Metadata Management in the Support of
Data Warehouse Development

by

Eric Klovning

A Research Paper
Submitted in Partial Fulfillment of the
Requirements for the
Master of Science Degree
in
Technology Management

Approved: 3 Semester Credits

Research Adviser

The Graduate School
University of Wisconsin-Stout

July, 2008
ABSTRACT

The purpose of this study is to identify areas for improvement in the metadata management processes of a data warehouse development group within the risk management department of a large commercial bank. This paper will give an overview of the role metadata plays in data warehouse development. Current and emerging practices in metadata management will be identified. The current practices of the development group will then be compared and contrasted to current practices identified through a review of literature. After a quantitative analysis is performed recommendations for process improvements will be suggested. This research may enable the bank to increase productivity within the development group which will result in increased analytical capability and may ultimately give the bank competitive advantages.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>v</td>
</tr>
<tr>
<td>Chapter I: Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Statement of the Problem</td>
<td>1</td>
</tr>
<tr>
<td>Purpose of the Study</td>
<td>2</td>
</tr>
<tr>
<td>Assumptions of the Study</td>
<td>2</td>
</tr>
<tr>
<td>Definition of Terms</td>
<td>3</td>
</tr>
<tr>
<td>Limitations of the Study</td>
<td>4</td>
</tr>
<tr>
<td>Methodology</td>
<td>4</td>
</tr>
<tr>
<td>Chapter II: Literature Review</td>
<td>5</td>
</tr>
<tr>
<td>Data Warehousing</td>
<td>5</td>
</tr>
<tr>
<td>The Role of ETL</td>
<td>6</td>
</tr>
<tr>
<td>System Development Life Cycle</td>
<td>6</td>
</tr>
<tr>
<td>What is Metadata</td>
<td>9</td>
</tr>
<tr>
<td>The Card Catalog Analogy</td>
<td>9</td>
</tr>
<tr>
<td>Classifying Metadata</td>
<td>10</td>
</tr>
<tr>
<td>Why Metadata Matters</td>
<td>11</td>
</tr>
<tr>
<td>Business Definition Matrix</td>
<td>14</td>
</tr>
<tr>
<td>Logical Data Map</td>
<td>14</td>
</tr>
<tr>
<td>Challenges and Cautions</td>
<td>15</td>
</tr>
<tr>
<td>Chapter III: Methodology</td>
<td>17</td>
</tr>
<tr>
<td>Research Methodology</td>
<td>17</td>
</tr>
<tr>
<td>Subject Selection and Description</td>
<td>17</td>
</tr>
</tbody>
</table>
List of Tables

Table 1: Metrics in Logical Data Map ................................................................. 26
Table 2: Derived Metric Business Rule Errors ....................................................... 27
Table 3: Derived Metric Business Rule Child Errors ............................................ 28
Table 4: Database Metrics .................................................................................... 29
Table 5: Database Metrics by Data Lineage .......................................................... 29
Chapter I: Introduction

Introduction

This study focuses on evaluating and improving the metadata management processes of a data warehouse development group that supports the risk management department in one of the 10 largest banks in the United States. The Bank provides services in over 20 states including lending, credit card, merchant services and mortgage banking.

Since it is essential that the business analysts in the risk management department have the ability to track numerous performance metrics of the credit products offered by the bank, a data warehousing group was created. This group is somewhat unique in that they are not part of the bank’s Information Technology department but instead they report to the risk management department. The department expects their dedicated development group to deliver projects faster than the corporate IT department could and also to provide increased value throughout the life of these reporting systems since the development group maintains a closer relationship with the risk management department and is expected to retain business line specific knowledge. The development group has been successful in meeting the needs of the risk management department. Evidence of this is the group’s increased headcount over the last 3 years.

Statement of the Problem

To meet ever increasing regulatory requirements, mitigate risk and remain competitive with financial products offered by other institutions The Bank must maximize the effectiveness of its business analysts. Data warehousing systems within the bank play a key role in enabling the analysts. This functionality is so critical that a
dedicated development group has been created within the risk management department. The development group may be reaching a point of diminishing returns, were each additional resource will no longer yield a proportional increase in productivity.

The development group’s current documentation processes used to collect and manage metadata such as business rules and reporting requirements is labor intensive and error prone. Documentation issues such as circular references, invalid or null references and failure to comply with defined naming conventions contribute to programming bugs, unnecessary system complexity and delays in system deployment. Since the data warehouse infrastructure is so critical to the analysts these issues ultimately limit the bank’s ability to quickly respond to changing regulatory requirements and the ever changing nature of the economy.

Purpose of the Study

The purpose of this study is to (1) Identify current and emerging practices in the discovery and management of domain specific metadata used to support data warehouse development, (2) evaluate the effectiveness of the development group’s current metadata management processes, (3) propose suggestions for enhancing the development group’s efficiency and effectiveness by applying lessons learned from this research.

