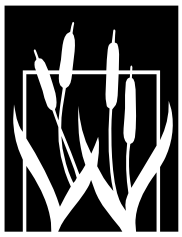

Whitemarsh
Information Systems Corporation

Quality Data-Centric Engineering and Management

Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716
Tele: 301-249-1142
Email: Whitemarsh@wiscorp.com
Web: www.wiscorp.com

Table of Contents

1. Objective

The objective of this short paper is to identify the key characteristics that databases and their supporting business information systems must exhibit to be judged as having quality data-centric engineering and management. The paper describes each of the characteristics and finishes with a set of conclusions that provide a way-ahead.

2. Topics Covered

The topics of this paper include:

- Validity
- Reliability
- Repeatability
- Auditability
- Single-Column Value-Domain Engineering
- Multi-Column Value-Domain Engineering
- Test Engineering
- Way Ahead Work Plan

3. Introduction and Scope

This is a concept paper on quality data-centric engineering and management for databases and supporting business information system projects. The quality characteristics addressed in this paper are:

- Validity
- Reliability
- Repeatability
- Auditability

In addition, two of the seven classes of data integrity rules are addressed. These are:

- Single Column Value Domain Engineering
- Multi-Column Value Domain Engineering

After presenting these six quality characteristics, this paper briefly presents how these six quality characteristics can be brought together to engineer and test the classes of business transactions that must be processed by data-centric business information systems. Within the context of this paper, a data centric business information system is one in which there is a database (not DBMS

(database management system such as Oracle)) core, and that the application software serves mainly to acquire, store, retrieve, compute/calculate, and report data. Included in this class are inventory, distribution, transportation, some manufacturing, customer management, human resources, and customer management business information systems. Not included would be direct manufacturing machine process control, guidance systems for automobiles, missiles, or weapons, and the like. These later systems are very “heavy” on process and “light” on data when compared to data centric business information systems.

This is a concept paper, not an engineering, design, or implementation paper. Once the concepts are accepted and/or modified by an organization, the engineering, design, and implementation can start. At the outset, the following are brief descriptions of the six characteristics that must be achieved to have quality data-centric engineering and management of databases and supporting business information systems.

Validity. Validity has two forms, deductive and inductive. To achieve deductive validity, a complete set of assertions would have to be made about core databases and processes executed to verify or refute assertions. Inductive validity can be seen as an assertion made about the total set of data based on the testing of a sample set of data.

Reliability. Reliability, from the Wikipedia, via the IEEE defines reliability as ". . . the ability of a system or component to perform its required functions under stated conditions for a specified period of time." Essentially, this means that an environment exists such that whenever the business information system is executed against the core database, the results are judged to be valid and/or acceptable.

Repeatability. Repeatability, means that when a set of transactions is re-executed, the results are the same. Reliability and repeatability go hand and glove. Together, these terms affect the engineering of the business information system and core database environment in that they require the existence of persistent "data-centric" starting points for the various key processes.

Auditability. Auditabilty, also from the Wikipedia, is "a non-functional requirement and concerns the transparency of a system with regards to external audits." To make this definition relevant, auditability simply means that outside auditors have the necessary and sufficient set of processes, guidelines, standards, and quality processes to determine that the results from the generation of values from the core database are the values that would be expected if the entire process was done "manually."

Single-Column Value-Domain Engineering. Single-Column Value-Domain Engineering, relates to the allowed values for the database columns inside the core database. For example, there is a column called Gender. If the allowed values are **M** and **F**, but a value **Q** is presented, that value and the context within which that value is presented must be rejected. It must be rejected because fields that have a restricted value domain are commonly used as the basis for valid values. Additionally, there are business rules that form the basis for policy-based decision making and computation. If unexpected values appear, the computations will be incorrect, or may fail to execute altogether.

Multi-Column Value-Domain Engineering. Multi-Column Value-Domain Engineering, relates to the allowed combinations of values for sets of database columns that may exist in one or more database tables. Examples include the monetary value for a core database

table column that must be equal to the sum of the values that are the "transaction details" for that monetary value.

Collectively, these six characteristics form the overarching framework for data-centric quality within databases and supporting business information systems. If a core database and supporting business information system have not been designed to support these characteristics, the likelihood of data-centric problems is virtually certain.

