The EngD Second Year Report

Ben Tagger

October 18, 2006

Preliminary EngD Title: A Framework for Managing Changes in Biological Data

Starting Date: 27th September 2004

1st Supervisor: Anthony Finkelstein

2nd Supervisor: Chris Clack

Industrial Supervisor: Delmiro Fernandez-Reyes

Assessor: Anthony Steed

Contents

List of Figures

The structure of the following report is different to the criteria set out for the second year report. We have included the whole literature review (rather than a list of the reviewed topics) in order to provide a more comprehensive overview of the research to date. We will now describe the structure of the report with respect to the official criteria so that those who wish to only review that material can do so.

Section one contains the problem statement. The entire literature review is contained in section two but for those wishing to see a list of reviewed material can refer to the table of contents on the previous pages. Section three provides conclusions from the literature review and sections four and five describe the proposed contribution and the scope of the thesis. Section six describes the research that has been carried out so far and the work that is going to be completed in order to fulfil the contribution. Finally, we provide a brief discussion of the validation of the research and a timetable of the remaining activities.

1 Problem Statement

Some years ago, as we entered the genomic era of biology, there occurred an explosion of publicly available biological data. The three principal stakeholders[1] in the collection, storage and curation of genomic data have since collected and disseminated over 100 gigabases of sequence data.

There arise many problems when attempting to deal with this magnitude of data; some have received more attention than others. One significant problem is that data within these publicly available databases are subject to change in an unannounced and unpredictable manner. Consider an e-scientist conducting experiments with such data. In the event that the data used for experimentation has changed, it may be important that the e-scientist be made aware of this change in order to establish the impact on their previously attained results. Couple this change with the experimentation, protocol and other changes that can occur, it is clear that there exists a complex environment of possible change that can affect the scientist's results.

The problem is how to manage this state of experimental change in a way that can benefit the experimenter. Some results may be more important than others, may require more effort to repeat or may be less affected by certain changes than other sets of results. Managing this environment of experimental change is a complex issue and, to date, has not been fully reconciled.

In some other areas of research, this problem has been addressed with the use of versioning. Versioning helps the user track changes to documents or other files and allows

[1]The three members of the INSDC (International Nucleotide Sequence Database Collaboration) are; **NCBI:** GenBank - maintained by the NCBI (National Center for Biotechnology Information); **EMBL-EBI:** European Molecular Biology Laboratory's European Bioinformatics Institute; **DDBJ:** DNA Data Bank of Japan.

the user to investigate previous iterations of work as well as the ability to roll back to them. This traditional view of versioning has served its purpose well for documents or other types of textual file tracking. However, when we consider these approaches for the application of biological experimentation, the techniques are inappropriate and unsuitable[2].

This unsuitability is not surprising. Traditional versioning techniques were devised to track text-based files and they have been successful and widespread over the last two decades. In order to address the problem of changes within a biological experimentation context, we must first identify the requirements for such a system so that we can establish the ways in which the genres differ.

This research aims to contribute in three distinct ways. Firstly, we aim to conduct an analysis on a biological experimentation problem (a test case) in order to identify the changes that can occur within such a system and track the impacts that occur as a result of these changes. Secondly, we will build a framework to aid e-scientists in their management of the changes that have been identified as occurring within the environment. Thirdly, we will conduct a suitability study to determine how the framework can applied to other areas of biological experimentation[3].

So far, we have developed an in-depth analysis of the example from which the framework is based. This consists of a detailed analysis of the example dataset (section 6.1) and of the example itself (section 6.2), a report on the sources of possible change (section 6.3) and an analysis of the impacts that occur as a result of those changes (section 6.4). There has been the opportunity to conceptualise the framework itself (sections 6.5 and 6.6) and to investigate some possible ways of implementing the framework (sections 6.7, 6.8 and 6.9).

2 Literature Review

This section provides a current edition of the literature review. It is intended that the literature review be a continually evolving document. While literature is added all the time as it becomes available, there may well be material not yet included (in a *pending* state).

2.1 Organisational Characteristics

There are three principal stakeholders involved in the collection, storage and curation of genomic information. The three members of the INSDC (International Nucleotide

[2]Refer to 2.4.7 for details of this unsuitability.

[3]Other areas refers to experimentation with other types of biological data or biological processes.

Sequence Database Collaboration) are; **NCBI:** GenBank - maintained by the NCBI (National Center for Biotechnology Information); **EMBL-EBI:** European Molecular Biology Laboratory's European Bioinformatics Institute; **DDBJ:** DNA Data Bank of Japan. Thanks to their exchange policy, these three members have recently reached the significant milestone by collecting and disseminating 100 gigabases of sequence data. These bases represent both individual genes and partial and complete genomes of over 165,000 organisms.

The NCBI (National Center for Biotechnology Information) provides a national resource of molecular biology information. Among other activities, the NCBI creates and maintains public databases, conducts research in computational biology, develops software tools for analysing genomic data and aids in the dissemination of biomedical information. The aim is to aid the understanding of molecular processes that affect human health and disease.

The EMBL (European Molecular Biology Laboratory) conducts basic research in molecular biology, aiming to provide essential services to its member states. The EMBL also endeavours to provide high-level training for scientists at all levels (including PhDs). Together with the development of new instrumentation for biological research, the EMBL seeks to further the fundamental understanding of the basic biological processes in model organisms. The EBI (European Bioinformatics Institute) is a non-profit organisation, which forms part of the EMBL. The purpose of the EBI is to provide freely-available data and bioinformatics services to all areas of the scientific community in a way that promotes scientific progress. It aims to contribute to the advancement of biology through basic investigator-driven research in bioinformatics. As part of the EMBL, the EBI also provides training to scientists and aims to disseminate relevant technologies to industry. Modern technologies have provided a vast amount of information on a variety of living organisms. There is a danger of being overcome by the size of this data. Therefore there is an ongoing need to collect, store and curate the information in ways that allow efficient retrieval and exploitation. The EBI aims to fulfill this important task.

THE DDBJ (DNA Data Bank of Japan) is a nucleotide sequence database, which collects, annotates and then releases the original and authentic DNA sequence data. The DDBJ aims to only release the sequence data after annotation. However, given the large amounts of data deposit and the time that it takes for annotation, a significant backlog has occurred. This has prompted the DDBJ to explore new ways of annotating DNA sequences.

The WTSI (Wellcome Trust Sanger Institute) aims to further the knowledge of genomes, in particular through large scale sequencing and analysis and is responsible for the sequencing of over a third of the human genome. The WTSI seeks to foster and promote research with the aim of improving the quality of human and animal health.

The NBII (National Biological Information Infrastructure) concentrates on the increasing the access to data and information on the nation's biological resources. The NBII links diverse, high quality databases, information products and analytical tolls maintained by its partners, government contributors, academic institutions, non-governmental organisations and private industry. More specifically, the NBII works on new standards, tools and technologies that make it easier to find, integrate and apply biological resources information.

PlasmoDB (the Plasmodium Genome Resource) is part of an NIH/NIAID funded Bioinformatics Resource Center to provide Apicomplexan Database Resources. PlasmoDB[4] is the official database of the Plasmodium falciparum genome sequencing consortium. This resource incorporates finished and draft genome sequence data and annotation emerging from Plasmodium sequencing projects. PlasmoDB currently houses information from five parasite species, and provides tools for cross-species comparisons [1].

2.2 Biological Data Sources

Before discussing aspects of biological data management, it is important to first talk about the various biological data sources currently available to biologists and researchers. One of the issues surrounding biological data management is the heterogeneity of the data and multiplicity of the data sources, so let us investigate this problem here.

2.2.1 Biological Sequence Formats

This section aims to provide a brief description of the principal data formats together with the sources for that data. Four types of data format are discussed below; sequence, microarray, clinical and mass spectrometry data. There are many other different types of biological data format (a point which this section aims to convey), far too many to adequately cover in this document. Similarly with respect to the data sources, only a selected sample will be presented here.

Sequence formats are the way in which biological data such as amino acid, protein and DNA sequences are recorded in a computer file. If you wish to submit some data to a data source, it must be in the format that that particular data source is prepared to receive. There are numerous sequence formats in the biological domain, too many to discuss their details and differences in this document. In general, each database will have its own sequence format. Some databases such as GenBank, EMBL and DDBJ (DNA Data Bank of Japan) share similar data formats (albeit with differing headers). FASTA is a simple text format commonly used by many pieces of bioinformatics software but there are many others (NEXUS, PHYLIP and many others).

The EBI (European Bioinformatics Institute) ensures that all the information from molecular biology and genomic research is made publicly and freely available in order to promote scientific progress. The EBI's databases and tools allow biological information to be stored, integrated, searched, retrieved and analysed. As mentioned above, each

[4]Refer to http://PlasmoDB.org

available database will require the appropriate tool in order to make successful queries and each database will require data to be submitted in the correct format.

Microarrays are one of the most important breakthroughs in experimental life sciences. They allow snapshots to be made of gene expression levels at a particular genomic stage[5]. Microarray data can be accessed though *Array Express* which is the public repository for microarray-based gene expression data. Although many significant results have been derived from microarray studies, one limitation had been the lack of standards for presenting and exchanging such data [2]. MIAME (Minimum Information About a Microarray Experiment) [2] describes the minimum amount of information necessary to ensure that the microarray data can be easily verified and enable the unambiguous interpretation and reproduction of such data.

The effective and efficient delivery of health care requires accurate and relevant clinical information. The storage of clinical information has traditionally been limited to paper, text or digitised voice. However, a computer cannot manipulate data in these formats. Clinical terminologies are also large, complex and diverse with respect to the nature of medical information that has been collected over 130 years of the discipline. There are numerous schemes which have been successful in supporting the collation and comparison of medical information. However, the problem arises when we try to transfer information between schemes. It is also hard to re-use schemes for purposes other than which they originally developed and this causes the proliferation of even more schemes. Galen (and the open source version OpenGalen [3]) provides a formal model of clinical terminology in an attempt to bridge the gap.

Mass spectrometry is a powerful analytical technique that is used to identify unknown compounds and quantify known compounds, even in very minute quantities. It is also used to establish the structure and chemical properties of molecules. Mass spectrometry can be used to sequence biopolymers such as proteins and oligosaccharides, determine how drugs are used by the body and perform forensic analyses such as drug abuse (athlete steroid abuse among others). Due to the large amounts of information that can be generated by mass spectrometry, computers are essential, not only to control the mass spectrometer but for spectrum acquisition, storage and presentation. Tools are available for spectral quantitation, interpretation and compound identification via on-line spectral libraries.

2.2.2 Biological Data Access

According to the EBI website, the general method for data manipulations (at least from the EBI databases) is the following;

Choose a tool: The EBI toolbox provides a comprehensive range of tools for bioinformatics. This refers to the method to which the submitted data will be subjected. EBI

[5]Taken from http://www.ebi.ac.uk/Databases/microarray.html

offers many tools including; similarity and homology search algorithms such as Blast and FASTA; protein function determination services such as CluSTr and GeneQuiz; proteomic services such as UniProt DAS [4]; sequence analysis (identifies similarities between unknown sequences and those sequences that have been functionally identified) such as ClustalW and pairwise-alignment tools; highly accurate structural analysis algorithms such as DALI and MaxSprout.

Select a database: There are a range of molecular biology databases to choose, depending on the type of data that you may wish to use, including; nucleotide sequence, protein sequence, protein function/interactions, gene expression and microarray, gene ontologies, biological structure, scientific literature and text mining and metabolic pathway databases.

Enter or upload the sequence: This will normally/often involve cut and pasting a sequence into a large text window. This sequence should be in the appropriate format[6].

Press run and wait for your results: Your results are either displayed "interactively" or emailed to your email account.

2.2.3 A Biological Data Problem

This subsection will describe a biological data problem, which highlights an area where data versioning may be of use. We also provide a scenario for determining the context of the problem. In this scenario, experiments are conducted using biological data sources and yield results. These results are, in turn, used as data sources in other experiments to yield further sets of results. This cycle of experimentation continues, presumably until the required set of results are achieved. This can often lead to many cycles of experimentation with complex sets of data sources from many heterogeneous sources. For the purposes of simplification and to aid explanation, it is assumed that the experimentation cycle occurs, for want of a better term, in a linear fashion. In other words, the results of one cycle form the entire data source for the next cycle with no other inputs.

Figure 1 illustrates this cycle of experimentation.

1. Importing biological data. Of all the scientific disciplines, biology has one of the most complex information-structures (concepts, data types and algorithms) [5]. Its richness and diversity provides many challenges for biological sciences, computational sciences and information technology. There exists a substantial amount of biological data which is stored in a variety of formats in a multitude of heterogeneous systems. Accessing the relevant data, combining data sources and coping with their distribution and heterogeneity is a very difficult task.

[6]Partially formatted sequences will be rejected.

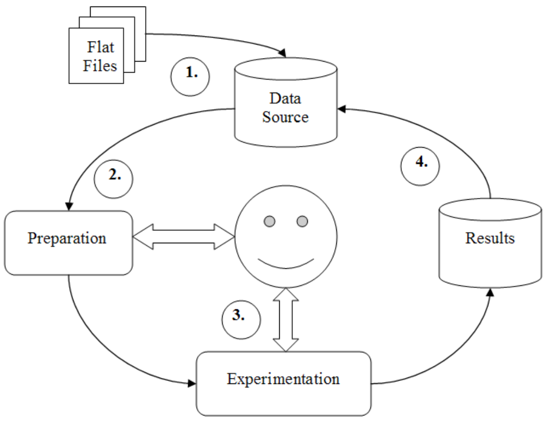


Figure 1: The cycle of experimentation and the re-use of results.

Data integration of geographically dispersed, heterogeneous, complex biological databases is a key research area [6]. One of the principal issues of data integration is the data format. Many current biological databases provide data in flat files which are poor data exchange formats. Worse still, each biological database has a different format. This makes the integration of such datasets difficult and time-consuming. In the example illustrated in figure 1, the flat file containing the biological data will be imported (usually downloaded over the Internet) and placed into whatever structure the bioinformaticien has devised for the purpose.

2. Preparation of the data. It is unlikely that, after importing the data, it will be ready for experimental use. It is also unlikely, although possible, that the data will be in the required format for the experimentation. Therefore, the data must be *wrapped*. *Wrapping* a data source refers to getting data from somewhere and translating it into a common integrated format [7]. The intended experimentation will seldom use the entire downloaded data source (this flat file including all the possible information), but rather a selected dataset. Therefore, the bioinformaticien may wish to manually select the dataset before starting the experiment. This can result in a considerable saving of experiment time. Manipulating and wrapping the data source can be a very timely process and may also depend on the nature of the experimentation to be done. It also involves considerable input from the scientist.

3. Experimentation. After the data has been prepared and the experiment is ready to be carried out, the scientist still needs to do a few more things. Firstly, the data source must be identified. Secondly, the method of experimentation itself must also be selected. This may be a third-party tool or a user-defined software program. Whatever the nature of the method, it is likely that the user will also want to specify some parameters. The parameters may relate to the form of the datasets being used in the experiment, the structure of the required output (results) or they may relate specifically to the bioinformaticien's experimental hypothesis. In any case, the state of the parameters will have an effect on the experiment and, therefore, the results.

4. Renewing the data source. When the experiment is complete, the results are placed in a data repository where they become available for further experimentation. This cycle will continue until the required results are obtained and this may result in many cycles.

The entire process can take a long time and that, itself, is a problem that requires attention. A more serious issue is the update to the initial biological data. The large biological data repositories, from which the data is taken as the initial data source, currently have no way of dynamically reflecting changes, updates, additions and deletions. Therefore, they release versions of the data (in flat files) every so often[7].

Clearly, it is preferable for the bioinformaticien's results to reflect the latest changes from the initial flat file data source. However, with the current methodology, every result is dependent on the initial data source. In order to get to the same stage (as before the new data version was released), the bioinformaticien must perform each experiment as before, only this time using the updated dataset. This will result in their having to re-do all previously conducted experiments, which also depends on their having accurately recorded the experimentation already done.

2.3 Biological Data Management

Over the past twenty or so years, biology and biological research has largely been dominated by what has become known as the genomic era. It was believed that the mapping of the human genome would herald a new world of discovery for the life sciences. Unfortunately, little (by comparison) has come to fruition due to the lack of understanding of the genome.

A large amount of genomic and proteomic data has been collected but there is still a lot to be done to allow its full understanding. The vastness of the data represents a part of the problem. There has been a significant amount of research on the problems attached to biological data and its management and I will endeavour to represent the most significant advances in this chapter.

Of all the scientific disciplines, biology has one of the most complex information-structures (concepts, data types and algorithms) [5]. Its richness and diversity provides many challenges for biological sciences, computational sciences and information technology. For the purposes of this document and with regard to my research, the term 'biological data' refers to data that is in digital format.

[7]The length of time between versions varies between database providers.

2.3.1 The Problems with Biological Data

This section will describe some of the currently observed problems surrounding biological data. There exists a vast amount of biological data, which is stored in a variety of formats in a multitude of heterogeneous systems. A large amount of biological data has been (and continues to be) collected. However, simply collecting and maintaining the data does not allow its understanding. The vastness of the data contributes to this problem. However, it is not only its volume that makes the management of biological data unique. Indeed, there are many problem domains that must deal with very large volumes of data.

Accessing the relevant data, combining data sources and coping with their distribution and heterogeneity is a very difficult task and, consequently, has received due attention. Specialist bioinformatics databases contain detailed information that is essential for the analysis of the relevant data. These specialist databases aim to; improve the level of annotation, clean the data, integrate from multiple data sources and integrate bioinformatics software tools. This is to address the issues with the volume, the disparate and heterogeneous data sources and, to some extent, the complexity of the data. However, there are problem domains that are data-intensive, source data from multiple sources and require the manipulation of complex data and data structures.

It has already been mentioned that biological data is considerably complex and this certainly contributes to part of the problem. Of all the scientific disciplines, biology has one of the most complex information structures (concepts, data types and algorithms). The richness and diversity of the data provides many challenges for biological and computational science and information technology.

These aspects alone do not illustrate the full extent of the problem with biological data. The problem with biological data becomes unique when the issue of data semantics is considered. Although there exists important issues with the information currently held in the data, there are also problems arising from the experimental information that is not being retained as part of the data (or metadata). One of the distinctions of biological data with respect to other types of scientific data, is the complexity and variety of the experiments that yield the data. Moreover, this complexity and variety have influences over the data generated, but are not recorded completely in the metadata. Metadata is a description of the content, quality, lineage, contact, condition and other characteristics of data. The aim is to retain information about the data, which is important but which is not reflected by the data itself. In the case of biological data, the complexity and heterogeneity of the experiments (and, therefore, the required metadata) is very substantial and this results in an increased difficulty in data management.

2.3.2 Data Retrieval and Access

There are a number of data sources from which scientists and researchers may wish to retrieve and access biological data, including[8]; GenBank, RefSeq, UniProt, Human Protein Reference Database (HPRD), Biomolecular Interaction Network Database (BIND), Database of Interacting Proteins (DIP), Molecular Interactions Database (MINT), IntAct, NCBI Taxonomy, Gene Ontology (GO), LocusLink, Entrez Gene, HomoloGene and many more.

In order to understand the problems associated with biological data management, it is necessary to first understand the current practice of this process. Currently, biological data management is largely conducted within specialist bioinformatic databases (data warehouses). Specialist bioinformatic databases contain detailed information, such as functional and structural properties and expert-enriched information that is essential for the in-depth analysis of the relevant data [10]. This level of detail is usually lacking in the primary databases such as SwissProt or GenBank. These specialist databases generally exhibit the following properties/qualities; Increased detail of annotation; cleaner data; integrated data from multiple sources; (often) integrated searches and analysis tools.

Schönback et al [11]defines such a biological data warehouse as a subject-oriented, integrated, non-volatile, expert-interpreted collection of data in support of biological data analyses and knowledge discovery. According to [10], the key steps to constructing a data warehouse (a Specialised Sequence data warehouse in the case of [10]) include the extraction of raw data from multiple sources and transformation to the data warehouse format (*data integration*), removing errors and discrepancies (*data cleaning*), enriching the data with information from relevant literature or from experimental results (*data annotation*) and, finally, the construction of a new database[9].

BioWare, a framework for the construction of a biological data warehouse, is described in [10]. As opposed to many SSDWs (Specialised Sequence Data Warehouse(s)), the users of BioWare do not require any specific programming or database expertise in order to retrieve, annotate and publish their data in the form of a searchable SSDW. However, BioWare does have some limitations. Given that the SSDW is designed to accessed via HTTP, a bottleneck is created with excessive downloading from public data sources. The current system, therefore, holds only 1000 entries. The data is still held in flat files, which impedes the speed of data access.