Assumptions of the Study

It is assumed that additional resources can not be added to the individual project teams solely to address the issues or implement suggestions of this study. Realization of increased efficiency and effectiveness must be possible with the current staffing levels.
Definition of Terms

Data Warehouse. A subject oriented, integrated, time variant and non volatile collection of data that supports the decision making process. (Inmon, Imhoff & Sousa, 2000)

ETL. Extract, transform and load. Extract copies the required data from the source. Transform can include merging data from multiple sources, cleansing duplicates and applying additional business rules. Load consists of transferring the quality assured data to a presentation area where end users can access it. (Kimball & Ross, 2002)

Four Ds of Data. A four step process to identify and understand critical data within an organization. The steps include discover, document, design and do. (“The Four Ds of Data,” 2008)

Metadata. Contextual information about data including business definitions, rules for creating the data, valid formats, system of record and ownership. (Adelman, Moss & Abai, 2005)

Spiral Model. A system development model that breaks projects into multiple iterations based on individual risks. Phases include determine objectives, identify risks, develop/test prototype and plan next iteration. Well suited for projects where initial requirements are incomplete. (Kuhl, 2002)

System Development Life Cycle (SDLC). A process used to create computing systems. Typically includes requirements gathering, analysis, design, programming, testing, integration, and implementation. (Inmon et al, 2000)
**System of Record.** The definitive source of data. It is often used to resolve data redundancy. If two or more systems have conflicting values for the same element the system of record takes precedence. (Inmon et al, 2000)

**Waterfall Model.** A classical system development model consisting of stages including requirements, design, development, testing and deployment. Well suited for projects with stable and clearly understood requirements. (Kuhl, 2002)

*Limitations of the Study*

The study is limited to the identification of existing and emerging industry practices in metadata management, evaluation of the effectiveness of the development group’s current document management processes and suggesting opportunities for improvement. New process implementation and analysis of effectiveness of changes are beyond the scope of this study.

*Methodology*

This study is based on a quantitative analysis of the development group’s project documentation generated during development of System XYZ. The project documentation will be compared and contrasted with current industry practices as summarized in the literature review.
Chapter II: Literature Review

*Data Warehousing*

Information is one of an organization’s most important assets. Within an organization information often resides in two forms (Kimball & Ross, 2002). The first is within the operational systems or systems of record. These are the systems that users enter data into while conducting day to day business operations. The second form in which information resides is the data warehouse. This form supports data retrieval for analysis and decision support purposes.

Data warehousing is a term that is often misunderstood due to inconsistencies in its definition (Kimball & Caserta, 2004). Some of the largest discrepancies are a result of inconsistencies in the classification of the components that make up the data warehouse, particularly in regard to physical design and implementation. Inmon et al (2000) defines a data warehouse as a subject oriented, integrated, time variant and non volatile collection of data. Subject oriented data is related to specific business processes. Integration implies that data was sourced from a variety of systems then merged into the data warehouse. Time variant indicates data has context and meaning based on the time period with which it corresponds. Non volatile indicates that once data is loaded it persists and is not removed. A simpler but still accurate definition of a data warehouse is a system that extracts and transforms source data and delivers it in a format that supports querying and analysis to enable decision making (Kimball & Caserta, 2004).

The goal of any data warehousing project is to make data available and easily accessible to people in the organization (Kimball & Ross, 2002). To do this the data warehouse must deliver data that is understandable and consistent. Attributes from one
area of the business, such as how the organization defines a customer, should match the
definitions of the same attribute from other areas. If multiple attributes share the same
name they must have the same meaning. Attributes that do not share the same meaning
cannot have the same name. These attributes and their common definitions must be
published and made available to the users.

The Role of ETL

Multiple internal and external sources contribute the data that comprises the data
warehouse (Adelman, Moss & Abai, 2005). This data must be cleansed and transformed
to provide fast delivery of data that supports the organization's analytical requirements.
For most data warehouse projects the ETL process will account for at least 70 percent of
the project timeline (Kimball & Caserta, 2004). The level of source system data cleansing
and transformation can vary from trivial to complex (Adelman et al, 2005). A simple
requirement might consist of adding zip codes to customer addresses. A more complex
requirement could consist of comparing customer records from multiple source systems
to identify redundant data that belongs to the same customer and subsequently load only
one record into the data warehouse.

System Development Life Cycle

Projects such as the creation of a data warehouse are by their nature large and
complex. Fortunately a number of system development life cycle (SDLC) models exist.
The oldest and best known SDLC model is the waterfall (Kay, 2002). It consists of
several stages where the output from one stage is the input to the next stage. A typical
waterfall project might consist of the following stages:
• Requirements definition. End user needs are analyzed to determine required functionality.

• Design. Features and functionality defined in detail. Business rules, process diagrams and other documentation created.

• Development. Actual code is written.

• Integration and Testing. System is deployed in a testing environment, technical resources and end users evaluate functionality, check for errors.

• Deployment. System is put into production.

While the waterfall model is used effectively in construction and manufacturing it is less effective for software development (Rajlich, 2006). The weakness of the waterfall model is its dependence on accurate requirements specifications prior to the design phase. During the development phase of a data warehousing project requirements can be volatile. They may not be fully known in advance and additional requirements are often added throughout the project.

An alternative to the waterfall model is the spiral model which focuses on the identification and management of risks ("Lifecycle Models," 2006). Under the spiral model a project is broken into multiple iterations that each deal with one or more specific risks. The first phase of spiral iteration begins with determining options and constraints for resolving a risk. Items with the greatest amount of risk exposure are usually dealt with first. In the second phase prototypes are often created to assist in evaluating the effectiveness of the options and constraints. Early iterations may generate paper models or simple prototypes with very limited functionality while later iterations may produce prototypes that may mimic a fully functional system. After reviewing the results of the
prototyping efforts a decision is made to either continue with the current approach or to choose a different option. Even if the initial approach is discarded the knowledge gained from building the prototype will be beneficial in determining other options. In the final phase of the spiral model the results are reviewed with the customer. If the outputs of the iteration are approved the cycle begins again with the next highest risk exposure item. If the iteration is not approved a decision may be made to evaluate new options for the specific risk. The steps in this cycle continue until the system is complete or the risks are determined to be too great and the project is canceled. A key advantage that the spiral model has over the waterfall model is its ability to handle changes early in the project where they are least expensive. Since complete system design and development have not yet taken place under the spiral model the extensive rework that would be necessary under the waterfall model can be avoided.