This paper also briefly addresses prototyping which is a critical and essential step prior to production implementation efforts. Prototyping enables requirements' discovery, the creation of a proven architecture, engineering, implementation approach, and proof of future success as well as comprehensive and correct implementation work plan development so as to prevent missed schedules, significant cost overruns, promised deliverables reductions, and an overall sense of project failure.

The remaining sections of this paper provide concrete examples regarding the core databases and support software systems with respect to these six data-centric characteristics, prototyping, a way-ahead activity list, summary and conclusions, and references to other Whitemarsh materials.

4. Validity

As stated in the first section, validity implies the "truth of the matter." There are two classes of validity here: Overarching, and detailed. Overarching validity is brought about by correctly identifying, engineering, and describing the enterprise missions, organizations, functions, and database domains so that database and business information systems efforts can be set within their correct enterprise context. The elements that must be defined are:

- **Missions.** Missions define the idealized and ultimate objectives of the enterprise. Its missions are what the enterprise ultimately strives to accomplish.

- **Organizations**. Organizations are the operating units of the enterprise that needs to be supported and/or will employ the data warehouse in their business intelligence activities.

- **Functions.** Functions are the activities of the various staff acting in their positions and they accomplish enterprise missions.

- **Database Domains**. Database domains are data-centric descriptions derived from each Mission "leaf."

Detailed validity requires "getting down into the weeds" of any effort. That can and should only be done once the overarching validity has been engineered, iterated, and accepted. Databases should be set within the boundaries of a selected set of database domains. Business Information Systems should be set within a boundary established by a selected set of functions.

A data-centric example of validity is information on the various database tables that enforce policy-driven correctness. In an absolute-accomplishment sense, however, this is impossible. There's an old mantra in the IT that achieving 95% data-centric quality requires a finite quantity of resources, the remaining 5% requires an additional infinite quantity of resources. 95% represents the practical achievable maximum.

Achieving validity requires the creation of assertions regarding all the facts contained in the database. For example, a single fact assertion is that the Gender for a given Person is correct. Assertions must also be made about combinations of facts. An example would be that the first employment project assignment date must be on or after the person's employment date.

Clearly it would be impossible to create all the assertions that databases should satisfy. Needed is a strategy to proceed from the most important assertions to the least important assertions. Assertions that are more important are those upon which decisions are being made, and assertions that are less important are merely those values that become "informational." This requires the identification, analysis, and specification of decision-based assertions, and their mapping to core database metadata.

Once the highest to lowest assertions are determined, the processes to determine the truth of the determinations can be specified in the database, the DBMS, and supporting business information systems. Once done, their implementations can be engineered. As these assertions are then executed, the identified problems because of failed assertions can be included in the assertion-execution report.

5. Reliability

As stated in the first section, reliability implies accomplishing the desired goal, as specified. There are three aspects here.

- Adequacy of the requirements that led to specifications
- Unambiguous implementation of the specification
- End-to-end integrated representation of requirements through to program code, to database design, and to supporting documentation.

The requirements must address not only the high-level requirements' statements, but also the supporting business-logic specification without regard to any form of technical implementation. The specifications must be such that reasonably subject-matter knowledgeable persons can understand what has to be done to achieve the desired result. Supporting all the requirements must be the supporting business policy specifications that provide the underlying validity to the requirements.

In data-centric efforts, there are sets of requirements that are initially expressed as statements representing expected outcomes. Each statement should have a business-based

description within which the business terms are identified and defined. Included also should be a mapping between the requirements' statement and the various database table columns. Finally, there should be a mapping between the requirements statements and the various business processes that must be accomplished to achieve the result. It is important to also have the business data model and the business process model interrelated one with the other, and both interrelated with the requirements statements.

It is important to distinguish between a business data model and a DBMS data model. The first represents the business-based groupings of data such that are clear and unambiguous from a business sense point of view. The data modeling convention of Third Normal Form is well suited to expressing business data models. DBMS data models are technology-based transformations of business data models into a form that is understood by a database management system (DBMS) such as Oracle or IBM's DB2.