The EnsMart system [12] purports to provide a generic data warehousing system for fast and flexible access to biological datasets as well as integration with third-party data and tools. EnsMart is a system that is capable of organising data from individual databases into one query-optimised system. The generic nature of EnsMart allows the integration of data in a flexible, efficient, unified and domain-independent manner [12]. EnsMart is

[8]Taken from [9]

[9]Incidentally many of these operations are included in this Biological Data Management section.

a self-contained addition to Ensembl software and data, providing access to commonly used parts of the genome data. One of the noted benefits of the EnsMart system is the ability to engineer your own version (with a little informatics expertise). This is useful is you wish to use data and queries that are not explicitly offered by the website. This will also overcome the problem of slowness or 'out-of-service' at www.ensembl.org when mining data in large quantities. EnsMart is reported to offer new levels of access and integration for the use of large quantities of genome data [13].

2.3.3 Data Annotation Systems

Annotations are features on the genome derived through the transformation of raw genomic sequences into information by integrating computational tools, auxiliary biological data and biological knowledge[10]. With the completion of the human genome and genome sequences being readily available, the task of manual annotation has become a large priority and increases to be so, as we look into the future [14].

In the beginning, bioinformatics data was stored in flat files as ASCII characters. This was sufficient as the number of known proteins were small and there was little thought of genomic data. With the advent of rapid gene sequencing, it was established that techniques such as dynamic programming would be needed to cope with the exponential growth of data [16].

Ensembl [17] began as a project to automatically annotate vertebrate genomes. Ensembl runs many commonly-used bioinformatics tools and combines the outputted data to produce genome-wide data sets such as protein-coding genes, genome-genome alignments and other useful information. Obviously, there is a considerable amount of processor power needed to produce the annotation for the vast amount of biological data. It is not feasible to compute this kind of problem on a single-CPU, but rather employ a distributed machine infrastructure.

Currently, most gene annotation is curated by centralised groups and little effort has been made to share annotations between multiple groups.The Distributed Annotation System (DAS) allows sequence annotations to be decentralised among multiple annotators and integrated by client-side software [18]. DAS has been successfully used to integrate protein annotations [4].

2.3.4 Data Integration Problems and Tools

Data integration of geographically dispersed, heterogeneous, complex biological databases is a key research area [6]. One of the principle issues of data integration is the data format. Ideally, a simple, self-describing format is best. However, many current biological

[10]Taken from ISMB '99 Tutorial 3: The challenge of annotating a complete eukaryotic genome: A case study in Drosophila melanogaster

databases provide data in flat files which are poor data exchange formats. Worse still, each biological database has a different format. Bio2X [6] is a system that gets around this problem by converting the flat file data into highly hierarchical XML data using rule-based machine learning techniques.

Data integration consists of wrapping data sources and either loading the retrieved data into a data warehouse or returning it to the user. *Wrapping* a data source means getting data from somewhere and translating it into a common integrated format [7]. There have been many systems designed for the specific purpose of biological data integration (too many to mention here). None of these approaches provides a seamless integration that permits the use of metadata about source content and access cost to permit the optimisation of the evaluation of the queries [7].

A wide variety of biological experimental techniques are available from the classic methods to high-throughput techniques such as gene expression microarrays. In the future, more disparate methods will be developed, which will continually push integrative technologies and continue to drive research in this area. MAGIC (Multisource Association of Genes by Integration of Clusters) is a general framework that uses formal Bayesian reasoning to integrate heterogeneous types of high-throughput biological data [19].

2.3.5 Data Cleaning

As the amount of biological data increases and becomes more pervasive, there arises various issues surrounding data quality, data legacy, data uniformity and data duplication. *Data cleaning* is the process by which biological data is corrected and standardised. However, due to the complex and diverse nature of the data, the problem of improving the data quality is non-trivial [20]. If quality of the data is not maintained, then the processes and operations that rely on this data will either fail or (arguably worse) provide skewed results. BIO-AJAX [20] is a toolkit that provides an extensible framework with various operations to address data quality issues through data cleaning within biological data repositories.

2.3.6 Metadata

Metadata literally means "data about data" and is a term that is understood in different ways by the diverse professional communities that design, create, describe, preserve and use information systems and resources [21]. In other words, metadata is structured information that describes, explains or locates, or otherwise makes it easier to retrieve, use or manage an information resource.

Metadata is key to ensuring that resources will survive and continue to be accessible in the future. It is generally understood that there are three main types of metadata[11].

1. *Descriptive* metadata describes a resource for the purpose of discovery and identification. It may include descriptive elements such as title, author, data, keywords, etc.

2. *Structural* metadata provides information as to how the data is structured. I.e. how pages are ordered into chapters.

3. *Administrative* metadata provides information to help manage the resource, such as when and how it was created, file type and other technical information.

According to [21], in an environment where a user can gain unmediated access to information objects over a network, metadata; certifies the authenticity and degree of completeness of the content; establishes and documents the context of the content; identifies and exploits the structural relationships that exist between and within information objects; provides a range of intellectual access points for an increasingly diverse range of users; provides some of the information an information professional might have provided in a physical reference or research setting.

Metadata plays an important role in providing semantic meaning to resources and promotes interoperability (the ability of multiple systems with different hardware and software platforms to exchange data with minimal loss of content and functionality).

The existence of information in digital form has heightened interest in the ability to create multiple and variant versions of those objects. Metadata is used to link multiple versions and capture the similarities and differences of each version. The metadata must also be able to distinguish what is qualitatively different between variant digitised versions and the hard copy original or parent object [21].

2.4 Generic Data Versioning

Although there has been substantial amount of research and publications concerned with the management and problems arising from the management of biological data, there is only a very limited amount on biological data versioning specifically.We have, therefore, decided to look at generic data versioning with a view to investigating the feasibility of applying existing data versioning techniques within the biological data domain. There has been a great deal of research done on data versioning. We cover only those areas that are appropriate to biological data.

[11]This is also referred to as content, context and structure [21].

2.4.1 Current Versioning Tools

There have been many approaches to the versioning of files and data and some of them are described here. There were several early file systems that provided versioning, such as the Cedar File System (CFS) [22] and 3D File System (3DFS) [23]. However, these systems were not transparent. That is to say, users had to manually create versions using special commands and tools. With CFS, users had to change a copy of the file on their local system. A file was only versioned when it was transferred to the remote server.

Unix and Unix-derivatives have a long history of using version management tools with the use of the original *diff* and *patch* programs. These led to a number of versioning tools, the most popular being RCS (Revision Control System) [24] and SCCS (Source Code Control System) [25]. SCCS [25] was one of the first versioning tools that stored differences between file versions as strings of *deltas* to show the progress history of the file. RCS [24] supported both forward and reverse deltas, allowing RCS the ability to either store an initial copy with subsequent changes or store the latest copy with previous changes. CVS [26] is also not transparent as users have to use specific commands to control the versioning of their files. CVS utilises a client-server architecture, whereby a client connects to the server in order to check-out a complete copy of the project, work on this copy and then later check-in their changes. CVS has become quite popular in the open-source world. Rational ClearCase [27] is another versioning tool, but requires specific administration and is also expensive.

BitKeeper [28] was developed more recently and aims to provide architecture to support globally distributed development. To work on a repository, the developer must first make a complete mirror of the original repository to work on, an idea similar to the CVS *sandbox*. When the developer wants to *bk push* (check in) their changes back to the tree, their changes are merged with changes, which have since been pushed in from other developers.

Subversion is a version control system with a working model similar to that of CVS, and was intended not only to replace CVS but to provide a versioned network filesystem over WebDAV (World Wide Web Distributed Architecture and Versioning). WebDav is aimed at providing a standard infrastructure for asynchronous collaborative authoring across the Internet. Subversion versions files and directories, provides support for metadata and can efficiently handle binary files.

Grid applications require versioning services to support effective management of constantly changing datasets and implementations of data processing transformations. V-Grid is a framework or generating Grid data services with versioning support from UML models that contain structural description for the datasets and schema [47].

There has been a considerable amount of research aimed at the versioning of XML data and documents. For this purpose, conventional versioning tools such as RCS and SCCS are inappropriate. This is largely due to the considerable computing overhead that occurs during the retrieval of a version [30]. Moreover, neither RCS nor SCCS preserve the logical structure of the original document. This somewhat negates the benefit of a hierarchical document such as XML.

There are also various snapshotting tools available such as WAFL [31] and Ext3cow [32]. With the snapshot method, incremental or whole snapshots of the file system are made periodically. When required, the entire file system can be restored from any recorded point in history. However, snapshotting makes no allowances for how often a file may be changed, it simply works on a discrete time period of data capture. The consequence of this is that files which do not change regularly will be continually 'snapshotted' regardless. Conversely a file that is being regularly updated may have updates that are not captured with the snapshot method. This suggests that it may be inappropriate for biological data versioning. Also, entire snapshots must be purged if the space is to be reclaimed.

Versioning employing the copy-on-write technique is used by Tops-20 [33], VMS [34], the Elephant File System [35] and CVFS [36, 37]. An ideal situation would be to have versioning automatically integrated into the operating system. However, this has yet to be implemented.

2.4.2 Current Object Versioning Methodology

One of the main issues in biological data versioning is how to deal with the sheer vastness of the data. Conventional versioning systems do not efficiently record large numbers of versions. Given the large amount of biological data, the prospect of storing multiple versions of the data (as well as the associated metadata) is an unpleasant one.

This subsection aims to explore the methodology behind current versioning techniques and describe some of the differences between various approaches. According to [38], there are two main methods for dealing with object versioning. The first technique stores versions of a particular object as complete objects and is referred to as *complete versioning.* This approach is fairly easy to implement, although the waste of storage space becomes more detrimental as the number of versions increases. The second technique stores a single object and each version is maintained as a difference between the current version and the previous version. This approach is more difficult to implement but it is more suitable for representing data that may be subject to continuous and dynamic changes [38].

Using this approach, changes in objects are handled using a version management system. Each version reflects a change in the object's attributes and/or behaviour. Subsequent changes in the object (versions) will generate related dynamic attributes and temporal links to the updated versions. This type of version management reduces the need for large storage space[12], since the current object is only stored once in its entirety [38].

[12]Storage is definitely a serious consideration given the vast amount of biological data that may be

Linear versioning is a technique where one version is stored as a complete object and the rest of the versions are represented as differences between the versions. This approach is based on one-to-one versioning. That is to say, that each parent or base object will have only one child or derived object. Linear versioning can be classified into two versioning strategies [38]. The first allows the current version to be calculated from the initial base version (with the addition of the subsequent versions representing the changes over time) and is referred to as *forward oriented versioning.* The second strategy stores the current version as the base version and previous versions are calculated as differences to the current version. This is known as backward oriented versioning [39].

Which method to choose varies greatly on the type of manipulation that the data is to be subject to. Forward oriented versioning takes longer to retrieve the current object (differences must be added to the base version to get the current version). However, it takes less time to retrieve earlier versions. Backward oriented versioning provides faster access for the newest versions. As a result, this strategy is usually more suitable for most applications [38].

Branching is a technique where one version is stored as a complete object and all other versions are stored as differences between that version and other versions. As all versions are just one 'difference' away from the current version, the *branch forward versioning* strategy provides the same access time for all versions.

2.4.3 The Use of Metadata in Versioning

Metadata is a description of the content, quality, lineage, contact, condition and other characteristics of data[13]. Biological metadata works in the same way. Information about the data, including content, quality, and condition makes the data more easily accessible to scientists and researchers. Unfortunately, conventional versioning systems do not efficiently record large numbers of versions. In particular, versioned metadata can consume as much space as the data itself [36]. Two space-efficient metadata structures for versioning file systems are examined in [36].

2.4.4 Schema Evolution and Temporal Versioning

A schema describes the structure of the data that is being stored. It is possible to evolve schema in response to the changing structure of the data. Schema evolution is a way of dealing with the changes that the data warehouse (DW) schema undergoes over time in response to evolving business requirements [40, 41].

effected by version management.

[13]According to the National Biological Information Infrastructure website - http://www.nbii.gov/datainfo/metadata/

2.4.5 Examples of Successful Data Versioning

The purpose of this section is not to look at versioning methodology as a whole, but rather look at specific examples of where versioning has worked in areas unrelated to the life sciences. Of particular importance are areas that seek to version data that is similar in structure to biological data. This section attempts to identify some specific areas where data has been successfully versioned and section four aims to explain how these techniques may (or may not) be applied to versioning biological data. It should be noted that this section is not intended to provide an exhaustive list of data versioning successes, but rather to present a few examples to demonstrate some different techniques of data versioning.

Business data is largely focussed on the most current version of the data. Previous versions are kept as a way of providing historical data for analysis. This is useful for documentary analysis and visualising business trends (showing improvement or decline), but is not as widely used as the current data version.

During the life cycle of a piece of software, different versions may be developed depending on the state of development and the purpose of the intended release. In order to effectively support software development, these versions as well as the relationships between them, must be stored [42]. These differing versions may be due to the development of the software (i.e. the adding of functionality). However, a specific set of versions may be needed to fulfill the requirements for a certain release of the software. For example, a shareware release of a software item may require a previous set of versions (with limited functionality). A beta release may require the latest set of versions, irrespective of the level of testing undergone. Software versioning is often conducted with a file versioning system, such as CVS [26]. CVS utilises a client-server architecture, whereby a client connects to the server in order to check-out a complete copy of the project, work on the copy and then later check-in their changes. Basic software versioning techniques typically provide versioning of large granularity objects (at the file level), whereas it may be more useful and appropriate to version at a finer granularity level (at the object and instance level).

An interesting example of successful data versioning in a scientific domain is presented by Barkstrom [43]. According to [43], large-scale scientific data production for NASA's Earth observing satellite instruments involves the production of a very vast amount of data from a very wide variety of data sources[14].It is noted that while software versioning requires tracking changes principally in the source code, versioning of the data requires the tracking of changes in the source code, the data sources and the algorithm coefficients (parameters). Barkstrom [43] notes that changes in any of these items can induce scientifically important changes in the data.

Shui et al. [44] present an XML-based version management system for tracking complex

[14]In fact, Barkstrom [43] quotes the production values at tens of thousands of files per day from tens or hundreds of different data sources.

biological experiments. Shui et al. [44] make a good case for the need for biological data versioning, similar to the one made in this report. The framework uses generic versioning operations (insert, delete) and defines three more (update, move and copy) in order to describe changes in the XML. The framework can store every single component of an entire experiment in XML. A change to any component will result in a new version. Users can then query the system to return sets of data so they can see the differences between sets of results according to the materials used. The title of the publication [44] reports to track complex biological experiments. However, the conclusion of the same publication states clearly that the framework only tracks changes to laboratory based data. This publication is important in assessing the need for data versioning of complex life-science data and experimentation, but proposes a framework that appears to deliver little more than generic XML versioning, applied in a scientific data context.

2.4.6 Other Versioning Technologies

Normally, versioning functionality has been implemented by high-level applications or at the filesystem level. [45] proposes a technique that pushes the versioning functionality closer to the disk by taking advantage of modern, block-level storage devices. Versioning at the block-level has several advantages over versioning at the filesystem or application levels. Firstly, it provides a higher level of transparency and is completely filesystem independent [45]. It also reduces complexity in the higher layers, in particular the filesystem and storage management applications [46]. Thirdly, it offloads expensive host-processing overheads to the disk subsystem, thus, increasing the overall scalability of the system [46].

However, there are disadvantages to versioning at the block level. Having offloaded complexity from one level to another, it is still picked up somewhere and the ramifications of this transference is unclear. The consistency of the data may be affected with the use of the same volume as the filesystem. But perhaps the most relevant disadvantage with respect to biological data versioning is the problem of versioning granularity. Since the data versioning is occurring at a lower system layer, information about the content of the data is unavailable. Therefore, access is only available to full volumes as opposed to individual files.

2.4.7 The Unsuitability of Current Versioning Technologies

Before examining the unsuitabilities of various versioning techniques, it seems prudent to examine the overall need for versioning. According to [36], the benefits of versioning can be grouped into three categories.

- Recovery from user mistakes
- Recovery from system corruption

- Analysis of historical change
- Merging of disparate work sources

The first two items above, although important in their own right, are not specifically applicable to the problem of biological data versioning. The analysis of historical change is important, however, as it is specifically the history of the data in which we are interested. Analysis of the history can help us answer questions about how a file reached a certain state [36]. For example, CVS [26] and RCS[24] keep a complete record of committed changes to specific files. It is reasonable to assume that due to the various needs for versioning, tools will be suited for different aspects of versioning.

It would take too long to explain how each of the aforementioned versioning technologies are individually unsuitable for the versioning of biological data. It is important to first note that, with a few noted exceptions [43] [44], the versioning technologies described above have not been designed with the intention of versioning scientific data. It is therefore unlikely that the versioning tools will provide a perfect solution for the prescribed problem. In short, current versioning technologies do not take into account the uniqueness of biological data and the constraints that versioning such data creates. However, this does not necessarily preclude these versioning tools from being used or modified for the purpose. The unsuitability of some of the more common versioning tools are described below.

When considering the unsuitability of a versioning tool, it is necessary to examine the function of the tool and measure that against the required functionality to solve the problem. In the case of biological data (and the problem of versioning such data), we must consider a scenario where multi-versioned data can be processed (experimented upon) in a variety of ways, producing multi-versioned results. When contemplating a problem of this kind, it seems obvious that a traditional file-versioning technology will not provide the required level of flexibility. It also follows that a snapshotting tool will also not provide the required level of flexibility.

Traditional document version management schemes, such as RCS and SCCS, are line-oriented and suffer from various limitations and performance problems. Both RCS and SCCS may read segments that are no longer valid for the required version [30]. Also, RCS and SCCS do not preserve the logical structure of the original document. This makes structured-related searches on XML documents difficult. It may require that the entire original document be reconstructed before such a search can take place [30].

XML versioning provides a closer match for the required purpose [30]. It is suggested that XML document versioning provides a sufficient solution for managing the versioning of biological data experiments [44]. However, this approach only addresses part of the problem by merely tracking the changes to the data. If biological data is to be adequately managed and versioned, the semantics of the experimental process must be encorporated into the solution. Given the self-describing nature of XML, it seems a promising method for describing structured data.

2.5 Versioning Frameworks

Various versioning frameworks such as the use of ontologies and RDF for the semantic web are described later in this document. The aim of this section is to include any other generic versioning strategies and frameworks that may not be specifically relevant to other areas. Therefore the content in this section may be quite limited (as the content will be included elsewhere).

Versioning systems are generally required to manage data objects that change frequently over time [47]. The objective is to achieve the ability of representing intermediate stages in the object's evolution paths as *versions*. [48] defines a version a version as a semantically meaningful snapshot of a design object (an artifact) at a point in time. One objective of versioning is to allow users access to these objects at previous points in time.

[49] investigates the use of versioning systems to facilitate the return to previous stages of work if a particular path of the design process doesn't work out. The versioning of an engineering process enables the reusability of a design path since a new design version can be derived from a previous design version. Object-oriented database management systems (OODBMS) can employ versioning at any level of granularity (e.g. schema, class, object). [49] presents a framework for checking each object version for data correctness and consistency.

2.6 Ontologies and the Semantic Web

There has been a great deal of research conducted in the areas of ontologies and the semantic web, far too much to be exhaustively investigated here. We provide a concise summary of the key points with respect to the problem of Biological data versioning.

An ontology is;

> An ontology is a formal, explicit specification of a shared conceptualization. Conceptualization refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine readable. Shared reflects that ontology should capture consensual knowledge accepted by the communities. [50]

2.6.1 Ontologies

There is a large amount of heterogeneous biological data currently available to scientists. Unfortunately, due to its heterogeneity and the widespread proliferation of biological databases, the analysis and integration of this data represents a significant problem. Biological databases are inherently distributed because the specialised biological expertise that is required for data capture is spread around the globe at the sites where the data originate. To make the best use of this data, different kinds of information must be integrated in a way that makes sense to biologists.

Biologists currently waste a great deal of time searching for available information in various areas of research. This is further exacerbated by the wide variations of terminology used by different researchers at different times. An ontology provides a common vocabulary to support the sharing and reuse of knowledge. The OBO ontology library [51] is a repository of controlled vocabularies, which has been developed in order to provide for improved communication across different biological and medical domains.

2.6.2 GO Consortium

The Gene Ontology (GO) project provides structured, controlled vocabularies and classifications that cover several domains of molecular and cellular biology. The project is driven and maintained by the Gene Ontology Consortium. Members of this consortium continually work collectively and, with the help of domain experts, seek to maintain, expand and update the GO vocabularies. Collaborations of this nature are difficult to maintain due to geography, misunderstandings and the length of time it takes to get anything done.

