DATASET VERSIONING THROUGH CHANGELOG ANNOTATION

By

Benno Lee

Prepared for:

Peter Fox, Thesis Advisor

Jim Hendler, Advisor

Deborah MacGuiness, Member

Beth Plale, Member

ii

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

ABSTRACT

Science is constantly changing and the data that drives it changes along with it. Understanding the behavior of data as it changes to correct errors and to add value, improves data quality and eases data integration. Current methods to track the evolution of data primarily focus on the collection of data provenance, but lacks the fidelity to evaluate the extent of a change. The conceptual model proposed in this document addresses this challenge by leveraging semantic technologies to automatically embed versioning information into changelog artifacts generated when changes are made to a dataset. A method to calculate change distance then follows using the resulting semantic graph by attaching weights to different change types. Initial applications begin with spreadsheets then databases with the ultimate goal in applying the model to online ontologies.

CHAPTER 1
INTRODUCTION

John C. Maxwell once said, "Change is inevitable. Growth is optional." While this inspirational quote refers to the human character, it also holds true for scientific datasets. With changing technology, data collected by researchers grew at an astounding rate. NASA's Atmospheric Science Data Center reported a growth from hosting around five million files to twenty million files between 2001 and 2004 [1]. The ATLAS project at CERN reports that it generates on the order of four thousand new datasets per day from experimental tests alone[2]. This explosion of data marks a change in the perspective of research from a software driven approach, where programs are used to confirm results using data, to a data driven approach, where vast quantities of data is integrated to demonstrate trends and produce results. Instead of simply using computers to assist in performing a complex computation, researchers now use broad data collections to inform advances in science. This means that data quality needs to be ensured in order to extract valid conclusions from large data sets as humans can no longer manually curate the data mining process. However, the need for quality assurance means that a portion of the files generated do not contribute to data set growth since data has not been added, simply modified. Many NASA datasets have required re-processing of their data, either to improve data quality or to correct for errors [3]. Data traceability now becomes particularly important to identify sources that contribute to improved data quality. It creates a need to understand not only that a data set has changed, but to also understand how much a data set has changed. Data versioning is the method of tracking the changes performed on a data set and determining the extent to which it has changed. In this document, data versioning is approached using technology provided by semantic technologies and applying them to artifacts currently generated by scientific data sets.

1.1 Data Set Proliferation

Data has existed long before computers, populating the storage space of filing cabinets and data closets the first transistor radio. It then comes as no surprise that libraries and library sciences provide the early methods of data management. The challenges and goals that face physical libraries remain valid even as data collection migrates to electronic alternatives [4]. Digital storage and the Internet has opened new opportunities and methods to administer book data by separating logical representations and physical representations [1]. It has also added a plethora of new content types such as wikis, blogs, and other document formats which have never seen physical print. All these new documents need a form of data management [5]. However, the migration has not been without its problems. Early citations used stagnant Uniform Resource Locators (URL) to refer to online documents, but this would lead to a condition known as link rot where moving the document would invalidate the URL [6]. This eventually led to the development of Persistent URLs (PURL) which also succumbed to link rot, and this eventually led to the distributed Digital Object Identifier (DOI) system used to track documents today. The DOI network provides a robust system to track documents, but when tracking data, it faces difficulty following the rate of change with some more volatile data sets. Distribution organizations assign a DOI whenever a new edition of a document becomes available, and due to the publication process, documents change very rarely so a new DOIs are rarely necessary. However, data sets are products and thus succumb to the iterative process of error correction and growth. Data collection often continues on after initial publication. DOI distributors treat new files like new sections to a paper and changes to files as edits so a new identifier must be issued to the data set.

For similar reasons, treating data as documents produces problems when applying technologies from software management [7][8]. Structure provides the most significant distinguisher between data and software since a data set with a removed file remains usable but a software project would break. The function of code comes from its content, but the function of data comes from its ability to store and organize data. This should not be confused with data formats which impose structure

onto data in much the same way programming languages provides a medium to express actions. However, exporting data in different formats is currently easier than exporting code into different languages. Data sets do not represent a single object, unlike a software project[9]. They are compact representations of all possible subsets of the data set, which are also datasets. This can be demonstrated simply in considering a data set where a user orders a file Y and over the course of a year all files but Y are modified and replaced. The user only wants to know when their file Y changes but remains unconcerned when other files change within the data set. This is not so with software projects. For this reason, the structures of data sets and software becomes incompatible and software versioning technologies are insufficient to capture this nuance.

The techniques employed by these technologies, however, can remain applicable to data sets and are often necessary when communicating change data to users. Version producers often refer to versions using numbers in the dot-decimal style [10]. While the values often signify the Major-minor numbers associated with the version, the names remain meaningless and can arbitrary assignment such as Ubuntu released numbered by Year-month values [11]. The arbitrary nature of the numbers often entails referring to versions by English nicknames instead. Such a regular method of naming release versions also means that determining the magnitude of change between two releases becomes impossible. Numbering the version this way, however, does allow computers and readers to quickly parse the version name and discern that a change has occurred, but little value exists beyond that [12]. The technique of distributed and federated employed by GIT does provide significant value to modern methods of versioning data [13]. As data workflows and data set dependencies grow, their volatility also expands, meaning that they become more likely to generate new versions. The federated approach available in the GIT environment allows developers to establish change dams that collect modifications and releasing the data at regular intervals,reducing the changes to a manageable flow.

1.1.1   Unified Systems

Working with data as documents leads to the shortfall of technologies, but working with the data of documents has led to significantly greater contributions. Many libraries often work in collaboration in order to provide a wide selection of texts over a limited number of physically available documents. The University of Virginia demonstrates the ability to achieve a unified library system using a combination of XML and web service technologies of their disparate assemblage of libraries [14]. The challenge involves providing a common landscape in order to compare the quality requirements imposed on the repositories. Versioning systems provide a notable mechanism to make this decision as quality determines when to generate new versions and what items belong to the same groupings of data. The comprehensiveness of XML and web technologies also allows this approach to apply to other systems and research areas as well. This becomes particularly relevant as innovations in computation technology generates small, volatile data sets to integrate into larger data managers [15]. In this application, the data food chain then becomes represented by smaller applications generated in situ and then unified with other data sets as they move up the food chain to a large, unified data distribution center.

Unified libraries represent a part of a larger collection of systems that rely on the propagation of data through heterogeneous systems to produce rapid complex solutions. The grid provided a unique environment that had to handle a variety of inputs, and therefore, different input data could run on distinct sets of grid services. This meant that different versions of the same data could be generated by differing services on the same grid [16]. The CERN grid for the Compact Muon Solenoid experiment separates the physical and logical storage of files, allowing multiple users to refer to the same file without needing to copy the file across the grid [17]. While the structure and construction of the grid reduced the uncertainty introduced by varying hardware, it raised questions on data quality by abstracting the transparency to underlying services. Cloud services have recently replaced the grid due to its flexibility in the services available to its autonomous systems. As the scale and complexity of autonomous systems grow, it becomes more difficult for one system

or organization to manage all the circles necessary to produce data deliverables. The ability to propagate relevant data change data across autonomous systems then assures valid quality in interactions between domains [18]. Not only does this ensure uniformity through system interaction, but it also ensures transparency with respect to the data and methods used to produce conclusions [19]. This often means that systems will need to negotiate a contract and establish a mutual interface to exchange data. Occasionally, this contract can be formal, but more ideally, the establishment of a standard lineage model or format would allow a greater variety of systems to interact with each other without needing lengthy contractual exchanges.