It is also import to distinguish a business-process model from a collection of computer language statements within a computer program. A business process model is multi-layered, and starts with high-level business process names and descriptions. The bottom layer has process-based statements that express the reading, writing, updating, and transformation of data to accomplish the business' requirements.

The business data model is "exercised" by the reviewer acting as one or more business processes. If the business data model does not correctly support the agreed-upon business processes, the business data model is deficient. In contrast, the business process model is exercised by following the data accesses, inserts, deletes and modifies. Again, if the processes do not provide sufficient support for the business data model, the processes are deficient.

Both business models, that is, the data and process, are an "implementation" of requirements' statements. If either or both of these models cannot be easily seen as a fulfillment of the requirements, either or both are deficient. Finally, the requirements, as evidenced by the business data and process models represent the required outcomes and policy of the enterprise. If the requirements cannot be seen as fulfilling enterprise outcomes and policies, the requirements are deficient.

Pure top-down process specification and engineering will not work because errors made at the outset are then just compounded as you move down the hierarchy of database and business information system development. Rather, what is needed is a composite approach of top-down for some aspects, bottom-up for other aspects, and an iterative process for other aspects such that all business data and process models are updated as appropriate during the life of the project.

As each iteration is accomplished, the total set of business data and process model specifications should be reviewed and approved. That is because it is the entire set of specifications must be reliable.

The only practical way to make an entire set of specifications reliable, integrated, nonredundant, and without inter and internal conflicts is to have these artifacts generated from a metadata repository, that is itself, an integrated, interrelated, and nonredundant database of the system's analysis and design work products. Not only is this practical, not to do this is unprofessional.

A key component is the business data model design. Many business data models actually exist across multiple layers of abstraction. The first layer represents the interface between the "outside world" and the core database. "Outside world" in this situation refers to the persons and/or systems that are the original sources of data. An example might be a person-based order-entry environment or an automated extract from some other business information system. This is referred to as the "data acquisition" layer. This data specification, and the actual data must be persistent, and it should be in the form of a collection of database tables. It is important so that this data can be processed and re-processed without having to rematerialize the original data by recollecting it from the "outside world." It is important that this originally collected data be in a database format so that it can be supported by archiving, transaction framing, security, query, update, and be analyzed for the various data-centric characteristics that can be easily accomplished through sql-language-based processes. If the source data is not in a database format, or is not persistent, it will be very difficult to achieve reliability.

A second layer of data represents the stored business data layer. That is, the data within the core database. Records in these tables are built directly from the data from the data acquisition layer.

The reliability aspect of the business data model is based on the fact that the data is traceable back to the data acquisition layer, which, in turn, is able to be mapped via the business rules directly to the "outside-world" source. Additionally, the complete set of fields in the "outside world" data record should be mapped to the business's data and processes. If this is accomplished, there is a complete collection of tables, starting with the data acquisition record all the way through to the core database tables that are able to be mapped to requirements statements. The transformation of the records of data from one table to the next is supported by the business processes.

If all the tables and processes are mapped, reviewed, revised as necessary, and agreed to as to content and transformation, the overall effort will be reliable.

While it may not be explicitly stated in various requirements' statements, data-centric quality must be sufficiently detailed down to the various business data model table columns. If the value domain for each column is restricted, each value must be specified and defined. Additionally, the business process logic decisions that depend on specific values must be clearly set out so that programmers can fully understand and know the consequences of these values. Without these levels of specification, reliability cannot be achieved.

As a final comment regarding reliability, there must not be any direct updates to any of the core database tables except through the formally defined, validated, and accepted business data and process model architecture.

6. Repeatability

As stated in the first section, there must be repeatability. Simply, that means that when sets of transactions are re-run from a known database state, the results must be the same.

In the architecture of the many data-centric business information systems, the underlying business data may be constantly changing on a day to day basis. For example, a business product order may be entered on Monday, revised on Tuesday, and again on Wednesday, and finally settled on Friday. Each day, the records from the data acquisition layer will be different. Some orders may be subsequently revised, weeks to months later. There must be a complete audit trail that supports all these changes. Otherwise, it will be very difficult to re-create the exactly same starting conditions through which reliability can be affirmed. Just having a database-table by database-table change log is not the same as business transaction log. Business transaction logs may have entries that span multiple databases, computers, DBMSs, and database tables.

