figure 16: Life cycle of web application.

Graphical illustration of the life cycle of web application (Fraternali, **1999**).

3.3.2. Apache web server

A web server is a program that accepts Hypertext Transfer Protocol (HTTP) requests from web clients and serves back the responses which may be accompanied with optional data contents. A survey by the NetCraft shows that the Apache web server is the most widely used web server driving over 60% of the Internet in the current times. (Nicholes, **2008**). Apache's popularity may be attributed to its open source nature (Open Source, **1998**). in the past five years, the stability, scalability and reliability of Apache has made it surpass the functionality of commercially-available web servers (Nicholes, **2008**). Apache executes the HTTP request life cycle as illustrated in Figure 17.

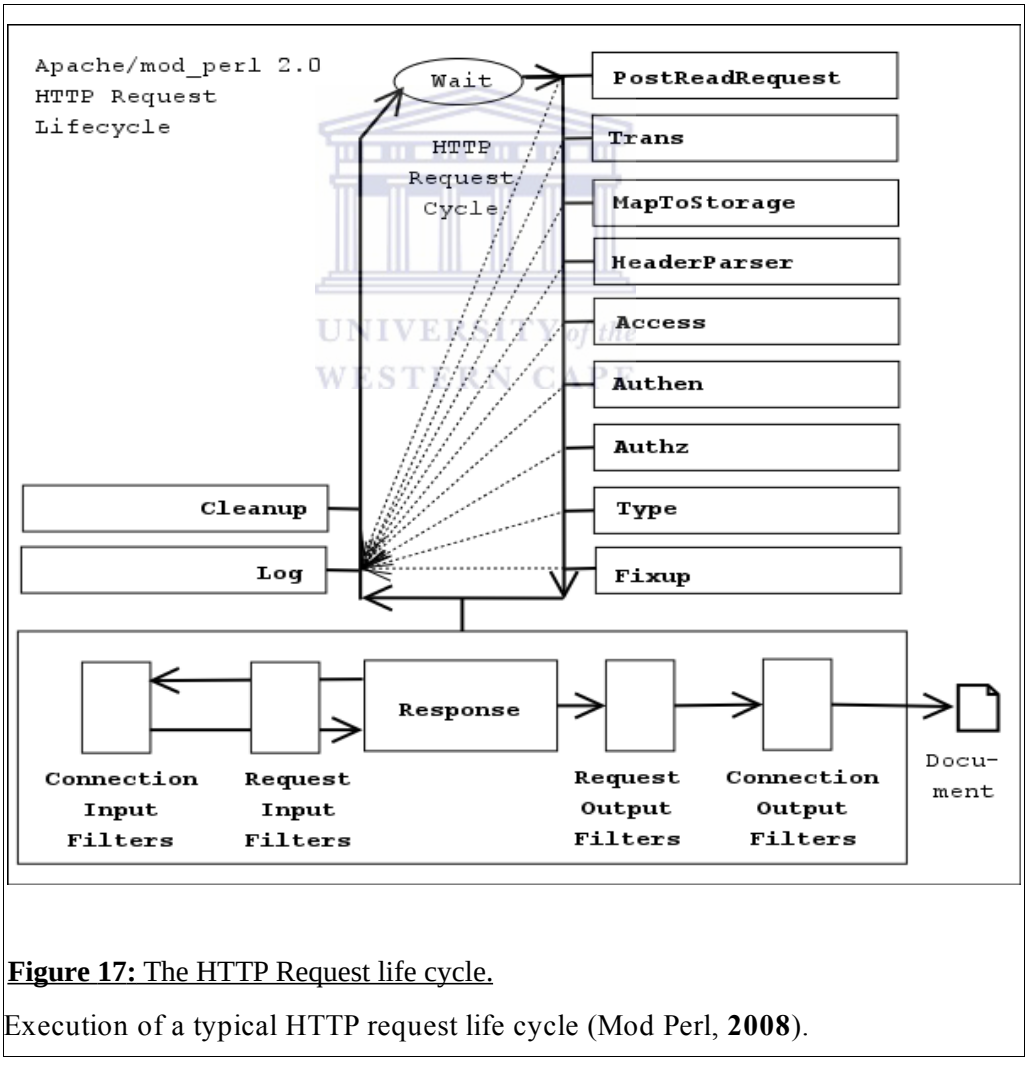

Figure 17: The HTTP Request life cycle.

Execution of a typical HTTP request life cycle (Mod Perl, **2008**).

3.3.3. TWiki

TWiki is a script that is written in Perl and is able to read a text file, hyperlink it and then convert it to Hypertext Mark-up Language (HTML). Built around a Wiki-Web concept, TWiki's components include a web server, Wiki engine, disk space, communication channel as well as its corresponding authors and readers. TWiki is built around a dynamic Intranet model that can be used as a knowledge base in a given academic or technical realm. The source code is available under the GNU general public license (GPL) and the software can be run in a variety of environments with Linux being the common choice (Raygan & Green, **2002**). Appendix VIII highlights all the key TWiki features.

3.3.4. Common Gateway Interface (CGI)

The CGI is a standard protocol that is applied when interfacing an external application software with a web server. As a technology, CGI facilitates the exchange of user-specific data, and the subsequent processed output, between a program and a web browser. (Gundavaram, **1997**). For many websites that apply CGI technology to provide dynamic web pages, Perl is the programming language of choice for a number of reasons which include; simple, but powerful string manipulation, quick prototype design goals, powerful application of regular expressions, a wide array of add-on libraries at the Comprehensive Perl Archive Network (CPAN), abstracted database access, web server integration, adaptable security measures and flexible multi-platform deployment. (Kelly, **2008**). Figure 18 shows how a typical Perl CGI session is mediated.

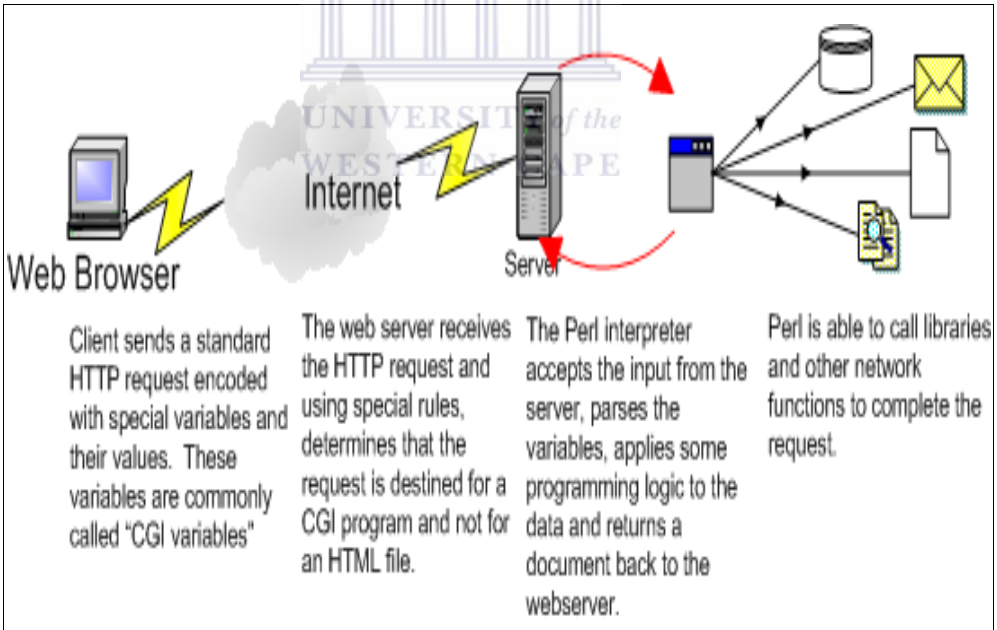

Figure 18: Perl CGI Session.

Graphical illustration of a typical Perl CGI session (Kelly, **2008**).

3.3.5. Cascading Style Sheets (CSS)

A Cascading Style Sheet (CSS) is a series of instructions commonly referred to as statements that identify the elements in a HTML document and then instruct a web browser on how to draw those elements. CSS can be contained in a special *.css* file or can be embedded in the head element of an HTML document (Web Standards Software and Learning, **2008**). As highlighted in Figure 19, the main application of the CSS web technology is in separating content from appearance while formatting web pages. CSS has a stringent syntax which must be adhered to, in order to obtain the required page format. Syntax errors may lead to either a dysfunctional web page display or an unexpected display.

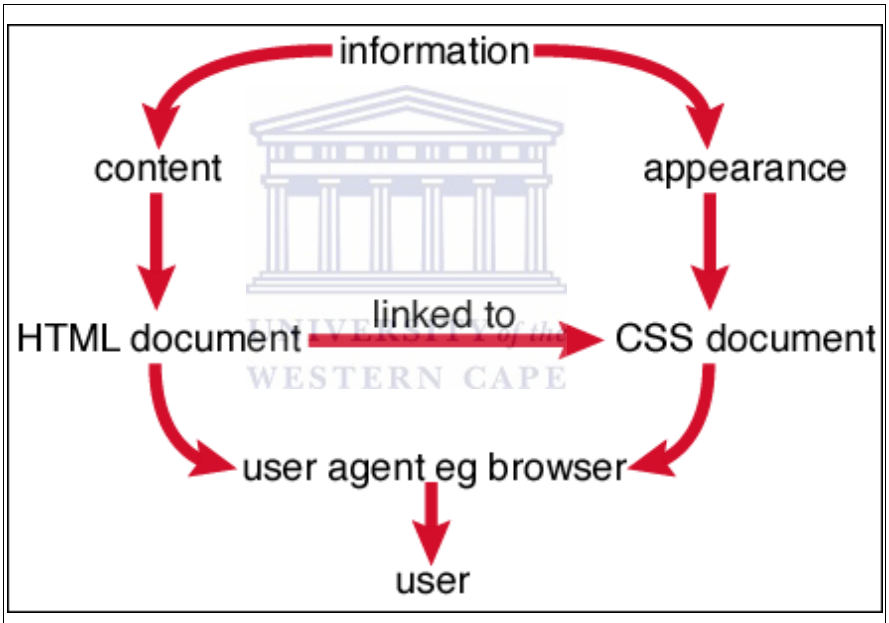

Figure 19: Cascading Style Sheets (CSS) application.

Using Style Sheets to separate content from appearance (Web Standards Software and Learning, **2008**).

Chapter 4: Research Design and Methodology

Each of the software component that constitute the mutation verification platform was assembled as per the guidelines outlined by Fraternali (**1999**), namely, *requirement analysis, conceptualization, prototyping and validation, design and implementation, evolution and maintenance*. Within this research design and methodology section, short code snippets have been used to illustrate a specific design approach, but where lengthy chunks of the source code may be required, appropriate attachments have been availed as appendices.