The iterative nature of the data warehouse development life cycle makes it substantially different from the classic waterfall model. Inmon (2000) refers to the data warehouse development life cycle as the "CLDS" due to its almost inverse relationship to the traditional waterfall SDLC. A typical data warehouse development life cycle will begin with an initial discovery process. This process will reveal the data used in an organization plus its associated metadata ("The Four Ds of Data," 2008). Findings such as the origin of the data, where it is stored and how it is used will be documented. Since this information is often technical and very detailed in nature is should be documented using terminology that is understandable to the business users. Once the development group familiarizes themselves with the data the formal design efforts can begin (Inmon, 2000). Several iterative development sessions may have to take place before the
requirements for the ETL process are fully understood. The metadata that is discovered and collected during this process provides critical context to the data that makes up the content of the data warehouse.

*What is Metadata*

Metadata becomes a very important element in the data warehouse (Inmon, 2000). Not only does it shape the ETL process but it also enables the business users to locate, understand and use the data once it is loaded into the data warehouse. One of the simplest definitions of metadata states that metadata is data about data. While this is true it is so vague that it provides little value in understanding the critical role metadata plays in data warehouse development. A more specific and useful definition describes metadata as contextual information about data that includes business definitions, valid formats and the system of record (Adelman et al, 2005). Another way to define metadata is that it is everything about data needed to promote its administration and use (Inmon et al, 2000). These definitions assist both the business user and the technical professional to better comprehend and identify the types of metadata they are already working with. It soon becomes obvious that business users are surrounded by domain specific metadata that adds context to the data they currently use and assists them in efficiently and effectively processing it and communicating with others. Metadata is very valuable to the business because it facilitates the understanding of data. Without this understanding the data would be useless.

*The Card Catalog Analogy*

A simple analogy to data warehouse metadata is the card catalog of a library (Stephens, 2003). Although most libraries now use computers instead of a physical card
catalog the underlying premise remains the same. A card catalog indexes information about the library’s holdings, including but not limited to books, journals, magazines, newspapers, etc. Some of the attributes captured in a card catalog include title, author, keywords and physical location. Once these attributes are defined and captured a relatively simple informational hierarchy soon emerges which enables library patrons to use the card catalog to do extensive relationship analysis of the metadata. In a data warehousing environment the assets would not be books and periodicals but instead might be entities, attributes, transformations, business rules and data stewards. For each of these items the description, business meaning, owner and any other attributes of interest would be captured and retained in a metadata repository.

Classifying Metadata

One basic distinction between different types of metadata is the differentiation between business and technical metadata (Inmon et al, 2008). Technical metadata is often used for data warehouse design, development and maintenance. Examples can include things such as the names of tables in a database or the status of current data warehouse activities such as the run time of an ETL process used to load the data warehouse. This data often exists in a format that would appear very cryptic to most end users and is of little use or interest to them. Business metadata on the other hand is valuable to the business person and can be used to support their daily operations. This data must exist and be expressed in the language of the business person. It provides context and meaning to the data that it accompanies and enables the business person to understand and interpret the data. Examples might include a rule or formula to calculate an account balance or a list of abbreviations and descriptions that represent an account’s status.
Business metadata is an important area to address because it supports the most important segment of data warehouse stakeholders, the users (Mundy, Thomthwaite & Kimball, 2006). Most users cannot get this information without assistance. End users often do not have the technical ability to find and retrieve the metadata required to understand the contents of the data warehouse.

A second critical distinction between types of metadata is the distinction between structured and unstructured data. Metadata that is structured has regular occurrences in a prescribed format (Inmon et al, 2008). This is true of the card catalog example. Each card would contain attributes such as title, author and call number and each card would have a similar physical layout. Structured metadata also exists in the data warehouse. The table names, column names and the formats or data types of the columns all exist in an organized and controlled format. Unstructured data does not have a predictable layout or structure and can exist in any format. An example is free form text like that found in the body of an email or in a written report. While no generalization is absolute, most structured metadata is technical metadata and most unstructured metadata is business metadata.

*Why Metadata Matters*

Without metadata the utility of business data is greatly reduced. Metadata brings context and meaning to business data (Adelman et al, 2005). The values 24, 38 and 41 have no meaning by themselves, however if you know the context is a company’s annual sales in billions of dollars you could see sales have increased. Without meaning data cannot be evaluated or used to make decisions. Without metadata and the context and
meaning it provides it becomes difficult for technicians to support systems and for business users to use the systems.

Some typical questions that may arise in a data warehousing environment that metadata can answer are as follows (Seiner, 1999).

- What entities/attributes exist and how are they defined?
- What is the standard definition for a certain attribute?
- How is the data value determined?
- Has the data value always been determined this way?
- Who do I contact if I have a question about the definition and use of data?
- What source system supplied the data?
- What changes occur in the movement of data from source to target?
- What actions are taken with data exceptions?
- What existing reports or queries give the results I need?

Answers to questions such as those above can serve multiple audiences (Inmon et al, 2000). This type of information is useful not only to the business users but also to the ETL developers. If changes are to be made to a system a developer will need to know what affect that change will have across the system. Metadata can assist the ETL developer in performing an impact analysis.