The Gene Ontology (GO) project is a collaborative effort that aims to address two aspects of information integration:

- Providing consistent descriptors for gene products in different databases.
- Providing a classification standard for sequences and sequence features.

The project began in 1998 as a collaboration between three model organism databases: FlyBase (Drosophila), the Saccharomyces Genome Database (SGD) and the Mouse Genome Database (MGD). Since then, the GO Consortium has grown to include many databases . The benefits of using GO increase as the number of participating databases increases. The GO project has grown enormously and is now a clearly defined model for numerous other biological ontology projects that aim to achieve similar results.

2.6.3 Ontology Methodology

Barry Smith is one of the leading characters in the development and analysis of Ontologies. His studies are not limited to biological Ontologies and have also addressed much of the methodology behind Ontologies. His latest publication [52] addresses the issue of relationships between biological Ontologies. [52] investigates the importance of vigorous, formal relations to ensure the maximally reliable curation of each single ontology while at the same time guaranteeing maximal leverage in building a solid base for life-science knowledge integration.

In [52], Smith also investigates some of the shortcomings of biomedical Ontologies at the moment. He also addresses the problem of under-specification of formal Ontologies in [53]. He has conducted study on data integration with Ontologies [54] and has also been involved in the development of tools to conduct such data integration [55].

2.6.4 Ontology Versioning on the Semantic Web

Ontologies are considered the basis building blocks of the Semantic Web as they allow machine supported data interpretation reducing human involvement in data and process integration [56]. Ontologies provide a reusable piece of knowledge about a specific domain. However, these pieces of knowledge are often not static, but evolve over time [57]. The evolution of ontologies causes operability problems, which hamper the effective reuse. Given that these changes occur and given that they are occurring within a constantly changing, decentralised and uncontrolled environment like the web, support to handle these changes is needed. This is especially prudent with respect to the semantic web, where computers will be using the data (humans are more likely (but still unlikely, however) to spot erroneous data due to unexpected changes).

Also, consider the nature, complexity and quantity of dependencies that exist between data sources, applications and ontologies. Changes in one area will have far-reaching effects [57]. [57] goes on to provide a framework for the versioning of ontologies and an interesting example of the versioning.

Traditional versioning systems (e.g. for the use of text and code) enable users to compare versions, examine changes, and accept or reject changes. However, an ontology versioning system must compare and present *structural* changes rather than changes in the text representation of the ontology. [58] presents the PROMPTDIFF ontology-versioning environment, which reportedly addresses these challenges. PROMPTDIFF includes an efficient version-comparison algorithm that produces a structural diff between ontologies.

2.6.5 The Semantic Web

The next generation of the web is often characterised as the *Semantic Web*. The semantic web refers to a system whereby information will no longer be easily-accessible for human readers, but also for processing by machines, enabling intelligent information services, personalised web-sites and semantically empowered search-engines [60]. The semantic web requires interoperability on the semantic level. *Semantic interoperability* requires standards not only for the syntactic form of documents, but also for the semantic information.

> "The Semantic Web will enable intelligent services such as information brokers, search agents, information filters etc. Such intelligent services on the Knowledgeable Web should surpass the currently available version of these services, which are limited in their functionality, and (most importantly), only work as stand-alone services that do not interoperate." [60]

2.6.6 XML and RDF

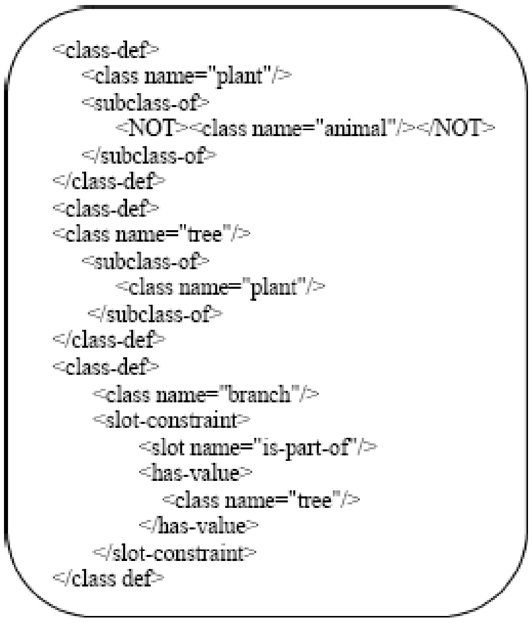
XML (Extensible Markup Language) and RDF (Resource Description Framework) are the current standards for establishing semantic interoperability on the Web. However, XML only describes document structure. RDF better facilitates interoperation because it provides a data model that can be extended to address sophisticated ontology representation techniques [60].

XML is intended as a markup-language for arbitrary document structure. An XML document consists of a properly nested set of open and close tags, where each tag can have a number of attribute-value pairs. One of the important aspects of XML is that the vocabulary is not set, but rather can be defined per application of XML. The following example XML shows a part of a defined ontology.

XML is foremost a means for defining grammars. Any XML document whose nested tags form a balanced tree is a well-formed XML document. A DTD (Document Type Definition) specifies the allowed combinations and nesting of tag-names, attribute-names, etc. using a grammar formalism. The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements. A DTD can be declared inline in your XML document, or as an external reference.

RDF was designed by the W3C in order to provide a standardised definition and use of metadata. *Resource Description Framework*, as its name implies, it is a framework for describing and interchanging metadata. A *Resource* is anything that can have a URI (Universal Resource Indicator) such as a web page or an element of an XML document. A *PropertyType* is a resource that has a name and can be used as a property. A *Property* is the combination of a Resource, a PropertyType and a value. A typical example of a Property might be: "http://www.cs.ucl.ac.uk/people/B.Tagger.html, Author, Ben Tagger." This would be represented as

```
<RDF:Description href='http://www.cs.ucl.ac.uk/people/B.Tagger.html'>
<Author>Ben Tagger</Author></RDF:Description>
```



```
<class-def>
    <class name="plant"/>
    <subclass-of>
        <NOT><class name="animal"/></NOT>
    </subclass-of>
</class-def>
<class-def>
<class name="tree"/>
    <subclass-of>
        <class name="plant"/>
     </subclass-of>
</class-def>
<class-def>
     <class name="branch"/>
     <slot-constraint>
          <slot name="is-part-of"/>
          <has-value>
             <class name="tree"/>
          </has-value>
     </slot-constraint>
</class def>
```

Figure 2: The XML representation of part of a defined ontology

RDF is carefully designed to have the following characteristics[15]:

Independence: Anyone can use RDF to invent their own types of metadata. The PropertyType can be anything and is not domain-specific.

Interchange: RDF properties can be converted into XML.

Scalability: As RDF properties are essentially made up of three distinct parts (see above), they are easy to handle and look up, even in large numbers. It is important that metadata is efficiently captured. With the large amount of data, it is important that the overhead caused by metadata is as minimal as possible.

In particular, XML falls down on the issue of scalability. Firstly, the order in which elements appear in an XML document is significant and can change the meaning of the document. When it comes to semantic interoperability, XML has disadvantages. Since XML only deals with the structure of the document, there is no way of recognising or extracting semantic meaning from a particular domain of interest.

2.6.7 LSIDs

Web standards, including RDF, make use of a family of identifiers collectively referred to as Universal Resource Identifiers (URIs). Members of this family include URNs

[15]According to http://www.xml.com/pub/a/98/06/rdf.html

(Unique Resource Name) and Universal Resource Locators (URLs). The latter is used for both specifying the identity and location of a resource. Life Science Identifiers (LSIDs) promise to uniquely and consistently identify data resources in life science, and provide a resolution protocol for the retrieval of that data and any associated metadata [61]. They have been initially developed under the umbrella of the I3C[16] and are now undergoing formal standardisation by OMG[17].

2.6.8 A Semantic Web Use Case for Biological Data Integration

The problem of interoperability between data sources is challenging for the following reasons (According to [59]):

1. The identification of interoperable data, let alone its functionality, is a difficult problem due to the lack of widely-accepted standards for describing the web sites and their contents.

2. Different resources make their data available in heterogeneous formats. Some data may be available in HTML format (readable by web browsers, but other data may only be made available in text (e.g. tab or comma-delimited) or binary formats (e.g. images). Such heterogeneity makes interoperability very, very difficult.

3. Data interoperability requires both syntactic and semantic translation. Both of these types of translation are hindered by the lack of standard data models, formats and ontologies/vocabulary.

YeastHub [59] demonstrates the use of RDF metadata/data standards and RDF-based technologies to facilitate integration of diverse types of genome data provided my multiple web resources in heterogeneous formats. Within YeastHub, a data warehouse has been constructed to store and query a variety of yeast genome data obtained from various sources. The application allows the user to register a dataset and convert it into RDF format. The datasets can then be queried. The goal of the application is to facilitate the publishing of life sciences data using the RDF format. The benefits of using the RDF format are [59]:

1. It standardises data representation, manipulation and integration using graph modelling methods.

2. It allows the exploration of RDF technologies and RDF query languages in order to integrate a wide variety of biological data.

3. It facilitates development and utilisation of standard ontologies to promote semantic interoperability between bioinformatic web services.

[16]Interoperable Informatics Infrastructure Consortium - http://www.i3c.org
[17]Object Management Group - http://www.omg.org

4. It fosters a fruitful collaboration between the artificial intelligence and life sciences research communities.

Below shows some of the feedback received for YeastHub.

> "Eagerly, I tested the data base but was a little let down: While I see the obvious benefits, many small problems appear, which taken together, make the system not really very helpful to the average bioinformatics user, let alone a biologist in the yeast community.
>
> The lack of descriptions of formats of the data sources makes it tough to create queries and I did not really get past trivial results. I also dearly missed capabilities to browse the data sets. However, it appears as if these technologies will become more and more important and the old tab separated tables hopefully disappear one fine day."[18]

2.6.9 Semantic Web Services and Processes

The main idea behind web services is to capture an organisation's functionality within an appropriate interface and advertise it online as a web service [56]. Web services promise universal interoperability and integration, but the key to achieving this relies on the efficiency of discovery of appropriate web services and composing them to build complex processes (*Semantic Web service integration*). This potential for businesses and organisations to be able to interact with each other on the fly is very appealing.

Within the web service domain, semantics can be classified into the following categories (According to [56]):

Functional Semantics: One of the most important aspects of Web services is their functionality. If the functionality is not correctly established, then the power of the Web service is neutralised. Web service functionality is established by the examination of the service's inputs and outputs [62]. This approach may not always provide an appropriate set of services. This can be remedied by having a specialised ontology for recognising functionality. Therefore, the intended functionality of each Web service can be represented as annotations using this ontology.

Data Semantics: Details of the input and output of a particular Web service are represented in the signature of the operations in a specification file. To effectively perform discovery of services, the semantics of the input/output data has to be taken into account. If the input/output data is annotated using an ontology, then the data semantics can be used in reference against the required data semantics.

QoS Semantics: After discovering the most appropriate Web service (or list of services), the next step is to select the most suitable service. Services are measured

[18] From "A Bioinformatics Blog" - http://binf.twoday.net/topics/Databases/

in terms of their QoS (Quality of Service) and this requires the management of QoS metrics for the services.

Execution Semantics: This encompasses the ideas of message sequence, conversation pattern of Web service execution, flow of actions, preconditions and the effects of Web service invocation. Using execution semantics can help in dynamically finding services that will match not only the functional requirements, but also the operational requirements (such as long-running interactions, complexity of operation).

These different types of semantics can be used to represent the capabilities, requirements, effects and execution of a Web service.

2.6.10 Distributed Computing

Distributed computing can be thought of as an environment where you can harness idle CPU cycles and storage space of tens, hundreds or even thousands of networked systems to work together on a particularly processing-intensive problem. The main goal of a distributed system is to connect users and resources in a transparent, open and scalable way.

Transparency: Any distributed system should hide its distributed nature from its users, appearing and functioning as a normal centralised system.

Openness: A property of distributed systems that measures the extent to which it offers a standardised interface that allows it to be extended and scaled. It is clear that a system that easily allows more computing entities to be plugged into it and more features to be easily added to it has an advantage over a perfectly closed and self-contained system.

Scalability: A scalable system is one that can easily be altered to accommodate changes in the amount of users, resources and computing entities affected to it.

2.6.11 Distributed Computing Architecture

There are various hardware and software architectures that exist for distributed computing. At a lower level, it is necessary to connect multiple CPUs with some sort of network, regardless of the nature of the network. At a higher level, it is necessary to interconnect processes running on different CPUs with some sort of communication system.

The following list describes the various types of architectures that can be employed for a distributed computing system [63].

Client-server: Smart client code contacts the server for data, then formats and displays it to the user. input at the client is committed back to the server when it represents a permanent change.

3-tier architecture: Three tier systems move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are 3-Tier.

N-tier architecture: N-Tier refers typically to web applications which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.

Tightly coupled (clustered): refers typically to a set of highly integrated machines that run the same process in parallel, subdividing the task in parts that are made individually by each one, and then put back together to make the final result.

Peer-to-peer: An architecture where there is no special machine or machines that provide a service or manage the network resources. Instead all responsibilities are uniformly divided among all machines, known as peers.

Service oriented: Where system is organized as a set of highly reusable services that could be offered through a standardized interfaces.

Mobile code: Based on the architecture principle of moving processing closest to source of data.

Replicated repository: Where repository is replicated amongst distributed system to support online/offline processing provided this lag in data update is acceptable.

2.6.12 Cluster Computing

A computer cluster is a group of loosely coupled computers that work together closely so that in many respects it can be viewed as though it were a single computer. Clusters are commonly (but not always) connected through fast local area networks. Clusters are usually deployed to improve speed and/or reliability over that provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or reliability [63].

A cluster is a type of parallel or distributed system that consists of a collection of interconnected whole computers and is used as a single, unified computing resource[64]. It may not be immediately clear that these definitions effectively separate clusters from other forms of aggregate computing. Cluster computing differs from parallel computing and SMPs (Symmetric Multi-Processor) computing in three principle areas [64]:

Scaling: To scale SMPs, you must do more than simply add processors. The entire system needs to be upgraded too. However, with clusters, you can simply add more computers as needed as each unit comes with its own complete system (memory, IO, etc.)

Availability: If any part of the SMP breaks, the entire SMP fails, no matter how many CPUs there are. However, clusters can (generally) survive the loss of single units as the processing load is simply transferred to the remaining working units.

System management: After you have programmed the processors to cooperate in a SMP, system management is generally simpler as you only have to do it once.

2.6.13 Grid Computing

Grid computing uses the resources of many separate computers connected by a network (usually the internet) to solve large-scale computation problems. Grid computing offers a model for solving massive computational problems by making use of the unused resources (CPU cycles and/or disk storage) of large numbers of disparate, often desktop, computers treated as a virtual cluster embedded in a distributed telecommunications infrastructure. Grid computing's focus on the ability to support computation across administrative domains sets it apart from traditional computer clusters or traditional distributed computing [63].

Grid computing has the design goal of solving problems too big for any single supercomputer, whilst retaining the flexibility to work on multiple smaller problems. Thus grid computing provides a multi-user environment. Its secondary aims are: better exploitation of the available computing power, and catering for the intermittent demands of large computational exercises.

This implies the use of secure authorization techniques to allow remote users to control computing resources.

Grid computing involves sharing heterogeneous resources (based on different platforms, hardware/software architectures, and computer languages), located in different places belonging to different administrative domains over a network using open standards. In short, it involves virtualising computing resources.

2.6.14 Functions of Middleware in Distributed Computing

Middleware consists of software agents acting as an intermediary between different application components. It is used most often to support complex, distributed applications. In a distributed computing system, middleware is defined as the software layer that lies between the operating system and the applications on each site of the system[19]. Applications use intermediate software that resides on top of the operating systems and communication protocols to perform the following functions:

1. Hiding distribution: the fact that an application is usually made up of many interconnected parts running in distributed locations.
2. Hiding the heterogeneity of the various hardware components, operating systems and communication protocols.

[19]Taken from ObjectWeb - http://middleware.objectweb.org.

3. Providing uniform, standard, high-level interfaces to the application developers and integrators, so that applications can be easily composed, reused, ported and made to interoperate.
4. Supplying a set of common services to perform various general purpose functions, in order to avoid duplicating efforts and to facilitate collaboration between applications.

These intermediate layers have come to be known under the generic name of middleware. The role of middleware is to make application development easier, by providing common programming abstractions, by masking the heterogeneity and the distribution of the underlying hardware and operating systems, and by hiding low-level programming details.

2.7 Data Provenance

The widespread nature of the Internet and the ease with which files and data can be copied and transformed has made it increasingly difficult to determine the origins of a piece of data. The term *data provenance* refers to the process of tracing and recording the origins of data and its movement between databases. With many kinds of data, provenance is not important. However, for scientists focussed on the *accuracy and timeliness* of the data, provenance is a big issue [65].

Provenance allows us to take a quantity of data and examine its *lineage*. Lineage shows each of the steps involved in sourcing, moving and processing the data [72]. In order to provide provenance, all datasets and their transformations must be recorded.

Frew and Bose [66] propose the following requirements for provenance collection:

1. A standard lineage representation is required so lineage can be communicated reliably between systems (currently there is no standard lineage format[20].
2. Automated lineage recording is essential since humans are unlikely to record all the necessary information manually.
3. Unobtrusive information collecting is desirable so that current working practices are not disrupted.

2.7.1 The Reasons for Provenance

Scientists are often interested in provenance because it allows them to view data in a derived view and make observations about its quality and reliability [67]. Goble [68] presents some notable uses for provenance:

[20]At least at the time of the publication of [66].

Reliability and quality: Given a derived dataset, we are able to cite its lineage and therefore measure its credibility. This is particularly important for data produced in scientific information systems.

Justification and audit: Provenance can be used to give a historical account of when and how data has been produced. In some situations, it will also show why certain derivations have been made.

Re-usability, reproducibility and repeatability: A provenance record not only shows how data has been produced, it provides all the necessary information to reproduce the results. In some cases the distinction between repeatability and reproducibility must be made. In scientific experiments results may be different due to observational error or processing may rely on external and volatile resources.

Change and evolution: Audit trails support the implementation of change management.

Ownership, security, credit and copyright: Provenance provides a trusted source from which we can procure who the information belongs to and precisely when and how it was created.

Zhao et al. [75] describes three further purposes for provenance[21] from the viewpoint of the scientist:

1. *Debugging.* Experiments may not produce the desired results. The scientist requires a log of events recording what services were accessed and with which data.
2. *Validity Checking.* If the scientist is presented with a novel result, he/she may wish to perform expensive laboratory-based experiments based on these results. Although sure that the workflow design is valid, she may still want to check how this data has been derived to ensure it is worthy of further investigation.
3. *Updating.* If a service or dataset used in the production of a result has changed, the scientist will need to know what implications that change has on those results.

2.7.2 Data Provenance for the Life Sciences

Among the sciences, the field of molecular biology has generated a wealth of biological data and is arguably one of the most sophisticated consumers of modern database technology [69]. The field of molecular biology supports many hundreds of public databases, but only a handful of these can be considered to contain ”source” data in that they receive experimental data. Many of the databases actually reference themselves. This sounds contradictory until you take into account that much of the value associated to a data source comes from the curation and annotation (conducted by experts) of the data.

[21] These are not entirely exclusive to those described by Goble [68].

Most implementers and curators of scientific databases would like to record provenance, but current database technology does not provide much help in this process as databases are typically rigid structures and do not allow the kinds of *ad hoc* annotations that are often needed to record provenance [65].

There have been some attempts to formally monitor data provenance. *Query inversion* is a process that attempts to establish the origin of data by inverting the query on the output tuple. Even for this basic operation, the formalisation of the notion of data provenance is a non-trivial problem [65]. [70] presents a data model where the location of any piece of data can be uniquely described as a path.

One of the most significant problems associated with data provenance can be seen with the following example[22]; Suppose A cites a (component of a) document B. Suppose the owner of B wishes to update it, thereby invalidating the citation in A. Whose responsibility is it to maintain the integrity of B? This is a common problem in scientific (in particular, biological) databases. The usual procedure is to release successive versions of a database as separate documents.

2.7.3 Data Provenance and Workflow Enactment for the Web and Grid

It is not only the biological science domain that is concerned with data provenance. Large-scale, dynamic and open environments such as the Grid and web services build upon existing computing infrastructures to supply dependable and consistent large-scale computational systems [71]. With both scientific experiments and business transactions, the notion of lineage and dataset derivation is of paramount importance since without it, information is potentially worthless.

[71] proposes an infrastructure level support for a provenance recording capability for service-oriented architectures such as the Grid and web services. Interestingly, [71] also proposes a method that uses provenance for determining whether previously computed results are still up to date.

A provenance capability in a Grid or web services environment has two principal functions: to record provenance on dataset transformations executed, e.g. during workflow enactment, and to expose this provenance data in a consistent and logical format via a query interface.