Most of the methods highlighted in this section involved writing programs using the Perl language. Good coding style was adopted, ensuring that the source code was easy to maintain, all the implementations ran efficiently and hence the resultant programs were robust.

4.1. The mutation server

There was a great need to isolate a UNIX system which would exclusively run all the applications related to the mutation project as well as host its website. The project coordinators would have the root access and hence the ability to install any software they wish, without interfering with other clients on the main system.

It was realized that the analysed requirements could be met by building a jail for the mutation project. This would be a lightweight virtual server, running on FreeBSD, which would be isolated from the rest of the system. This concept was seen as a good compromise between a virtual domain on a shared server and a private dedicated server.

An MSI laptop running Ubuntu 7.10 was used as a prototype to simulate the hosting of the mutation verification platform. This was carried out bearing in mind that there could be logical and practical limitations for this prototype to match the

typical performance of a dedicated server. This was aimed at identifying potential problems with the overall design requirement analysis and hosting activities.

After assessing the functional capability of the prototype, a jail was set up and configured for running this project. The host name identified for it was *mutation* and a unique Internet Protocol (IP) address was also allocated. Other system administration tasks, such as mail and web server set-up, were implemented. Installation and updating of all the needed software which included modules from the Comprehensive Perl Archive Network (CPAN), was done as well.

The very nature of a jail within the UNIX environment makes it easy to evolve in functionality as well as software update and maintenance. This is due to its modular design which enhances porting and has support for multiple servers and a fully customizable user shell.

4.2. Mutation Checker

Mutation Checker is a program that aims at verifying the transcription and translation effects of molecular level sequence variations. The user requirements included the need for a clear web interface and an easy to interpret command-line options. System requirements included the computer hardware, a correctly configured LINUX OS running Perl and an Apache web-server.

The key task was to design Mutation Checker version 3.0, which could be run optimally from the LINUX command-line and equally from across a network connection as a web application. The new version of the Mutation Checker also ensured that all the prime features of version 1.0 and version 2.0 were retained. These concept features include; the ability to process multiple Coding Sequences (CDS) in a genomic sequence, automatic detection of a molecule type from the European Molecular Biology Laboratory (EMBL) reference sequence file, ability to validate mutations against reference data and output mutation information in Extensible Mark-up Language (XML) tag.

A prototype of the Mutation Checker was built and ran on the local server to access input processing, navigation and presentation of the results. Once the prototype satisfied the desired layout and functionality, design and implementation was successfully carried out.

Like the previous version, Mutation Checker Version 3.0 was re-written in a modular fashion that separates mutable object-oriented gene model, mutation event, and storage and printing of mutations. From the BioPerl class library, Bio::LiveSeq and Bio::Variation are the two name spaces, developed by Heikki Lehväslaiho, upon which gene building, mutability and presentation is hinged. Illustration 1 highlights the loading of those name spaces together with other Perl modules.

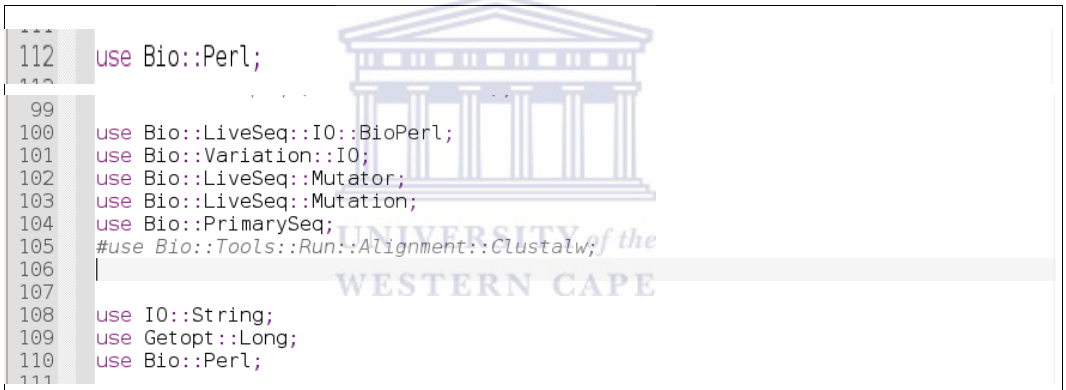

Illustration 1: Loading of name spaces

Perl code snippet illustrating the loading of name spaces from the BioPerl library

During the implementation of a command-line driven Mutation Checker, all the key parameters were encoded as Perl scaler variables (Table 2).

Table 2: Mutation Checker parameters

Perl code representation of the Mutation Checker parameters

Parameter	**Perl Code Representation**
Acc (Accession Number)	$acc_nt
Numbering	$numbering
CDS No.	$cdsnumber
Sequence	$mut_seq
Type	$mutation
Start Number	$ntnumber
Reference Seq	$ntori
Variant Seq	$ntmut
Output format	$format

Johan Vromans' Getopt::Long module (Illustration 1) obtained from the CPAN was used to implement the LINUX command-line options from the Mutation Checker parameters. This enables the programs behaviour modification, depending on the user requirements .Due to the high number of options parameters needed, a hash reference data structure was implemented as the option destination (Illustration 2). The option takes, as value, strings of the key and value form.

```
my $results;
my %options = ();
my $result = GetOptions (\%options,
                         "acc_nt:s",
                         "mut_seq:s",
                         "db:s",
                         "ntnumber:i",
                         "mutation:s",
                         "ntmut:s",
                         "ntori:s",
                         "action:s",
                         "cdsnumber:i",
                         "format:s",
                         "numbering:s");

my $acc_nt = $options{"acc_nt"};
my $mut_seq = $options{"mut_seq"};
my $db = $options{"db"};
my $ntnumber = $options{"ntnumber"};
my $mutation = $options{"mutation"};
my $ntmut = $options{"ntmut"};
my $ntori = $options{"ntori"};
my $cdsnumber= $options{"cdsnumber"};
my $format = $options{"format"};
my $numbering = $options{"numbering"};
my $action = $options{"action"};
```

Illustration 2: Implementation of the command-line options

In order to download sequences automatically from public data-banks, the *get_sequence* BioPerl method was employed (Illustrations 3). A temporary directory is created in the application server where initial sequence downloads are cached for subsequent usage. This optimization techniques ensures efficiency and an increase in the processing speed of the Mutation Checker.

```
562  # make sure the temporary directory exist
563    my $tmpdir = '/tmp/check';
564    unless (-e $tmpdir) {
565        mkdir $tmpdir;
566        chmod (0777, $tmpdir);
567     }
568
569     my @current_files = glob("/tmp/check/*");
570
571    #download the file and save it with the accession number as its name
572
573     my $outputfile = $acc_nt;
574
575     my $seq_object = get_sequence('embl',$acc_nt);
576
577     write_sequence(">/tmp/check/$outputfile", 'embl', $seq_object);
578
579
580    $target_file = "/tmp/check/$outputfile";
581    $loader = Bio::LiveSeq::IO::BioPerl->load(-file =>$target_file);
582
```

Illustration 3: Automatic sequence download

A log file (Illustration 4) was created with the aim of compiling usage statistics. Appendix VI shows sample log file contents.


```
120  our $LOGFILE = "/tmp/check.log";
```

Illustration 4: Creating a log file for usage statistics

The performance of the Mutation Checker version 3.0 is being analysed keenly to gauge any need for maintenance as well as to document its systematic evolution. During the version upgrade, corrective maintenance was performed but the program still remains open for adaptive and perfective maintenance as well.

4.3. Database of Mutation Databases and Related Resources (MutRes)

The need to establish an internet accessible data bank of mutation information was identified during the requirement analysis phase. The data bank would be an up-to-date and complete reference source for details regarding on-line mutation resources. The design concept involved a relational database with Perl object layer and the provision of a web interface to the database. Once the relationships and dependencies between the various entities constituting MutRes had been determined, it was possible to aggregate the data into a logical structure which could then be mapped into tables. To validate the design of MutRes, a prototype consisting of six tables, was developed and ran successfully on An MSI laptop running Ubuntu 7.10.

The six tables in MutRes are; *Resource, Gene, Reference, Link* and *Contact.* Normalization was carried out, hence breaking down the tables to their constituent parts. Each table represents only one piece of information with the columns serving to fully describe only that information.

The Resource Table (Illustration 5) carries the name and the type of the on-line mutation resource. The 'acc' field, whose data-type is an integer, carries a unique identifier for each resource in the database and has been set as the primary key. The default has been set to a zero to ensure that no record could be entered with a blank 'acc' field. The 'name', 'species', 'format', 'platform', 'size', 'funding' and 'status_com' fields have been set with variable character data-types each with a specified byte size. This textual size limit ensures optimal database performance. The 'type' and 'status' fields carry an *enum* data-type which ensures that the user can only fill in the record by selecting from a given list of possible choices. The date fields carry the 'time stamp' data-type which ensures uniformity in the date and time formats within the database.

```
DROP TABLE IF EXISTS `resource`;
CREATE TABLE `resource` (
  `acc` int(7) NOT NULL default '0',
  `name` varchar(255) NOT NULL default '',
  `type` enum('database, gene','database, ethnic','database, spectra','database,
core','database, SNP','p
rogram','website','journal') default NULL,
  `disease` text,
  `species` varchar(64) default NULL,
  `format` varchar(255) default NULL,
  `platform` varchar(255) default NULL,
  `size` varchar(255) default NULL,
  `funding` varchar(255) default NULL,
  `comment` text,
  `last_update` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP,
  `update_com` varchar(100) default NULL,
  `_timestamp` timestamp NOT NULL default '0000-00-00 00:00:00',
  `status` enum('ok','unavailable','inactive','not found','other','unchecked') default NULL,
  `status_com` varchar(100) default NULL,
  PRIMARY KEY  (`acc`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Illustration 5: SQL code snippet highlighting the resource table design

The Gene table (Illustration 6) contains the name, symbol and the synonyms for a list of genes, as per the guidelines and nomenclature laid out by the Human Genome Organization (HUGO).

```
DROP TABLE IF EXISTS `gene`;
CREATE TABLE `gene` (
  `table_id` int(10) unsigned NOT NULL auto_increment,
  `acc` int(7) default NULL,
  `symbol` varchar(255) default NULL,
  `synonyms` varchar(255) default NULL,
  `name` varchar(255) default NULL,
  KEY `resource_acc` (`acc`),
  KEY `internal` (`table_id`),
  KEY `symbol` (`symbol`),
  KEY `synonyms` (`synonyms`),
  CONSTRAINT `gene_ibfk_1` FOREIGN KEY (`acc`) REFERENCES `resource` (`acc`)
ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Illustration 6: SQL code snippet highlighting the gene table design

The 'table_id', which is an integer, is a unique field which identifies each record. The 'acc' field within the gene table has been set as the foreign key and establishes a relationship between the gene table and the resource table.