As end users and technicians interact with the data warehouse they need to retrieve descriptive information to better understand areas they are unfamiliar with or to find an authoritative source of information to resolve disputes about data usage or business rule interpretation. Since metadata adds descriptive context to an object it can be used to assist in locating and retrieving that object (Franks & Kunde, 2006). Locating and
using data without the help of metadata would be analogous to trying to find a book in a library without the aid of a card catalog or without knowing the title of the book or the author's name (Adelman et al., 2005). In data warehousing environment that lack metadata, users are often unaware of and unable to find existing reports that have been created and validated to meet their reporting needs. This leads them to create and maintain their own library of redundant queries and reports (Seiner, 1999). In such organizations with large and complex infrastructures metadata can enhance information discovery (Franks & Kunde, 2006). It allows for consistent representation of data which enables employees to locate and use the data. Since organizations can now find, reuse and repurpose existing data they save the time, expense and opportunities for error involved in recollecting it. This increased speed and accuracy along with reduced cost of information retrieval speeds the decision making process.

Once an organization recognizes the benefits of harnessing metadata, whether it is in support of a data warehousing project or simply to support current operations, the challenge is now to locate the metadata and persist it in a way that enables people to find and use it. One of the richest sources of metadata within any organization is the people in the organization (Seiner, 1999). Whenever data is touched there are unwritten rules that influence how data is defined, interpreted and processed. Metadata discovery is primarily a process of interviewing business users and technical people (Mundy et al., 2006). One on one interviews are often more effective than group interviews as they allow each person to communicate their individual experiences with and knowledge of data operations. These interviews are also often less demanding of the business user’s time as they can usually be accomplished in about an hour. At the end of the interview it is
critical to summarize the discussion. This gives both parties a chance to review the subject matter discussed and to clarify misunderstandings or address omissions.

**Business Definition Matrix**

Business definitions are a critical requirement in building the data warehouse. They can be captured in a business definition matrix (Kimball & Caserta, 2004). Creation of a business definition matrix does not require extensive technical infrastructure. A simple three column spreadsheet would suffice. A basic business definition matrix contains three main components. The first components are physical names. Although the physical column names might not be exposed to the business users the ETL process needs this information to relate the physical database objects to the corresponding business attributes. The second component is the business name. The business name is a translation of the physical column name to a format that provides context and meaning to the business users. The business name is often the representation that is used to present data elements to the users. The final component is the business definition. The business definition is a sentence or two that describes the attribute. If users cannot provide a business definition that may be an indicator that the attribute has no analytical value and should be excluded from the data warehouse. Once the key business definitions have been captured the logical modeling process can begin.

**Logical Data Map**

The logical data map is the cornerstone of the ETL development process (Kimball & Caserta, 2004). It documents the data lineage from its origin in the source system, through the application of business rules and transformations during the ETL process, and finally to its ultimate location in the data warehouse. The logical data map serves
several purposes during data warehouse development. It first serves as a functional specification and is used to create the ETL jobs. It is then used as a reference document as users test and validate data during Quality Assurance and User Acceptance Testing activities. Once the system is deployed the logical data map remains an important reference document both for the technical team who oversee the maintenance of the data warehouse ETL processes and for the end users to answer questions about data lineage.

A spreadsheet is a simple and practical tool to capture the key elements required to build the logical model (Mundy et al, 2006). Appendix A is an example spreadsheet that captures key elements needed to build the logical model. The spreadsheet can easily be modified to include other attributes that will be required during physical database design. Information such as source system, business rules, sample values and comments can be captured. Upon completion the logical data mapping spreadsheet will contain the source to target data lineage required to begin building the ETL jobs.

**Challenges and Cautions**

Metadata has been referred to as the Bermuda Triangle of data warehousing (Mundy et al, 2006). Just as the Bermuda Triangle, a region in the Atlantic Ocean off the Southeast coast of the United States that has allegedly been the site of numerous mysterious vanishings of ships and planes, metadata management projects can easily evolve into overwhelming and seemingly endless projects with unbounded scope and limited returns. The challenge of metadata management is not unique to data warehousing. While the software industry recognizes metadata management is a difficult problem initial efforts to build metadata repositories are often marginally successful. Even initial successes often fade as repositories go unmaintained until they become too
outdated to be of any business value. If metadata is not captured during system
development it would be a daunting task to try and reconstruct it after the fact (Inmon et
al, 2000). Funding and resources, including the business users and developers who were
involved in the system’s creation may not be available. Changes to a system of record or
to business processes can create multiple versions of metadata across time. Even if an
effort were made to retroactively capture metadata changes such as these can make it
impossible to create a definitive metadata repository that applies to all historical data.
Chapter III: Methodology

The Bank must maximize the effectiveness of its business analysts in order to mitigate risk, meet ever increasing regulatory requirements and remain competitive with other financial institutions. Data warehousing systems are so critical to the day to day duties of the business analysts that a dedicated development group exists within the risk management department to create and maintain these systems. The development group’s current documentation processes are labor intensive and error prone which cause unnecessary system complexity and delays in system deployment.

Research Methodology

The research methodology for this field problem begins with a review of literature pertaining to data warehousing, system development life cycle models and metadata as it relates to data warehousing. Next a quantitative analysis of documentation from System XYZ is conducted. Finally the findings from the analysis of the System XYZ documentation are compared and contrasted to current data warehousing practices as revealed by the literature review. The results of this comparison will identify strengths as well as opportunities for improvement in the development group’s metadata management processes.

Subject Selection and Description

The data warehouse system and supporting documentation to be analyzed was selected from among the existing systems maintained by the development group. Since an analysis of each system would not have been practical System XYZ was selected for three primary reasons.
• Supporting documentation exists for System XYZ. This is not true of all systems maintained by the development group.

• System XYZ is a relatively mature and stable system.

• System XYZ was delivered in six major releases instead of a single release to production. This is unique to System XYZ and therefore will test the effectiveness of the development group's processes to maintain documentation over multiple releases.