1.2 Data Quality/Provenance

The fundamental challenge to determining data quality, its subjectivity, needs clarification. Conceptually, a data set on desert climate likely has very poor data quality and relevancy to a study in whale biology. However, in a more quantitative sense, that same desert climate data can have excellent quality with respect to its correctness, expression, and traceability. With the hybridization of data sets from disparate agencies to provide big data solutions, collaborations plays an ever present role in achieving broad, valid findings. This requires good quality data and the ability to determine when data with better quality becomes available. [20]. The primary focus, generally, involves tracing the lineage of artifacts and activities that lead to the current data. This provides insight into possible sources of error as well as validating the assumptions made in generating a data set for future use.

There are several characteristics that can describe a data's quality, but the one most relevant to data versioning is provenance. It describes the sequence of events that lead to the construction of an object [21]. In art this describes the sequence of people who have had ownership of a piece of artwork. For data, provenance relates the history inputs and operations that result in a data object such as a plot or data set. NASA defines three levels of data processing, for example, that encompass the stages required to turn a raw signal from satellite instruments into physical measurements into global aggregate summaries [22]. Each stage computes a dataset using a collection of input data, processing scripts, and calibration values. This

collection forms the provenance for the resulting level of data, but because lower level data become inputs for higher level data, its provenance assimilates into the lineage of all resultant data sets. For example, Level 1 data's provenance is described by Level 0 data and the scripts and the calibration inputs that were required to turn Level 0 data into Level 1. Because Level 1 data is then used to generate Level 2 data, which in turn is used to generate a Level 3 data set, the Level 1 provenance is also included in Level 3 data's provenance. It is part of the sequence of events that led to the generation of the Level 3 data set.

As data sets grow, this process becomes even more confusing to coordinate so version control systems often manage provenance. Current research endeavors to provide high quality data clearly becomes more formalized as data becomes concentrated in massive data warehouses [23]. The focus of a majority of versioning research focuses on lineage retrieval which becomes ever important as evidence grows that researchers generate data faster than they can reasonably track [24]. This poses a particularly difficult problem as provenance provides a potent means of data auditing. With provenance, data producers ensure the trustability of their data inputs, either ingested from external sources or integrated from internal data sets. Fairly reassuring results have been found when combining lineage management and error reporting systems [25]. The errors provide a context for the changes made to advance the lineage of the data set and the version manager demonstrates that a problem has been addressed and how it was corrected. This system becomes extremely important when considering that agency funding often depends on the ability to account for the value of a project's dataset [26]. The data analytics required to determine the value of data collected by a project also requires the provenance to ensure that the analysis is also reproducible. The basic provenance often collected by hand now needs to be collected automatically in order to facilitate collaboration, especially with projects that are farther away from the data.

Early attempts at encoding provenance data into semantic models include the development of the Proof Markup Language [27]. While this was originally developed to express inference reasoning through traceable graph relations, the model can also be used to express the provenance of products using the same transitions.

The power is that it is able to use terms defined on the Semantic Web to construct inferences. This early demonstration of the ability for web based semantic technologies also expresses complex relations in a way that can be reasoned over and computed. It then allows for autonomous solutions to understanding change as data freshness begins playing a significant role in successful system function [28].

Not long after began the development of the Open Provenance Model (OPM) [29]. Driven by the uncertain needs and sometimes conflicting conventions of different scientific domains, the model sought to find a method to standardized the way in which provenance data is captured while also keeping the specification open to accommodate current data sets through the change. In an experimental case, the model has been applied to sensor networks to automate and unify their provenance capture even as they grow [30]. To aid in the adoption of the OPM, the framework Karma2 was developed to assist integrating provenance capture into scientific workflows [31]. It reduces the amount of modifications required to adopt the OPM through web services and, more importantly, integrates into scientific workflows. With the magnitude of data collection endeavors, it is no longer feasible for scientists to stay close to the data and must take a more abstract view of their data collection activities. Scientific workflows provides this high level view of complex data collection, curation, and analysis [32]. The value then of integrating provenance capture and workflow design is that lineage planning can then take place at a high level of scientific work. This gives insight into how different parts of the workflow fit together and how new exploratory expansions may occur.

Following the OPM, PROV is a W3C recommendation that deliniates a method to express data provenance with semantic technologies that has been accepted as a World Wide Web Consortium (W3C) Recommendation [33] [34] [35]. The recommendation uses a conceptual model relating activities, agents, and entities together to describe data production lineage [36] [37] [38]. Intended as a high level abstraction, it describes data as entities that are generated by activities enacted by agents. This basic relationship is very powerful in its ability to describe data production activities. The expression of the conceptual model occurs through the PROV Ontology (PROV-O), which can be conveyed through various resource description languages

[39] [40] [41]. The ontology is further formalized into a functional notation for easier human consumption [42] [43]. One particular strength that has contributed to the adoption of PROV is its ability to link into other ontlogies, making it easier for existing semantically enriched data sets to adopt PROV [44] [45]. Like the OPM, a framework has also been developed to alleviate workflow integration through Komadu [46]. The framework improves over its predecessor, Karma, by no longer utilizing global context identifiers that were no necessarily shared throughout the workflow.

PROV has played a significant contribution in maintaining the quality and reproducibility of datasets and reporting in the National Climate Assessment (NCA) [47, ?]. This implication signifies that there is an increased likelihood of adoption through other scientific fields as a result of this reporting. The Global Change Information System, which houses the data used to generate the NCA, uses PROV to meticulously track the generation of its artifacts and results as they are used in the report [48]. This means that not only does the data have a traceable lineage to verify quality, but the content of documents can have the same verifiability [49].

1.2.1 Changelogs

Changelogs, sometimes called patch notes, are artifacts resulting from the versioning process often found in major software projects. They document the changes made within the system and seek to explain, in human language, the motivations behind changes [50]. The logs provide significant utility to both users and producers as it can serve as both documentation and tutorials. Many users will often refer to the patch notes in order to decide how to adapt to changes made to the system they use, either data or software. Meanwhile, changelogs aid producers through team transitions by keeping a history of decisions made to improve the project. This is particularly evident in the realm of open-source projects as developers can contribute without having been part of the original development team. The need for documentation to bring new programmers up to speed for a project drives the ability to keep the project alive.

Open source projects have much more consistent adoption of changelogs than

data sets, possibly resulting from complex code techniques emerging earlier than large data methods. These logs provide a great source of value to developers as they can be used to give insight to the health of a software project [51]. These projects have a tendency to die rather quickly after initial enthusiasm and with the rather low overhead cost to start new open-source projects, some automated methods of determining the progress of a project is needed. It would give insight into the maturity of a project's development team as well as the likelihood that team members will correct errors within the code. However, readability proves to remain a significant hurdle as current development change logs contain solely human readable text. While machines may still be significantly removed from the ability to comprehend the impact of changes made to a data set or software code, they are currently opaquely blocked from consuming any of the content within logs more than understanding they contain text. The transition between different versions of large datasets is then left largely up to the human user's ability to understand and process the modifications mentioned within the change log.