The curator contact details for each database has been provided in the Contact table (Illustration 7). The 'table_id' field is a unique identifier for each record within the contact table. It has also been set up as the primary key. The 'acc' field within the contact table has been set as the foreign key and establishes a relationship between the contact table and the resource table.

```
DROP TABLE IF EXISTS `contact`;
CREATE TABLE `contact` (
  `table_id` int(10) unsigned NOT NULL auto_increment,
  `acc` int(7) default NULL,
  `name` varchar(64) default NULL,
  `address` varchar(255) default NULL,
  `email` varchar(64) default NULL,
  `phone` varchar(64) default NULL,
  `fax` varchar(64) default NULL,
  PRIMARY KEY  (`table_id`),
  KEY `resource_acc` (`acc`),
  CONSTRAINT `contact_ibfk_1` FOREIGN KEY (`acc`) REFERENCES `resource` (`acc`)
ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```


Illustration 7: SQL code snippet highlighting the contact table design

All the cross-reference details between MutRes and other on-line sequence databases has been provided in the Link table (Illustration 8). The 'acc' field within the link table has been set as the foreign key and establishes a relationship between the link table and the resource table.

```
DROP TABLE IF EXISTS `link`;
CREATE TABLE `link` (
  `table_id` int(10) unsigned NOT NULL auto_increment,
  `acc` int(7) default NULL,
  `symbol` varchar(20) default NULL,
  `db_name` varchar(64) default NULL,
  `db_id` varchar(64) default NULL,
  `refseq` varchar(50) default NULL COMMENT 'EMBL:dna/rna',
  `numbering` enum('coding region','gene','entry') default 'coding region',
  `offset` int(10) default NULL,
  `comment` varchar(64) default NULL,
  KEY `resource_acc` (`acc`),
  KEY `internal` (`table_id`),
  CONSTRAINT `link_ibfk_1` FOREIGN KEY (`acc`) REFERENCES `resource` (`acc`) ON
DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Illustration 8: Code snippet highlighting the link table design

For each mutation resource within MutRes, the Uniform Resource Locator (URL) table (Illustration 9) carries the URL address of that resource as well as its availability status. The 'table_id' field, which is a unique identifier for each record within the URL table, has been set as the primary key.

```
CREATE TABLE `url` (
  `table_id` int(11) NOT NULL auto_increment,
  `acc` int(10) NOT NULL,
  `url` text NOT NULL,
  `url_type` enum('Website','Data','Manual','Other') default NULL,
  `url_status` enum('ok','failed') default NULL,
  `status_date` date default NULL,
  `symbol` varchar(50) default NULL,
  PRIMARY KEY  (`table_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=690 ;
```

Illustration 9: Code snippet highlighting the URL table design

The reference table (Illustration 10) carries the literature supporting any given resource within MutRes, hence its validity. The 'acc' field within the reference table serves as the foreign key and establishes a relationship between the reference table and the resource table.

```
DROP TABLE IF EXISTS `reference`;
CREATE TABLE `reference` (
  `table_id` int(10) unsigned NOT NULL auto_increment,
  `acc` int(7) default NULL,
  `reference` varchar(255) default NULL,
  KEY `resource_acc` (`acc`),
  KEY `internal` (`table_id`),
  CONSTRAINT `reference_ibfk_1` FOREIGN KEY (`acc`) REFERENCES `resource`
(`acc`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Illustration 10: Code snippet highlighting the reference table design

Through the Perl object layer, mechanisms were programmed into the database, that would allow full and boolean key word searches, creation of new entries, facilitate on-line database update and allow text output from database entries. Appendix V shows a sample textual output from the database.

Perl scripts were used to check and weed out duplicates from the database, hence ensuring a stable normalization level as well as eliminating redundant datasets. Appendix I shows a Perl script which was applied for removing duplicates from the gene table. Other Perl script usage was in checking the database for invalid on-line resources (Appendix III) as well as for gathering database statistics (Appendix II). Appendix IV shows a sample output file showing the URL status of the database resources within MutRes.

Appropriate SQL queries were run to collect database statistics. With MutRes having being designed using MySQL, the data manipulation language (DML) component of SQL facilitates data modification and hence database maintenance, while the data definition language (DDL) component would facilitate future database expansion.

4.4. Federated Mutation Database (FMD)

During the requirement analysis, the need to integrate multiple autonomous mutation database systems into a single database was identified. The main concept would involve the establishment of a virtual and fully-integrated database system composed of disparate mutation database systems with a unified front end user-interface. The heterogeneous databases would be exposed to the end-user through a web service abstraction layer.

With the federated database being a virtual entity, prototyping involved the use of Perl wrappers to standardize queries aimed at different potential mutation databases. This is because various database management systems employ different query languages.

Once satisfied with the prototype, MySQL was used to loosely couple several pre-existing mutation databases systems by importing data from this databases and integrating it into one resource, hence forming a federated schema. Illustration 11 shows the use of a MySQL *join query* being applied to compared two records in separate databases based on their accession number (primary key) field.

```
mysql> select u.acc from mutres.url join mutraw.source s on u.table_id = s.mutresid where s.dbid = 75;
```

Illustration 11: Use of MySQL Join Query.

The MySQL Join query was used to link up resources in a Federated Database

The fact that a federated database system is able to decompose a query into sub-queries for submission to the relevant constituent Database Management System adds a degree of flexibility to it. This flexibility enhances patching, maintenance and upgrading of the constituent Database Management Systems.

4.5. Project website

There was need for a publicly accessible, secure and reliable website to host the mutation verification platform. A concept was developed to assemble a dynamic website with a robust structure that would govern the retrieval, analysis and management of sequence variation data. The website had to be secure, scalable and easily accessible. It would also be able to support the running of web applications within it. TWiki, which is a secure program and whose components include a web server, Wiki engine, disk space and a communication channel, was decided upon to run the prototype.

After downloading the TWiki distribution, full installation was done on the mutation server which had been configured earlier. The installation was done with the root (administrator) privileges and file access rights were appropriately set. All the Perl modules which TWiki is dependent on (Table 3) were installed using the CPAN user agent. Appendix VII tabulates optional Perl modules for TWiki. A local library configuration file that controls library resource allocation for TWiki was created using a pre-existing template. The file attributes were retained by running the Linux copy command with the *-p* flag.

The web server configurations, that run TWiki on Apache, were done using the *twiki_httpd_conf.txt* file that is shipped with the TWiki distribution. For security reasons, this configuration script was limited only to the local host which was done using basic Apache authentication. Server security settings were enhanced by turning off any kind of PHP, Perl, Python or Server Side Includes in the *pub* directory. Access was denied to all other TWiki directories, other than the *bin* and

the *pub.*

Customization and interface design was done through modification of the colours, styles and layout Cascading Style Sheets (CSS) code files.

Table 3: Critical Perl modules for running TWiki

MODULE	PREFERRED VERSION
Algorith::Diff (Included)	
CGI::Carp	>=1.26
Config	>=0
Cwd	>=3.05
Data::Dumper	>2.121
Error (Included)	
File::Copy	>=2.06
File::Find	>=1.05
File::Spec	>=3.05
FileHandle	>=2.01
IO::File	>=1.10
Text::Diff (Included)	
Time::Local	>=1.11

The ease of evolution and maintenance was achieved by use of Cascading Style Sheets (CSS) which enables the separation of the web layout from the content.

4.6. Other tools

Two additional tools were incorporated into the mutation verification platform. The Reverse Translator is a web program that calculates the probable DNA point mutations that could have caused a given amino acid substitution. Translation Tables displays all known codon tables.

Chapter 5: Results: Presentation and Discussion

The results obtained from this work have been discussed based on three major design dimensions of a web application, as defined by Offutt (**2002**). The first one is the *structure*, which highlights the organization of the content managed by the application and their semantic relationships. The second one is *navigation*, which involves the facilities for accessing information and browsing through the application content. The last dimension which is equally fundamental is the *presentation*. This dictates the way in which the application content and navigation styles are presented to the end user.

5.1. The mutation server

Mutation is a light weight virtual server which was established as a jail and is hosted on a FreeBSD main server. It runs under the name *mutation* which was branded for all external access and usage. An IP address was also assigned to the server to which it has exclusive access. The web server running within *mutation* is Apache. Users who have been granted access to mutation can log in using the SSH from any Unix/Linux terminal (Illustration 12). A password is required.


```
ssh frederick@mutation.sanbi.ac.za
```

Illustration 12: Login into the mutation server

Authorized users can have root access to *mutation* and are able to install any program they deem useful. File transfer from other remote servers within the South African National Bioinformatics Institute (SANBI) is facilitated through the Secure Shell Protocol (SCP) (Illustration 13).

```
scp file_1 frederick@mutation.sanbi.ac.za:
```

Illustration 13: File transfer

Login into the *mutation* server with the correct password gives the user access to the server and shows the terminal from which they have gained access (Illustration 14).

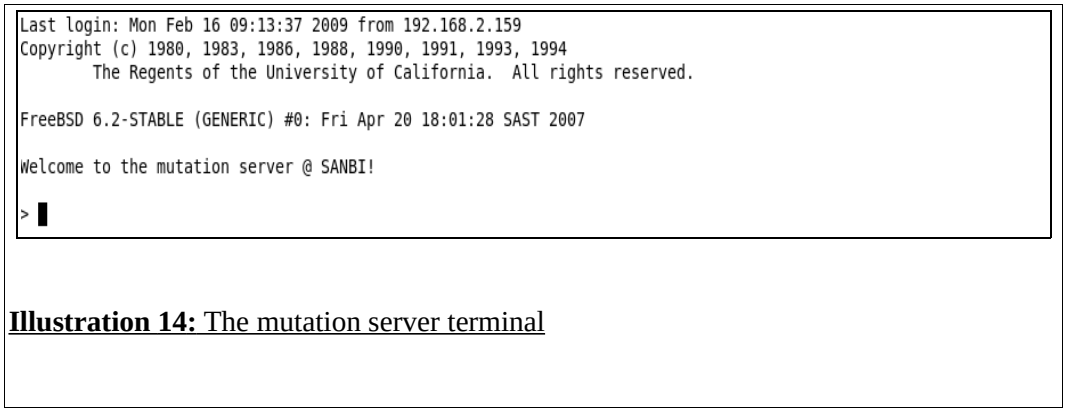

Illustration 14: The mutation server terminal

5.2. Mutation Checker

The Mutation Checker program enables the verification of transcription and translation effects of molecular level sequence variations. Written in Perl, the mutation checker can be run either as a web application or from the LINUX command-line, by typing in all the required parameters as command-line options. The command-line options were implemented using the Get-Opt Perl module which is available from the CPAN. The required fields have been explained within the *Navigation* section. The robustness of the program has been attained through consistent use of subroutines for abstraction. All the constructs within the mutation checker have been verified for scalability and errors arising from incorrect usage are easy to read and understand.

To access the on-line mutation checker from outside SANBI, a user needs Internet connection and any web browser of choice. By typing the URL (Illustration 15) to the sequence variation project, the user is taken to the project home page where there are a number of tools to choose from and mutation checker is the second in the list.

http://mutation.sanbi.ac.za

Illustration 15: Server web access

Figure 20 shows the mutation checker web form. Alternatively, the user could run the program by entering command-line parameters (Illustration 16) from a Linux shell. Table 4 highlights the command-line parameters for the Mutation Checker.