Business Processes

The logical data map is the primary document used throughout the development process. It is a spreadsheet that contains one row for each reporting metric. That row contains the name of the metric plus many other attributes, the most critical to this study being the business rule and descriptions of source system data used to create the metric. Appendix A contains a logical data map that is similar to the logical data maps used by the development group.

Content in the logical data map is added, updated and sometimes deleted throughout the system development life cycle. As new metrics emerge additional rows are added to the logical data map which contains the name of the new metric as well as its business rule and data describing its source system. If a metric's business rule contains other metrics the logical data map must be reviewed to ensure those child metrics exist. Appendix E summarizes the process used to manage the content of the logical data map.

The development group does not strictly adhere to any single system development life cycle however their development processes tend to be iterative in nature. Once a new project begins the discovery and documentation processes take place concurrently. Business processes are analyzed to find essential data entities then the relationships
between the entities are identified and analyzed. These findings are documented primarily in the logical data map. Once a critical mass of business knowledge is captured a sample data set is analyzed and the findings are compared to the contents of the logical data map. This will reveal discrepancies in the documentation or oversights in the discovery phase. Any errors are corrected and the logical data map is again compared to a sample data set. These processes are summarized in Appendix B.

Once the development group and the business lines have an adequate understanding of the domain the design and development phases can begin. In these phases a physical database is created and ETL code is written to process the source data and create the required reporting metrics. Again this is done in smaller units of work using an iterative process. An ETL component is written and run against a sample set of data and the actual results are compared to the expected results. If discrepancies are found the cause is identified and remedied which can include making modifications to the ETL code and to the business rules in the logical data map. Appendix C outlines the design and development process.

Once the development process is complete and all reporting metrics have been created the system is ready for formal testing by the business users. At this stage the ETL code runs against a data set similar to what will exist in the production environment. If the new code executes successfully the business users will compare the actual result against the expected result. Any discrepancies will cause the test to fail. If this occurs the project may iterate through discovery and design as well as development before returning to user acceptance testing. During these iterations the ETL code will be modified and the
logical data map may have to be updated. User acceptance testing is outlined in Appendix D.

**Data Collection Procedures**

The researcher collected two sets of data. The first set consisted of all reporting metrics from the System XYZ database that were available to the business users. While the techniques used to collect this data will be platform specific, this data set was obtained by querying the database’s system catalog and returning only the column names accessible to the business user security role. The second data set consists of all reporting metrics and their definitions retrieved from the System XYZ logical data map that was published to the business users. Since the logical data map exists in spreadsheet format the first data set was added as a new worksheet within the spreadsheet file that contained the logical data map.

**Data Analysis**

The two data sets were analyzed to identify:

- Metrics that exist in both data sets
- Metrics that exist in only one data set
- Metrics in the logical data map whose business rule contained references to child metrics that did not exist in the logical data map

The first two comparisons listed above were relatively simple comparisons and were performed using Microsoft Excel with built in functions such as vlookup and countif as well as Excel pivot tables. Since the only attributes required for the comparisons were the name of the metrics and a description of their data lineage these comparisons were simple matching exercises based on discrete values in individual
columns. Once the two sets were compared and metrics were identified as being common to both sets or unique to either set counts and summary data could easily be assembled using pivot tables.

The third comparison was more complex and required additional data processing. In this step the business rule for each derived metric was evaluated to see if it contained one or more child metrics. Since the business rule is a text string, or unstructured metadata, the analysis cannot easily be performed with native spreadsheet functionality. A three step process was developed to conduct this analysis. Since the logical data map has no means of enforcing formatting conventions on the business rule the first step was to identify characters that indicate word breaks. These characters would serve as delimiters and would indicate positions where the business rule should be split into multiple rows. A brief analysis of the business rules revealed that in addition to spaces candidates for delimiters included symbols representing mathematical operations, single and double quotes, brackets, braces, parenthesis and ascii characters for carriage return and line feed. An example business rule for metric A might be (A=true if B=30 or C>10 else false). This string would parse into ten rows and the data set would have two columns. The first column would contain an A to represent the row is derived from metric A’s business rule. The second column would represent the term extracted from the string. Those values are as follows: A, true, if, B, 30, or, C, 10, else, false. Once the string was parsed the second step was to identify and eliminate noise words. A list of noise words quickly emerged after analyzing a distinct list of all parsed terms. Some examples included program flow words such as if, then, else, greater, less, true and false. Any rows where the parsed term could be converted to an integer value were also excluded. Self
references were removed as well. In the example above the ten rows would be reduced to two rows, A, B and A, C indicating metric A has dependencies on metrics B and C. The final step was to join the two-column parsed data set to the metric names in the logical data map. Any rows that did not return a match would either be noise words that escaped the filtering process or the error condition being searched for, child metrics that had no entry in the logical data map.

Limitations

The review is limited to business metadata that defines the reporting metrics. The processes related to the collection and distribution of technical metadata is outside the scope of this research.
Chapter IV: Results

This study has three primary goals. They are to first identify current and emerging data warehouse industry practices in regard to metadata management. Second is to evaluate the effectiveness of the development group’s current metadata management processes. Finally suggestions will be made to apply lessons learned from this research to increase the efficiency and effectiveness of the development group’s metadata management processes.

Data Warehouse Industry Practices

Metadata adds context and meaning to data. In a data warehousing environment there are two primary types of metadata, technical and business. Technical metadata often focuses on the physical implementation of the data warehouse and supports its maintenance but is of little interest to the end users. Business metadata is of much greater interest to end users as it enables them to find, understand and interpret the data. Since business metadata is a key enabler for data warehouse users and due to the fact that users typically cannot gather this information without assistance is it critical that processes exist within any data warehousing environment to capture and publish this information.