Provenance needs to be stored and [71] envisages two possible solutions for storage.

1. Data provenance is held alongside the data as metadata.
2. Data provenance can be stored in a dedicated repository, made accessible as a Grid or web service.

[22]Taken from [65].

The first solution requires the holders of any such data to maintain the integrity of the provenance records as transformations take place, and it imposes significant changes to any existing data storage structures. Such a kind of provenance can be useful for a user to remember how a result was derived, and what steps were involved. It is unclear that such provenance can be trusted by any third party, since the data and provenance owner has to be trusted to have recorded provenance properly.

Workflow enactment is the automation of a process during which documents, information or tasks are passed from one participant to another for action, according to a set of declarative or procedural rules [71]. In Grid applications, this task is often performed by a "workflow enactment engine", which uses a workflow script, such as WSFL or BPEL4WS, to determine which services to call, the order to execute them in and how to pass datasets between them.

2.7.4 The ^{my}Grid Project

^{my}Grid, as a pilot e-science project, aims to provide middleware services not only to automate the execution of *in silico* experiments as workflows in a Grid environment, but also to manage and use results from experiments [73]. The ^{my}Grid project is currently being developed using several molecular biological scenarios.

^{my}Grid is being used to automate complex and tedious *in silico* experimentation[23] by wrapping each web-based analysis tool and data resource as a web service. Within this process, often discarded intermediate results are assigned identifiers and published locally, so that they become resources of the personal web of science [74].

If provenance information is to be shared within ^{my}Grid, we need to overcome their heterogeneity and agree a common understanding (or semantics) as to what the contents of each data item and service represents and the relationships we provide between resources [75].

The following figure from Zhao [75] shows the architecture of the provenance providing components in ^{my}Grid and how provenance data are drawn together.

> From Zhao [75]:
>
> First the user starts a workflow using the Taverna workflow workbench[24]. The workbench holds organisation in- formation such as user identity, which is passed to the workflow enactor together with input data and workflow specification. As the workflow is run, the enactor stores data (using the mySQL

[23]Often, the mundane nature of the task means not all possible avenues are explored, because the scientist either misses possible routes or discards apparently uninteresting results.

[24]Taverna and FreeFluo are both open source projects available from http://taverna. sourceforge.net and http://freefluo.sourceforge.net

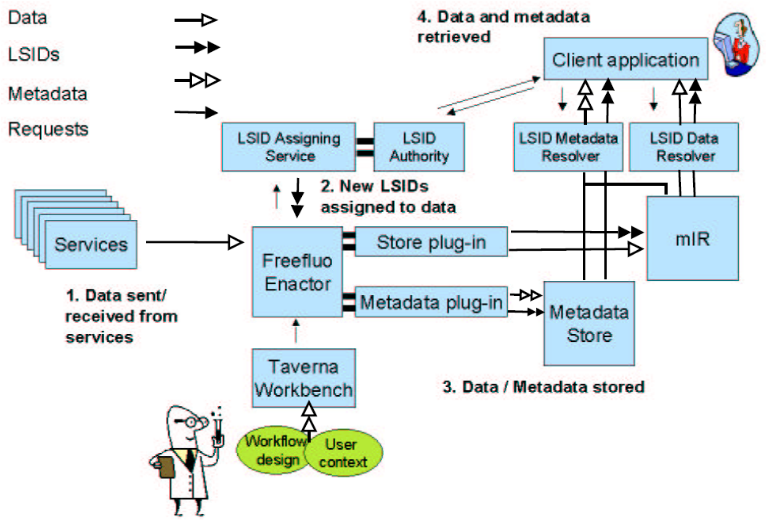


Figure 3: Architecture for Provenance Generation and Visualisation in ^{my}Grid.

RDBMS), and metadata (in a Jena RDF repository)[25] corresponding to our four views of provenance. Each resource (including person, organisation, data and service) is assigned an LSID and made available via an LSID authority implemented using an open source framework[26]. Client applications can then visualize both metadata and data using the LSID protocol.

2.7.5 The Taverna e-Science Workbench

This subsection is taken from the ^{my}Grid user guide (bundled with the ^{my}Grid download) and provides a nice description of the Taverna workbench.

The main user interface to myGrid is the Taverna e-science workbench. Taverna provides a language and software tools to enable the design and implementation of workflows. In a bioinformatics context, a workflow is the entire process of collecting relevant data, performing any number of analyses over the data and extracting biological results. Often, in bioinformatics the result of one experiment can form the input values of the next. Designing workflows in Taverna allows services (or bioinformatics analyses) to be scheduled to run in series. If results are not dependent upon one another, services can be designed to run concurrently, allowing for faster, more efficient processing.

[25] Jena is an open source semantic web framework for Java including an RDF repository http://jena.sourceforge.net/

[26] A LSID Java server stack available for download at: http://www-124.ibm.com/ developerworks/projects/lsid

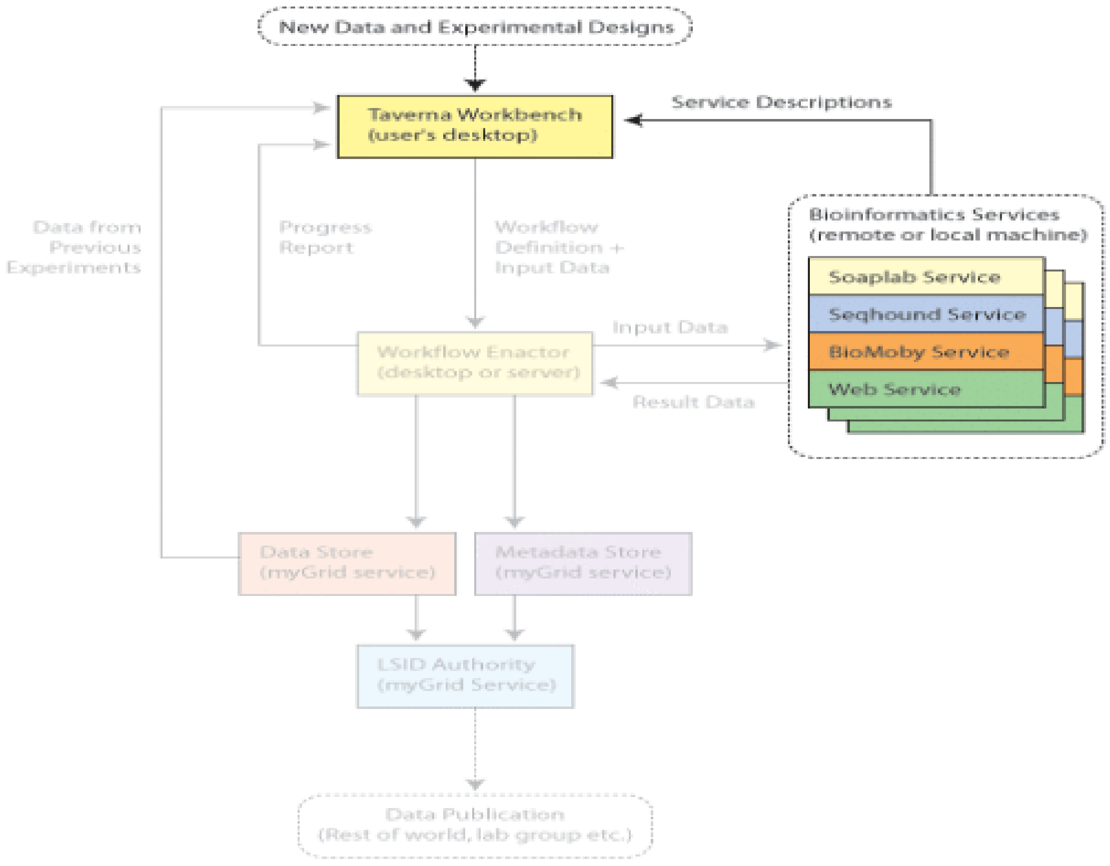

New Data and Experimental Designs
Taverna Workbench (user's desktop)
Service Descriptions
Bioinformatics Services (remote or local machine)
Soaplab Service
Seqhound Service
BioMoby Service
Web Service
Data from Previous Experiments
Progress Report
Workflow Definition + Input Data
Input Data
Result Data
Workflow Enactor (desktop or server)
Data Store (myGrid service)
Metadata Store (myGrid service)
LSID Authority (myGrid Service)
Data Publication (Rest of world, lab group etc.)

Figure 4: Service selections from remote and local machines

Workflows are designed in Taverna by selecting services and inserting them into a workflow diagram. Services are any bioinformatics applications that are available as web services. More and more applications are being provided as web services, but 'legacy' applications, either command-line tools, or simple web interfaces, can also be transformed into web services using wrapping tools, such as Soaplab and Gowlab respectively. This functionality means that any bioinformatics experiment that can be conducted elsewhere, regardless of the number of different processes or data sources required, can be run as a workflow in Taverna.

The following sections are pending. They have been researched but not yet been included in the review.

2.8 Data Provenance Tools

Data provenance is receiving an increasing amount of attention and this chapter describes some of the tools that are currently available. Chimera [77] is a virtual data system for representing, querying and automating data derivation. ESSW [78] is a nonintrusive data management infrastructure to record workflow and data lineage for computational experiments. Tioga is a proposal to modify an existing DB visualiser built over POSTGRES, where user functions are registered and executed by DBMS, to provide fine grained lineage: lineage is computed from ancillary, user supplied weak inversion and verification functions. CMCS [79, 80] is an informatics-based approach to synthesising multi-scale chemistry information that uses WebDAV. MyGrid [73]is a high-level service-based middleware to support the construction, management and sharing of data-intensive in silico experiments in biology. Trio [81] is a system for the integrated management of data, accuracy and lineage. With Trio, database views (modelled as query trees) can be supplied with inversion queries in order to determine the source tables used for that derived data.

2.9 Hashing

This chapter will cover the various aspects of cryptographic hashing. We will describe the problems that hashing is designed to address, a description and survey of the hashing techniques currently available and suitable for our purpose and a discussion of their respective limitations.

3 Conclusions from the Literature Review

Biological data has three principal (unfavourable) characteristics; high volume, disparate heterogeneous sources, and the complexity of the data semantics. Although the first two have received considerable attention [10, 12, 17], comparatively little effort has been made in managing the semantic complexity and stability. There have been various attempts at investigating the versioning of scientific data [43, 44] but these have, so far, failed to provide an acceptable framework[27].

It is clear from the literature that the constant unannounced changes to biological data represent a significant problem and whilst there are some tools proposed to address this issue, they have only been addressed in part. It is also clear that establishing the provenance or lineage of a set of results is an important aspect of successful science. However most methods for recording provenance rely on having a set of pre-defined, explicit processes that can occur on the data. With particular respect to workflow

[27]For more information, please refer to the Literature Review document which is available on request.

enactment, there is very limited flexibility in what can be done with the data if it is to be recorded accurately. Most tools for workflow enactment rely on a set of application-specific functions for the data within a bespoke database. For a more detailed survey of current provenance-based tools, please refer to [82].

4 Proposed Contribution

We have already made clear in the previous section that continuing unannounced changes to biological data represents a significant problem. Indeed, we have a real world example where such changes can and do occur and the problems arising can be observed. The main contribution of this EngD is to provide a framework for managing changes that can occur during biological data experimentation. It is not only the source data that may change, but the experimental method and parameters, preprocessing techniques, system configuration parameters, data access protocols. Obviously, some changes will have more significant impacts than others but all the changes have the ability to affect the results in some way.

By means of a test case example of an experimentation environment, it is the aim of this EngD to identify a set of possible changes that can occur during experimentation along with an analysis of the impacts of these changes. From here, we can establish the requirements for a framework for managing experimental change. We can then design and build the framework to satisfy these requirements.

We can then validate the framework by applying it to the original example case. This provides us with the opportunity to root the framework within a real-world example in order to identify the successes and shortcomings. There may then be a further opportunity to apply the framework to other experimental environments which may, themselves, use other types of biological data.

5 Scope of the Thesis

Aside from the specification of the problem and a review of the relevant literature, the first part of the thesis will contain a detailed description and analysis of the test case. It is important to characterise the test case as a genuinely applicable real-world example. It is not sufficient to have an example from which to base the framework unless it is clear that the example is not a "one-off" but is representative of a wider set of experimental environments.

Figure five illustrates the initial basic stages of the research plan. The initial phase is to select a suitable working example from which to base the framework. At this point, it is

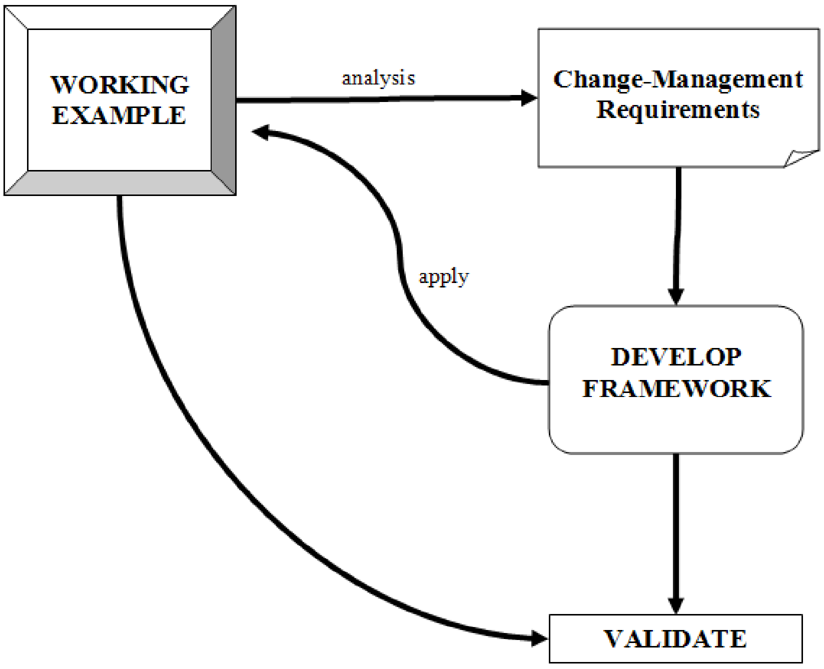


Figure 5: Basic stages for the thesis

important to provide an adequate analysis of the example in order to extract the requirements from which to build the framework. Given that the purpose of the framework is to deal with changes to the experimental environment, the analysis should reflect this requirement. We propose a detailed schematic analysis of the example, followed by an investigation into the changeability of the environment and an impact analysis, detailing the impacts that results from the identified changes.

Once the changes and their impacts have been identified and adequately investigated, we can begin to develop the framework for managing those changes. Each change identified during the requirements analysis (change analysis) will provide a different set of problems to address. Broadly speaking, the framework will bundle together the tools that address these different problems.

The next stage is to introduce and integrate the framework into the real-world example from which the framework was based. Following a successful integration of the framework, we will conduct a comparative evaluation of the new frameworked test case and an analysis of the changes exhibited by the revised system. We can then compare the impact analyses for the two systems (pre- and post- framework versions) in order to validate the framework for further use. At this point, we can investigate the possibility of using the framework for other experimentation environments. If the framework is to be widely accepted, we must demonstrate its suitability or potential applicability for a number of varying scientific environments. This study of suitability shall form the concluding section of the thesis.

6 Research Progress

The main aim of this report is to focus on the research that has been completed so far and to highlight the areas that require further work in order to complete the contribution described in section 4. Section 6, which constitutes the bulk of this report, concentrates on addressing these points and the following sections detail the work that has been conducted so far during the EngD programme. Section 6.1 describes the example dataset that will be used in the example described in section 6.2. Sections 6.3 and 6.4 provide details of the change and impact analyses that were conducted on the example. Section 6.5 provides a description of the basic building blocks of a data change management framework and section 6.6 provides a first look at the implementation of such a framework. Section 6.7 describes the hashed data model (HDM) and section 6.8 describes an example of the HDM operation. Section 6.9 describes some of the issues surrounding provenance and the use of provenance logs and section 6.10 describes the work to be done in order to complete the EngD contribution.

6.1 The Example Dataset

The dataset that is to be initially used is a set of proteomic fingerprinting data generated at St. George's Hospital[28] and the NIMR. The original format for the data were Excel flat files. These were then edited in Excel to produce CSV (Comma-Separated-Variable) files to be used with ObjectDB. These CSV files were then tokenized and marshalled to create JDOs (Java Data Objects). ObjectDB is a powerful Object Database Management System (ODBMS) written entirely in Java.

6.1.1 Proteomic Data

Proteomics is the analysis of complete complements of proteins. It is concerned not only with the identification and quantification of proteins, but also the determination of their localisation, modifications, interactions, activities and function. Proteomics is a useful tool for the analysis of various diseases. A disease can arise; when a protein (or gene) is over or under-expressed, when a mutation in a gene results in a malformed protein, or when a protein's function is altered through post-translational modifications. Therefore, in order to understand the biological process and to aid disease diagnosis, the relevant proteins can be studied directly.

[28]Refer to http://www.st-georges.org.uk/

There are several things to look at when considering the validity of proteomic data. Proteomic data, like data obtained from almost any biological experiment, is subject to any number of external factors. A common technique for improving the reliability of a set of data (or to demonstrate the validity by reducing the effects of any external variables) is to repeat the experiment. How many times an experiment has been repeated lends weight to the validity of that resulting data. It is also important to look at the approach taken for protein quantitation and explain how this approach may impact the interpretation of any results obtained.

6.1.2 The Data Structure

People were selected to form part of a clinical study at St. George's hospital in order to diagnose patients with tuberculosis. The dataset consists of data from 349 patients, each with a set of 219 protein quantitations. Each patient contains 56 additional data descriptors. These are illustrated in figure 6.

From the dataset, *Patient*, other sub-datasets are created with the purpose of investigating various hypotheses. For example, tuberculosis and HIV are inextricably linked [15]. HIV progressively weakens the immune system, making people vulnerable to infections such as tuberculosis. Consider a dataset to investigate the correlation between HIV status and tuberculosis is produced from Patient with the following filter;

```
tb_study_label == "TB" && hiv_status == "Positive"
```

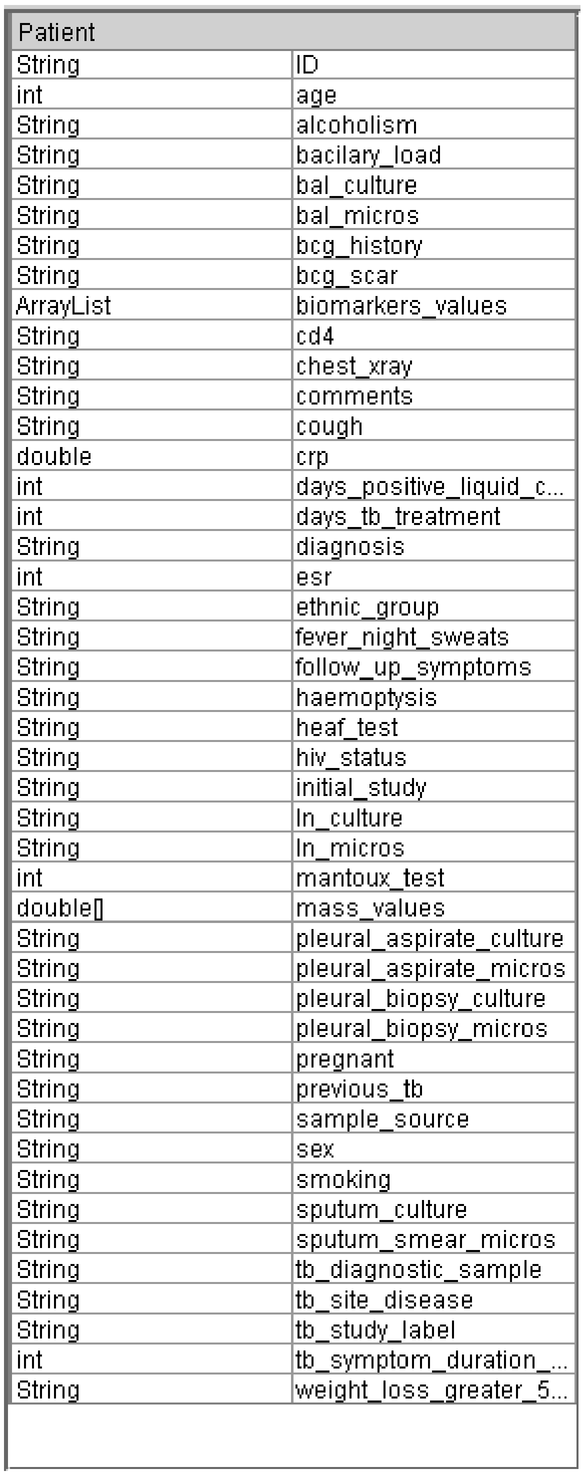
to find those patients with tuberculosis and who are HIV positive.

```
tb_study_label == "TB" && hiv_status == "Negative"
```

to find those patients with tuberculosis and who are not HIV positive. This data can then be tested and trained upon using any number of machine learning techniques to find correlations in the data.