As mentioned previously, changelogs also allow developers to link bugs and errors with their corrections in new versions of the code [52]. This gives feedback to the user community that corrections have been addressed as well as ensuring that modifications to the code base are driven by improving the project. It also has the added benefit of creating a system that can be used to link the introduction of new features with the emergence of new bugs [53]. The resulting discoveries help reveal patterns of development and prevent further occurrences of problematic code. Therefore, providing an machine consumable changelog would accelerate and assist in navigating through dataset changes and error corrections.

1.3   Provenance Distance

Data workflows have become complex and sized in such a way that even subtle changes may have meaningful effects [54]. One approach is following the provenance of the new data and identifying potential differences between how the data has become generated. Since provenance graphs have both directed edges and labeled nodes, there are a few methods of measuring similarity available beyond basic iso-

morphism [55] [56] [57] . This similarity measure is known as the provenance distance between the two data objects. A concern that arises with this approach is that, as previously stated, provenance only provides a context for versioning. Consider an example where there are versions 1 and 2 of a data object, both compiled from a reference source. Since there are two versions, it is understood that the versions are different so the comparison extends to their reference sources. If both versions were compiled from the same source, it can only be concluded that the compilation methods were different. There also exist methods that compare workflows based on quality criteria that leverages provenance to bound quality of service [58]. In the case of the NASA datasets, those methods should not have changed and no further insight has been found. If the reference sources have been changed, then it is expected that there would be a new version or data object and no new information has been found. This reasoning motivates the development of a versioning method that utilizes changelogs and RDFa to determine versioning distance.

1.4   Data Versioning Operations

Architecture has a principle that says form follows function, but, for data, form equals function. As a result, data has as many different forms as it has functions. Biological experiments often use data within cyclical data workflows where outputs are immediately fed back into new experiments [19]. Even though the goal of the experiment is the final data set, all the intermediary data sets provide significant value in reaching the goal. Libraries store data about their collections in large databases where both old and new versions of literature need to be maintained [20]. Some data exist in such a highly constrained environment that it must be managed at near the hardware level [59]. The challenge no longer becomes generating data, but instead, fitting the data into a format that users find useful and can consume.

The challenge of data versioning systems is to provide a unifying environment that can handle the plethora of forms and functions of its data. At its core, versioning systems only need to concern themselves with three operations: addition, deletion, and modification. Most literature surveys do not realize the significance of this commonality as this means that versioning methods can be described by

delineating how each operation is approached by a system [19] [4]. Data addition generally constitutes the least complicated versioning operation because it interacts the least with pre-existing data. However, new data does share context with pre-existing data and provides a method of measuring data set growth. Since data sets no longer have to be used in its entirety and can be freely subsetted, a data set's complexity increases significantly with its growth. Every new file added to a data set doubles the number of available subsets.

Data deletion, however, has a more philosophical difference between systems. From the perspective of a versioning specialist, data should never be deleted since knowing why data was excluded is as important as knowing why data was included. The software versioning manager GIT uses a method of compressing older data to conserve space without deleting the data [9]. Pragmatically, this is not always possible due, generally, to the physical constraints of storage space. In high energy physics, observational data often cannot be re-collected due to cost, and as a result, poor quality data cannot be re-processed or replaced [26]. The decision in this document is to use the term invalidation when referring to data removal operations as it implies that whether permanently deleted or not, there exists a more valid alternative.

Data modification encompasses the most involved data versioning operation. As a result, it often comprises a majority of the description of a data versioning service. In truth, data modification can be summarized as the invalidation of an instance of a data object - which can be a file, a record, or anything in a data set - followed by the addition of a new instance of that data object. However, this kind of operation is used so often to fix errors and update data sets that it is considered a unique operation. Modification owes its complexity to interacting with both pre-existing data from the invalidation stage and new data from the addition stage. However, this compound relationship fully contextualizes the relationship the operation has in relating the old data and the new data. In some cases, this only provides forward or backwards references between data versions, but having both gives users context for data's current state and update to new data [60].

Due to the ubiquity of the data addition, invalidation, and modification oper-

ations in versioning systems, the conceptual versioning model presented in Chapter 3 centers around capturing the relationships established by each of the operations. While other functions exist commonly in versioning systems such as object locking to prevent simultaneous conflicting changes, viewing to see the version an object belongs to, and branching to allow distributed modifications, these functions comprise the space of utility operations that support the three core processes.

1.4.1 Types of Change

Focus on change types because focusing on the values has comparably less value. When working with large data sets, individual changes are less important than changes made to the set abstractly.

Change is the fundamental characteristic in the study of versioning. Continuing the comparison with software, the measure of change in applications is primarily functional. Data, on the other hand, has not only function but also structure. While swapping the ordering of columns in a spreadsheet may not effect the usability of the data, it could cause issues with codes that use the data. This highlights the idea that many different forms of change exist with differing degrees of intensity. In this document, change is viewed as having three general categories: scientific, technical, and lexical. The most impactful change, scientific, denotes changes that have modified the fundamental science used to generate the dataset. Changes in algorithms or sampling methods generally fall under scientific modifications as they effect the base assumptions and possible error within the dataset. Technical impacts do not change the underlying science of the data, but impose a large enough change as to warrant notice. Structure alteration and unit conversions count as technical changes since the dataset now needs to be consumed differently but remains valid for use. Lexical changes belie the transformations that can best be described as corrections. Filling in previously missing values or fix erroneous values may be lexical changes.

The exact category that a particular change falls into can be controversial. The decision to change concentration units from parts per million to milligrams per milliliter poses a Technical change for a data producer. However, for a data consumer, the change may be viewed as a Scientific change as it invalidates the meth-

ods they had previously used. This conflict in view illustrates the data consumer-producer dynamic. In general, data producers are in control of the methods of versioning, but data consumers determine the classification of a data change. Reference 19 demonstrates the viewpoint as producers tend to use versioning systems to ensure data quality of service through audits and recovery tools. Meanwhile, a consumer will analyze the historical changes and determine the impact this may have to their data use. As a result, this means that data versioning systems must communicate a dynamic view of the changes in a system contextualized by the user of that data.

1.5   Conclusion - Thesis Statement

The priority of provenance tracking is understanding where and how a data object was generated, not it's relationship to other data products [24].

1.6   NEEDS EDITING

1.6.1   RDFa

The modifications to implement a machine readable changelog should still remain in a form viewable by a human user. The subjectivity of versioning information means that domain scientists should still be able to interpret changes as they relate to their application. Resource Description Framework in Attributes (RDFa) allows web developers encode information XHTML and HTML documents using standardized ontologies. The developer would use the attributes in RDFa to augment the existing HTML tags to provide extra meaning that a web crawler can extract. The displayed text would remain unchanged signifying that if a changelog could be produced in HTML, then data producers can embed machine readable versioning information into a changelog using RDFa while still leaving it human readable as well.

CHAPTER 2
PREVIOUS WORK

2.1 Spreadsheets

In this project, spreadsheets were chosen for study as they resemble text-like data objects while still maintaining a level of complexities. Though not as well encapsulated as other data format types such as the Hierarchical Data Format (HDF) or Network Common Data Form (NetCDF), spreadsheets provide many helpful tools that scientist favor for quick data storage and distribution over comma separated values (CSV). There also exists other document-like formats that are not discussed in this paper such as eXtensible Markup Language (XML). The initial work was done with the "Noble gas isotope abundances in terrestrial fluids" workbook (Noble Gas) [61]. The "Paragenetic Mode for Copper Minerals" workbook (Copper Data) was used to give better insight into data changes due to collaboration with the data set's author [62].