A common practice in the data warehousing industry is the use of a logical data map to capture domain specific metadata. Appendix A contains a sample logical data map. The development group uses several spreadsheet templates to capture business metadata during the system development life cycle. The primary document is similar to Appendix A. Some of the key metadata elements captured in the development group’s logical data map include the common name for each reporting metric plus its description, business rule and physical database name.
Data lineage is critical to build, construct, maintain and use a data warehouse. For this reason data lineage information is recorded for each metric. Metrics can be grouped into one of three categories based on their data lineage. These categories are:

- **Source system metrics.** Metrics that are loaded directly from the system of record with little or no data cleansing or manipulation. Source system information such as database name, table name and column name are recorded. Example metrics in this category might include account number or account balance.

- **Dimension table metrics.** This category consists of metrics that come from dimension tables within the data warehouse. Dimension tables often hold additional attributes that further define or describe a metric that was loaded directly from a source system. A common example of a dimension table metric is a date attribute. A calendar date stored in month/day/year format can map to dimensional attributes such as the fiscal period the date belongs to or to a true/false flag to indicate if it is a holiday.

- **Derived metrics.** The final category consists of metrics that are calculated or derived. These metrics are built from other metrics and therefore do not exist in source systems outside of the data warehouse. Their business rules can be simple and contain a single source system metric or they can be extremely complex and nest several levels of source system, dimensional and derived metrics.

*Effectiveness of Development Group Practices*

Understanding the relationships between metrics that support the creation of derived metrics is critical to interpreting this research. The model represented in Figure 1 will be referred to throughout the data analysis.
Figure 1: Relationship Model

The circles in Figure 1 represent metrics and the lines indicate a relationship that exists between metrics. Metrics B, D, and E represent source system metrics and have no dependencies on other metrics. Metrics A and C are derived metrics meaning they are created from other metrics and therefore have direct dependencies and may also have indirect dependencies. Metric C has direct dependencies on D and E. Since D and E are both source system metrics the dependency chain for metric C terminates at the direct dependency level. Metric A has direct dependencies on B and C. Since C is also a derived metric, metric A is indirectly dependent on any dependencies of metric C.

To evaluate the effectiveness of the development group's current processes the following data analysis was conducted.

1. Analysis of the logical data map, including
   - Classification of metrics into data lineage categories
   - Analysis of derived metrics and their direct dependencies
   - Review of errors in derived metric direct dependencies

2. Comparison of database metrics against logical data map, including
   - Match database metrics to logical data map
• Classify matching database metrics based on data lineage categories

The first phase of the analysis summarizes the metrics contained in the logical data map (LDM). Metrics are first categorized based on the three primary data lineage categories of source system, dimension table or derived metrics.

Table 1
Metrics in Logical Data Map

<table>
<thead>
<tr>
<th>Data Lineage Category</th>
<th>Count of Metrics</th>
<th>Duplicates</th>
<th>Unique Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source System</td>
<td>154</td>
<td>23</td>
<td>131</td>
</tr>
<tr>
<td>Dimension Table</td>
<td>43</td>
<td>2</td>
<td>41</td>
</tr>
<tr>
<td>Derived</td>
<td>210</td>
<td>6</td>
<td>204</td>
</tr>
<tr>
<td>No Business Rule</td>
<td>42</td>
<td>1</td>
<td>41</td>
</tr>
<tr>
<td>Totals</td>
<td>449</td>
<td>32</td>
<td>417</td>
</tr>
</tbody>
</table>

The summary information in Table 1 reveals two interesting findings. First is the fact that there were a total of 32 metrics with duplicate entries in the logical data map. Each duplicate entry was reviewed and in every case the duplicate entry contained only a logical name while the primary entry for that metric consisted of a complete or nearly complete record in the logical data map. It cannot be determined with certainty why these duplicate records existed but their impact to business or technical users would most likely be minimal since they are easily recognizable and therefore can be disregarded. All duplicate entries were excluded from further analysis.

The second interesting finding was the existence of 41 metrics with no business rules defined. The records in the logical data map for these metrics were blank with the
exception of the physical name column. It is interesting to note that these records are identical in format to the duplicate records mentioned above with the only exception being that they have no corresponding complete record. The presence of metrics without business rules has a more severe impact on business and technical users as both groups are left without an authoritative source to provide definitions and context to these metrics.

The business rules of the 210 derived metrics were further analyzed. Their business rules were parsed to identify the metrics they contain. A simple example is as follows. For the derived metric A the business rule is \( A = (B + C) \). The business rule is treated as a text string and when parsed it returns two rows: A, B and A, C. These rows represent metric A which has two child metrics B and C. Once the business rules are parsed the child metrics can then be evaluated against the logical data map and categorized by data lineage. This will identify derived metrics which have dependencies on child metrics that either do not exist in the logical data map or have no business rule defined.

Table 2

<table>
<thead>
<tr>
<th>Total Metrics With Errors</th>
<th>No Business Rule</th>
<th>No LDM Entry</th>
<th>Both Errors Exist</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>1</td>
<td>22</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2 summarizes the findings from parsing and analysis of the derived metric business rules. Of the 204 derived metrics 24 had business rules which contained errors including metrics that existed in the logical data map but had no business rule and metrics that did not exist in the logical data map at all. Both of these conditions are severe since...
users can not resolve them using the project documentation alone. Errors of this nature are often resolved only with the assistance of a knowledgeable business user who can provide supplemental or historical information.

The child metrics that caused the error conditions in Table 2 were further analyzed to determine if each occurrence was an isolated incident or if a few child metrics were responsible for the majority of the errors. Table 3 summarizes these findings.