6.1.3 Problems and Challenges with the Dataset

Many of the problems and challenges with this type of dataset have been discussed in section 2.3. The first thing to consider when looking at the example dataset above is the size of the dataset. Although the number of patients included in the dataset are not too large, the size and complexity of the derived collection of datasets grows considerably. Given the possibilities for derived datasets, it is easy to see how the size and complexity can get to an unmanageable point. The current example dataset has twelve sub-datasets. Figure 7 shows some of the derived datasets from *Patient*, but there are many more possibilities.



Patient	
String	ID
int	age
String	alcoholism
String	bacilary_load
String	bal_culture
String	bal_micros
String	bcg_history
String	bcg_scar
ArrayList	biomarkers_values
String	cd4
String	chest_xray
String	comments
String	cough
double	crp
int	days_positive_liquid_c...
int	days_tb_treatment
String	diagnosis
int	esr
String	ethnic_group
String	fever_night_sweats
String	follow_up_symptoms
String	haemoptysis
String	heaf_test
String	hiv_status
String	initial_study
String	ln_culture
String	ln_micros
int	mantoux_test
double[]	mass_values
String	pleural_aspirate_culture
String	pleural_aspirate_micros
String	pleural_biopsy_culture
String	pleural_biopsy_micros
String	pregnant
String	previous_tb
String	sample_source
String	sex
String	smoking
String	sputum_culture
String	sputum_smear_micros
String	tb_diagnostic_sample
String	tb_site_disease
String	tb_study_label
int	tb_symptom_duration_...
String	weight_loss_greater_5...

Figure 6: Data Descriptors for Patient

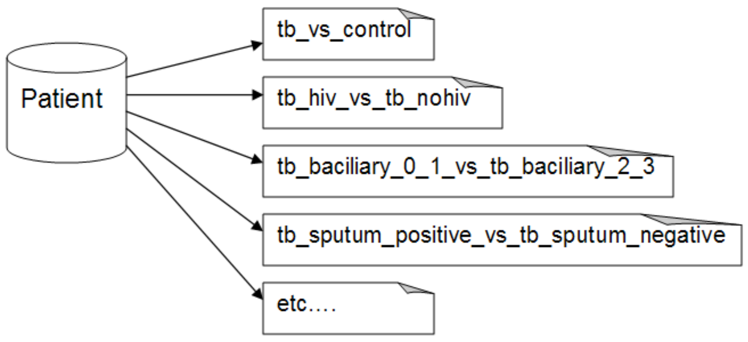

Patient
tb_vs_control
tb_hiv_vs_tb_nohiv
tb_baciliary_0_1_vs_tb_baciliary_2_3
tb_sputum_positive_vs_tb_sputum_negative
etc....

Figure 7: Some of the derived datasets from *Patient*.

The example dataset exhibits an above average level of complexity due to the dependencies on both clinical and molecular data. These types of data are significantly different in that they focus on different levels of detail of the processes occurring within the organism. Nevertheless, the level of complexity of the example dataset is representative of the level of complexity generally found in proteomic datasets.

The semantics of the data are more difficult to identify. There is less metadata than one might expect of a dataset of this size and type. There is little to suggest how the data was gathered[29]. Data from various experimental techniques are present in the dataset. These include protein quantitations, chest x-rays, baciliary loads, biopsies and FNAs (Fine Needle Aspirations). However, there is no associated metadata that describes any of the details of these tests. It is therefore difficult to assess the error margins or validity of the data obtained. However, further metadata is generated as more and more datasets are generated and experimented upon. For example, taking the case of the tuberculosis-HIV correlation dataset, the following metadata is recorded; the algorithm, algorithm parameters (hypothesis), source and destination datasets, time and date, selected features, feature selection method, kernel, kernel parameters, model criteria and validation method. For this particular dataset, there are 18 classification experiments, each with differing data and metadata. This demonstrates how the amount of data and associated metadata can easily multiply at a rapid pace. How does an experimenter keep track of these multiple versions of experiments?

6.2 Analysis of Current Working Example

The aim of this section is to provide an analysis of an existing biological data process. The process uses data from a private collaborative database consisting of clinical and proteomic data from 350 patients. The data is mined using various machine learning

[29]The clinical data for the dataset was actually collected using a collection of surveying and interviewing techniques.

techniques in order to derive patterns and relationships from within the data.

We start with an overview of the example which has been illustrated as a non-structured rich picture in figure 8. This is the first step for the analysis and is used to identify the areas of interest and sub-systems present in the working example. Some of the parts illustrated are not within the machine domain of the example but do, however, affect the operation of the system[30].

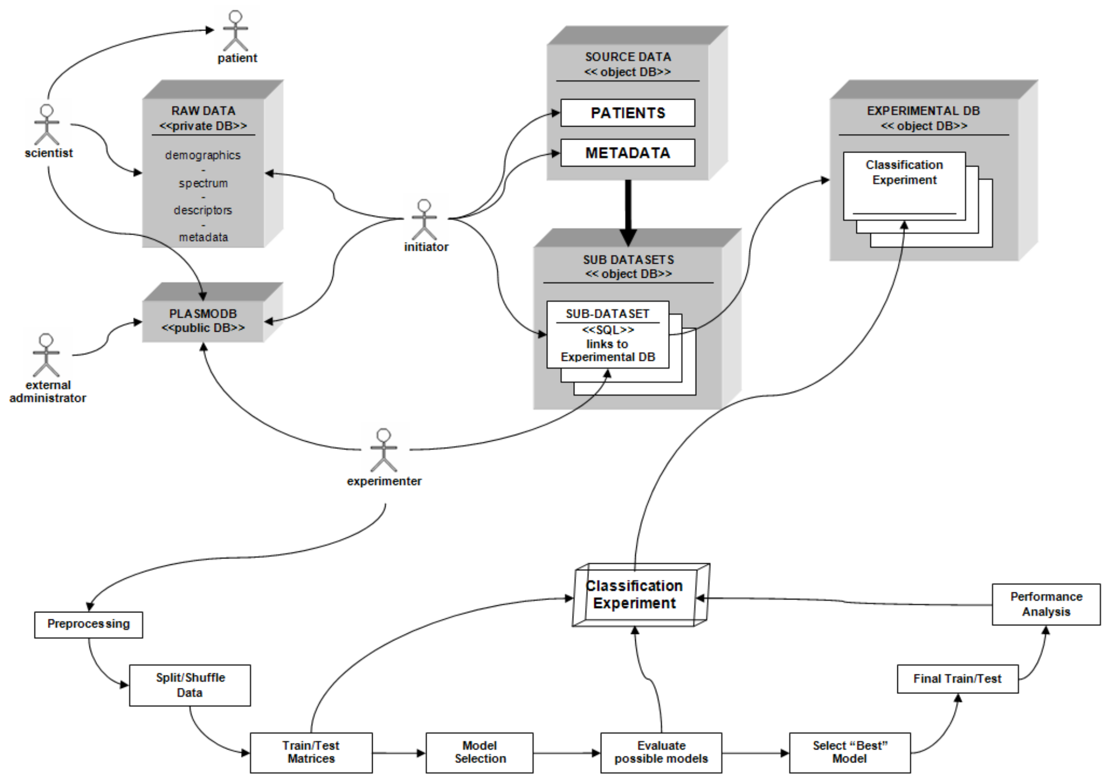


Figure 8: An overview of the working example.

From figure 8, we can identify five actors that interact with the system either directly or indirectly. The *scientist* maintains the private collaborative database which he/she populates with data from the *patient*. The *scientist* may also publish and submit data to a publicly available database, which is maintained by the *external administrator*, or they may use such a database to validate values from their own data. The *initiator* uses the private database and, possibly, the public database to initiate the bespoke database within the experimentation environment[31]. Once the relevant datasets have been initialised in preparation for experimentation, the *experimenter* can use one of the datasets

[30]Please refer to the change and impact analyses in sections 6.2 and 6.3.

[31]In this case, ObjectDB is used to store and maintain the initial data and any subsequent sub-datasets. For more information, see http://www.objectdb.com/

in an experiment. The precise nature of the experiment depends on the hypothesis, method and parameters employed by the *experimenter*. Some of the generic stages are illustrated in figure 8. The experiment is run and a *ClassificationExperiment* object is created and populated with, not only the results of the experiment, but the model that has been used to attain the results, the method used and other parameters specific to the experiment. The *ClassificationExperiment* object is maintained within an Experimental database, which itself is linked to the previously-initialised sub-dataset.

From figure 8, we can identify the important aspects of the system and choose the precise areas of the example to analyse. The analysis of the working example consists of a set of descriptive schematics detailing the various parts of the system. Figure 9 provides a map of the schematics that have been produced for the analysis.

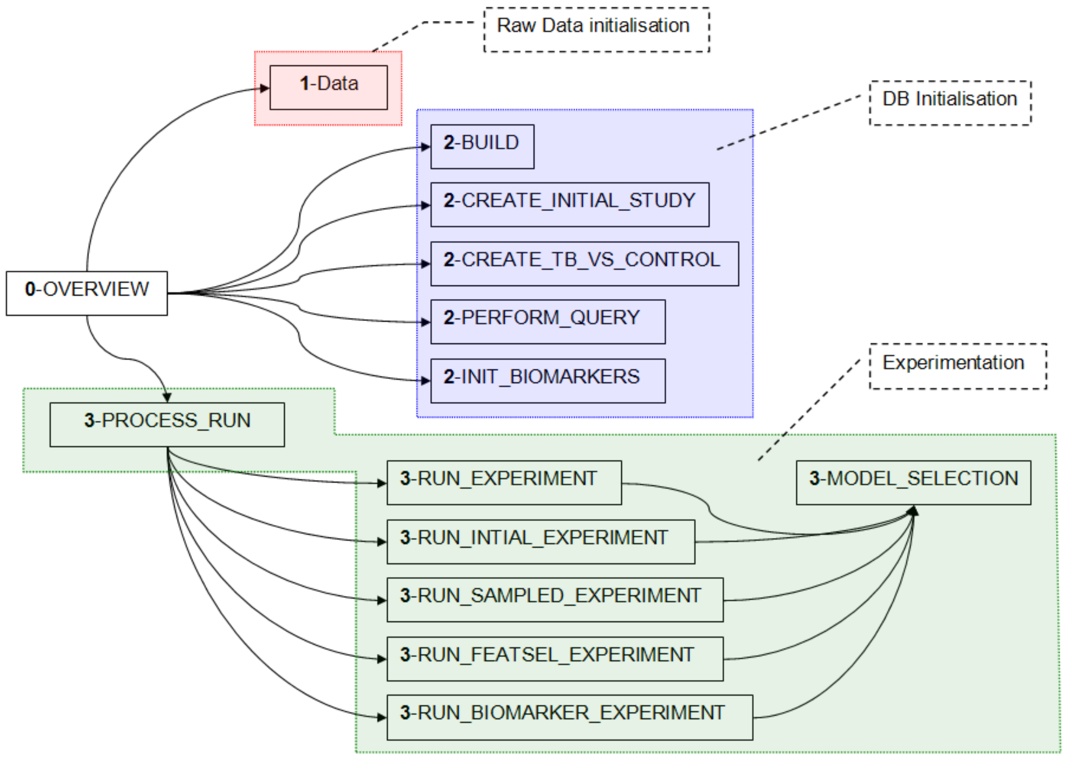
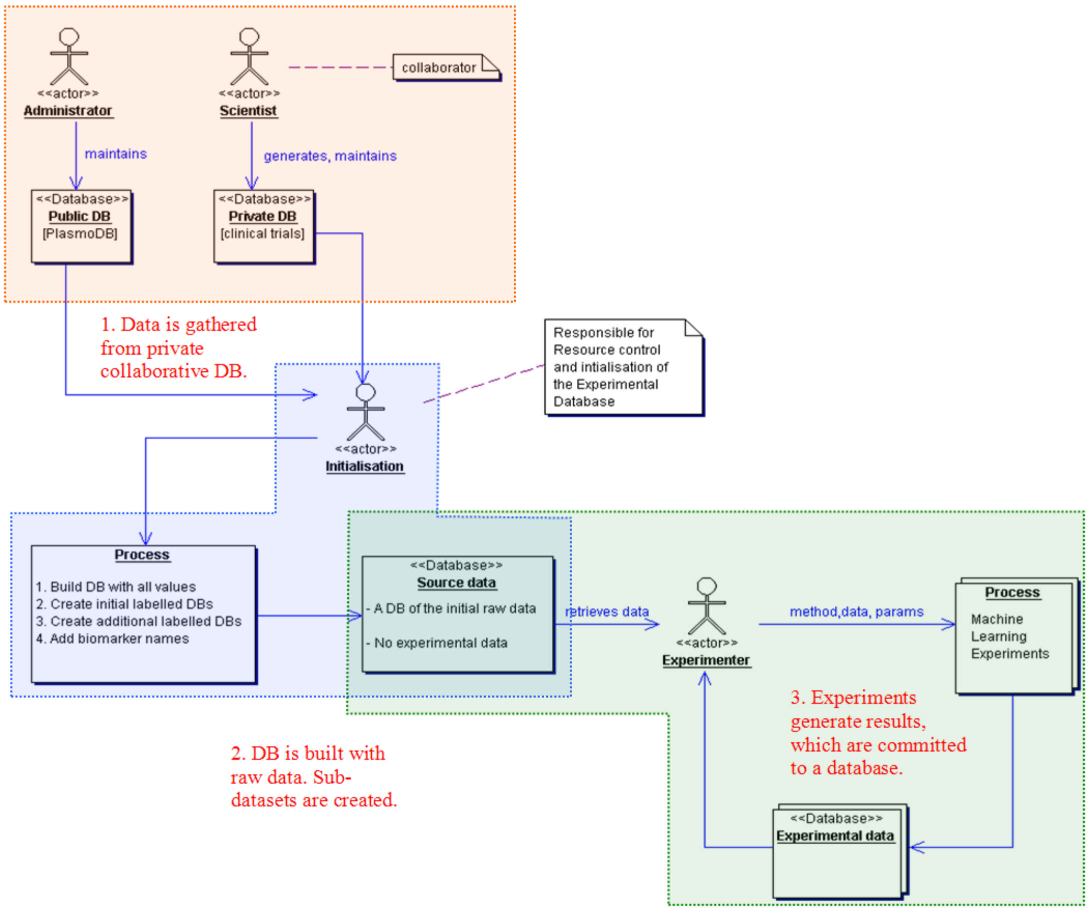


Figure 9: A map of the schematics that make up the analysis of the example.

As can be seen from figure 9, there are three constituent parts to the example; raw data initialisation, database initialisation and experimentation. It is not necessary to describe the schematics in the detail that is present in the full report (available on request) but some of the analysis will be described so that the reader can achieve a rough idea of the working example and the level of work involved in the analysis.

The following diagram (0-OVERVIEW from figure 9) is the revised overview for the structured schematics. This schematic forms the base from which all of the other schematics are drawn (refer to figure 9).

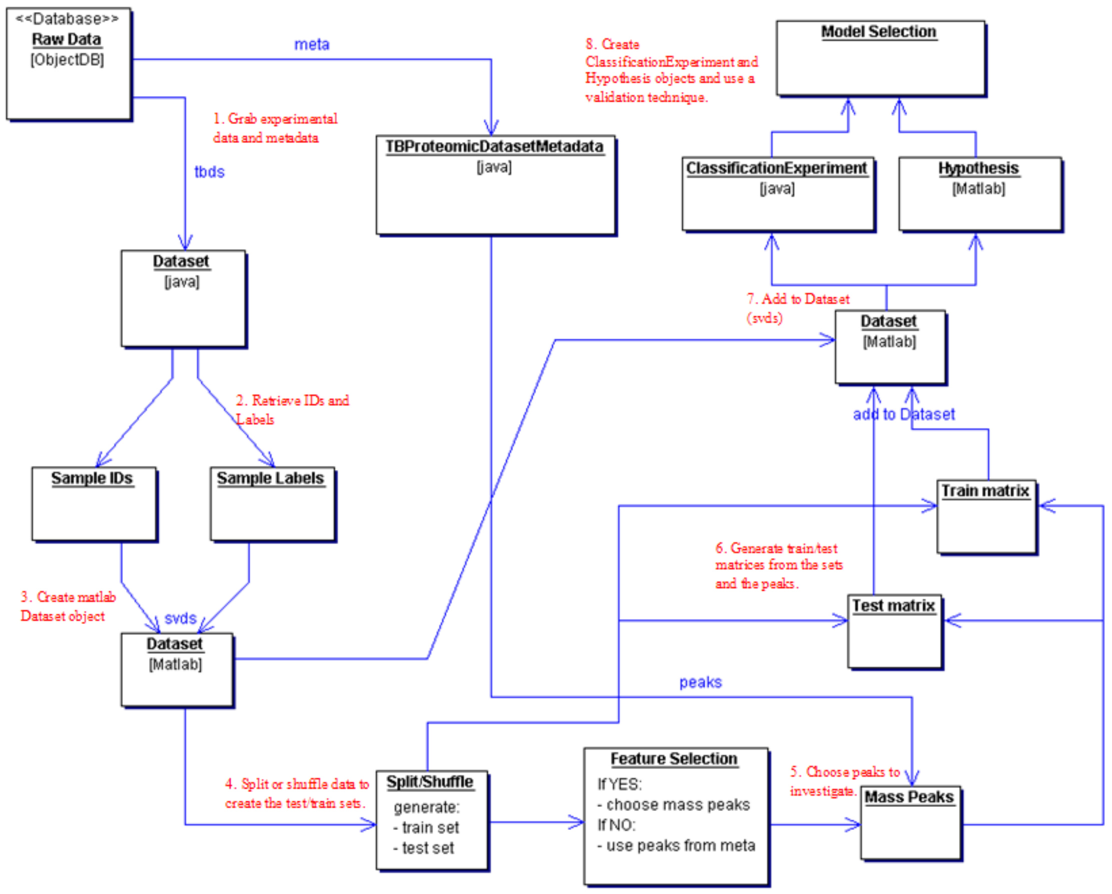


<<actor>> Administrator
<<actor>> Scientist
collaborator
maintains
generates, maintains
<<Database>> Public DB [PlasmoDB]
<<Database>> Private DB [clinical trials]
1. Data is gathered from private collaborative DB.
Responsible for Resource control and intialisation of the Experimental Database
<<actor>> Initialisation
Process
1. Build DB with all values
2. Create initial labelled DBs
3. Create additional labelled DBs
4. Add biomarker names
<<Database>> Source data
- A DB of the initial raw data
- No experimental data
retrieves data
<<actor>> Experimenter
method,data, params
Process
Machine Learning Experiments
3. Experiments generate results, which are committed to a database.
2. DB is built with raw data. Sub-datasets are created.
<<Database>> Experimental data

Figure 10: The revised overview.

This diagram illustrates the overview of the entire system from the collection and processing of raw data to the experimentation and generation of results. There are three basic stages to the system. The first stage is the sourcing of the data. This is accomplished by a collaborative scientist that generates and maintains the private, clinical data and an external administrator that administers the public database. The second stage is the generation and initialisation of the source database in preparation for experimentation. The third stage describes various machine learning experiments, the results yielded and the storage and administration of those results.

The test case analysis consists of a variety of collaboration diagrams which describe the various parts of the biological process. Interactions between the actors and the system are described and the principal parts of the system implementation are illustrated at a determined granularity. It is not feasible to include all of the schematics in this report. We have shown the overview above in figure 10, which provides a basic analysis of the example. Figure 11 describes the processes involved in the running of a basic experiment. It refers to the matlab function RUN_EXPERIMENT.



<<Database>> Raw Data [ObjectDB]
meta
tbds
1. Grab experimental data and metadata
TBProteomicDatasetMetadata [java]
8. Create ClassificationExperiment and Hypothesis objects and use a validation technique.
Model Selection
ClassificationExperiment [java]
Hypothesis [Matlab]
Dataset [java]
7. Add to Dataset (svds)
Dataset [Matlab]
2. Retrieve IDs and Labels
add to Dataset
Sample IDs
Sample Labels
Train matrix
6. Generate train/test matrices from the sets and the peaks.
3. Create matlab Dataset object
svds
Test matrix
Dataset [Matlab]
peaks
4. Split or shuffle data to create the test/train sets.
Split/Shuffle
generate:
- train set
- test set
Feature Selection
If YES:
- choose mass peaks
If NO:
- use peaks from meta
5. Choose peaks to investigate.
Mass Peaks

Figure 11: The processes involved in running a basic experiment.