```
> perl checker.pl --acc_nt=AF029980 --ntmut=g --ntnumber=100 --cdsnumber=1 --mutation=point --format=flat
--action=check
```

Illustration 16: Command-line options for the Mutation Checker

Table 4: Command-line options for the Mutation Checker

Command-line Option	Usage
-- acc_nt	A valid EMBL/GenBank/DDBJ accession number to use as a reference sequence.
--ntmut	A nucleotide added into the reference sequence by the mutation.
--action	The action must be set to 'check' in order for the program to run.
--ntnumber	The start number which is the reference sequence location that is first affected by the mutation.
--format	The format of the results which could be either flat or in XML.
--cdsnumber	Number of the CDS for the gene of interest in the entry sequence.
--mutation	Specifies the type of mutation. This could be either an insertion, a deletion, a point mutation or a complex mutation.

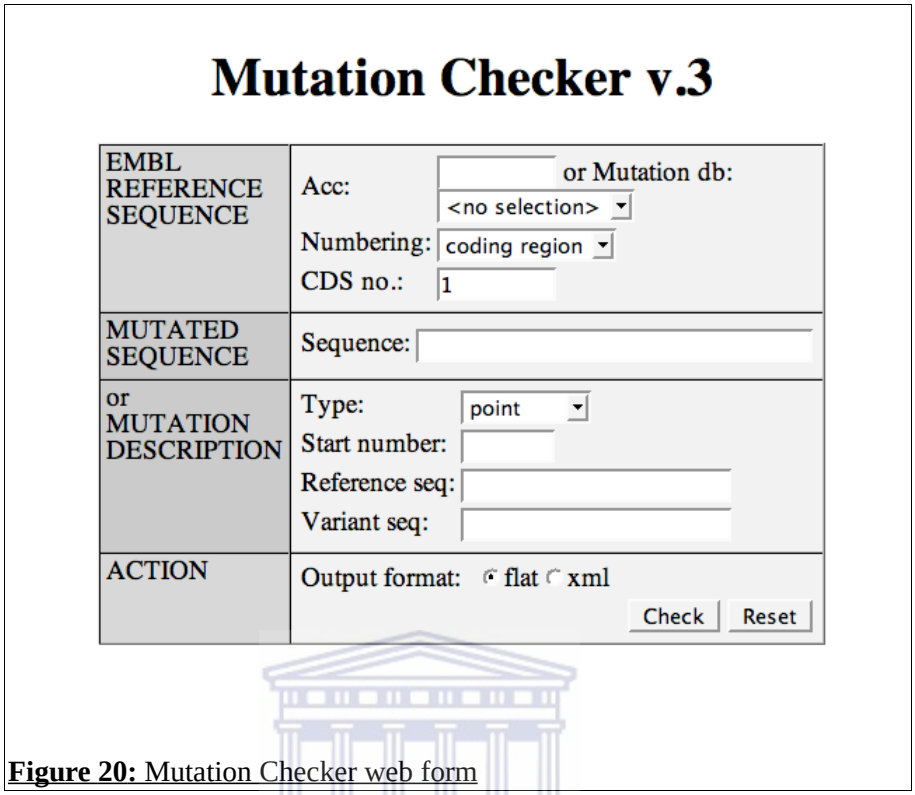

Figure 20: Mutation Checker web form

When ran through a web form, one has to enter a valid EMBL/GenBank accession number into the *Acc* field or select a locus specific mutation database from the *Mutation db* pop-up menu. A numbering schema must then be selected from the *Numbering* pop-up menu which must adhere to the human genetics non-zero convention i.e, any location preceding one (1) is designated as negative one (-1). The order number of the coding sequence for the gene of interest has to be entered in the *CDS no* field. The *type* of mutation is then selected and a *start number* is indicated. Both the *Reference seq* and the *Variant seq* are optional, but the output format for the results has to be chosen. This could be either *flat* or *xml*.

When running the Mutation Checker through the Linux command-line, the same parameters must be passed into the program in the form of command-line options.

The Mutation Checker program enables the verification of transcription and

translation effects of molecular level sequence variations. When ran with all the right parameters, the program simulates the mutation and reports the changed restriction enzyme sites for point mutations, affected mRNA region, affected codon position and calculates the amino acid change. The flanking sequence and restriction enzyme changes for Ribonucleic Acid (RNA) and DNA molecules signifies differences at around the intron/exon region.

Depending on the user specification, the output format for the results could either be in a flat (text) format (Illustration 17) or in XML format (Illustration 18)

```
ID          AF029980:(219)c.+100A>G; K34E
Feature     DNA; 1
Feature       /label: point, transition
Feature       /proof: computed
Feature       /location: 100
Feature       /upflank: gaagaaaaaccatccgttctctgtt
Feature       /change: a>g
Feature       /dnflank: aggtcggattagctcaggtacttcg
Feature       /re_site: +MnlI, -MseI
Feature     RNA; 1
Feature       /label: missense
Feature       /proof: experimental
Feature       /location: 100 (AF029980::319)
Feature       /upflank: gaagaaaaaccatccgttctctgtt
Feature       /change: a>g
Feature       /dnflank: aggtcggattagctcaggtacttcg
Feature       /re_site: +MnlI, -MseI
Feature       /codon_table: 1
Feature       /codon: aag>gag; 1
Feature       /region: coding
Feature     AA; 1
Feature       /label: substitution, conservative
Feature       /proof: computed
Feature       /location: 34
Feature       /change: K>E
//
```

Illustration 17: Mutation Checker flat format results

