Addition to the EngD First Year Report

Ben Tagger

January 31, 2006

Contents

1 Introduction **3**

2 Biological Data Background **3**

2.1 Organisational Characteristics . . . 3

2.2 Data Format Variety and Location Heterogeneity . . . 5

2.3 Problems with Biological Data . . . 6

3 The Example Dataset **7**

3.1 Proteomic Data . . . 7

3.2 The Data Structure . . . 8

3.3 Problems and Challenges with the Dataset . . . 8

4 The Contribution **10**

5 Current Versioning Methodologies **12**

5.1 Versioning Tools . . . 12

5.2 Examples of Successful Data Versioning . . . 13

5.3 Other Versioning Technologies . . . 15

5.4 The Unsuitability of Current Versioning Technologies . . . 15

6 System Properties **16**

1 Introduction

The purpose of this document is to provide an additional report to the one submitted on the 16th of November 2005 to address matters arising from that report and from the meeting that was held on the 5th of December 2005. Upon reading this report, three things should be clear; the problem to be addressed, the contribution to address the problem and the validation to show that the problem has successfully been addressed.

Section two will address some further issues concerning the background to the biological data. The major organisational stakeholders involved with biological data will be documented together with a description of the data with which they are explicitly involved. This will be followed by some observed problems with their biological data. Section three will describe the specific data set that will be used for the contribution together with the potential challenges and problems with reference to biological data in general. Section four will discuss the contribution that will be made and investigate how the contribution will address the problems discussed in the previous sections. Section five will provide some descriptions of current versioning methodologies and illustrate how these methodologies are unsuitable for the problems that are to be addressed by the proposed contribution. Section six will address some of the system properties and how these properties may apply to other areas.

2 Biological Data Background

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 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[1] 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].

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

2.2 Data Format Variety and Location Heterogeneity

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 and too many to discuss the details and differences in this document. In general, each database will have its own sequence format. Some databases such as GenBank, EMBL and DDBJ 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).

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[2]. 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. MIAME (Minimum Information About a Microarray Experiment) 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 the 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.

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

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

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 and 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.

There are a number of data sources from which scientists and researchers may wish to retrieve and access biological data, including; 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.

2.3 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[3]. 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.

[3]Please refer to section 2.2

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.

3 The Example Dataset

The dataset that is to be initially used is a set of proteomic fingerprinting data generated at St. George's Hospital[4] and the NIMR. Refer to section 2.2 for descriptions of other types of biological data. 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.

3.1 Proteomic Data

Proteomics is the analysis of complete complements of proteins. It is concerned not only with the identfication 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.

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

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

approach taken for protein quantitation and explain how this approach may impact the interpretation of any results obtained.

3.2 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 1.

From the dataset, *Patient*, other sub-datasets are created with the purpose of investigating various hypotheses. For example, tuberculosis and HIV are inextricably linked [4]. 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.

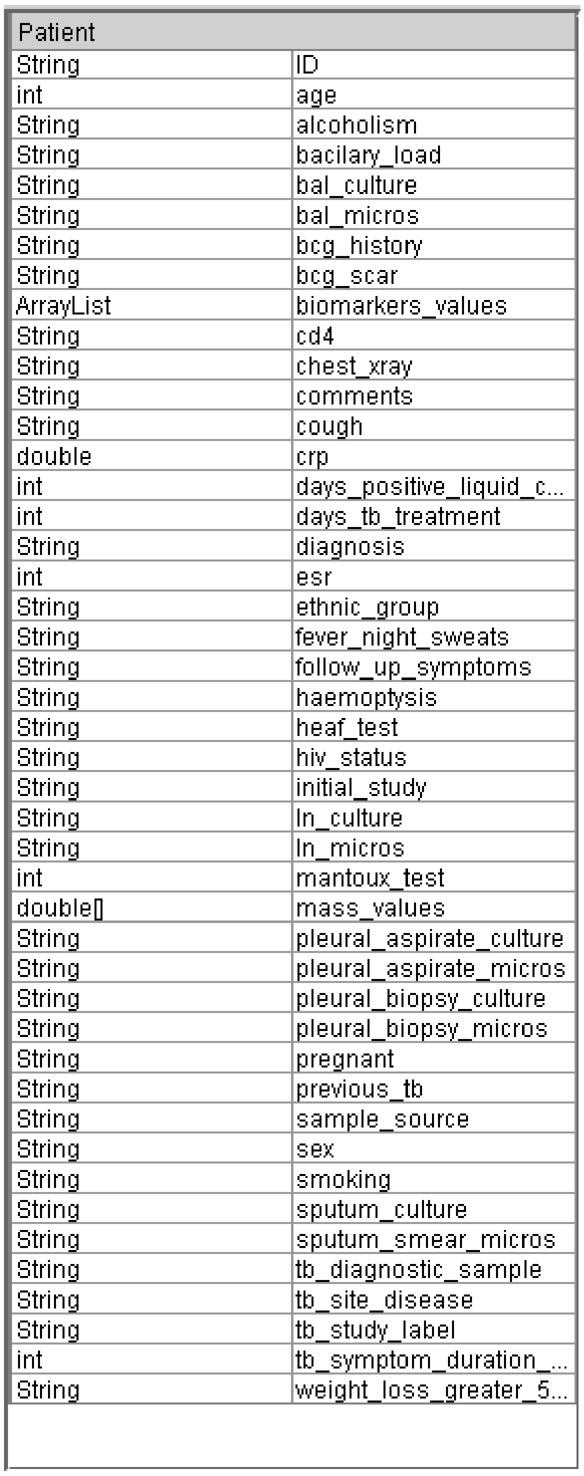
```
tb_study_label == "TB" && hiv_status == "Negative"
```