Repeatability is essential to support auditability because without repeatability, the starting conditions cannot be depended upon to be the same. Every data acquisition layer record must clearly identify the “outside world” record from which it was generated. This traceability along with the business processes enables the manual re-processing of data to ensure that the business processes are being properly carried out.

If a multiple-layered approach exists, as is conceptually described above, the correctness of the processing of business transactions can be traced.

7. Auditability

As stated in the first section, auditability is the ability of outside auditors to exercise the necessary and sufficient set of processes, guidelines, standards, and quality processes to determine that the results from the generation of business values are the values that would be expected if the entire effort was done "manually."

With the existence of fully implemented reliability and repeatability, the ability to perform an audit is relatively easy. There would be a direct set of trace materials from requirements to the business data and process model to the actual DBMS data model tables that represent the transformation of the data acquisition layer record into core database data via the business rules contained in the business process model.

8. Single-Column Value-Domains Engineering

The first addressed data integrity rule, Single Column Value Domain Engineering relates to the allowed values for the database columns inside the core database. The business data model, generated in consequence to the requirements, contains columns. Columns have either a restricted value domain, or an unrestricted value domain. Examples of restricted value domains include Gender, that is, (M)ale, (F)emale, or (U)nknown. An unrestricted value domain, which while it might have a maximum or minimum value, might have an unrestricted set of numbers between these limits. Each restricted value domain column must be fully defined. Included with the column's definition are the values that are permitted, and the meaning of each such value.

A key use of restricted values is decision making. Most restricted values are employed in various logic decisions including value breaks for sorting. The requirements specification must be examined to determine if all the values have been specified. Then, the business process logic must be examined to know if all the values are being employed in decisions.

The business data model must be examined to ensure to prohibit the storage of any value that is not a member of the restricted value set. The prohibition can be accomplished in one of two ways: 1) as an explicit Valid Values clause in the database schema, or 2) as a table of valid values with a referential integrity clause between the business data table that contains the restricted value as part of a business data record and the external table wherein the restricted value is the sole basis for the record's existence.

From a core database point of view, knowledge of the restricted values must be employed to exhaustively check the acquisition source records. If an acquisition source record does not conform to the restricted value set, it must be rejected. SQL queries could be constructed to check batches of acquisition source records before they are transformed into updates to the core database. This class of processing ensures that no acquisition source record can ever be processed with invalid data.

For the columns that have unrestricted value domains, two other classes of value domain checking may be appropriate. First, these values may be restricted in other databases. If so, then there should be a centralized place where all these data values are stored such that they can be accessed for validation purposes. Examples might include checking the validity of postal codes from a master data source, or the validation that an expense report from an employee is related to an actual employee. Master sets of reference data are a critical component in the engineering of enterprise data-centric quality. Another example might be the exercise of a business rule that says that values cannot exceed, for example, three standard deviations above or below an average value. Such values would represent fewer than 3% of the total set of values. If these tests are run routinely, either unacceptable records will be prevented, or if the tests are run randomly or say, only on the weekend, not only will data be flagged but so also will records from the rest of the database tables so they can subsequently be examined for quality.

9. Multi-Column Value Domain Engineering

The second data integrity rule, Multi-Column Value Domain Engineering relates to the allowed combinations of values for sets of database columns that may exist in one or more database tables.

The first step in determining multi-column dependencies is to examine requirements. For example, suppose there is an order that has an Order-Header record and a set of Order-Detail records. The Order-Header record might include two columns, Total Order Amount, and Total Shipping Weight. Clearly, both these columns are calculated from data from within the set of order-detail records. This is a kind of multi-column and multi-table dependency. Other examples are the Birth-Date and Death-Dates of a person. The death date must occur on or after the birth date. Additional order related examples would include Order-Date, Ship-Date, and Backorder-

Date. These are examples of single table multi-column dependencies as these columns would all be within the Order-Header, but would also have their values set through direct updates to data records within these tables.