The following is a description of the processes illustrated in figure 11. The relevant dataset (specified in the matlab function, RUN) is retrieved and split in sample IDs and labels. These are used to create a matlab object called *dataset*. The *dataset* is then split or shuffled according to pre-set variables to generate the train and test sets. If the experimenter has selected feature selection, the mass peaks of those features are selected, otherwise the mass peaks of all the features are selected from the metadata. These mass peaks, together with the train and test sets are used to generate train and test matrices, which are in turn added to the matlab dataset. This dataset object is used to create a *ClassificationExperiment* java object and a *hypothesis* matlab object. These are passed to a validation process.

The schematics and their descriptions are bundled in a full report and forms the basis for the change and impact analyses, which are covered in the next sections.

6.3 An Investigation into the Sources of Change

We are dealing with a system that exhibits a high degree of change over a variety of levels. There are multiple points of possible change and there can be various consequences as a results of those changes. Figure 12 provides a very simple view of a generic biological experimentation process and highlights the possible areas of change.

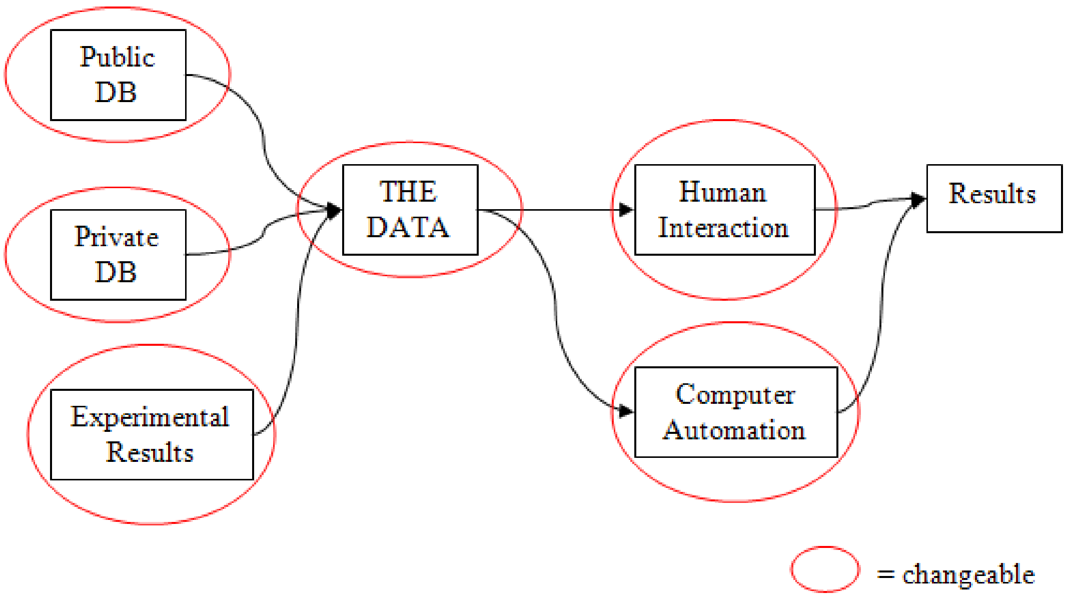


Figure 12: Overview of changes

Clearly, figure 12 does not provide the necessary detail from which to base an analysis of the system. It does, however, broadly highlight the areas that can change and provide a basis to complete the more detailed change analysis.

The change and impact analyses were based around the schematic analysis of the working example carried out and described in the previous section. The changes were identified using a working knowledge of the example and postulating the presence of possible changes within the system.

For the change analysis, we identify four types of change that occur within the example.

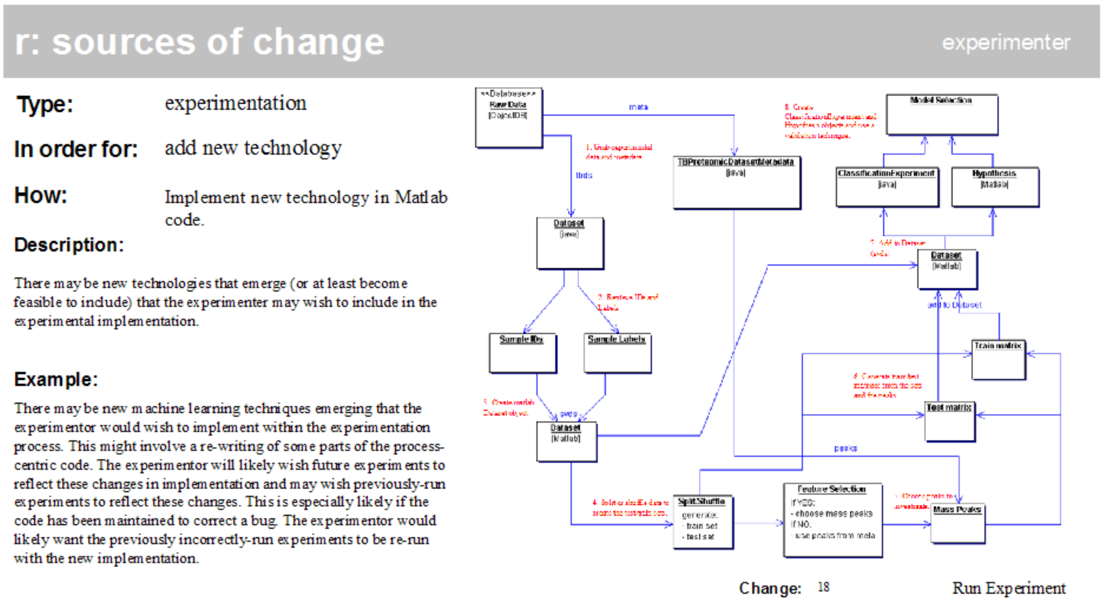
Content Refers to a change to the content of the data itself. This refers to the values of the specific data points.

Semantic This refers to a change that is semantic in origin such as the modification of data headings or the parameters of experimentation.

Protocol Refers to a change that occurs from the running and operation of the system. This is not a change to either the data or the implementation of that data. It refers to the operation of a part of the system that may have an effect on another part of that, or related, systems.

Experimentation Refers to a change that occurs as part of the experimentation process such as the addition of experimental technology or the addition or removal of an existing experiment.

The change analysis documents the points at which changes can occur within the system. It describes where the change occurs, the actor responsible for the change, the reason for the change and a real-world example of the change. The change number in the bottom right-hand corner corresponds to the impact number in the impact analysis. Figure 13 provides an extract from the change analysis report and an example of the type of changes that are recorded.



r: sources of change

experimenter

Type: experimentation

In order for: add new technology

How: Implement new technology in Matlab code.

Description:

There may be new technologies that emerge (or at least become feasible to include) that the experimenter may wish to include in the experimental implementation.

Example:

There may be new machine learning techniques emerging that the experimentor would wish to implement within the experimentation process. This might involve a re-writing of some parts of the process-centric code. The experimentor will likely wish future experiments to reflect these changes in implementation and may wish previously-run experiments to reflect these changes. This is especially likely if the code has been maintained to correct a bug. The experimentor would likely want the previously incorrectly-run experiments to be re-run with the new implementation.

Change: 18

Run Experiment

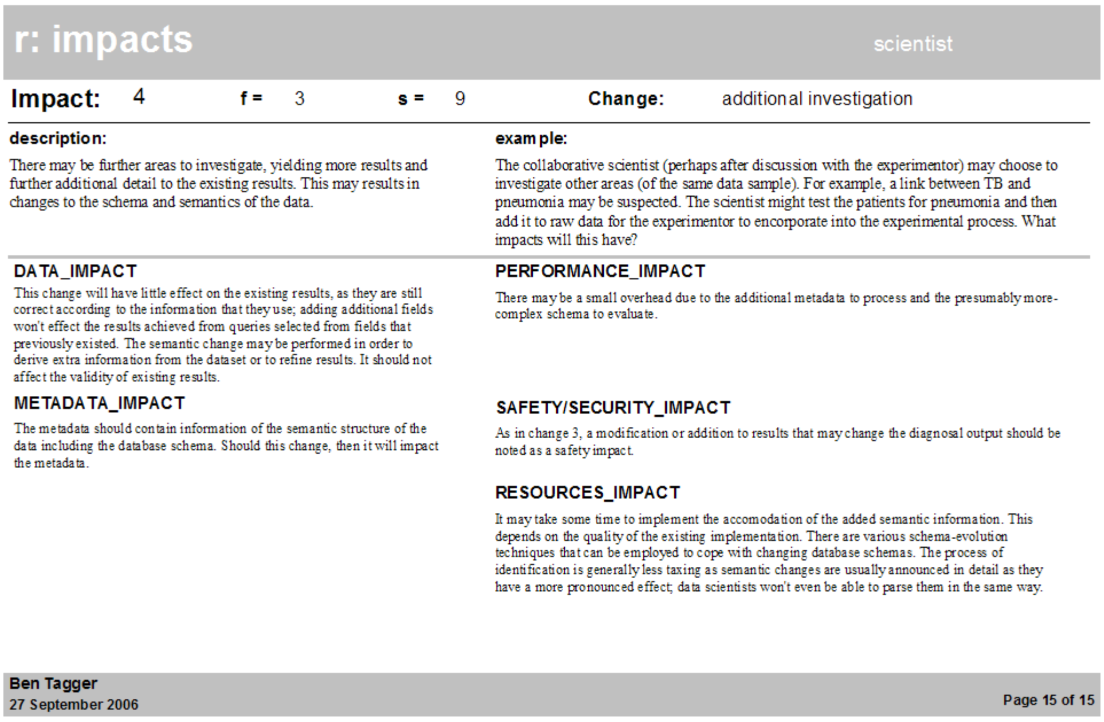
Figure 13: An extract from the change analysis report.

From figure 13, we describe the change that is occurring, an example of that change and illustrate the location of the change by providing the diagram of the part of the system where the change could occur. Incidentally, the diagram in figure 13 is the same schematic from figure 11.

The purpose of the change analysis was to investigate the areas of potential change and link those changes to a real-world example. It is not sufficient to merely claim that the changes occur and that we must deal with them. The changes must be identified, located, described and rooted within a real context. Only then can we begin to hypothesise solutions to the problems.

6.4 Impact Analysis

Having analysed the sources of possible change, we can look at each change and begin to discuss the impacts that the change may have on the system and its component parts. This analysis was completed using the analysis of the working example along with the change analysis to hypothesise the possible impacts that each change may have.



r: impacts

scientist

Impact: 4 **f =** 3 **s =** 9 **Change:** additional investigation

description:

There may be further areas to investigate, yielding more results and further additional detail to the existing results. This may results in changes to the schema and semantics of the data.

example:

The collaborative scientist (perhaps after discussion with the experimentor) may choose to investigate other areas (of the same data sample). For example, a link between TB and pneumonia may be suspected. The scientist might test the patients for pneumonia and then add it to raw data for the experimentor to encorporate into the experimental process. What impacts will this have?

DATA_IMPACT

This change will have little effect on the existing results, as they are still correct according to the information that they use; adding additional fields won't effect the results achieved from queries selected from fields that previously existed. The semantic change may be performed in order to derive extra information from the dataset or to refine results. It should not affect the validity of existing results.

METADATA_IMPACT

The metadata should contain information of the semantic structure of the data including the database schema. Should this change, then it will impact the metadata.

PERFORMANCE_IMPACT

There may be a small overhead due to the additional metadata to process and the presumably more-complex schema to evaluate.

SAFETY/SECURITY_IMPACT

As in change 3, a modification or addition to results that may change the diagnosal output should be noted as a safety impact.

RESOURCES_IMPACT

It may take some time to implement the accomodation of the added semantic information. This depends on the quality of the existing implementation. There are various schema-evolution techniques that can be employed to cope with changing database schemas. The process of identification is generally less taxing as semantic changes are usually announced in detail as they have a more pronounced effect; data scientists won't even be able to parse them in the same way.

Ben Tagger
27 September 2006

Page 15 of 15

Figure 14: An extract from the impact analysis.

The impact analysis examines each source of change in the change analysis and describes the impacts on the system and its actors as a result of those changes. The impacts are measured in five categories[32];

Data Impacts: The impact to the values and/or structure of the data.

Metadata Impacts: The impact to the system metadata.

Performance Impacts: The impact to the performance to the system after the change has occurred.

Safety/Security Impacts: The impact on safety or to any security policies as a result of the change.

Resource Impacts: The computational, financial or human cost incurred as a result of the change.

Figure 14 provides an extract from the impact analysis report. We can see the various parts of the impact analysis, how it is linked to the change analysis report and a description of the impact as separated by the aforementioned five categories.

[32]It is important to note that, unlike the previous reports, the impacts report is not limited by the limitations of the system that is being analysed. For example, a change that should impact the system but doesn't due to an implementation limitation is considered nevertheless.

6.5 Developing the Framework

The purpose of this section is to provide a rough outline of a possible data change management framework and a description of the components that might make up such a framework. We are trying to develop a framework for managing experimental environments with changing data. We assume that the data changes with sufficient frequency and significance to warrant such a framework.

Given that the data changes with a frequency and strength that causes a user to require management of those changes, we can conclude some requirements for the data-change management framework. Firstly, it is necessary to be able to detect whether a change has occurred or not within the dataset. This should not be a computationally expensive process. Secondly, we must be able to conclude the significance of the detected changes. Some changes will have greater impact than others and it is important to measure this impact in order to decide how to manage the change. Thirdly, we would like to know the location of the change. The location of the change can help by first, adding to the desiderata for the significance of the change and second, to provide provenance information for the dataset.

For the moment, this framework does not allow for an extreme, fine-grained change management system. It is possible to conceptualise a framework that records the specific details of each change in the data and allow a traceable versioning of the dataset in which every version can be reassembled. However, in my opinion this approach is not feasible.

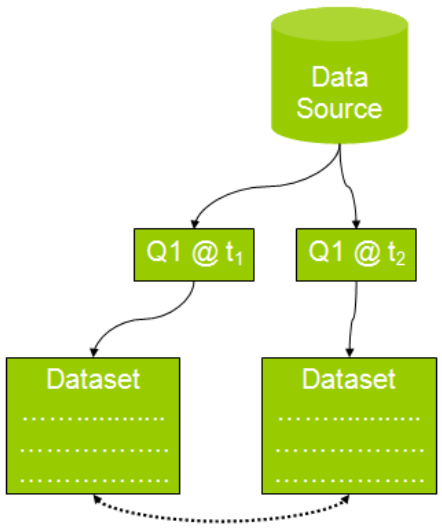


Figure 15: Query 1 at t1 and again at t2.

Figure 15 provides a simplified view of the process of querying some datasources to get the experimental datasets. We start with a dataset created by querying the data source at t1 with Query 1. At some later time, t2, we use the same query to create another dataset. We would like to know whether, during t1 and t2, the dataset has changed. This appears to be a fairly simple problem. However, when we consider that from a single dataset we may possibly have thousands of experiments, each conducted at a distinct time and, therefore, a distinct experimental dataset[33], we must face the possibility of managing many versions of the same dataset.

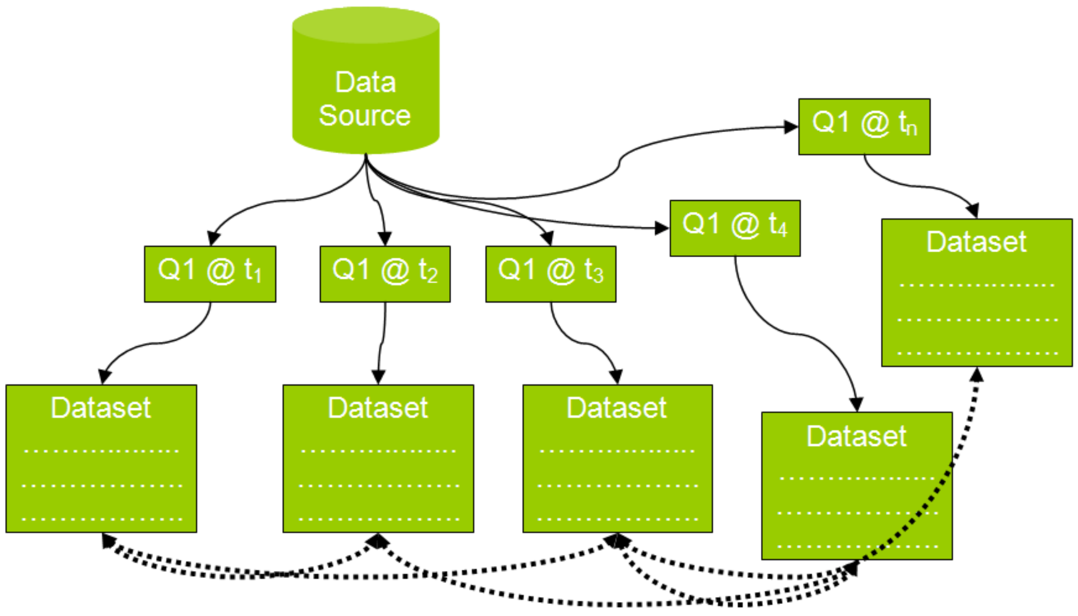


Figure 16: Query 1 over five time periods on the same datasource.

With regard to figure 16, there may be multiple experiments generating multiple Datasets over a range of time periods. This depends on the temporal granularity of the system. There is considerable overhead in keeping and controlling these datasets and analysing the changes between them.

In particular the storage overhead for n versions of a dataset that are x bytes in size can be represented as $x\ n$. The computational complexity for comparing n versions can be represented by the triangular number T_n[34], *where*;

[33]Note that we have already suggested that the data source changes over time.

[34]More formally, a triangular number is a number obtained by adding all positive integers less than or equal to a given positive integer n.

$$T_n = \sum_{k=1}^{n} k$$

$$= \frac{1}{2}n(n+1)$$

(an n^2*problem*)

Ideally, we would like to have some way of not storing all the data many times and still be able to fulfill the requirements already described.

6.6 A First Look at the Framework

We start by querying the source databases, which may possibly be a collection of databases, to form our experimental dataset. This is the data, from which the experiment or set of experiments will be executed. At the first instance before any experimentation has occurred, the experimental datasets may be created and stored ready for use[35]. However, it is important to note that the nature of the experimental datasets relate specifically to the experiments that are to use that dataset. Therefore, an experimental dataset should not exist if there are no corresponding experiments for that dataset or there shall be none in the future.

The query that generates the experimental dataset is experiment-specific[36] and the resultant dataset is created at time t1. The experimental dataset is returned to the experiment, the subsequent experiment is performed, and results are generated. Meanwhile, a hashed data model of the experimental dataset is generated at the time of experimentation. Also, provenance logs are generated that record the instantiation of the dataset. This may contain various information such as who created the dataset, when and why, etc.

When the results are generated, the hashed data model is stored with the experimental results as a record of the data upon which the results are based. The experimental dataset is discarded. Only the hashed values of the dataset and the provenance logs remain within the system.

Figure 16 illustrates the operation of updating the dataset depending on the changes that have occurred on a dataset over a given time period. As before, a dataset is created by Q1 (a query) at time t1. At a later time t2, the same query creates a new dataset. Given these two datasets, we would like to know whether there has been a change between t1 and t2. We can achieve this by examining the hashed data models of the two datasets. Depending on the outcome of this comparison, we can perform various

[35]In particular, this occurs in the case example.

[36]The details of the dataset are supplied by the experiment (in the metadata possibly).