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.

3.3 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 2 shows some of the derived datasets from *Patient*, but there are many more possibilities.

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.



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 1: Data Descriptors for Patient

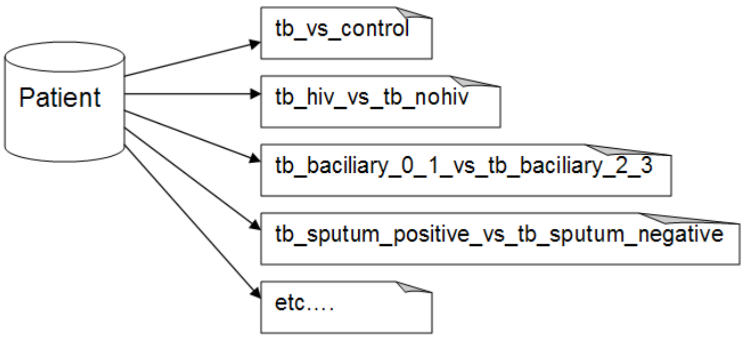

Patient
tb_vs_control
tb_hiv_vs_tb_nohiv
tb_baciliary_0_1_vs_tb_baciliary_2_3
tb_sputum_positive_vs_tb_sputum_negative
etc....

Figure 2: Some of the derived datasets from *Patient*.

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[5]. 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?

4 The Contribution

Given the vastness, its heterogeneity and other characteristics of biological data, the management of such data has become a key research area. To date, much of the research has focused on dealing with the size, heterogeneity and multiple sources of the data. Much of this work has been centered on improving the integration of the data, concentrating on data formats and standards. In contrast, little by comparison has been done on one aspect of data management, namely the versioning of biological data. But why is there a need for the versioning of life sciences data?

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

Consider the generation of life sciences data. Not only is the source data vast, heterogeneous and disparate, but the experiments that yield further results are complex and highly varied. Furthermore, with the advent of high-throughput computation, thousands of experiments can now be conducted in the same time it took to do only a handful several years ago (manually). Experiments may be repeated so as to verify results or to update data that has been subsequently updated itself. E-scientists may want access to previous versions of results in order to check repeatability or to verify other items of data. Previous versions of data may be used to check similarity with updated results or simply to re-run older experiments.

It is important to note that versioning for this contribution differs to the versioning usually found in a computer science environment. Traditional versioning is commonly aimed at resolving differences in documents[6] written by multiple authors. Here, versioning is concerned with allowing multiple users to work on a set of resources whilst minimising the loss of the total work performed. In the context of biological data, versioning refers to something a little different. It has been identified that certain biological experimentation processes will result in multiple versions of results. In this context, versioning means to literally manage the available versions from an ego-centric position (i.e., assuming that there is no experimentational concurrency).