The Noble Gas data set was initially published on June 11, 2013 and then released a second version on March 8, 2015. Many significant changes were made to the data set between the two versions, which makes this data set particularly challenging to version. The physical structure of the data set changed from eight separate Excel spreadsheets to a single spreadsheet. The second version also trimmed 195 columns to 54 columns in the second release. In addition, many new locations were surveyed and added to the second release. Documentation accompanied the data set explaining different components of the spreadsheet and its usage, but it included no versioning information. This lack of versioning or transitioning information indicates a focus on data usage rather than data maturation, which is not a particularly bad approach. It makes logical sense to simply download the latest data set when it becomes available and not worry about the format of the invalidated data set. This approach convenient for new users of the data as the cost to consumer the new version of the data is the same cost they would have spent to acquire the data in the first place. However, users of the old data are disproportionately effected by

the change in versions since old code and workflows may need to be updated to accommodate the changes in addition to the cost of consuming the architecture of the old data set. In this case, users would need to read the documentation to understand whether 182 from the June data set is still available in the March data and, if it is, in which column it resides in the March spreadsheet. This brings to light the additional concern for the Noble Gas data that the documentation is not easily machine consumable, meaning that all mapping activities will need to be performed manually. Not only is this approach time consuming, but it also does not scale well into larger data sets.

The Copper data set was acquired during the process of a workshop to generate new methods of visualizing mineralogy data, initially on June 8, 2016. The process entailed trying various orders and organization for the data and results in various new versions of the data that depend on varying filtering requirements, acquired on August 21, 2016. Unlike the Noble Gas data set, the Copper Data had no accompanying documentation, since the primary consumers of the data at the workshop were also mineralogy experts. However, this data set had more stable characteristics including physical and logical structure. Only two columns were removed from the transition to the second version, but sixteen new columns were added to the data collection. It also demonstrates a change in orientation with respect to data usage since the previous data set was designed to be distributed for general usage and discovery. In this case, the structure and organization of the data within the set was driven for a specific purpose in the development of more expressive visualizations. As a result, versioning information is driven by developmental needs instead of the other way around with versioning information bridging the gap between software migrations.

The data files from both data sets can easily be tracked using standard version management services such as GIT or SVN. Likewise, there exist comparison tools like Spreadsheet Compare from Microsoft Corporation that can generate diff-like outputs for each of the data sets. In conjunction with commit logs, the comparison outputs provide a basic versioning methodology that describes the data set's evolution. However, these applications rely on human attention and interaction

to operate, and with larger data sets, proper documentation becomes difficult to maintain. With the Copper Data, the demand for new versions of the spreadsheets exceeded the time necessary to document version history as a result of rapid product evolution during the workshop. In consequence, the process to manually commit and annotate changed data impairs the natural progression of scientific development.

2.2   Database Systems

Databases remain the most relied upon technology for storing and searching large quantities of data rapidly. While the dynamic combination of tables means that data bases remain flexible enough to represent complex objects, it also means that they represent a much more complicated case for attribution. Since tables may be combined in different ways to answer complex queries, indexes do not remain constant across requests to the database. The approaches to database versioning typically focus on ensuring the reproducibility of queries to the database. This can often be difficult as with spreadsheets since changes to the content or structure can result in different solutions from the database for the same query even using time stamps. For example, consider the query to select all columns of row A from a database on March 1st, then the database undergoes a schema change to add a new column to the table on April 1st. A subsequent request for all columns of row A would include the new column which does not represent the response on March 1st. In addition, even if the data is timestamped, the time signature is associated with the row and not the schema, meaning that the query may still return row A with the new column with a NULL value, depending on the distribution. The query, not the data, would need to be modified to exclude the new column.

This presents as a challenge because unlike data files and spreadsheets, databases are generally not instanced. Databases often store massive quantities of data and replication of that data to archive snapshots or distribution frequently proves too costly to be feasible. Instead, interaction with the database occurs from a centralized source through transactions. Various methods have been studied to manage changes within these systems focusing primarily on schema versioning, emphasizing data's structural component [63]. This provides a method to enact a transactional rollback

on the database to execute queries in an environment reminiscent of the original execution. The framework of the resulting database environment can become quite complicated as a result of the complexity of the tables representing intricate data objects [64]. This results from the need to manage the time instances of realization, storage, and validity. The datum becomes realized at collection, then stored upon entry into the database, and finally valid until the present or new data replaces it. More recently, new methods have been developed to adjust to the enormous quantities of data populating modern databases, focusing on query citation rather than data citation [65] [66]. Citation by query avoids the complexities involved with referencing data that can grow and move. However, this method relies on the existence of a versioning system for data. This method also recognizes that modifying queries to operate on the current state of the database may often be easier than rolling back transactions or schema to reproduce the results of a query [67]. As a result, to versioning a database system may be more feasible as data size increases by applying methods to the query results and not to the data.

The RRUFF Database is "an integrated database of Raman spectra, X-ray diffraction and chemistry data for minerals" [68]. It features a web accessible change log using the transactional log generated by the database software[1]. As the records in specific tables change, the log reports these changes, supplying persistent access to the modifications made to the RRUFF data. The approach to this alteration information highlights the always on-line approach to modern databases where changes to the data do not constitute a new database. The log demonstrates strong versioning characteristics with not only a breakdown of the change components, but also a commentary on the motivation for the difference. In addition, its HTML structure allows automated web crawlers to systematically consume the version information. With the integration of web ontologies, the change log would also be intelligible to automated agents.

---

[1]`http://rruff.info/index.php/r=rruff_log_display`

2.3   Ontologies

On-line ontologies are a different way of storing data than relational databases that has found significant traction within Semantic Web applications. They form graphs, relating a vocabulary of terms and relationships together to model complex interactions within an application's domain. Since the ontology is represented as a graph, it has more expressiveness than relational databases. The objects no longer need to share uniform structure and fields when entered into the database. Ontologies improve interoperability between scientific data sets by allowing differing data to share a common vocabulary and be comparable. Like other data, ontologies change regularly as definitions and relationships update to better represent their source material. As a connected graph, they easily lend themselves to providing mappings between changes and versions within the ontology. New transitions would be represented as a simple link between new and old concepts. This is particularly important on the Semantic Web since most reasoning and interactions are handled automatically by underlying services. Ontologies, thus, benefit the most when providing both forward and backward mapping as it allows more up to date systems to interact with entities that haven't migrated yet [60]. Incomplete mappings, where transitions exclude either forward or exclude backward mappings, retain value as backward mappings inform traceability and forward mappings communicate advances in the domain. However, the uncertain landscape of web services means that full ontology mappings prove invaluable to making data inter-operable.

The Global Change Master Directory (GCMD) is a metadata repository used by NASA to story records of its available data sets [69]. It employs a keyword ontology to search for Earth science data in NASA data sets. These keywords tag and label datasets into strictly defined categories in order to make them more discoverable [70]. Version 1.0.0 of the GCMD Keywords was published on April 24, 1995, and as of the time of writing, the most recent version of the keywords is 8.4. As can be seen, the naming scheme of the versions changed since the first publication of the keywords. In the initial scheme, each part of the decimal system represented a different level of the GCMD Keyword hierarchy: category, topic, and term, respectively. Incrementing a number in the version name indicates a change

occurring in that level of the ontology. However, this gave very little information on compatibility between versions, and the ontology currently employs a more standard Major.minor release naming scheme.