Table 3

<table>
<thead>
<tr>
<th>Derived Metric Business Rule Child Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Business Rule</td>
</tr>
<tr>
<td>Metric Errors</td>
</tr>
<tr>
<td>Child Metric Error Count</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>Distinct Child Metric Errors</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

The No Business Rule errors were both separate occurrences. For the No LDM Entry errors half of the 28 errors were caused by six metrics while the remaining 14 errors were isolated occurrences. While no effort was made to reconcile these errors 5 of the 20 No LDM error metrics contained spelling or formatting errors that may have prevented them from matching to a valid logical data map metric.

The second phase of the analysis was to compare the metrics in the database that are exposed to the end users against the metrics in the logical data map and to classify them by data lineage category. Tables 4 and 5 summarize the findings.

Table 4
The percentage of database metrics that do not exist in the logical data map was surprisingly high. While no effort was made to reconcile omissions the metrics that did not match the logical data map appear to be free from spelling and formatting errors so there is not an obvious or simple explication for the high rate of mismatches.

Table 5
Database Metrics by Data Lineage

<table>
<thead>
<tr>
<th>Data Lineage Category</th>
<th>Count of Database Metrics</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source System</td>
<td>42</td>
<td>21.32%</td>
</tr>
<tr>
<td>Dimension Table</td>
<td>17</td>
<td>8.63%</td>
</tr>
<tr>
<td>Derived</td>
<td>137</td>
<td>69.54%</td>
</tr>
<tr>
<td>No Business Rule</td>
<td>1</td>
<td>0.51%</td>
</tr>
<tr>
<td>Total</td>
<td>197</td>
<td>100%</td>
</tr>
</tbody>
</table>

With 197 of 221 database metrics matching to the logical data map there are 196 of 417 unique metrics remaining in the logical data map that do not match to a metric in the database. While this number seems high it may be attributable to indirect relationships (see Figure 1). It may also indicate that a substantial percentage of the
metrics in the logical data map have no relationship to the database. Without a complete analysis of the hierarchies created by indirect relationships, which is beyond the scope of this research, this question cannot be fully answered.

Suggestions for Improvement

The development group has a solid foundation of business processes and documentation. They follow a structured system development life cycle and use supporting documentation such as the logical data map to support the data warehouse development process. A suggestion for improvement would be to conduct periodic reconciliations between the logical data map and the physical database as well as auditing the business rules of any derived metrics to ensure that child metrics exist in the logical data map. Processes to perform these actions are outlined in the methodology section of this research. Conducting this type of analysis periodically during the discovery, documentation, design and development phases would result in more accurate and complete supporting documentation which may result in decreased error rates during development and may yield time savings due to reducing the unplanned occurrences where business and technical users must spend time reconciling errors or omissions in documentation to complete their scheduled tasks.
Chapter V: Discussion

The goals of this study were to identify industry practices in regard to metadata management used to support data warehouse development, evaluate the effectiveness of the development group’s current metadata management processes and to identify opportunities for improvement in the development group’s current processes.

Limitations

This study is limited to identifying industry practices in metadata management, evaluating the effectiveness of a specific development group’s current document management processes and suggesting opportunities for improvement. New process implementation and analysis of effectiveness of changes are beyond the scope of this study. Research for this study is based on analysis of documentation. Bias may have occurred based on the researcher’s perceptions.

Conclusions

The development group’s current documentation and metadata management processes are consistent with practices recommended by industry experts. The existing documentation provides a framework that will support the successful development and deployment of data warehouse projects. The results of the data analysis show some errors and inconsistencies in the content of the particular project documentation that was reviewed. These errors complicate the tasks required of both the technical and business line users of the data warehouse and must be minimized. The structured nature of the development group’s processes would support regular analysis and review processes that could identify and correct errors within the logical data map as well as omissions of metrics from the logical data map that exist in the data warehouse.
Recommendations

The methodology section of this research outlines processes the development team could apply to audit their project documentation. By applying the data analysis processes used in this research to their project documentation the development group will be able to increase the accuracy and completeness of key supporting documents. It should be possible to automate some of the review processes so an analysis and subsequent reconciliation can be conducted on demand with minimal effort. The analysis of metrics common to both the database and logical data map is a relatively simple process that can be conducted with spreadsheet software. The internal analysis of the logical data map is a slightly more complex task. The development group could allocate time from one of its developers to write a relatively simple program in the language of their choice to perform the internal logical data map analysis discussed in this research. Any team member could then execute that program on demand to identify errors in the logical data map. That team member could then work with business users to reconcile the errors.

Future Research

Areas for future research include identifying the indirect relationships for all derived metrics. This would complete the dependency chain and enable a more thorough understanding of relationships between metrics. Such an understanding would support business users when investigating apparent data anomalies and would also support impact analysis if system changes are required.

Another suggestion for future research would be developing an automated process to create a visualization of the logical data map. This representation could be a graph where each metric in the logical data map becomes a node and relationships between
metrics are edges on the graph. This may provide a more user friendly presentation of the
data currently contained in the logical data map. The benefits may include increased
comprehension for all users and could simplify the identification of business rules with
needless complexity. Just as a mathematical equation can often be rewritten in a
simplified format it may be possible to rewrite business rules in a simpler format. This
would make the business rules easier to interpret and would also reduce the complexity of
the ETL code.
References