```
<seqDiff id="AF029980" moltype="rna" offset="219" sysname="c.+100A&gt;G" trivname="K34E">
    <DNA number="1" start="100" end="100" length="1" isMutation="1">
        <label>point</label>
        <label>transition</label>
        <proof>computed</proof>
        <upFlank>gaagaaaaaccatccgttctctgtt</upFlank>
        <allele_ori>a</allele_ori>
        <allele_mut>g</allele_mut>
        <dnFlank>aggtcggattagctcaggtacttcg</dnFlank>
        <restriction_changes>+MnlI, -MseI</restriction_changes>
    </DNA>
    <RNA number="1" start="100" end="100" length="1" isMutation="1">
        <label>missense</label>
        <proof>experimental</proof>
        <upFlank>gaagaaaaaccatccgttctctgtt</upFlank>
        <allele_ori>a</allele_ori>
        <allele_mut>g</allele_mut>
        <dnFlank>aggtcggattagctcaggtacttcg</dnFlank>
        <codon codon_ori="aag" codon_mut="gag" codon_pos="1"></codon>
        <restriction_changes>+MnlI, -MseI</restriction_changes>
        <region>coding</region>
    </RNA>
    <AA number="1" start="34" end="34" length="1" isMutation="1">
        <label>substitution</label>
        <label>conservative</label>
        <proof>computed</proof>
        <allele_ori>K</allele_ori>
        <allele_mut>E</allele_mut>
    </AA>
</seqDiff>
```

Illustration 18: Mutation Checker XML format results

5.3. Database of Mutation Databases and Related Resources (MutRes)

MutRes is a relational database made up of six (6) tables. Figure 22 is an Entity Relationship Diagram (ERD) that shows the MutRes database schema. The diagram was generated using the Dbdesigner version 4.0 software. Relationships between any two or more entities, are represented by a line that joins the entity boxes. Each relationship has two ends, for each of which there is a name.

There are two ways through which users can access the MutRes on-line database. The first one is by accessing the mutation project website at http://mutation.sanbi.ac.za and then choosing MutRes from the left bar menu. Alternatively, one may, straight away, access the database web form at; http://mutation.sanbi.ac.za/mutres/query. The user-friendly search page contains buttons for executing typed search command which are interpreted as database queries. The home page allows the user to search the database, edit database entries, submit mutation data and access links to other databases.

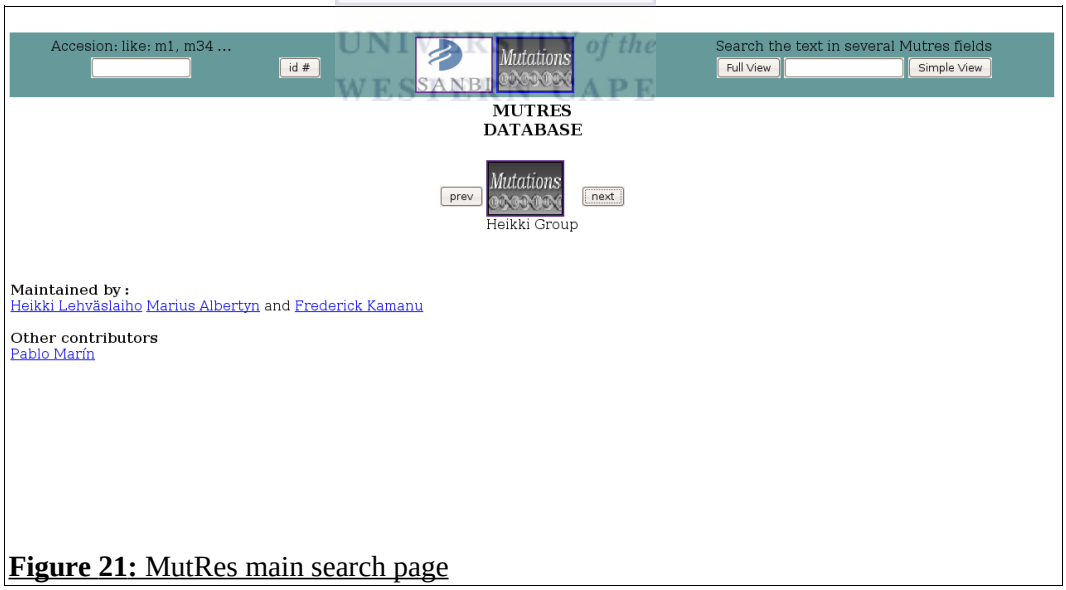

Figure 21: MutRes main search page

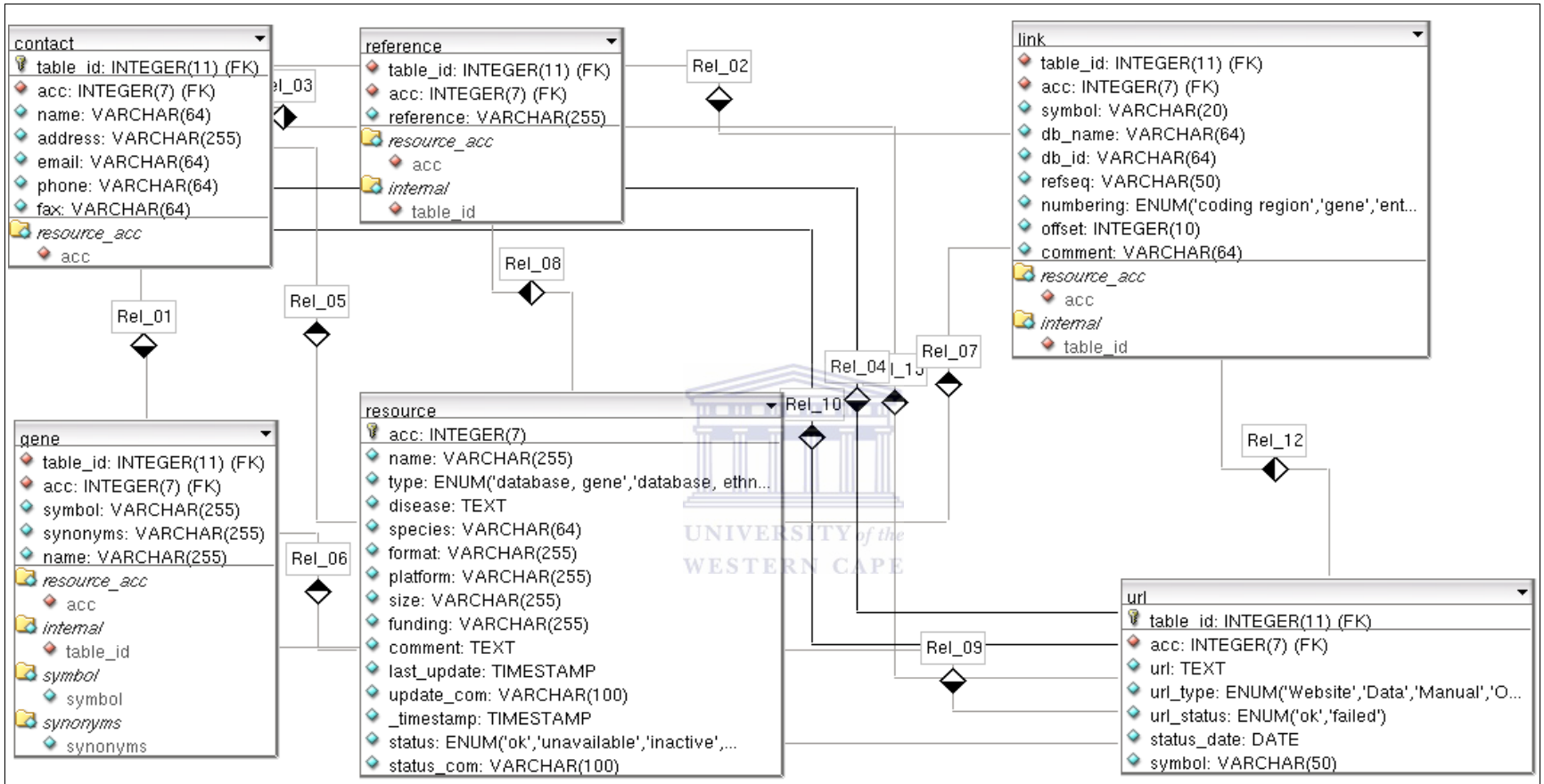

Figure 22: An ERD highlighting MutRes database schema

All pages are presented in a dynamic fashion that enables the user to interact with the database effectively. Figure 21 shows the main search page for the MutRes database web form. Figure 23 highlights sample results displayed from the MutRes database.

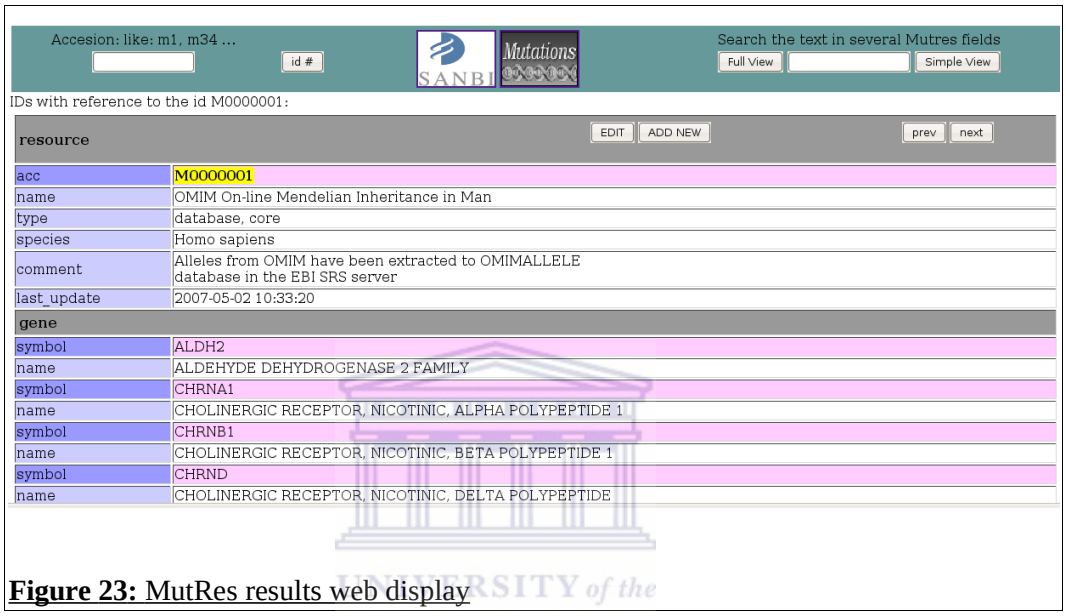

Figure 23: MutRes results web display

The web interface for the MutRes database (Figure 24) allows the database curators to edit existing mutation information entries. This could be as a result of a published correction in the mutation data. The web interface also offers a page (Figure 25) that enables the curators to catalogue new mutation information following a published discovery or reporting.

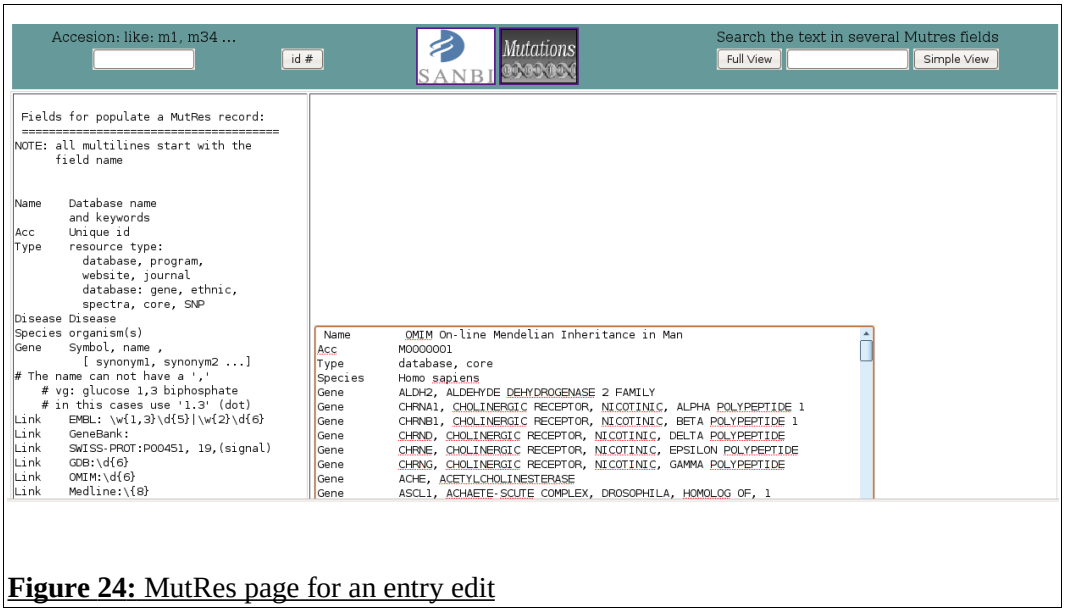

Figure 24: MutRes page for an entry edit

Figure 25: Mutres addition web page

Table 5 shows the number of entries in the URL table categorized by their availability status.

Table 5: URL table information for the MutRes database

ENTITY	COUNT
Functional (available) URLs	456
Non-functional (unavailable) URLs	152
Re-directional URLs	37

Table 6 highlights a comparison in the proportion of HUGO approved genes contained in the LSDBs listings found at the Human Genome Variation Society (HGVS) website and those in the MutRes database.

Table 6: Distribution of Hugo approved genes in MutRes and the LSDB listing

ENTITY	COUNT
HUGO approved genes	28160
HUGO approved genes found LSDBs at the HGVS website	678
HUGO approved genes found in MutRes	256

There are a number of significant differences between MutRes and the mutation resource maintained by the HGVS. Unlike MutRes, this resource does not provide a reference sequence for mutation verification purposes. It also lacks in the provision of adequate mutation information at the codon level, hence making it inconsistent and redundant. On the other hand, MutRes facilitates validation of putative mutations against a reference sequence and enables the creation of standardized mutation representation. Other advantages that MutRes presents to its users include the ability to parse mutation entries from distribution files and the writing out of new data for submission to the the Human Genome Variation database of Genotype-to-Phenotype information (HGVbaseG2P).

Being a collection of on-line mutation databases and resources, the main shortcoming with MutRes is the need for constant updating. This is due to the fact that individual databases constituting MutRes are bound to undergo changes in both structure and content. Fortunately, this potential disadvantage could be overcome through the development of web crawlers aimed at gathering new mutation information from specified web pages. The MutRes database schema caters for development and proper integration of new datasets, hence facilitating expansion.

5.4. Federated Mutation Database (FMD)

FMD is a virtual and fully-integrated database system composed of heterogeneous mutation database systems with a unified front end user-interface. Database access and query execution is done via a user-friendly web form. Table 7 highlights the fraction of mutation databases that have been processed to constitute the Federated Mutation Database.

Table 7: Federate Mutation Database.
Fraction of mutation databases within MutRes that have been processed to constitute the Federated Mutation Database

ENTITY	COUNT
Number of databases in MutRes	331
Number of processed databases	97

A sample listing of the databases that have been already verified, and constitute the Federated Mutation Database has been provided in Figure 26.

ID	Records	Resource Name	Database URL	Status
1	13778	OMIM On-line Mendelian Inheritance in Man	omim.txt.Z	New
2	8	Harvard Medical School Hereditary Deafness database	gja1dream.htm	Verified
3	2	Harvard Medical School Hereditary Deafness database	tmprss3dream.htm	Verified
4	1	Harvard Medical School Hereditary Deafness database	pou4f3dream.htm	Verified
5	10	Harvard Medical School Hereditary Deafness database	tmc1dream.htm	Verified
6	2	Mutations of the Adaptin b3a Gene	adtb3mut.htm	Verified
7	3	Mutations of the Usher Syndrome Type 3 Gene (USH3)	ush3mut.htm	Verified
8	23	Mutations of the Bardet-Biedl Syndrome Type 2 Gene	bbs2mut.htm	Verified
9	1	Mutations of the Neuroretina-linked Leucine Zipper Gene	nrlmut.htm	Verified
10	93	Mutations of the Human Crumbs Homologue 1	crb1mut.htm	Verified
11	30	Mutations of Nyctalopin	nyxmut.htm	Verified
12	5	Mutations of the Harmonin Gene	ush1cmut.htm	Verified
13	1	Mutations of the P-Cadherin Gene	cdh3mut.htm	Verified
14	20	Mutations of the 11-cis Retinol Dehydrogenase Gene	rdhmut.htm	Verified
15	10	Mutations of the Small Nucleotide-binding Protein 27a Gene	rab27mut.htm	Verified
16	3	Mutations of the USH2a Gene	ush2amut.htm	Verified
17	3	Mutations of the Inosine Monophosphate DehydrogenaseType 1 Gene	impdhmut.htm	Verified
18	105	Mutations of the RPE65 Gene	rpe65mut.htm	Verified
19	108	Mutations of the RDS/Peripherin Gene	rdsmut.htm	Verified
20	5	Mutations of the Tissue Inhibitor of Metalloproteases 3 Gene	timpmut.htm	Verified
21	1	Mutations of the HPS3 Gene	hps3mut.htm	Verified
22	659	Mutations of the ATP-binding Cassette Transporter Retina	abcrmut.htm	Verified
23	2	Mutations of HRG4	hrg4mut.htm	Verified
24	4	Mutations of the EGF-containing fibulin-like extracellular matrix protein 1	efempmut.htm	Verified

Figure 26: Sample listing of verified mutation databases

Sample listing of verified mutation databases forming the Federated Mutation Database

5.5. Project website

The project web site has been built upon a TWiki framework. Appendix IX shows the TWiki directory structure and the data types contained in those directories. Navigating around the website is done through pointing and clicking the available hyper-links. TWiki utilizes a Perl script and hence is able to read text files, hyper-link them and convert them to HTML.

Figure 27 presents the project website which is accessible through an internet connection by typing the website URL into a browser. The URL is 'http://mutation.sanbi.ac.za'. By clicking web links on the left bar, one can smoothly navigate through the site.

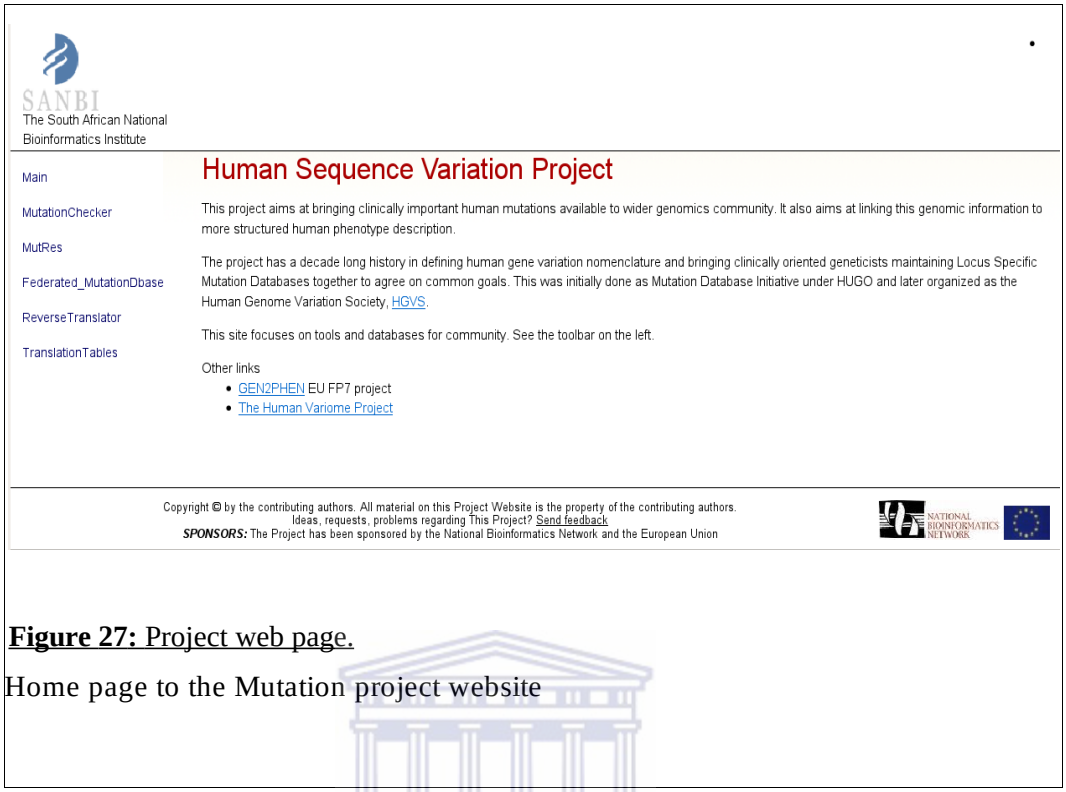

Figure 27: Project web page.

Home page to the Mutation project website

Chapter 6: Conclusion and Recommendations