The data set provides a very interesting case to study because of its history of medium storage. The GCMD originally distributed the keywords in a spreadsheet format, but later migrated onto database services as the scope and demand grew. The data may currently be accessed through a dynamic web service that can provide results in a variety of linked data formats. As a result, it leverages the endeavors made towards the environments mentioned in the previous two sections. The keywords have an accompanying change log, but due to the variety of mediums, the early logs are difficult to interpret. Since they attempt to use web technologies, the keywords each have unique identifiers that can be dereferenced using a Universal Resource Identifier (URI). Attribution, therefore, has mostly addressed by the source material. This is to be expected as a result of curated application of linked data principles. Due to the shift from spreadsheet to databases, there exists a disconnect between the early versions of the ontology and modern editions. The work done in this project will be able to link them and provide a road map through the evolution and migration of the vocabulary as well as guide future evolutions of the keywords.

CHAPTER 3
CONCEPTUAL MODEL

The conceptual model used within this thesis is built around the expression of three core versioning operations: addition, invalidation, and modification. These three activities can be represented by interacting with three types of concepts: versions, attributes, and changes. Versions represent the data entities being compared. These could be two different editions of a book or versions of software. It is important to understand that a version is an abstraction as it can be represented by multiple physical files. In the sections that follow, operations will only consider the interaction between two versions and will be explained later in the chapter. Versions then contain attributes representing a quantity being modified. Specifically for tabular data, attributes would correspond to an identifier that refers to particular rows or columns within the data. Attributes of the two versions are then connected by a change. This link functions as a very general concept which can be subclassed into more informative types such as unit changes, improving the expressiveness of the model beyond PROV's revisionOf concept.

3.1   ADDITION

When a change adds a new attribute to a version, it only needs to refer to version two and its corresponding attribute. The reasoning should be fairly obvious as the attribute never existed in version one, and therefore, there is nothing to refer to and no need to form a relationship between the change and version one. However, by linking the addition change to version one, we address a difficulty with comparing provenance graphs. When two data objects have identical structures, it is difficult to determine what time the objects were added to the dataset and which version they belong to. As a result, determining the compatability of the two objects becomes difficult. The change contributions to the dataset evolution appears naturally using this construction. The resulting model can be seen in Figure 3.1. Some relationships are specifically left out, such as that between Change A and Version 2, to not confuse
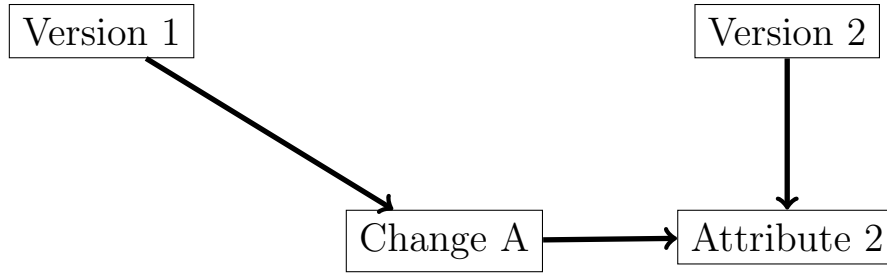
Figure 3.1: Model of the relationships between Versions 1 and 2 when adding an Attribute 2 to Version 2 as a result of Change A

identification of other types of changes. The relationship between Change A and Version 2 can still be implied from Attribute 2.
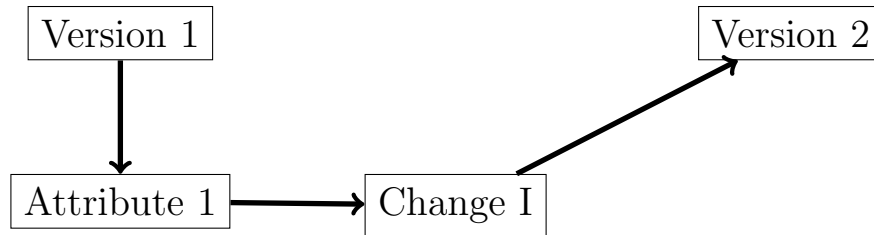
3.2 INVALIDATION

The Invalidation operation corresponds to the delete concept found in other applications. The choice of invalidation over delete results from the policy that, in versioning, data should never be deleted. In practicality, this may not be particularly feasible due to space limitations and relative validity. In either case, the change invalidates an attribute in version one, resulting in version two. Unlike the Addition operation, Invalidation forms a clear relationship between both versions, which can be seen in Figure 3.2. Notice again that since Attribute 1 no longer exists in Version 2, there is no corresponding Attribute 2 to refer to.

From Figure 3.1, we can see the confusion that could result from requiring explicit relationships between versions and changes in both the Addition and Invalidation operations. Linking Change A to Version 2 would create a duplicate connection and provides a mechanism to identify when items specifically enter or leave a version.

3.3 MODIFICATION

The final operation is Modification, and it maps a change from one attribute from version one to its corresponding attribute in version two. The particular type of change in this case is purposely left out in order to allow data producers to subclass and customize the resulting graph to properly reflect the versioning that they desire.

Figure 3.2: Model of the relationships between Versions 1 and 2 when invalidating Attribute 1 from Version 1 as a result of Change I

3.4   MULTIPLE LINKED VERSIONS

Using the construction outlined in the previous three sections, many changes can be compiled together into a graph in a changelog. After all additions, invalidations, and modifications have been compiled into a single graph, a complete mapping from version one to version two may be developed. The orientation of the relationships in the graph allows a flow to be created from attributes in version one to corresponding attributes in version two. Taking version two and performing the same graph construction to a version three results in not only a flow from version two to version three, but also from version one to version three. As a result, the flow can be used to construct a mapping from version one to version three or any future version.
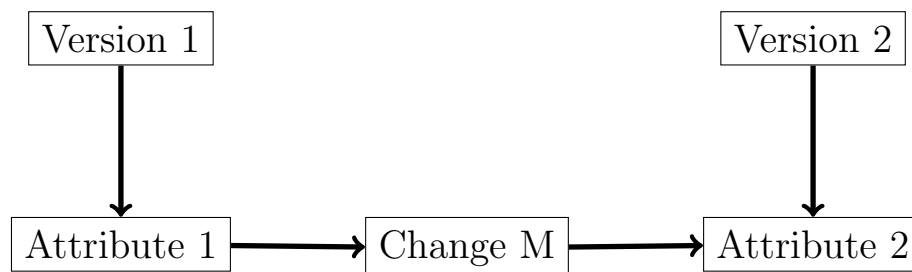
Figure 3.3: Model of the relationships between Versions 1 and 2 when modifying Attribute 1 from Version 1 as a result of Change M, resulting in Attribute 2 from Version 2

CHAPTER 4
VERSIONING SPREADSHEETS

Two datasets were initially studied. The "Noble gas isotopes in hydrocarbon gases, oils and related ground waters" database, These are the most challenging

The Paragenetic Mode for Copper Minerals database produced two versions, one at a workshop on June 8, 2016 and another at a following workshop on August 21, 2016. These both take the form of Excel spreadsheets, which has the benefit of having strict row and column numbering. This allows unique identifiers to be used when referring to individual pieces of data and providing a level of abstraction. The structural changes made to the Copper Dataset resemble those found in the Noble Gas Dataset.