All cases of multi-column dependency must be determined and classified as to dependency type. The complete set of data integrity rules is:

- Single table, single column
- Single table, multiple column
- Single table, single column, multiple row
- Single table, multiple column, single row derived data
- Single table, single column, multiple row derived data
- Multiple table derived data
- Multiple table, multiple column (referential integrity)

Each data integrity rule must be identified and cross-referenced back to requirements, the business data and process models, and finally, to the DBMS data models and the business information systems program logic code that enforce these data integrity rules to ensure that they are being properly handled. Properly engineered and implemented metadata repositories can both support, update, and report this network of interconnections.

Special processes must be established that execute completely outside normal processing to check the proper value determination of these multi-column value domain dependencies. While it would be ideal, that these processes are executed 100% of the time against the acquisition source records to ensure that no invalid data is able to be entered into the main database, that is not always possible as some of these values, such as the Total-Order-Amount, or Shipping-Date and Back-Order-Dates can only be determined over time.

10. Test Engineering

Most data-centric business information systems consist of critical business processes and their resulting business transactions that are used to load, update, calculate, and retrieve data. From the scenarios set out in Section 3, Reliability, the three main areas for data-centric quality engineering are:

- Acquisition source data creation
- Core database creation
- Ongoing database update

The first area consists of the processes necessary to originally capture data from the “outside world.” The result is the set of acquisition source records. The process of creating the necessary and sufficient data for the proper creation of acquisition source records requires a through understanding just how these source records are used. Thus, there is a back and forth effort

between understanding gathering the data from the "outside world" into acquisition source data records, and having the right data to support the creation of the core database.

Once this process is complete, testing the acquisition source data records consists of being able to successfully create all the different types and classes of core database data.

Comprehensive data-centric quality engineering must be incorporated into the acquisition source data creation effort. Every value domain for every single-column must be known. Every multi-column value domain interaction must also be known. Additionally, decisions must be made about what to do for every single-column and multi-column value domain error that can be encountered. The error reports need to be engineered not only to report the error but also the full context of the error so that it can be addressed in a timely manner.

Analysis must be made of the kinds of data-centric errors that may exist across the entire data life cycle, and what corrections to that data need to be made to make the complete set of data useful for the core database.

Ideally, for the creation of acquisition source records, there has to be a comprehensive set of testing against specially engineered subsets of data records from the "outside world." Engineering this environment takes considerable effort and planning to accomplish, execute, and once the acquisition source data records are all corrected, reexecuted. This comprehensive effort is required because the result is the set of acquisition data records used to load and update the core database.

The second step, core database creation, includes the identification, engineering, and testing of the complete set of database update transactions. These transactions have both source and destination requirements. The source requirements related to ensuring that all the necessary data exists within the acquisition source data records so that all the appropriate database transactions can be built. The destination requirements related to ensuring that all the data necessary to build and/or update the database records exists.

As an acquisition source data record is processed, the database transaction creation process must build all the appropriate transactions. Knowledge must therefore exist about all the possible database transactions that need to exist to update the core database. The data requirements for each database transaction type must be identified and designed. Once designed, the required data that must be present in the acquisition source data. Once finished, there can be comprehensive testing of all the different valid and invalid sets of data.

The third step, core database creation and update, ensures that all the database tables contained within the core database are properly updated. To ensure this, data requirements from the core database tables must be known and then reflected back onto the data contained in the database transactions. Once that analysis is complete, then there can be appropriate updating of the core database tables.

Unique to the core database tables are two concepts: Referential Integrity, and uniqueness. Some tables have strict parent-child relationships. It should never be possible to install records of data without the proper parents already being established. In database, the relationship between the parent and child record is most often based on common data values. For example, there might be an Order-Number in the Order-Header table and the same Order-Number value in each of the Order-Detail records. If the values are the same between the Order-

Header and in the Order-Detail records, the Order-Header record and the value-based selected set of Order-Detail records comprise the complete order.

Sometimes the referential integrity between database tables is not based on business data values. That is, not based on shared values such as Order-Numbers. Rather, the shared values are based on computer-generated artificial numeric values. In a sense, the referential integrity is artificially generated through the use of these computer-generated values versus business values such as Order-Number. Because of this difference, there must be sufficient business-data-based columns in, for example, the Order-Header table, to be able to select a single record based solely on business-related data values. This is needed in the creation of a new instance of a "detail" record. There must be sufficient business-value-based data to determine whether a specific "header" record exists or not. If the "header" record exists, the "detail" record can be created, otherwise it cannot.