The mutation verification platform has been established as a comprehensive on-line suite of tools and databases for analysing sequence variation and relating this information to important phenotypes. The need for a continuously updated mutation database will aid in the management, handling and processing of the sequence variation data. Currently, users can download textual mutation profile data from the available databases. Data distribution within the Federated Mutation Database will lead to increased availability and a reduction in access time, and this will translates to reliable mutation data access. In future, graphical rendering tools may be developed which would enable the viewing and downloading of three dimensional mutation representations. Having being released to the broader scientific community, the mutation verification platform has been hailed as a good initiative.

References

Ambler, S.W. (2008), '*Relational Databases 101: Looking at the Whole Picture*'; [Cited July 21, 2008]; Available from: http://www.agiledata.org/essays/relationalDatabases.html, .

Argueello, J.R.; Little, A.M.; Pay, A.L.; Gallardo, D.; Rojas, I.; Marsh, S.G.; Goldman, J.M. & Madrigal, J.A. (1998), 'Mutation detection and typing of polymorphic loci through double-strand conformation analysis.', *Nat Genet* **18**(2), 192--194.

Bodmer, W. & Bonilla, C. (2008), 'Common and rare variants in multifactorial susceptibility to common diseases.', *Nat Genet* **40**(6), 695--701.

Buch, V. (2002), 'Database Architecture: Federated vs. Clustered', *Oracle*.

Bunce, T. (2008), '*DBI - Database Independent Interface for Perl*'; [Cited 22 July, 2008]; Available from: http://search.cpan.org/~timb/DBI/DBI.pm, .

Chappel, M. (2008), '*Database Normalization Basics*'; [Cited July 21, 2008]; Available from: http://databases.about.com/od/specificproducts/a/normalization.htm, .

Collins, F.S.; Brooks, L.D. & Chakravarti, A. (1998), 'A DNA polymorphism discovery resource for research on human genetic variation.', *Genome Res* **8**(12), 1229 1231.

Collins, F.S.; & McKusick V. A. (2008), 'Implications of the Human Genome Project for medical science'.,JAMA

Condit, C.M.; Achter, P.J.; Lauer, I. & Sefcovic, E. (2002), 'The changing meanings of "mutation:" A contextualized study of public discourse.', *Hum Mutat* **19**(1), 69--75.

Eichler, E.E. (2006), 'Widening the spectrum of human genetic variation.', *Nat Genet* **38**(1), 9--11.

Fraternali, P. (1999), 'Tools and Approaches for Developing Data-Intensive Web Applications: A Survey', *ACM Computing Surveys* **Vol.31, No.3**, 227-263.

George, R.A.; Smith, T.D.; Callaghan, S.; Hardman, L.; Pierides, C.; Horaitis, O.; Wouters, M.A. & Cotton, R.G.H. (2008), 'General mutation databases: analysis and review.', *J Med Genet* **45**(2), 65 70.

Gundavaram, S (1996). *CGI Programming on the World Wide Web*. Cambridge: O'Reilly. 1.

Gunderson, K.L.; Huang, X.C.; Morris, M.S.; Lipshutz, R.J.; Lockhart, D.J. & Chee, M.S. (1998), 'Mutation detection by ligation to complete n-mer DNA arrays.', *Genome Res* **8**(11), 1142--1153.

Haden, R. (2008), '*General Unix Information*'; [Cited July 18, 2008]; Available from: http://www.rhyshaden.com/unix.htm, .

Human Genome Variation Society (2008), '*Nomenclature for the description of sequence variants (Mutation nomenclature)*'; [Cited August 04, 2008]; Available from: http://www.genomic.unimelb.edu.au/mdi/mutnomen/recs.html, .

Human Genome Project (2008), '*Human Genome Project*'; [cited June 06, 2008]; Available from: http://www.iscid.org/encyclopedia/Human_Genome_Project, .

Iafrate, A.J.; Feuk, L.; Rivera, M.N.; Listewnik, M.L.; Donahoe, P.K.; Qi, Y.; Scherer, S.W. & Lee, C. (2004), 'Detection of large-scale variation in the human genome.', *Nat Genet* **36**(9), 949--951.

Joshi, J.B.; Aref, W.G.; Ghafoor, A. & Spafford, E.H. (2001), 'Security Models for Web-Based Applications', *Communications of the ACM* **Vol.44, No.2**, 38-44.

Kelley, C. (2008), '*Perl CGI Tutorial Overview*'; [cited July 17, 2008]; Available from: http://inconnu.isu.edu/~ink/perl_cgi/index.html, .

Kidd, J.M.; Cooper, G.M.; Donahue, W.F.; Hayden, H.S.; Sampas, N.; Graves, T.; Hansen, N.; Teague, B.; Alkan, C.; Antonacci, F.; Haugen, E.; Zerr, T.; Yamada, N.A.; Tsang, P.; Newman, T.L.; TÃ¼zÃ¼n, E.; Cheng, Z.; Ebling, H.M.; Tusneem, N.; David, R.; Gillett, W.; Phelps, K.A.; Weaver, M.; Saranga, D.; Brand, A.; Tao, W.; Gustafson, E.; McKernan, K.; Chen, L.; Malig, M.; Smith, J.D.; Korn, J.M.; McCarroll, S.A.; Altshuler, D.A.; Peiffer, D.A.; Dorschner, M.; Stamatoyannopoulos, J.; Schwartz, D.; Nickerson, D.A.; Mullikin, J.C.; Wilson, R.K.; Bruhn, L.; Olson, M.V.; Kaul, R.; Smith, D.R. & Eichler, E.E. (2008), 'Mapping and sequencing of structural variation from eight human genomes.', *Nature* **453**(7191), 56--64.

Lehväslaiho, H. (2000), 'Human sequence variation and mutation databases.', *Brief Bioinform* **1**(2), 161--166.

Lehväslaiho, H.; Ashburner, M. & Etzold, T. (1998), 'Unified access to mutation databases.', *Trends Genet* **14**(5), 205--206.

Lizardi, P.M.; Huang, X.; Zhu, Z.; Ward, P.B.; Thomas, D.C. & Ward, D.C. (1998), 'Mutation detection and single-molecule counting using isothermal rolling-circle amplification.', *Nat Genet* **19**(3), 225--232.

Lucas, M.W. (2008), *Absolute FREEBSD*, No Starch Press.

Lupski, J.R. (2007), 'Structural variation in the human genome.', *N Engl J Med* **356**(11), 1169--1171.

Lupski, J.R. (2006), 'Genome structural variation and sporadic disease traits.', *Nat Genet* **38**(9), 974--976.

modperl (2008), '*mod_perl: HTTP Handlers*'; [cited July 17, 2008]; Available from: http://perl.apache.org/docs/2.0/user/handlers/http.html, .

Moorhouse, M. & Barry, P. (2004), *Bioinformatics Biocomputing and Perl*, John Wiley & Sons, Ltd.

Nature Genetics Editorial (2007), 'What is the human variome project?', *Nat Genet* **39**(4), 423.

Ng, P.C.; Levy, S.; Huang, J.; Stockwell, T.B.; Walenz, B.P.; Li, K.; Axelrod, N.; Busam, D.A.; Strausberg, R.L. & Venter, J.C. (2008), 'Genetic variation in an individual human exome.', *PLoS Genet* **4**(8), e1000160.

Nicholes, B. (2008), '*How to Use NDS eDirectory to Secure Apache Web Server for Netware*'; [Cited July 17, 2008]; Available from: http://support.novell.com/techcenter/articles/ana20010202.html, .

Offutt, J. (2002), 'Quality Attributes of Web Software Applications', *IEE Sofware* March/April, 25-32.

Okou, D.T.; Steinberg, K.M.; Middle, C.; Cutler, D.J.; Albert, T.J. & Zwick, M.E. (2007), 'Microarray-based genomic selection for high-throughput resequencing.',

Nat Methods **4**(11), 907--909.

Open Source Initiative (2008), '*About the Open Source Initiative*'; [cited June 04,2008]; Available from: http://www.opensource.org/about, .

Patrinos, G.P. & Brookes, A.J. (2005), 'DNA, diseases and databases: disastrously deficient.', *Trends Genet* **21**(6), 333--338.

Raygan, R.E. & Green, D.G. (2002),'Internet Collaboration: TWiki', Technical report, UAB; Birmingham, Alabama.

Scriver, C.R.; Nowacki, P.M. & Lehväslaiho, H. (1999), 'Guidelines and recommendations for content, structure, and deployment of mutation databases.', *Hum Mutat* **13**(5), 344--350.

Sherry, S.T.; Ward, M.H.; Kholodov, M.; Baker, J.; Phan, L.; Smigielski, E.M. & Sirotkin, K. (2001), 'dbSNP: the NCBI database of genetic variation.', *Nucleic Acids Res* **29**(1), 308--311.

Stajich, J.E.; Block, D.; Boulez, K.; Brenner, S.E.; Chervitz, S.A.; Dagdigian, C.; Fuellen, G.; Gilbert, J.G.R.; Korf, I.; Lapp, H.; LehvÃ¤slaiho, H.; Matsalla, C.; Mungall, C.J.; Osborne, B.I.; Pocock, M.R.; Schattner, P.; Senger, M.; Stein, L.D.; Stupka, E.; Wilkinson, M.D. & Birney, E. (2002), 'The Bioperl toolkit: Perl modules for the life sciences.', *Genome Res* **12**(10), 1611--1618.

Stenson, P.D.; Ball, E.V.; Mort, M.; Phillips, A.D.; Shiel, J.A.; Thomas, N.S.T.; Abeysinghe, S.; Krawczak, M. & Cooper, D.N. (2003), 'Human Gene Mutation Database (HGMD): 2003 update.', *Hum Mutat* **21**(6), 577--581.

Stout, G.A. (2001),'Deploying a Website: Best Practices', Technical report, The Revere Group.

Sunyaev, S.; Ramensky, V.; Koch, I.; Lathe, W.; Kondrashov, A.S. & Bork, P. (2001), 'Prediction of deleterious human alleles.', *Hum Mol Genet* **10**(6), 591--597.

The Perl Foundation (2008), '*The Perl Directory*'; [Cited July 18, 2008]; Available from: http://www.perl.org/, .

Tito, B.J.D.; Poff, H.E.; Novotny, M.A.; Cartledge, D.M.; Walker, R.I.; Earl, C.D. & Bailey, A.L. (1998), 'Automated fluorescent analysis procedure for enzymatic mutation detection.', *Clin Chem* **44**(4), 731--739.

Wagner, R.; Debbie, P. & Radman, M. (1995), 'Mutation detection using immobilized mismatch binding protein (MutS).', *Nucleic Acids Res* **23**(19), 3944--3948.

Web Standards Software and Learning (2008), *'Complete CSS Guide*'; [Cited July 17, 2008]; Available from: http://www.westciv.com/style_master/academy/css_tutorial/introduction/how_they_work.html, .

Wheeler, D.A.; Srinivasan, M.; Egholm, M.; Shen, Y.; Chen, L.; McGuire, A.; He, W.; Chen, Y.; Makhijani, V.; Roth, G.T.; Gomes, X.; Tartaro, K.; Niazi, F.; Turcotte, C.L.; Irzyk, G.P.; Lupski, J.R.; Chinault, C.; zhi Song, X.; Liu, Y.; Yuan, Y.; Nazareth, L.; Qin, X.; Muzny, D.M.; Margulies, M.; Weinstock, G.M.; Gibbs, R.A. & Rothberg, J.M. (2008), 'The complete genome of an individual by massively parallel DNA sequencing.', *Nature* **452**(7189), 872--876.

Appendices

Appendix I: Multi-line remover script

A script for removing duplicate entries in the gene table of MutRes database