4.1   Provenance Analysis

The first approach to determining the provenance distance for the datasets began with the Noble Gas Dataset. The dataset provides a set of references from which the values were extracted and compiled into the dataset. As a result, a simple provenance mapping was constructed, using PROV, from each reference to its corresponding row in the spreadsheet. After this was done for each version of the dataset, we can generate graphs to compare objects from the two versions like the one found in Figure 4.1. We can tell from the labels that different Activities were used to compile the data entry, but the structure of the graph does not provide any information as to how extensive this change was for this version. Here we can see how the wasRevisionOf relationship would break down in determining the magnitude of change between the two versions. The relationship must, therefore, be expanded in order to provide the desired data necessary to make an evaluation.

4.2   Versioning Comparison

The initial challenge for both of these datasets is producing an appropriate mapping between their previous and latest versions. Through the second compila-
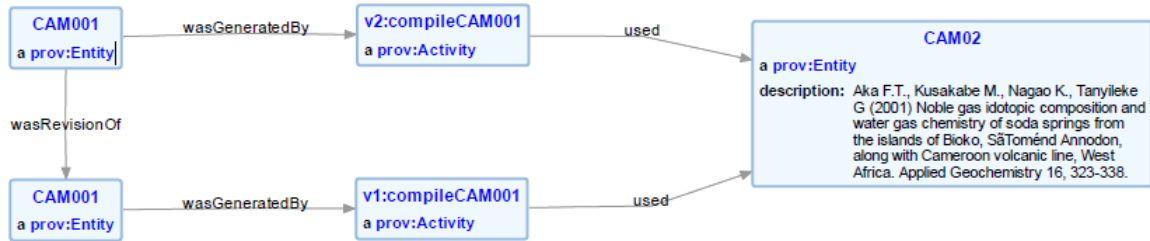
Figure 4.1: Provenance graph for the entry CAM001 entry of the Noble Gas Database. Other than the labels, the structure of each of the data objects is very much the same.

tion process, many columns were removed from the dataset or moved around, and as a result, the identifiers associated with the Attribute in version two did not necessarily correspond to their identifier in version one. The column headings in the Noble Gas Dataset is stored in the spreadsheet as data and makes it difficult for a general system to automate mapping columns between the two versions since any number of rows may contain metadata information. Additionally, while the second release of the Noble Gas Dataset did have a document detailing the organization of the columns contained within the dataset, it did not have any information to map entries from the old dataset to the new dataset. This limits the ability to map and update changes to human actors. Immediately apparent is that any columns that have a mapping from version one to version two means that these columns (Attributes from the model) will only undergo Modification operations since there exists an associated Attribute in both the previous and current version. The conclusion then follows that all remaining Attributes (including both columns and rows) belonging to version one were Invalidated and those belonging to version two were Added. Using this conclusion, we can pre-calculate the Added and Invalidated Attributes and separate any report into changes grouped by operator.

Once a mapping exists between the two versions, a comparison was performed to determine whether an Attribute of the data object changed. In this case, a simple equality operator was used to determine if anything was modified. In practice, more advanced or complicated methods can be used to determine equality. Since, all Additions and Invalidations have already been predetermined, we can output each Modification as we see them. As is common in versioning, the changes

were outputted to a changelog document formatted using HTML. The idea here is that by providing the changelog in HTML, the changes can be made available online and therefore accessible by data consumers. Another benefit of providing a changelog in HTML is that the document can be additionally enriched by RDFa. The changelog document becomes no longer restricted to human consumption, but allows autonomous agents to more intelligently interact with dynamic data systems. The conceptual model detailed in Chapter 3 becomes encoded into the changelog and the graph resulting from the log can be extracted automatically.

The resulting graph provides a structure upon which a flow may be calculated. This becomes an alternative method to determine the provenance distance between two datasets with a much higher fidelity. As mentioned previously, the Change concept is meant to be sub-classed to provide more freedom to represent the particular change bridging the two versions. For example, the He Count entry for the Noble Gas Database changed its units from parts per million to cc STP of given gas specie per cc STP of the total gas. This would be better qualified as a unit change and would be associated with a certain weight in contribution to the total change to the dataset.

CHAPTER 5
DATABASE VERSIONING

The framework for communicating change information resembles the spreadsheet context. However, the main difference is the method to refer to the Attributes as they are mapped across versions. Database tables do not have strict identifiers for rows and columns as spreadsheets do. Row identification relies on indexed keys to uniquely identify entries, but these entries can also be arranged and presented in different orderings depending on the queries used to view the database. From Reference 24, elements within the database do not have to be digitally organized in the same way that it is physically presented.

Databases are also meant to be kept online for extended periods of time. Spreadsheets must be entirely republished in order for a change to the data to be made public. Changes in a database are therefore more sparse in between each version. In order to make changelogs more comprehensive, they can describe changes by day or other increment of time or describe the latest particular subset of time.

Once the changes within the database are encoded into the versioning conceptual model, publishing the changes can follow the methods detailed in Chapter 4

CHAPTER 6
ONTOLOGY VERSIONING

Since there exists an implementation of the GCMD Keywords in RDF, the URIs can be used as references for the Attributes in the concept model. As mentioned in Reference 8, in order for a data consumer to understand how to use a new version of an ontology, they need not only understand what concepts are new and what concepts are old, but also how to map the old ontology onto the new ontology. The mapping then informs the migration of Keyword labeling of datasets between versions.

CHAPTER 7
CONCLUSION

REFERENCES

[1] B. R. Barkstrom and J. J. Bates, "Digital library issues arising from earth science data," 2006.

[2] M. Branco, D. Cameron, B. Gaidioz, V. Garonne, B. Koblitz, M. Lassnig, R. Rocha, P. Salgado, and T. Wenaus, "Managing atlas data on a petabyte-scale with dq2," *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062017, 2008. [Online]. Available: http://stacks.iop.org/1742-6596/119/i=6/a=062017

[3] B. Barkstrom, *Earth Science Data Management Handbook: Users and User Access.* CRC Press, April 2014, vol. 1. [Online]. Available: https://books.google.com/books?id=pI3rTgEACAAJ

[4] S. Burrows, "A review of electronic journal acquisition, management, and use in health sciences libraries," *Journal of the Medical Library Association*, vol. 94, no. 1, pp. 67–74, 01 2006, copyright - Copyright Medical Library Association Jan 2006; Document feature - Graphs; Tables; ; Last updated - 2016-11-09. [Online]. Available: http://search.proquest.com/docview/203517273?accountid=28525

[5] K. Berberich, S. Bedathur, T. Neumann, and G. Weikum, "A time machine for text search," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '07. New York, NY, USA: ACM, 2007, pp. 519–526. [Online]. Available: http://doi.acm.org/10.1145/1277741.1277831

[6] S. Lyons, "Persistent identification of electronic documents and the future of footnotes," *Law Library Journal*, vol. 97, pp. 681–694, 2005.

[7] W. F. Tichy, "Rcsa system for version control," *Software: Practice and Experience*, vol. 15, no. 7, pp. 637–654, 1985.

[8] S.-Y. Chien, V. J. Tsotras, and C. Zaniolo, "Version management of xml documents," in *Selected Papers from the Third International Workshop WebDB 2000 on The World Wide Web and Databases.* London, UK, UK: Springer-Verlag, 2001, pp. 184–200. [Online]. Available: http://dl.acm.org/citation.cfm?id=646544.696357

[9] S. Chacon, *Pro Git*, 1st ed. Berkely, CA, USA: Apress, 2009.