This EngD aims to contribute in three distinct ways. Firstly, and most trivially, the e-scientist should be presented with a method for simply viewing and having easy access to multiple versions of data. Secondly, the e-scientist should be able to compare multi-versioned input data and map the multiple versions of output data to which it relates. The first two aspects of the contribution are based on presenting the versioning of the biological data to the user in an agreeable manner.

The third contribution of the EngD will be to record the user's experimental process. The experiments and processes that the user performs with the data will be recorded and versioned alongside the data itself. The record of the experimentation process will aid:

1. The repeatability of results. It will be possible to re-run experiments to verify results or by means of a backup to recover lost results[7].
2. The ability to keep track of constantly changing source data. An update to a source data can require the user to update their results. However, if many experiments have been run in the mean time, it may not be possible to reproduce the 'working dataset' to reflect the changes in the source data. However, if the experimentation process is accurately recorded and fine-grained enough to repeat the experiments without human interaction[8], then updates can be dealt with in a far less painful

[6]In this context, a 'document' can refer to any kind of object that can be versioned, such as a file, document or piece of work (code).

[7]Technically, this would not be a recovery operation, but a replacement of lost results.

[8]This statement refers to having substantially less human interaction, rather than no interaction whatsoever.

manner.

3. The improvement of the scientific documentation of the experimentation process. A record of the experiment process will provide an ongoing log of user actions.
4. Allow the user a more dynamic work environment. With access to the user's entire experimental history, it may become easier to identify promising areas of further experimentation.

5 Current Versioning Methodologies

5.1 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) [5] and 3D File System (3DFS) [6]. 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) [7] and SCCS (Source Code Control System) [8]. SCCS [8] 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 [7] 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 [9] 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 [10] is another versioning tool, but requires specific administration and is also expensive.

BitKeeper [11] 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 [26].

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 [13]. Moreover, neither RCS nor SCCS preserve the logical strucutre 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 [14] and Ext3cow [15]. 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 [16], VMS [17], the Elephant File System [18] and CVFS [19, 20]. An ideal situation would be to have versioning automatically integrated into the operating system. However, this has yet to be implemented.

5.2 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 [21]. 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 fulfil 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 [9]. 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 [22]. According to [22], 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[9].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 [22] notes that changes in any of these items can induce scientifically important changes in the data.

Shui et al. [23] present an XML-based version management system for tracking complex biological experiments. Shui et al. [23] 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 [23] 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

[9]In fact, Barkstrom [22] quotes the production values at tens of thousands of files per day from tens or hundreds of different data sources.

little more than generic XML versioning, applied in a scientific data context.

5.3 Other Versioning Technologies

Normally, versioning functionality has been implemented by high-level applications or at the filesystem level. [24] 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 [24]. It also reduces complexity in the higher layers, in particular the filesystem and storage management applications [25]. Thirdly, it offloads expensive host-processing overheads to the disk subsystem, thus, increasing the overall scalability of the system [25].

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.

5.4 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 [19], 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 [19]. For example, CVS [9] and RCS[7] 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 [22] [23], 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 verisoning technologies do not take into account the uniqueness of biological data and the contraints 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 [13]. 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 [13].

XML versioning provides a closer match for the required purpose [13]. It is suggested that XML document versioning provides a sufficient solution for managing the versioning of biological data experiments [23]. 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.

6 System Properties

In the first case, the contribution (discussed in section 4) will be applied to the example dataset (discussed in section 3). However, proteomic data is only one of the types of biological data available. If the contribution is to be considered as a contribution for biological data in general, then it must be applicable for more than just proteomic data. With this in mind, at least two further types of biological data will be selected upon which to apply the contribution. It has not yet been confirmed exactly what types of data will be used, but the likely candidates will be sequence data and microarray data[10].

These three case studies will be sufficiently large and representative of biological data in general to suggest, given that the contribution proves successful for each case study, then the contribution will provide a solution for biological data versioning. This does not mean that the framework will be capable of versioning any type of biological data at that time. Rather, it argues that the framework is capable of being applied for any (or most other) types of biological data.

The specific properties that each case-study should exhibit are explained in section 4 ("The Contribution"). The criteria documented in the contribution (view data versions, manage and compare data versions, record experimentation process) describes the MLOS (minimum level of success) for the versioning framework.

[10]Refer to section 2.2 for a description of these types of data.

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

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

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

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

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

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

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

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

[12] 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.

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

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

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

[28] 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.