The point to this is that there must be the necessary and sufficient set of testing with highly engineered test data to ensure that there have been the proper analysis and design of the set of core database table business columns.

11. Prototyping

As stated in Section 3, Introduction and Scope, prototyping is a critical and essential step prior to production implementation efforts. Prototyping enables requirements' discovery, the creation of a proven architecture, engineering, implementation approach, and proof of future success as well as comprehensive and correct implementation work plan development so as to prevent missed schedules, significant cost overruns, promised deliverables reductions, and an overall sense of project failure.

Today, virtually all the computing facilities necessary to create completely valid prototypes exist. The prototype computing facilities not only exist but are very cost effective to use. For example, production application software development effort that would take a programming staff of five six months to create, that is, about 2.5 staff years, can be completely prototyped in a PC-server environment by one person in less than one staff month. That's a ratio of 30 to 1.

The value of a prototype cannot be over estimated. It serves to prove all the quality data-centric engineering and management characteristics in a very short time with a minimum of effort. That is, you can prove validity, reliability, repeatability, and auditability as well as two of classes of data integrity rules well in advance of any serious production-class development efforts.

The prototype, once complete, can serve as the architecture and engineering construct for the entire production class environment. Prototypes also serve to ferret out undiscovered requirements and enable comprehensive production-effort work breakdown structures to be fully known.

Quality data-centric engineering and management efforts should always have prototypes, possibly end-to-end as critical path elements. Not to engineer, deploy, and fully use prototypes is

simply unprofessional. Without fully developed prototypes validity, reliability, repeatability, and auditability of production class efforts are all put at risk because of the "unknowns" that are almost always discovered during prototypes.

12. Way Ahead Work Plan

Attachment 1 to this paper provides a way-ahead work plan. This plan is intended to set out the key activities associated with quality data-centric engineering and management. The activities are merely listed without explanation. These activities do not present either a detailed "how-to" of the activities themselves, or the methodological "how-to" for the actual development processes such as "how to develop/implement a database design," nor "develop/implement a process model design." References to detailed explanations are provided in Section 14, References.

13. Summary and Conclusions

The practical application of the points made in this paper include:

- An outlining of the architecture and engineering quality characteristics that must be addressed in the various database tables and data sources involved in databases and supporting business information systems.

- The identification and description of the six key components of quality data-centric engineering and management, that is:

    - Validity
    - Reliability
    - Repeatability
    - Auditability
    - Single Column Value Domains Engineering
    - Multi-Column Value Domain Engineering

Once the concepts contained in this paper are accepted and/or revised, the various follow-on detailed engineering, implementation, and testing activities can be planned, staffed, and accomplished.

14. References

The following references to Whitemarsh materials provide a more detailed exposition practical application of the significant content of this paper.

The following documents are available free from the Whitemarsh website:

Paper	URL
Comprehensive Metadata Management	http://www.wiscorp.com/ComprehensiveMetadataManagement.pdf
Metabase Overview	http://www.wiscorp.com/Metabase.zip
Whitemarsh Data Modeler, Architecture and Concept of Operations	http://www.wiscorp.com/MetabaseDataModelerArchitectueandConceptofOperations.zip
Metabase User Guides	http://www.wiscorp.com/MetabaseUserGuides.zip
Iterations of Database Design	http://www.wiscorp.com/iterations_of_database_design.pdf
Data Management Conferences	http://www.wiscorp.com/dama2002.zip http://www.wiscorp.com/dama2003.zip http://www.wiscorp.com/wrad2000.zip

The following documents are available for Whitemarsh Website Members. The URLs that follow provide descriptions of the pages. Members should log in and proceed to the appropriate page, e.g., Enterprise Database, find the book, paper, or course and perform the download.