[10] A. Stuckenholz, "Component evolution and versioning state of the art," *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 1, pp. 7–, Jan. 2005. [Online]. Available: http://doi.acm.org/10.1145/1039174.1039197

[11] "Common questions: Ubuntu release and version numbers," Canonical Ltd., accessed: December 12, 2016. [Online]. Available: https://help.ubuntu.com/community/CommonQuestions##Ubuntu% 20Releases%20and%20Version%20Numbers

[12] J. Dijkstra, *On complex objects and versioning in complex environments.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 13–23. [Online]. Available: http://dx.doi.org/10.1007/BFb0024353

[13] P. Cederqvist, R. Pesch *et al.*, *Version management with CVS.* Network Theory Ltd., 2002.

[14] S. Payette and T. Staples, *The Mellon Fedora Project Digital Library Architecture Meets XML and Web Services.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 406–421. [Online]. Available: http://dx.doi.org/10.1007/3-540-45747-X_30

[15] K. S. Baker and L. Yarmey, "Data stewardship: Environmental data curation and a web-of-repositories," *The International Journal of Data Curation*, vol. 4, no. 2, pp. 12–27, 2009.

[16] J. Kovse and T. Härder, "V-grid-a versioning services framework for the grid," in *Berliner XML Tage*, 2003.

[17] K. Holtman, "CMS Data Grid System Overview and Requirements," CERN, Geneva, Tech. Rep. CMS-NOTE-2001-037, Jul 2001. [Online]. Available: http://cds.cern.ch/record/687353

[18] R. Rantzau, C. Constantinescu, U. Heinkel, and H. Meinecke, "Champagne: Data change propagation for heterogeneous information systems," in *In: Proceedings of the International Conference on Very Large Databases (VLDB), Demonstration Paper, Hong Kong*, 2002.

[19] B. Tagger, "A literature review for the problem of biological data versioning," Online, July 2005. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/btagger/LitReview.pdf

[20] U. K. Wiil and D. L. Hicks, "Requirements for development of hypermedia technology for a digital library supporting scholarly work," in *Proceedings of the 2000 ACM Symposium on Applied Computing - Volume 2*, ser. SAC '00. New York, NY, USA: ACM, 2000, pp. 607–609. [Online]. Available: http://doi.acm.org/10.1145/338407.338517

[21] D. Dai, Y. Chen, D. Kimpe, and R. Ross, "Provenance-based object storage prediction scheme for scientific big data applications," in *Big Data (Big Data), 2014 IEEE International Conference on.* IEEE, 2014, pp. 271–280.

[22] B. R. Barkstrom, *Data Product Configuration Management and Versioning in Large-Scale Production of Satellite Scientific Data.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 118–133. [Online]. Available: http://dx.doi.org/10.1007/3-540-39195-9_9

[23] P. Vassiliadis, M. Bouzeghoub, and C. Quix, *Towards Quality-Oriented Data Warehouse Usage and Evolution.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 164–179. [Online]. Available: http://dx.doi.org/10.1007/3-540-48738-7_13

[24] R. Bose and J. Frew, "Lineage retrieval for scientific data processing: A survey," *ACM Comput. Surv.*, vol. 37, no. 1, pp. 1–28, Mar. 2005. [Online]. Available: http://doi.acm.org/10.1145/1057977.1057978

[25] M. Fischer, M. Pinzger, and H. Gall, "Populating a release history database from version control and bug tracking systems," in *Proceedings of the International Conference on Software Maintenance*, ser. ICSM '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 23–32. [Online]. Available: http://dl.acm.org/citation.cfm?id=942800.943568

[26] R. Cavanaugh, G. Graham, and M. Wilde, "Satisfying the tax collector: Using data provenance as a way to audit data analyses in high energy physics," in *Workshop on Data Lineage and Provenance*, Oct. 2002.

[27] P. P. da Silva, D. L. McGuinness, and R. Fikes, "A proof markup language for semantic web services," *Information Systems*, vol. 31, no. 45, pp. 381 – 395, 2006, the Semantic Web and Web Services. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306437905000281

[28] M. Bouzeghoub and V. Peralta, "A framework for analysis of data freshness," in *Proceedings of the 2004 International Workshop on Information Quality in Information Systems*, ser. IQIS '04. New York, NY, USA: ACM, 2004, pp. 59–67. [Online]. Available: http://doi.acm.org/10.1145/1012453.1012464

[29] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. Paulson, "The open provenance model: An overview," in *International Provenance and Annotation Workshop.* Springer, 2008, pp. 323–326.

[30] Y. Liu, J. Futrelle, J. Myers, A. Rodriguez, and R. Kooper, "A provenance-aware virtual sensor system using the open provenance model," in *2010 International Symposium on Collaborative Technologies and Systems*, May 2010, pp. 330–339.

[31] Y. L. Simmhan, B. Plale, and D. Gannon, "Karma2: Provenance management for data-driven workflows," *Web Services Research for Emerging Applications: Discoveries and Trends: Discoveries and Trends*, p. 317, 2010.

[32] F. Casati, S. Ceri, B. Pernici, and G. Pozzi, *Workflow evolution.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 438–455. [Online]. Available: http://dx.doi.org/10.1007/BFb0019939

[33] K. Belhajjame, H. Deus, D. Garijo, G. Klyne, P. Missier, S. Soiland-Reyes, and S. Zednik, *PROV Model Primer*, W3C Working Group, Apr. 2013, 30. [Online]. Available: https://www.w3.org/TR/prov-primer

[34] ——, "Prov model primer," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/NOTE-prov-primer-20130430/

[35] "Prov-overview," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/NOTE-prov-overview-20130430/

[36] K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, S. Miles, J. Myers, S. Sahoo, and C. Tilmes, "Prov-dm: The prov data model," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/REC-prov-dm-20130430/

[37] T. D. Nies, "Constraints of the prov data model," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/REC-prov-constraints-20130430/

[38] P. Missier, L. Moreau, J. Cheney, T. Lebo, and S. Soiland-Reyes, "Prov-dictionary: Modeling provenance for dictionary data structures," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/NOTE-prov-dictionary-20130430/

[39] K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, and J. Zhao, "Prov-o: The prov ontology," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/REC-prov-o-20130430/

[40] L. Moreau, "Prov-xml: The prov xml schema," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/NOTE-prov-xml-20130430/

[41] L. Moreau, O. Hartig, Y. L. Simmhan, J. Myers, T. Lebo, K. Belhajjame, S. Miles, and S. Soiland-Reyes, "Prov-aq: Provenance access and query," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/NOTE-prov-aq-20130430/

[42] J. Cheney and S. Soiland-Reyes, "Prov-n: The provenance notation," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/REC-prov-n-20130430/

[43] "Ssemantic of the prov data model," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/NOTE-prov-sem-20130430/

[44] S. Miles, C. M. Trim, and M. Panzer, "Dublin core to prov mapping," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/NOTE-prov-dc-20130430/

[45] "Linking across provenance bundles," April 2013, accessed: December 17, 2016. [Online]. Available: https://www.w3.org/TR/2013/NOTE-prov-links-20130430/