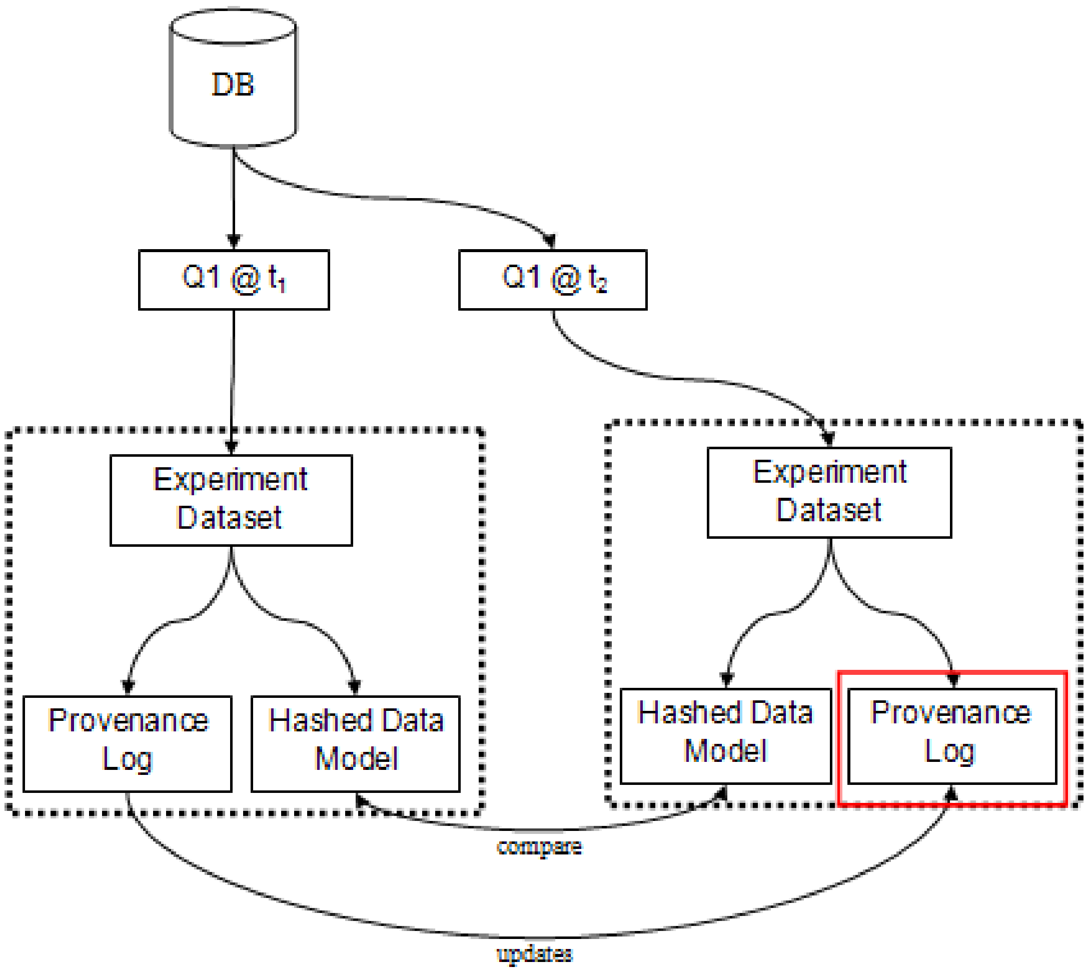


Figure 17: Comparing and updating datasets.

operations. If there is no change detected or there is a change but it is deemed to be of minimal significance (at some minimum threshold), then the new dataset (Q1 @ t2) can be discarded and the original experimental dataset can be used for the experiment.

If a significant change is detected, then the new dataset (Q1 @ t2) is used for the experiment and the provenance logs are updated with details of the changes to the dataset. This is of limited importance for new experiments that use the same experimental dataset; we might as well use the new dataset even if there is the slightest change since we have already expended the computational effort of querying the database. There is no saving to be made by using the original. The saving comes when evaluating an existing experiment. We may want to know whether we should re-run the experiment and update the results. In this case, we can decide to use the original despite a slight redundancy in the data to save running the experiment again.

The provenance logs contain the history of the data. By maintaining a record of the changes undergone by a particular dataset, we are keeping a record of all previous, significant versions of the dataset, albeit not on a data-specific granularity.

6.7 The Hashed Data Model (HDM)

In order to detect difference within the datasets, we must compare one dataset to another. Given the high number of experiments that can come from a single dataset, it is possible that we may have to consider many versions of the dataset at any given time. It is, therefore, not feasible to store a complete version of the dataset for each experiment; consider a dataset with 10,000 dependant experiments. Are we to store the precise details of the dataset used in each experiment? We would like to find an abstraction of each version of the dataset which can allow us to fulfill the requirements criteria outlined in section 6.4.

Hashes could be used to model the datasets. A hash is a way of creating a small digital "fingerprint" of some item of data. Good hash functions typically have two important characteristics. Firstly, The hash function should be *deterministic*. The same item of data (if unchanged) should always hash to the same value. Secondly, any change in the data will be very likely to produce a different hash value. You can therefore infer that if two hash values are different, it is very likely that the data that produced the hashes are also different. The ideal hash function has very low collisions (but we cannot remove these collisions altogether).

At the most basic level, if we applied a hash function to the entire dataset, we can then look for any changes to the dataset. Any change in the hash value will depict a change to the dataset. Even at this level, we have satisfied the first requirement (from section 6.4).

However, given the second property of hash functions (see above), we cannot imply the significance or size of the change. Therefore, we cannot satisfy the second requirement.

The problem lies with the nature of the hash functions, which have been primarily developed for security purposes. In the security domain, it is a positive aspect of a hash function that exhibits wildly different hashes for very similar (although slightly different) data items. This aims to reduce the feasibility of a comparative analysis attack on the hashed value. For our purpose, we would ideally like to have a comparative hash function. So, very different datasets hash to very different hash values but datasets that differ only slightly exhibit the same slight difference in their hashed values.

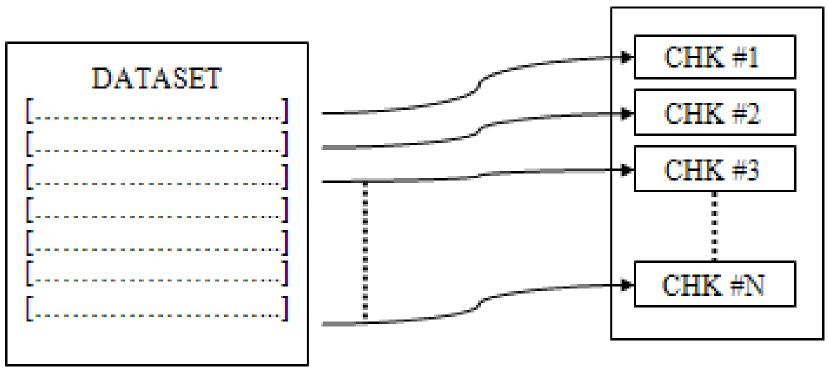
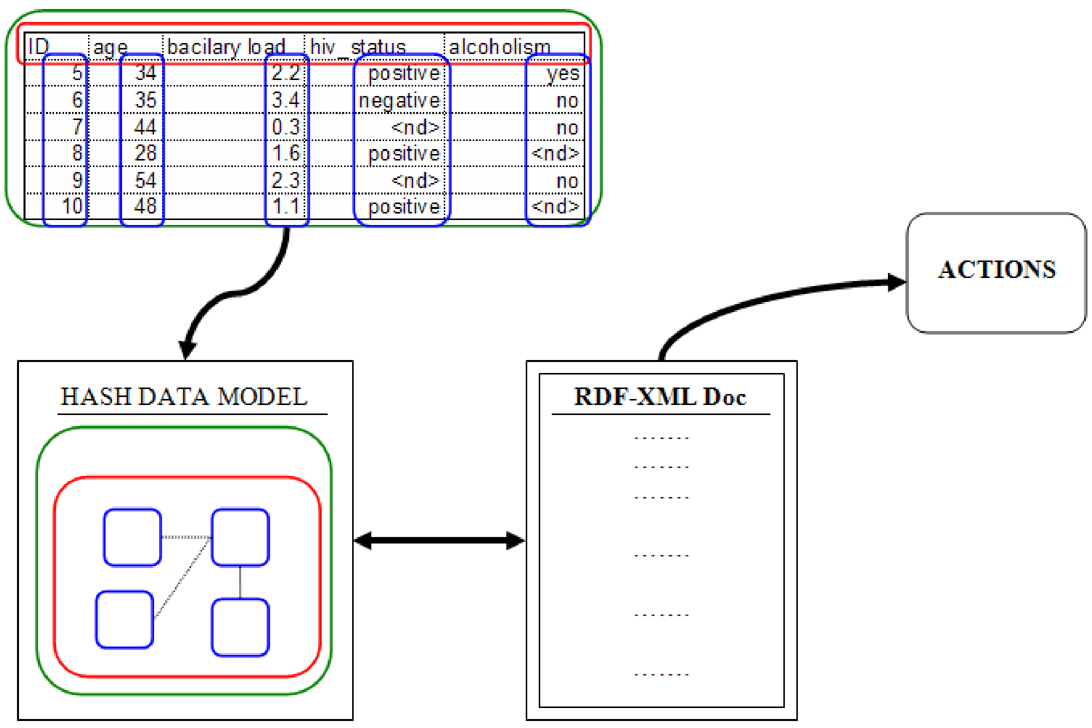


Figure 18: Splitting the data and applying a checksum.

One possible solution for a comparable hash is to split the data into determined data windows and applying a hash function to each portion of data. This functions much in the same way as a checksum operation. A checksum is a very simple measure for protecting the integrity of the data and detecting errors in a piece of data over time (or over a data stream traditionally). In this way, we could determine the approximate size of the changes as well as their approximate location. The benefits of this approach are determined by the implementation of the hash and the size of the data windows. If the windows are too large, we find the hashed value to be as defunct as before, detecting a change but giving no information as to its size or location. If the data windows are too small, we may find ourselves storing almost as many hashes as the original size of the dataset.

This methodology has other drawbacks. If we simply split the dataset into data windows of a set length (which is necessary in order to provide the comparative analysis later), we can make small changes that have large knock-on effects on the other parts of the checksum hashes. Consider the insertion of a single digit at some point in the dataset. This change renders the above methodology useless, as each subsequent data window following the insertion will be "off by one".

It seems clear that we cannot simply cut the data up blindly into data windows as we must accommodate many different types of change. Rather than splitting the data into



ID	age	bacilary load	hiv_status	alcoholism
5	34	2.2	positive	yes
6	35	3.4	negative	no
7	44	0.3	<nd>	no
8	28	1.6	positive	<nd>
9	54	2.3	<nd>	no
10	48	1.1	positive	<nd>

Figure 19: An overview of the Hashed Data Model

arbitrarily sized windows, it seems reasonable to segregate the dataset into semantically appropriate sections and detect changes there so we can infer more intelligently as to the nature of the change.

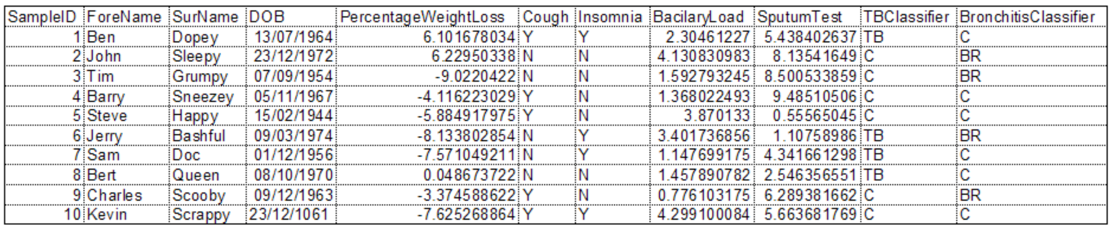
Consider the diagram illustrated in figure 19. If we identify important parts of the dataset, we can create an abstraction of the dataset in the form of a hashed data model. Of course, the success and usefulness of the hashed data model relies on the algorithms we use to detect and record the important parts of the dataset.

By selecting from what we consider the important aspects of the data and recording their relationships, we can build up the nodes and relationships of the data model. The HDM abstracts and records the aspects of the data by taking the hash of each data aspect. We build this model using RDF (Resource Description Framework). RDF facilitates interoperation because it provides a data model that can be extended to address sophisticated ontology representation techniques.

We will be using RDF to model the semantic representations of the data. Once we have we have built our RDF-XML document of the data model, we have the operations and functionality that have been developed for use with XML. There are XML comparison algorithms, versioning tools and many visualisation tools. We have used one such tool for this example[37].

6.8 A Brief Example of the HDM

There are three main steps in the example presented below. Figure 20 illustrates the example dataset. We have selected a small randomly generated dataset to represent our data[38]. We have some generic patient data (Name, age, etc.). We also have some measured attributes (symptoms, signs, etc.) as well as some derived classes. Derived classes are those that have been formed from other attributes present in the dataset. In this example the derived classes (TBClassifier and BronchitisClassifier) have been derived from a pre-experimental machine learning technique. From this dataset, we present a refined RDF graph that expands the model to include some semantic data, which is not included in the dataset itself. This is likely to require some input from the user, although we make no assumptions for how this may happen.



SampleID	ForeName	SurName	DOB	PercentageWeightLoss	Cough	Insomnia	BacilaryLoad	SputumTest	TBClassifier	BronchitisClassifier
1	Ben	Dopey	13/07/1964	6.101678034	Y	Y	2.30461227	5.438402637	TB	C
2	John	Sleepy	23/12/1972	6.22950338	N	N	4.130830983	8.13541649	C	BR
3	Tim	Grumpy	07/09/1954	-9.0220422	N	N	1.592793245	8.500533859	C	BR
4	Barry	Sneezey	05/11/1967	-4.116223029	Y	N	1.368022493	9.48510506	C	C
5	Steve	Happy	15/02/1944	-5.884917975	Y	N	3.870133	0.55565045	C	C
6	Jerry	Bashful	09/03/1974	-8.133802854	N	Y	3.401736856	1.10758986	TB	BR
7	Sam	Doc	01/12/1956	-7.571049211	N	Y	1.147699175	4.341661298	TB	C
8	Bert	Queen	08/10/1970	0.048673722	N	N	1.457890782	2.546356551	TB	C
9	Charles	Scooby	09/12/1963	-3.374588622	Y	N	0.776103175	6.289381662	C	BR
10	Kevin	Scrappy	23/12/1061	-7.625268864	Y	Y	4.299100084	5.663681769	C	C

Figure 20: The example dataset.

The final stage of the example is to show the RDF-XML for the refined graph. We must then add the hash values for each node to the XML although this has not yet been completed. It is intended that this XML can be stored either together with the experimental results or in a separate repository. We can therefore record a model of the dataset that was used for a particular experiment and use already-established methods for storing, comparing and versioning the XML data model.

[37]IsaViz can be found at http://www.w3.org/2001/11/IsaViz/.

[38]It is important to note that the data used is entirely fictional and any resemblance to any person either living or dead is entirely coincidental.

The following is an example of the metadata associated with the dataset.

```
USER: btagger
LASTUPDATED: 06/06/2006
QUERY: SELECT * FROM SOURCE_DATA WHERE BACILARY_LOAD > 0
HYPTHESIS: To investigate the relationship between BacilaryLoad
and TB and Bronchitis Classification.
FIELDS: SampleID, ForeName, SurName, DOB, PercentageWeightLoss,
Cough, Insomnia, BacilaryLoad, SputumTest, TBClassifier, BronchitisClassifier
```

We can model the dataset as a structural, RDF graph. There are further relationships that can be modelled from the dataset. We can class the data aspects into different subclasses and we show this revised graph in figure 21. We can also refine the relationships between nodes to give more structural information about the data model. We use the standard RDF and RDFS syntax to extend the data model.

The following is an extract of the XML that is generated by the graph in figure 21.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:gss="http://www.w3.org/2001/11/IsaViz/graphstylesheets#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdf:Description rdf:about="Generic">
    <rdf:object>
      <rdf:Description rdf:about="Name">
        <rdf:object rdf:resource="SurName"/>
        <rdf:object rdf:resource="ForeName"/>
      </rdf:Description>
    </rdf:object>
    <rdf:object rdf:resource="SampleID"/>
    <rdf:object>
      <rdf:Description rdf:about="Age">
        <rdf:object rdf:resource="Date"/>
        <rdf:object rdf:resource="BronchitisClassifier"/>
        <rdf:object rdf:resource="DOB"/>
      </rdf:Description>
    </rdf:object>
  </rdf:Description>