```
#! /usr/bin/perl -w

use strict;
use warnings;

use DBI;

my ($dbh, $sth_1, $sth_2);

#my $host = "mosh.sanbi.ac.za";
my $db = "DBI:mysql:database=mutres;host=mosh.sanbi.ac.za";
my $user = "mutres";
my $password = "mut";

main();

#=======================================================
sub main
{
   print "Getting data...\n";
   GetFileData();
   print "...processing data...\n";
   ProcessFileData();
   print "Done!\n";
}

#=======================================================
sub GetFileData
{
   my $dbh = DBI->connect($db, $user, $password, { RaiseError => 1 });

   open (OUTFILE, ">formated_names.txt") || die "Cannot open product file: $!";
   my $sth_1 = $dbh->prepare("select * from mutres.gene where table_id = 5484");
   $sth_1->execute();
      while (my($table_id, $acc, $symbol, $synonymes, $name) = $sth_1->fetchrow)
      {
            $name =~ s/\n/ /g;
            $name =~ s/;;/,/g;
            $name =~ s/;/,/g;
            print OUTFILE $table_id, ";", $name, "\n";
      }
   $sth_1->finish();
   close(OUTFILE);

   $dbh->disconnect();
}
```

```
#=============================================================
sub ProcessFileData
{
   my $rv;
   my $dbh = DBI->connect($db, $user, $password, { RaiseError => 1, AutoCommit =>
0 });

   open(INFILE, "formated_names.txt" ) || die "Cannot open product file: $!";
   while ( <INFILE> )
   {
      chomp;

      my ($table_id, $new_name) = split /;/ ;

      $new_name =~ s/XXX//g ;
      my $sql = "update mutres.gene g set g.name = '" . $new_name . "' where
g.table_id = " . $table_id;

         $dbh->prepare($sql);

print "b4 - $sql\n"; getc;

      eval { $rv = $dbh->do($sql); }; warn "ERR: $@" if $@;

print "RV = $rv\n";

print "af"; getc ;

   }
   close(INFILE);

   $dbh->disconnect();
}
```

Appendix II: MutRes statistics gathering script

```
package mutres;

use strict;
use DBI;
use Data::Dumper;

#=================================================================
=
sub new
{
	my $class = shift;
	my $self = bless {}, $class;
	$self -> _Connect (@_);
	return $self;
}


#=================================================================
=
sub _Connect
{
	my ($self) = @_;
	my $db;

	$self->{DATASETNAME} = "DBI:mysql:database=confirm;host=localhost";
	$self->{USER} = "fred";
	$self->{PASSWORD} = "stillfred";

	eval {$db = DBI->connect($self->{DATASETNAME}, $self->{USER}, $self-
>{PASSWORD},
			{PrintError => 0, RaiseError => 1}); };
	if ($@) { warn $@; return 0; }

	return $db;

}

#=================================================================
sub CountTableEntries
{
	my ($self, $table) = @_;
	my ($rv, $sth, $sql);
	my $db = $self->_Connect();

	$sql = 'SELECT COUNT(*) FROM '. $table;
```

```
        eval
        {
            $sth = $db->prepare($sql);

            $sth->execute();

            $rv = $sth->fetchrow_array();

            $sth->finish();
        };

        if ($@) { warn "ERROR: " . $@; return; }

        return $rv;
}

#============================================================
sub CountGeneEntries
{
        my ($self, $gene) = @_;
        my ($rv, $sth, $sql);
        my $db = $self->_Connect();

        $sql = 'SELECT symbol FROM gene where symbol = ?';

        eval
        {
            $sth = $db->prepare($sql);

            $sth->execute($gene);

            $rv = $sth->fetchrow_array();

            $sth->finish();
        };

        if ($@) { warn "ERROR: " . $@; return; }

        return $rv;
}


1;
```

Appendix III: MutRes database validity script

A sample script for checking the validity of URL entries in MutRes

```
#!/usr/bin/perl -w
use strict;
use DBI;
use LWP::UserAgent;
use LWP::Simple;

my $dsn = "DBI:mysql:database=mutres;host=mosh.sanbi.ac.za";
my $du = "mutres";
my $dp = "mut";
my $db;

#=============================================================
sub OpenDB
{
   my ($datasetname, $duser, $dpassword) = @_;
   #my $db;
   eval { $db = DBI->connect($datasetname, $duser, $dpassword, {RaiseError => 1}); };
   if ($@) { warn $@; return 0; }
   bless (\$db,"DBI::mysql");
   return $db;
}

#=============================================================
sub CloseDB
{
   my $db = shift;
   eval { $db->disconnect(); }; warn $@ if $@;
}