[46] I. Suriarachchi, Q. G. Zhou, and B. Plale, "Komadu: A capture and visualization system for scientific data provenance," *Journal of Open Research Software*, vol. 3, no. 1, mar 2015. [Online]. Available: http://dx.doi.org/10.5334/jors.bq

[47] X. Ma, J. G. Zheng, J. C. Goldstein, S. Zednik, L. Fu, B. Duggan, S. M. Aulenbach, P. West, C. Tilmes, and P. Fox, "Ontology engineering in provenance enablement for the national climate assessment," *Environmental Modelling & Software*, vol. 61, pp. 191 – 205, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364815214002254

[48] C. Tilmes, P. Fox, X. Ma, D. L. McGuinness, A. P. Privette, A. Smith, A. Waple, S. Zednik, and J. G. Zheng, *Provenance Representation in the Global Change Information System (GCIS)*, ser. Lecture Notes in Computer Science.   Santa Barbara, CA, USA: Springer Berlin Heidelberg, June 2012, vol. 7525, ch. Provenance and Annotation of Data and Processes, pp. 246–248. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34222-6_28

[49] X. Ma, P. Fox, C. Tilmes, K. Jacobs, and A. Waple, "Capturing provenance of global change information," *Nature Clim. Change*, vol. 4, no. 6, pp. 409–413, Jun 2014, commentary. [Online]. Available: http://dx.doi.org/10.1038/nclimate2141

[50] A. Capiluppi, P. Lago, and M. Morisio, "Evidences in the evolution of os projects through changelog analyses," in *Taking Stock of the Bazaar: Proceedings of the 3rd Workshop on Open Source Software Engineering*, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani, Eds., May 2003, citation: Capiluppi, A., Lago, P., Morisio, M. (2003). ?Evidences in the evolution of OS projects through Changelog Analyses.? in Feller, P., Fitzgerald, B., Hissam, B. Lakhani, K. (eds.) Taking Stock of the Bazaar: Proceedings of the 3rd

Workshop on Open Source Software Engineering ICSE?03 International Conference on Software Engineering Portland, Oregon May 3-11, 2003. pp.19-24.. [Online]. Available: http://roar.uel.ac.uk/1037/

[51] D. German, "Automating the measurement of open source projects," in *In Proceedings of the 3rd Workshop on Open Source Software Engineering*, 2003, pp. 63–67.

[52] K. Chen, S. R. Schach, L. Yu, J. Offutt, and G. Z. Heller, "Open-source change logs," *Empirical Softw. Engg.*, vol. 9, no. 3, pp. 197–210, Sep. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:EMSE.0000027779.70556.d0

[53] K. Herzig and A. Zeller, "Mining cause-effect-chains from version histories," in *2011 IEEE 22nd International Symposium on Software Reliability Engineering*, Nov 2011, pp. 60–69.

[54] C. Tilmes, Y. Yesha, and M. Halem, "Distinguishing provenance equivalence of earth science data," *Procedia Computer Science*, vol. 4, pp. 548 – 557, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050911001153

[55] B. Cao, Y. Li, and J. Yin, "Measuring similarity between gaphs based on the levenshtein distance," *Applied Mathematics & Information Sciences*, vol. 7, no. 1L, pp. 169–175, 2013.

[56] X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," *Pattern Analysis and Applications*, vol. 13, no. 1, pp. 113–129, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10044-008-0141-y

[57] W. Goddard and H. C. Swart, "Distances between graphs under edge operations," *Discrete Math.*, vol. 161, no. 1-3, pp. 121–132, Dec. 1996. [Online]. Available: http://dx.doi.org/10.1016/0012-365X(95)00073-6

[58] Y. Ma, M. Shi, and J. Wei, "Cost and accuracy aware scientific workflow retrieval based on distance measure," *Information Sciences*, vol. 314, no. C, pp. 1–13, Sep. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2015.03.055

[59] M. D. Flouris, "Clotho: Transparent data versioning at the block i/o level," in *In Proceedings of the 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies (MSST 2004)*, 2004, pp. 315–328.

[60] M. Klein and D. Fensel, "Ontology versioning on the semantic web," in *Stanford University*, 2001, pp. 75–91.

[61] B. Polyak, E. Prasolov, I. Tolstikhin, L. Yakovlev, A. Ioffe, O. Kikvadze, O. Vereina, and M. Vetrina, "Noble gas isotope abundances in terrestrial

fluids," 2015. [Online]. Available:
https://info.deepcarbon.net/vivo/display/n6225

[62] S. Morrison, R. Downs, J. Golden, A. Pires, P. Fox, X. Ma, S. Zednik,
A. Eleish, A. Prabhu, D. Hummer, C. Liu, M. Meyer, J. Ralph, G. Hystad,
and R. Hazen, "Exploiting mineral data: applications to the diversity,
distribution, and social networks of copper mineral," in *AGU Fall Meeting*,
2016.

[63] J. F. Roddick, "A model for schema versioning in temporal database systems,"
Australian Computer Science Communications, vol. 18, pp. 446–452, 1996.

[64] P. Klahold, G. Schlageter, and W. Wilkes, "A general model for version
management in databases," in *Proceedings of the 12th International
Conference on Very Large Data Bases*, ser. VLDB '86.   San Francisco, CA,
USA: Morgan Kaufmann Publishers Inc., 1986, pp. 319–327. [Online].
Available: http://dl.acm.org/citation.cfm?id=645913.671314

[65] S. Pröll and A. Rauber, "Citable by design - A model for making data in
dynamic environments citable," in *DATA 2013 - Proceedings of the 2nd
International Conference on Data Technologies and Applications, Reykjavík,
Iceland, 29 - 31 July, 2013*, 2013, pp. 206–210. [Online]. Available:
http://dx.doi.org/10.5220/0004589102060210

[66] M. Helfert, C. Francalanci, and J. Filipe, Eds., *DATA 2013 - Proceedings of
the 2nd International Conference on Data Technologies and Applications,
Reykjavík, Iceland, 29 - 31 July, 2013*.   SciTePress, 2013.

[67] S. Proell and A. Rauber, "Scalable data citation in dynamic large databases:
Model and reference implementation," in *IEEE International Conference on
Big Data 2013 (IEEE BigData 2013)*, 10 2013.

[68] B. Lafuente, R. T. Downs, H. Yang, and N. Stone, "1. the power of databases:
The RRUFF project," in *Highlights in Mineralogical Crystallography*,
T. Armbruster and R. M. Danisi, Eds.   Walter de Gruyter GmbH, 2015, pp.
1–30. [Online]. Available: http://dx.doi.org/10.1515/9783110417104-003

[69] Z. B. Miled, S. Sikkupparbathyam, O. Bukhres, K. Nagendra, E. Lynch,
M. Areal, L. Olsen, C. Gokey, D. Kendig, T. Northcutt, R. Cordova,
G. Major, and N. Savage, "Global change master directory: Object-oriented
active asynchronous transaction management in a federated environment
using data agents," in *Proceedings of the 2001 ACM Symposium on Applied
Computing*, ser. SAC '01.   New York, NY, USA: ACM, 2001, pp. 207–214.
[Online]. Available: http://doi.acm.org/10.1145/372202.372324

[70] "Keyword faq," Earthdata, 2016, accessed: December 12, 2016. [Online].
Available: https://wiki.earthdata.nasa.gov/display/CMR/Keyword+FAQ