Appendix A: Logical Data Map

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Datatype</th>
<th>Source System</th>
<th>Source Table</th>
<th>Source Field Name</th>
<th>ETL Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerIDName</td>
<td>Customer full name (Last, First, Middle) prepended with CustomerID</td>
<td>varchar(100)</td>
<td>Derived in ETL</td>
<td></td>
<td></td>
<td>AcctNum + ' ' + DW FullName</td>
</tr>
<tr>
<td>CustomerTitle</td>
<td>Courtesy title</td>
<td>char(5)</td>
<td>AW</td>
<td>Contact</td>
<td>Title</td>
<td></td>
</tr>
<tr>
<td>FirstName</td>
<td>Customer's first name</td>
<td>varchar(30)</td>
<td>AW</td>
<td>Contact</td>
<td>FirstName</td>
<td></td>
</tr>
<tr>
<td>MiddleName</td>
<td>Customer's middle name (often NULL)</td>
<td>varchar(30)</td>
<td>AW</td>
<td>Contact</td>
<td>MiddleName</td>
<td></td>
</tr>
<tr>
<td>LastName</td>
<td>Customer's last name</td>
<td>varchar(30)</td>
<td>AW</td>
<td>Contact</td>
<td>LastName</td>
<td></td>
</tr>
<tr>
<td>CustomerFullName</td>
<td>Customer's full name as Last, First, Middle</td>
<td>varchar(100)</td>
<td>Derived in ETL</td>
<td></td>
<td></td>
<td>LastName + ' ' + FirstName + ' ' + MiddleName</td>
</tr>
<tr>
<td>BirthDate</td>
<td>Customer's date of birth</td>
<td>datetime</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;BirthDate&gt;</td>
</tr>
<tr>
<td>MaritalStatus</td>
<td>Customer's marital status</td>
<td>char(7)</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;MaritalStatus&gt; Decode to Single/Married</td>
</tr>
<tr>
<td>Gender</td>
<td>Customer's gender</td>
<td>char(7)</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;Gender&gt; Decode to Female/Male/Unknown</td>
</tr>
<tr>
<td>EmailAddress</td>
<td>Customer's email address</td>
<td>varchar(50)</td>
<td>AW</td>
<td>Contact</td>
<td>EmailAddress</td>
<td></td>
</tr>
<tr>
<td>IncomeRange</td>
<td>Customer's annual income</td>
<td>varchar(50)</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;YearlyIncome&gt;</td>
</tr>
<tr>
<td>TotalChildren</td>
<td>Customer's total number of children</td>
<td>tinyint</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;TotalChildren&gt;</td>
</tr>
<tr>
<td>NumberChildrenAtHome</td>
<td>Customer's number of children at home</td>
<td>tinyint</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;NumberChildrenAtHome&gt;</td>
</tr>
<tr>
<td>Education</td>
<td>Customer's education level</td>
<td>varchar(30)</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;Education&gt;</td>
</tr>
<tr>
<td>Occupation</td>
<td>Customer's general occupation (eg Managerial)</td>
<td>varchar(30)</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;Occupation&gt;</td>
</tr>
<tr>
<td>HomeOwnerStatus</td>
<td>Is the customer a homeowner?</td>
<td>varchar(13)</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;HomeOwnerFlag&gt; Decode to Homeowner / Not Homeowner</td>
</tr>
<tr>
<td>NumberCarsOwned</td>
<td>Number of cars the customer owns</td>
<td>tinyint</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;NumberCarsOwned&gt;</td>
</tr>
<tr>
<td>DateFirstPurchase</td>
<td>Date person first purchased a bike (self-reported)</td>
<td>datetime</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;DateFirstPurchase&gt;</td>
</tr>
<tr>
<td>CommuteDistance</td>
<td>Customer's average commute distance</td>
<td>varchar(15)</td>
<td>AW</td>
<td>Individual</td>
<td>Demographics</td>
<td>Shred Demographics: &lt;CommuteDistance&gt;</td>
</tr>
<tr>
<td>CustomerValueScore</td>
<td>Customer's current lifetime value score to AdventureWorks</td>
<td>varchar(15)</td>
<td>Derived in ETL</td>
<td></td>
<td></td>
<td>Out of scope for Phase 1</td>
</tr>
<tr>
<td>Phone</td>
<td>Customer's phone number</td>
<td>varchar(20)</td>
<td>AW</td>
<td>Contact</td>
<td>Phone</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B: Discovery and Documentation Flowchart

Domain knowledge emerges

Knowledge is captured

Knowledge is tested, refined, verified

Adequate understanding exists to begin development

Analyze Business Process

Identify data used and created (entities/attributes)

Identify relationships between entities

Document / revise business rules

Test business rules against sample data

Business rules consistent with sample data

Many iterations will take place at both stages

Investigate Major Discrepancies

Resolve Minor Discrepancies

No

Yes

Begin Design Process
Appendix C: Design and Development Flowchart

1. Create database schema (tables)
2. Write single ETL component
3. Test component against sample data
   - Component executes successfully
     - Yes: Output matches expected result
       - Yes: Ready for system test
       - No: Additional ETL components required
     - No: Component contains bug
       - Yes: Revise component
       - No: Business rule is incorrect
4. Edit / update business rule
   - Yes: Component contains bug
   - No: Ready for system test
Appendix D: User Acceptance Testing Flowchart

Assemble all ETL components

Acquire / create large data set

Execute ETL process against large data set

ETL code executes successfully

Yes

Users review data created by ETL process

Pass

Deploy to production

No

Error in business rule

Bug in code

Failure caused by Bug in code

Update business rule (Logical Data Map)

Repeat process from beginning

Modify code

Pass
Appendix E: Logical Data Map (LDM) Flowchart

New metric emerges (at any point in the SDLC)

Add metric to LDM (name, description, business rule)

Do new metric business rules contain metrics? (child metrics)

Yes

Do all child metrics exist in the LDM?

Yes

LDM update complete

No

Add child metrics to LDM

LDM update complete

No