Paper	URL
Data Management Program - Metadata Architecture For Data Sharing Data Management Program - Database Interface Architectures Data Management Program - Projects And Data-Asset Product Specifications Data Management Program - Work Breakdown Structures Knowledge Worker Framework Database Objects Managing Database - Four Critical Factors	http://www.wiscorp.com/wwmembr/mbr_products_edb.html
Work Breakdown Structures	http://www.wiscorp.com/wwmembr/mbr_products_dp.html
Data Architecture Classes Guidelines for Data Architecture Class - Data Warehouse Iterations of Database Design	http://www.wiscorp.com/wwmembr/mbr_products_dd.html
Work Breakdown Structures Database Project Work plan Templates Information Systems Development Methodology Phases 1 and 2 Whitemarsh Project Estimating Work plan Development	http://www.wiscorp.com/wwmembr/mbr_products_dp.html

Attachment 1
Way Ahead Work Plan
for
Quality Data-Centric Engineering and Management

1. Identify the scope of the database and business information system
   a. Description of effort
   b. Description of what's in and out of scope
2. Create business data and process models
   a. Create overall requirements documents
      i. Mission requirements
      ii. Organizational requirements
      iii. Functional requirements
      iv. Database domain requirements
      v. Validity requirements
      vi. Reliability requirements
      vii. Repeatability requirements
      viii. Auditability requirements
      ix. Technical requirements
      x. Performance requirements
      xi. Auditing requirements
   b. Acquire external documents
      i. Business domain specific industry best practices
      ii. Business domain specific state and local regulations and requirements
      iii. Business domain specific federal regulations and requirements
   c. Create business data model specification
      i. Engineering or design materials
      ii. Engineered data flows
      iii. Accomplish data transformation engineering
         (1) Intra table rules, processes, audits, and rollbacks
         (2) Inter table rules, processes, audits, and rollbacks
         (3) Insert-rules, processes, audits, and rollbacks
         (4) Modify-rules, processes, audits, and rollbacks
         (5) Delete-rules, processes, audits, and rollbacks
      iv. Create data element model
         (1) Concepts
         (2) Conceptual value domains
         (3) Value domains and interrelate to conceptual value domains
         (4) Data element concepts,
         (5) Concepts and Conceptual value domains interrelationships
         (6) Data elements, data element concepts and value domains interrelationships

v. Create data model tables
   (1) Acquisition source data tables
   (2) Database transaction tables
   (3) Database tables

vi. Maps to validity, reliability, repeatability, and auditability requirements

d. Create business process model specification
   i. Create source data creation processes
      (1) Process descriptions
      (2) Process business logic specifications
      (3) Computer program code
      (4) Error checking and reporting
      (5) Integrity checking and reporting
      (6) Validity, reliability, repeatability, and auditability requirements mappings
   ii. Create transaction creating processes
      (1) Process descriptions
      (2) Process business logic specifications
      (3) Computer program code
      (4) Error checking and reporting
      (5) Integrity checking and reporting
      (6) Maps to validity, reliability, repeatability, and auditability requirements
   iii. Create database creation/updating processes
      (1) Process descriptions
      (2) Process business logic specifications
      (3) Computer program code
      (4) Error checking and reporting
      (5) Integrity checking and reporting
      (6) Maps to validity, reliability, repeatability, and auditability requirements

e. Create interaction mappings between business data and process models

f. Create integrity specification
   i. Create business data model
      (1) Single-column value domains
      (2) Multi-column value domains
         (a) Single table integrity checks
         (b) Multi-table integrity checks
      (3) Referential integrity across tables
      (4) Maps to validity, reliability, repeatability, and auditability requirements
   ii. Business process model (software module classes)
      (1) Acquisition source data creation

(a) Full context transaction reporting
(b) Program process flow reporting
(c) Decision taken reporting
(d) Single-column value error reporting
(e) Multi-column value error reporting
(f) Transaction processing statistics
(g) Maps to validity, reliability, repeatability, and auditability requirements

(2) Database transaction creation
(a) Full context transaction reporting
(b) Program process flow reporting
(c) Decision taken reporting
(d) Single-column value error reporting
(e) Multi-column value error reporting
(f) Transaction processing statistics
(g) Maps to validity, reliability, repeatability, and auditability requirements

(3) Database creation/updating
(a) Full context transaction reporting
(b) Program process flow reporting
(c) Decision taken reporting
(d) Single-column value error reporting
(e) Multi-column value error reporting
(f) Transaction processing statistics
(g) Maps to validity, reliability, repeatability, and auditability requirements