  ...[continues]...
```

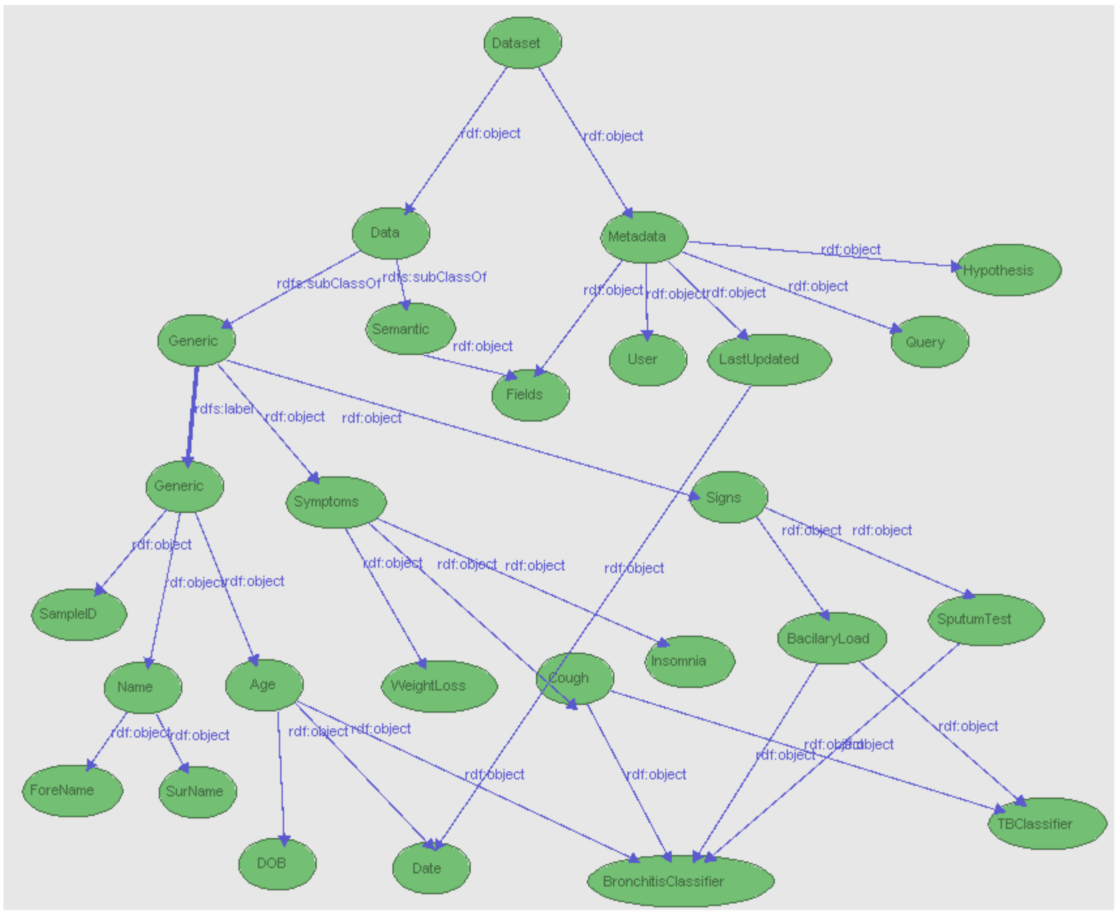


Figure 21: The refined graph representation

6.9 Provenance Logs

The provenance logs are designed to record the history of a piece of data, in this case, the experimental dataset. Such history might contain information as; who constructed the dataset query, when it was first constructed, the experiments that run from the dataset, and other items.

It may also be important to record how the dataset is used and changed over time. In the example in this report, the provenance is modified during the experimentation phase when the dataset has changed significantly enough to warrant an update. The provenance log is also stored alongside the results. This is, perhaps, not necessary and a provenance database would be more appropriate so that the provenance for each dataset is stored only once. By using the hashed data model to detect changes, we can easily record these changes in the provenance logs to register a new version of the dataset. The experimental dataset then becomes versioned based on significant changes but the present version does not have an effect on previously run experiments unless a significant change has occurred since that experiment.

The benefit of this is that each experiment records its own 'version' of the experimental dataset. We can update a particular set of results, which are sensitive to dataset changes without affecting other experiments, which are perhaps less sensitive to data changes. As each experiment contains its own version, we can investigate the changes between exactly that version and the current dataset and then decide whether an update is required. Experiments are not limited to released versions of the dataset.

Conceptually, a new version of the dataset is created whenever it is experimented upon, resulting in a very flexible versioning system. The extent to which we record the versioning is an implementation question and would be reflected in the provenance logs.

6.10 Future Work

It does not appear to be feasible to successfully record each and every change that occurs within a dataset given the large number of experiments that can exist upon that dataset. As demonstrated in section 6.4, the computational and storage complexity of maintaining the multiple versions are very demanding[39], especially if the system is to be used for personal use (on a personal workstation). Therefore, we need to create a model of the dataset with which we can detect, quantify, monitor and record changes made to the dataset. The approach proposed in this report is simply one possible way of managing the change. There is still a considerable amount of work to be done to ensure the suitability of using hashes for creating the data abstraction and this is planned for the very near future.

In particular, we would like to be able to infer various aspects of the data by comparing

[39]An O(n2) problem will suffer from exponential complexity.

the hashes. We may want to identify the precise location of the change or the predicted impact in order to decide whether we need to act upon the change. We may like to use a comparable hash function in order to compare changes to model significance. To have a meaningful comparable hash function, we find ourselves moving towards a finer and finer grained change detection[40] until we are recording the same amount of hash values as there are items of data in the dataset and we are providing no abstraction.

There is still much work to be done investigating how the data model and the data provenance interact with one another and where and how each of these are generated and maintained. It may be unwise to make assumptions as to the progress that can be made with regard to provenance management. There has been a considerable and increasing amount of work towards handling provenance and these have generally been pioneered by large groups or consortiums[41]. That does not mean we cannot make a contribution to the problem, but a fully functioning provenance-handler may be beyond the scope of this research.

The work on the framework so far has concentrated on managing changes in the data and the semantic structure. The change and impact analyses identified other types of change that occur within the example. Other experimental aspects must be managed such as; systems configuration parameters, experimental parameters, protocol changes and more. These must be incorporated into the framework before we try and integrate it into the example.

After the framework has been developed to a reasonable standard, we can begin to integrate it into the example environment. There is likely to be impacts on the example environment as there would be with most new systems. It is important that we identify the impacts so that these can be minimised and accounted for so that the framework can be used successfully by the users. Therefore, we will conduct an impact analysis on the example as a result of the implemented framework, the outcome of which will be an operating manual or a guide, at least.

At the completion of the integration stage, we will conduct a suitability study to determine how we might apply the framework to other biological processes. It is not clear whether this will take the form of a comparative study of integrated examples[42] or a report based on the impact analysis above and a hypothesised look at the suitability of the framework. This will depend primarily on the time available at this point.

[40] I.e., we are comparing smaller and smaller parts of the data.
[41] Please refer to section 2.8
[42] I.e., actually applying the framework to a variety of examples.

7 Validation of Contribution

Having completed such a comprehensive change and impact analysis on the real-world example, we can compare it against the post-integration impact analysis (see section 6.9) in order to validate the contribution. By identifying what needs to be done and evaluating what has been achieved, we can measure the extent to which the contribution has addressed the problems.

To date, there has been little publishable content produced. The analyses of the working example, although important for the basis of the research, are not suitable for publication. It is expected, upon further refinement of the hashed data model, we will have some good publishable content very shortly.

8 Timetable

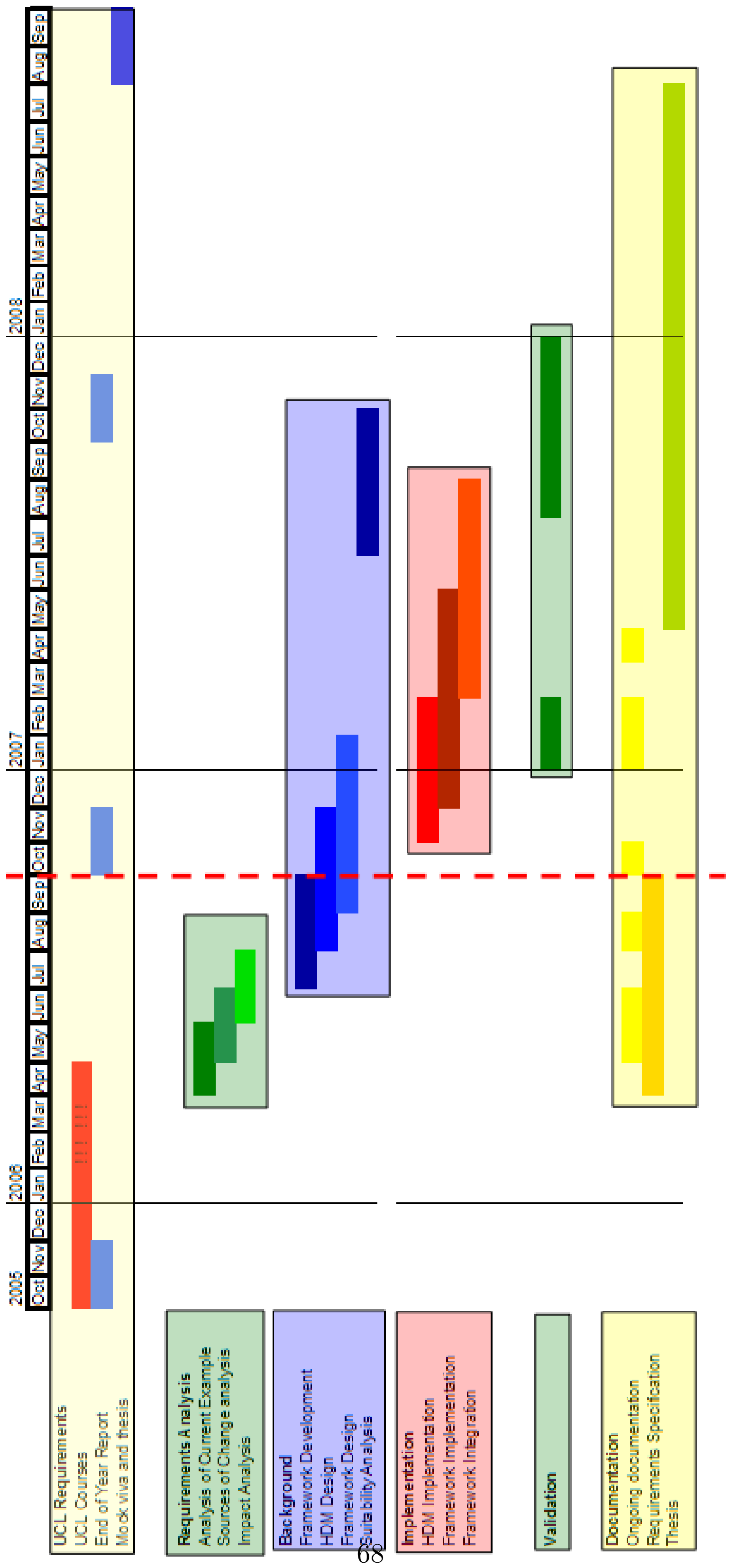


Figure 22: Project timetable detailing the activities for the next two years.

References

[1] A. Bahl, B. Brunk, R. L. Coppel, J. Crabtree, S. J. Diskin, M. J. Fraunholz, G. R. Grant, D. Gupta, R. L. Huestis, J. C. Kissinger, P. Labo, L. Li, S. K. McWeeney, A. J. Milgram, D. S. Roos, J. Schug, and C. J. Stoeckert, Jr. PlasmoDB: the Plasmodium genome resource. An integrated database providing tools for accessing, analyzing and mapping expression and sequence data (both finished and unfinished). *Nucleic Acids Res, vol. 30, pp. 87-90*, 2002.

[2] A. Brazma et al. Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nat Genet, vol. 29, pp. 365-71*, 2001.

[3] A. L. Rector, J. E. Rogers, P. E. Zanstra, and E. Van Der Haring. OpenGALEN: open source medical terminology and tools. *AMIA Annu Symp Proc, pp. 982*, 2003.

[4] P. I. Olason. Integrating protein annotation resources through the Distributed Annotation System. *Nucleic Acids Res, vol. 33, pp. W468-70*, 2005.

[5] J. Futrelle, S. Chen and K. C. Chang. NBDL: a CIS framework for NSDL. *In Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries (US).* JCDL 2001.

[6] S. Yang, S. S. Bhowmick, and S. Madria. Bio2X: a rule-based approach for semi-automatic transformation of semi-structured biological data to XML. *Data and Knowledge Engineering, vol. 52, pp. 249-271*, 2005.

[7] Z. Lacroix, Biological data integration: Wrapping data and tools. *Ieee Transactions on Information Technology in Biomedicine, vol. 6, pp. 123-128,* 2002.

[8] W.H. Wolberg, W.N. Street, O.L. Mangasarian. Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates. *Cancer Letters 77:163-171*, 1994.

[9] S. P. Shah, Y. Huang, T. Xu, M. M. Yuen, J. Ling, and B. F. Ouellette, Atlas - a data warehouse for integrative bioinformatics. *BMC Bioinformatics, vol. 6, pp. 34*, 2005.

[10] J. Koh et al. BioWare: A framework for bioinformatics data retrieval, annotation and publishing. *SIGIR 2004 Workshop*

[11] C. Schönbach, P. Kowalski-Saunders, and V. Brusic. Data warehousing in molecular biology. *Brief Bioinform, vol. 1, pp. 190-8*, 2000.

[12] A. Kasprzyk, D. Keefe, D. Smedley, D. London, W. Spooner, C. Melsopp, M. Hammond, P. Rocca-Serra, T. Cox, and E. Birney. EnsMart: A generic system for fast and flexible access to biological data. *Genome Research, vol. 14, pp. 160-169*, 2004.

[13] D. Gilbert. Shopping in the genome market with EnsMart. *Briefings in Bioinformatics, software review submission*, June 2003

[14] S. M. J. Searle, J. Gilbert, V. Iyer, and M. Clamp. The Otter annotation system. *Genome Research, vol. 14, pp. 963-970*, 2004.

[15] J. P. Narain, E. Pontali, S. Tripathy. Epidemiology and Control Strategies. *Symposium on HIV TB. Indian Journal of Tuberculosis.* 2000.

[16] J. A. Cuff, G. M. P. Coates, T. J. R. Cutts, and M. Rae. The Ensembl computing architecture. *Genome Research, vol. 14, pp. 971-975*, 2004.

[17] T. Hubbard. Ensembl 2005. *Nucleic Acids Research, vol. 33, pp. D447-D453* 2005.

[18] R. D. Dowell, R. M. Jokerst, A. Day, S. R. Eddy, and L. Stein. The distributed annotation system. *BMC Bioinformatics, vol. 2, pp. 7*, 2001.

[19] O. G. Troyanskaya, K. Dolinski, A. B. Owen, R. B. Altman, and D. Botstein. A Bayesian framework for combining heterogeneous data sources for gene function prediction (in Saccharomyces cerevisiae). *Proceedings of the National Academy of Sciences of the United States of America, vol. 100, pp. 8348-8353,* 2003.

[20] K. G. Herbert, N. H. Gehani, W. H. Piel, J. T. L. Wang, and C. H. Wu. BIO-AJAX: An extensible framework for biological data cleaning. *Sigmod Record, vol. 33, pp. 51-57*, 2004.

[21] A.J. Gilliland-Swetland. Setting the Stage. *Introduction to Metadata, Pathways to Digital Information - http://www.getty.edu/research/institute/standards/intrometadata/*, 2000

[22] D. K. Gifford, R. M. Needham, and M. D. Schroeder. The Cedar File System. *Communications of the ACM, 31(3):288298*, 1988.

[23] D. G. Korn and E. Krell. The 3-D File System. *In Proceedings of the USENIX Summer Conference, pages 147156.* Summer 1989.

[24] W. F. Tichy. RCS-A System for Version Control. *Software-Practice Experience 15, 7, 637-654*, July 1985.

[25] M. J. Rochkind. The Source Code Control System. *IEEE Transactions on Software Engineering, SE-1, 4, 364-370* December 1975.

[26] D. Grune, B. Berliner, and J. Polk. Concurrent Versioning System, *http://www.cvshome.org.*

[27] BM Rational. Rational clearcase. *www.rational.com/products/clearcase/index.jsp.*

[28] V. Henson and J. Garzik. BitKeeper for Kernel Developers. *Ottawa Linux Symposium, Ottawa, Ontario Canada,* 2002.

[29] J. Kovse, T. Härder. V-Grid - A Versioning Services Framework for the Grid. *Proc. 3rd Int. Workshop Web Datenbanken, Berliner XML Tage, Berlin, pp. 140-154*, October 2003.

[30] S. Y. Chien, V. J. Tsotras, and C. Zaniolo. XML document versioning. *Sigmod Record, vol. 30, pp. 46-53*, 2001

[31] D. Hitz, J. Lau, and M. Malcolm. File System Design for an NFS File Server Appliance. *In Proceedings of the USENIX Winter Technical Conference, pages 235245*, January 1994.

[32] Zachary N. J. Peterson and Randal C. Burns. Ext3cow: The design, Implementation, and Analysis of Metadata for a Time-Shifting File System. *Technical Report HSSL-2003-03*, Computer Science Department, The Johns Hopkins University, 2003. http://hssl.cs.jhu.edu/papers/peterson-ext3cow03.pdf.

[33] Digital Equipment Corporation. *TOPS-20 Users Guide (Version 4)*, January 1980.

[34] K. McCoy. VMS File System Internals. *Digital Press*, 1990.

[35] D. S. Santry, M. J. Feeley, N. C. Hutchinson, A. C. Veitch, R.W. Carton, and J. Ofir. Deciding When to Forget in the Elephant File System. *In Proceedings of the 17th ACM Symposium on Operating Systems Principles, pages 110-123*, December 1999.

[36] Craig A. N. Soules, Garth R. Goodson, John D. Strunk, and Greg Ganger. Metadata efficiency in versioning file systems. *Conference on File and Storage Technologies (San Francisco, CA).* 31 March–02 April 2003.

[37] J. D. Strunk, G. R. Goodson, M. L. Scheinholtz, C. A. N. Soules, and G. R. Ganger. Self-Securing Storage: Protecting Data in Compromised Systems. *In Proceedings of the 4th Symposium on Operating Systems Design and Implementation, pages 165-180*, October 2000.

[38] S. Khaddaj, A. Adamu and M. Morad. Object Versioning and Information Management. *Information and Software Technology, Volume 46, Issue 7.* June 2004.

[39] Dadam, V. Lum and H.D. Werner. Integrating of time versions into relational database systems. *Proceeding of the Conference on Very Large Database* (1984) pp. 509522.

[40] F. Grandi, F. Mandreoli, A formal model for temporal schema versioning in object-oriented databases. *A Timecenter Technical Report TR-68*, 2002.

[41] M. Golfarelli, J. Lechtenborger, S. Rizzi, and G. Vossen. Schema versioning in data warehouses. *Conceptual Modeling for Advanced Application Domains, Proceedings, vol. 3289, pp. 415-428,* 2004.

[42] K. R. Dittrich, D. Tombros and A. Geppert. Databases in Software Engineering: A Roadmap. *The Future of Software Engineering, Anthony Finkelstein (Ed.), ACM Press* 2000

[43] B. R. Barkstrom. Data product configuration management and versioning in large-scale production of satellite scientific data. *Software Configuration Management, vol. 2649, pp. 118-133*, 2003.

[44] W. M. Shui, N. Lam, R. K. Wong. A Novel Laboratory ersion Management System for Tracking Complex Biological Experiments. *bibe, p. 133, Third IEEE Symposium on BioInformatics and BioEngineering (BIBE'03)*, 2003.

[45] M. D. Flouris and A. Bilas. Clotho: Transparent Data Versioning at the Block I/O Level. *citeseer.ist.psu.edu/flouris04clotho.html*, 2004.

[46] N. C. Hutchinson, S. Manley, M. Federwisch, G. Harris,D. Hitz, S. Kleiman, and S. OMalley. Logical vs. Physical File System Backup. *In Proc. of the 3rd USENIX Symposium on Operating Systems Design and Impl. (OSDI99)*, Feb. 1999.

[47] J. Kovse and C. Gebauer. VS-Gen: A case study of a product line for versioning systems. *Generative Programming and Component Engineering 2004, Proceedings, vol. 3286, pp. 396-415*, 2004.

[48] R. H. Katz. Toward a Unified Framework for Version Modeling in Engineering Databases. *Computing Surveys, vol. 22, pp. 375-408*, 1990.

[49] J. S. Goonetillake, T. W. Carnduff, and W. A. Gray. An integrity constraint management framework in engineering design. *Computers in Industry, vol. 48, pp. 29-44*, 2002.

[50] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies, vol. 43, pp. 907-928*, 1995.

[51] OBO: Open Biomedical Ontologies. *http://obo.sourceforge.net*

[52] B. Smith et al. Relations in biomedical Ontologies. *Genome Biology* 2005

[53] W. Ceusters, I. Desimpel, B. Smith, S. Schulz. Using Cross-Lingual Information to Cope with Underspecification in Formal Ontologies. *Proceedings of Medical Informatics Europe* 2003.

[54] J. Verschelde, M. Casella Dos Santos, T. Deray, B. Smith and W. Ceusters. Ontology-assisted database integration to support natural language processing and biomedical data-mining. *Journal of Integrative Bioinformatics*

[55] W. Ceusters, B. Smith, J. M. Fielding. LinkSuite: formally robust ontology-based data and information integration. *Database Integration in the Life Sciences* 2004.

[56] J. Cardoso and A. Sheth. Introduction to semantic web services and web process composition. *Semantic Web Services and Web Process Composition, vol. 3387, pp. 1-13*, 2005.

[57] M. Klein and D. Fensel. Ontology versioning for the Semantic Web. *In Proceedings of the International Semantic Web Working Symposium (SWWS), Stanford University, California, USA*, July 30 – Aug. 1, 2001.

[58] N. F. Noy, S. Kunnatur, M. Klein, and M. A. Musen. Tracking changes during ontology evolution. *Semantic Web - Iswc 2004, Proceedings, vol. 3298, pp. 259-273*, 2004.

[59] K. H. Cheung, K. Y. Yip, A. Smith, R. deKnikker, A. Masiar, and M. Gerstein. YeastHub: a semantic web use case for integrating data in the life sciences domain. *Bioinformatics, vol. 21, pp. I85-I96*, 2005.

[60] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The semantic Web: The roles of XML and RDF. *Ieee Internet Computing, vol. 4, pp. 63-74*, 2000.

[61] S. Martin, M. Senger and M. Niemi. Life Sciences Identiers second revised submission. *Technical report, I3C* (2003)

[62] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. Importing the semantic web in UDDI. *Web Services, E-Business, and the Semantic Web, vol. 2512, pp. 225-236,* 2002.

[63] Wikipedia Encyclopedia - http://en.wikipedia.org

[64] G. F. Pfister. In search of clusters (2nd ed.). *Prentice-Hall, Inc.* 1998

[65] P. Buneman, S. Khanna, and W. C. Tan. Data provenance: Some basic issues. *Fst Tcs 2000: Foundations of Software Technology and Theoretical Computer Science, Proceedings, vol. 1974, pp. 87-93*, 2000.

[66] J. Frew and R. Bose. Lineage issues for scientific data and information. *Data provenance/derivation workshop.* October 2002.

[67] P. Buneman, S. Khanna, and Wang-Chiew Tan. Computing provenance and annotations for views. *Data provenance/derivation workshop.* October 2002.

[68] C. Goble. Position statement: Musings on provenance, workflow and (semantic web) annotations for bioinformatics. *Data provenance/derivation workshop.* October 2002.

[69] S. Davidson, C. Overton and P. Buneman. Challenges in Integrating Biological Data Sources. *Journal of Computational Biology, 2(4):557-572*, Winter 1995.

[70] P. Buneman, A. Deutsch, and W. Tan. A Deterministic Model for Semistructured Data. *In Proc. of the Workshop On Query Processing for Semistructured Data and Non-standard Data Formats, pages 14-19*, 1999.

[71] M. Szomszor and L. Moreau. Recording and reasoning over data provenance in web and grid services. *On the Move to Meaningful Internet Systems 2003: Coopis, Doa, and Odbase, vol. 2888, pp. 603-620,* 2003.

[72] D. Pearson. Data requirements for the grid. *Scoping Study Report.* February 2002. Status Draft.

[73] R. Stevens, A. Robinson and C. Goble. myGrid: Personalised Bioinformatics on the Information Grid. *International Conference on Intelligent Systems in Molecular Biology.* (2003)

[74] J. Hendler. Communication: Enhanced Science and the Semantic Web. *Science 299* (2003) 520-521

[75] J. Zhao, C. Wroe, C. Goble, R. Stevens, D. Quan, and M. Greenwood. Using semantic web technologies for representing e-Science provenance. *Semantic Web - Iswc 2004, Proceedings, vol. 3298, pp. 92-106*, 2004.

[76] J. Shawe-Taylor, N. Cristianini. Kernel methods for pattern analysis *Cambridge University Press, Cambridge*, 2004.

[77] I. Foster et al. Chimera: A virtual data system for representing, querying, and automating data derivation. *Proceedings of the 14th International Conference on Scientific and Statistical Database Management()SSDBM '02). July 2002*

[78] J. Frew, R. Bose. Earth system science workbench: A data management infrastructure for earth science products. *Proceedings of the 13th Conference on Scientific and Statistical Database Management()SSDBM '01). July 2001*

[79] J. Myers et al. Multi-scale science: Supporting emerging practice with semantically derived provenance. *Proceedings of the Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data.* 2003

[80] C. Pancerella et al. Metadata in the collaboratory for multiscale chemical science. *Proceedings of the Dublin Core Conference (DC '03)* 2003

[81] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. *CIDR*, 2005.

[82] R. Bose and J. Frew. Lineage retrieval for scientific data processing: a survey. *ACM Comput. Surv. 37, 1* (Mar. 2005), 1-28.