#if database opens successfully
if ($db = OpenDB($dsn, $du, $dp)) {
        #prepare the SQL
        my $sth = $db->prepare('SELECT resource.acc,resource.name,url.url FROM
        resource LEFT JOIN url ON resource.acc=url.acc') or die "Can't prepare SQL: "
         . $db->errstr();

        #execute the SQL
        $sth->execute() or die "Can't execute SQL: " . $sth->errstr();

        #loop through each url in the row, check its status and print it out in a file
        my($acc, $dbase_name, $url_string);
        open (OUTPUT, ">>dbase_status_report") ||die "can't open Output file";
        while(($acc, $dbase_name, $url_string) = $sth->fetchrow()) {
```

```
        if (head($url_string)) {
                print OUTPUT "VALID: [$acc] $dbase_name\t$url_string\n";
                }
        else {
                print OUTPUT "INVALID:[$acc] $dbase_name\t$url_string\n";
                }
        }
}

close (OUTPUT);
CloseDB($db);
```

Appendix IV: Sample status output file

Sample output file showing the database URL status

```
VALID: [1] OMIM On-line Mendelian Inheritance in Man
http://www3.ncbi.nlm.nih.gov/omim/
INVALID:[1] OMIM On-line Mendelian Inheritance in Man
http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=OMIM
VALID: [2] GDB Genome DataBase        http://www.gdb.org/
INVALID:[2] GDB Genome DataBase       http://gdb.infobiogen.fr
INVALID:[2] GDB Genome DataBase       http://gdb.weizmann.ac.il
INVALID:[2] GDB Genome DataBase       http://www-gdb.caos.kun.nl/gdb/
VALID: [2] GDB Genome DataBase        http://www.hgmp.mrc.ac.uk/gdb/
VALID: [3] Protein mutant DB (PMD)    http://pmd.ddbj.nig.ac.jp
VALID: [3] Protein mutant DB (PMD)    ftp://spock.genes.nig.ac.jp/pub/pmd
VALID: [3] Protein mutant DB (PMD)    http://pmd.ddbj.nig.ac.jp/~pmd/pmdkey.html
VALID: [3] Protein mutant DB (PMD)    http://pmd.ddbj.nig.ac.jp/~pmd/pmdseqblt.html
VALID: [4] The IARC p53 Mutations Database    http://www-p53.iarc.fr/index.html
INVALID:[4] The IARC p53 Mutations Database   http://www.iarc.fr/p53/Germline.html
INVALID:[4] The IARC p53 Mutations Database
http://www.iarc.fr/p53/Polymorphism.html#description
INVALID:[4] The IARC p53 Mutations Database
ftp://ftp.ebi.ac.uk/pub/databases/p53/datar4.txt
VALID: [4] The IARC p53 Mutations Database    http://srs.ebi.ac.uk/srs7bin/cgi-
bin/wgetz?-page+LibInfo+-id+1mTim1LkkUq+-lib+P53
VALID: [5] p53/APC      ftp://ftp.ebi.ac.uk/pub/databases/p53APC/MUTATION.TXT
INVALID:[5] p53/APC     http://www.mayo.edu/papers/P53%20Mutations
INVALID:[5] p53/APC     http://www.mayo.edu/papers/P53%20Mutations/readme.txt
VALID: [6] BTKbase      http://www.uta.fi/laitokset/imt/bioinfo/BTKbase/index.html
VALID: [6] BTKbase      http://bioinf.uta.fi/BTKbase/btkpub.html
VALID: [6] BTKbase      ftp://protein.uta.fi/pub/btkpub.dat
VALID: [6] BTKbase      http://srs.ebi.ac.uk/srs7bin/cgi-bin/wgetz?-page+LibInfo+-id
+2ke4_1KrZMK+-lib+BTKBASE
VALID: [7] von Willebrand Factor Database     http://www.shef.ac.uk/vwf/
VALID: [7] von Willebrand Factor Database     http://srs.ebi.ac.uk/srs7bin/cgi-
bin/wgetz?-page+LibInfo+-id+2ke4_1KrZMK+-lib+VWF
VALID: [7] von Willebrand Factor Database
http://www.shef.ac.uk/vwf/downloads/vwfmutations.xls
```

Appendix V: Sample text output from MutRes database

Sample text output from the MutRes database

```
//
Name        Neuronal Ceroid Lipofuscinoses (NCL) Mutations
Acc         M0000128
Type        database, gene
Species     Human
Disease     Neuronal Ceroid Lipofuscinoses, NCL
Gene        PPT
Link        OMIM: 256731
Link        OMIM: 256730
Gene        CLN1
Gene        CLN2
Gene        CLN3, ceroid-lipofuscinosis, neuronal 3, juvenile (Batten, Spielmeyer- , [NULL]
Gene        CLN5, ceroid-lipofuscinosis, neuronal 5 , [NULL]
Contact     Sara Mole
Address     Department of Paediatrics and Child Health, Royal Free and University
College Medical School, University College London, 5 University Street, London. WC1E
6JJ. United Kingdom.
Email       S.Mole@ucl.ac.uk
Format      HTML tables data
URL         http://www.ucl.ac.uk/ncl/
URL_type    Website
URL         http://srs.ebi.ac.uk/srs7bin/cgi-bin/wgetz?-page+LibInfo+-id+4kuF81Krb_B+-lib
+NCL
URL_type    Manual
Last_update 2004-05-18 00:00:00
//
```

Appendix VI: Mutation server sample output

Sample server log output

```
 from 192.168.2.119 on 2008-02-01 10:15:58
 from 192.168.2.119 on 2008-02-01 10:19:55
 from 192.168.2.119 on 2008-02-01 12:42:48
 from 192.168.2.140 on 2008-02-04 12:04:18
 from localhost on 2008-02-07 10:24:16
 from 192.168.2.121 on 2008-02-07 10:35:27
J02933 from 192.168.2.121 on 2008-02-07 10:35:57
 from 192.168.2.121 on 2008-02-07 10:36:34
J02933 from 192.168.2.121 on 2008-02-07 10:37:06
J02933 from 192.168.2.121 on 2008-02-07 10:37:21
 from 192.168.2.121 on 2008-02-07 10:39:25
J02933 from 192.168.2.121 on 2008-02-07 10:41:26
 from 192.168.2.119 on 2008-02-07 14:14:45
X58957 from 192.168.2.119 on 2008-02-07 14:15:39
X58957 from 192.168.2.119 on 2008-02-07 14:17:34
 from 193.167.195.60 on 2008-02-12 10:07:13
O60500 from 193.167.195.60 on 2008-02-12 10:12:17
 from 193.167.195.60 on 2008-02-12 10:12:45
 from 193.167.195.60 on 2008-02-12 10:12:54
 from 193.167.195.60 on 2008-02-12 10:12:58
 from 193.167.195.60 on 2008-02-12 10:13:05
AAG17141 from 193.167.195.60 on 2008-02-12 10:30:01
AAG17141 from 193.167.195.60 on 2008-02-12 10:30:09
 from 193.167.195.60 on 2008-02-12 10:30:15
 from 193.167.195.60 on 2008-02-12 10:30:18
 from 192.168.2.105 on 2008-02-18 08:23:13
 from 192.168.2.226 on 2008-03-18 11:52:29
 from 192.168.2.226 on 2008-03-18 11:53:16
J02933 from 192.168.2.226 on 2008-03-18 11:53:44
 from 192.168.2.226 on 2008-03-18 11:54:10
NM_000310 from 192.168.2.226 on 2008-03-18 11:54:37
 from 192.168.2.226 on 2008-03-18 12:49:17
 from 192.168.2.226 on 2008-03-18 12:51:46
 from 192.168.2.226 on 2008-03-18 12:51:49
 from 192.168.2.226 on 2008-03-18 12:51:53
```

Appendix VII : Table of optional Perl modules required by TWiki

Optional Perl modules required by TWiki

Module	Preferred version	Description
Archive::Tar		May be required by the Extensions Installer in configure if command line tar or unzip is not available
CGI::Cookie	>=1.24	Used for session support
CGI::Session	>=3.95	Highly recommended! Used for session support
Digest::base		
Digest::SHA1		
Jcode		Used for I18N support with perl 5.6
Locale::Maketext::Lexicon	>=0	Used for I18N support
Net::SMTP	>=2.29	Used for sending mail
Unicode::Map		Used for I18N support with perl 5.6
Unicode::Map8		Used for I18N support with perl 5.6
Unicode::MapUTF8		Used for I18N support with perl 5.6
Unicode::String		Used for I18N support with perl 5.6
URI		Used for configure

Appendix VIII : Table summarizing TWiki features

FEATURE	COMMENT
Any web browser	Edit existing pages or create new pages by using any web browser. There is no need for ftp or http put to upload pages.
Edit link	To edit a page, simply click on the Edit link at the bottom of every page.
Auto links	Web pages are linked automatically. You do not need to learn HTML commands to link pages.
Text formatting	Simple, powerful and easy to learn text formatting rules. Basically you write text like you would write an email.
Webs	Pages are grouped into TWiki webs (or collections). This allows you to set up separate collaboration groups.
Search	Full text search with/without regular expressions. See a sample search result.
Email notification	Get automatically notified when something has changed in a TWiki web. Subscribe in WebNotify.
Structured content	Use TWiki Forms to classify and categorize unstructured web pages and to create simple work-flow systems.
File attachments	Upload and download any file as an attachment to a page by using your browser. This is similar to file attachments in email, but it happens on web pages.
Revision control	All changes to pages and attachments are tracked. Retrieve previous page revisions and differences thereof. Find out who changed what and when.
Access control	Define groups and impose fine-grained read and write access restrictions based on groups and users.
Variables	allows you for example to dynamically build a table of contents, include other pages; or show a search result embedded in a page.
TWiki Plugins	Easily install program enhancements using external plug-in modules. Developers can create plug-ins in Perl, with the TWiki Plugin API.
Templates and skins	A flexible templating system separates program logic and presentation. Skins overwrite template headers and footers; page content is unaffected.

FEATURE	COMMENT
Managing pages	Individual pages can be renamed, moved and deleted through the browser.
Managing users	Web based user registration and change of password.
Statistics	Create Statistics of TWiki webs. Find out most popular pages and top contributors.
Preference	Three levels of preferences - TWikiPreferences for site-level; WebPreferences for each web; and user level preferences.
Topic locking	Users are warned if another person is editing a page. This is to prevent contention, e.g. simultaneous page editing.
Referred-By	Find out back-links to a page.

Appendix IX : TWiki directory structure

TWIKI DIRECTORY	CONTENTS
twiki	Start-up pages
twiki/bin	CGI bin
twiki/lib	Library files
twiki/locale	Language files
twiki/pub	Public files
twiki/data	Topic data
twiki/templates	Web templates
twiki/tools	TWiki utilities