3. Evaluate business data and process models
   a. Business data element model (ISO 11179 based)
      i. Concepts
      ii. Conceptual value domains
      iii. Data element concepts
      iv. Value domains
      v. Data elements
      vi. Evaluate adequacy of mapping to validity, reliability, repeatability, and auditability requirements
   b. Evaluate business data model
      i. Table definitions and supporting specifications.
      ii. Column definitions and supporting specifications.
      iii. Column to iso 11179 data element mapping
      iv. Column to iso 11179 value domain mapping
      v. Primary keys and supporting specifications.
      vi. Unique keys and supporting specifications.

vii. Referential integrity and supporting specifications.
viii. Column data types and supporting specifications.
ix. Single-column restricted value domains
    (1) Identification
    (2) Enumeration of value domains
x. Existence of multi-column restricted value domains
    (1) Identification
    (2) Enumeration of value domains
xi. Evaluate adequacy of mapping to validity, reliability, repeatability, and auditability requirements

c. Evaluate business process model
    i. Existence of subsystem architecture and description documents and charts for each software module class.
    ii. Existence of software logic flow artifacts sufficient to verify against actual program logic for each software module class
    iii. Existence of actual program language listings with sufficient self-contained documentation to verify conformance to software logic flow artifacts for each software module class
    iv. Existence of all software input and output specifications for each software module class
    v. Existence of all success and error message specifications for each software module class
    vi. Evaluate adequacy of mapping to validity, reliability, repeatability, and auditability requirements

d. Prototyping
    i. Engineered prototype for significant components of database and business information system functionality
    ii. Created mappings between prototype components and the business data and process model validity, reliability, repeatability, and auditability characteristics
    iii. Implemented prototype and incorporate selected subsets of production-class data to provide 100% realism.
    iv. Built database and business information based scenarios for demonstrating prototypes
    v. Selected and demonstrate to key user communities affected by database and business information system development
    vi. Iterated prototype
        (1) Identify and formalize prototype findings
        (2) Categorize each deficiency by critical, important, or nice to have
        (3) Resolve findings as changes to business and process models
    vii. Adequate evaluations of mapping to validity, reliability, repeatability, and auditability requirements

4. Perform production development and implementation audit
    a. Perform database audit
        i. Determined adequacy of all primary keys
        ii. Determined adequacy of all unique keys
        iii. Determined adequacy of DBMS controlled single-column value domains
        iv. Determined adequacy of DBMSs controlled multi-column value domains
    b. Perform software subsystem audit
        i. Determined existence of all software modules
        ii. Determined adequacy of processing logic of all software to ensure proper construction and mapping to both data model and to specified business rules
        iii. Determined adequacy of software encoded business rules
        iv. Determined adequacy of all reporting to ensure appropriate success and failure interpretation and remediation.
    c. Evaluate mapping between production development implementation components and validity, reliability, repeatability, and auditibility business and process model specifications
5. Business database and business information system testing
    a. Develop testing requirements
        i. Develop transactional testing requirements
            (1) Identify and describe each test case
                (a) Input data required
                (b) Output data expected
                (c) Outcomes expected
                (d) Overall success results reporting
                (e) Overall failure results reporting
                    (i) Context
                    (ii) Strategies to correct
                    (iii) Impact on subsequent processing
                    (iv) Rollback and recovery as necessary
                (f) Ensure that the testing requirements are appropriately mapped to the business data and process model validity, reliability, repeatability, and auditability characteristics
        ii. Develop business database tests
            (1) Engineer “decision making” column-based testing
                (a) Determined decisions
                (b) Identification of tables and columns involved in test case
                (c) Identification and quantify risks associated with column value failures.
                (d) Identification of test cases for each decision-based test
            (2) Engineer restricted value domains testing
                (a) Identify value domains

(b) Identify all columns employing the value domain
(c) Identify and quantify risks associated with column value failures.
(d) Specify test for value domain assurance

(3) Ensure that the database tests are appropriately mapped to the business data and process model validity, reliability, repeatability, and auditability characteristics

iii. Develop business information system tests

(1) For each involved software system
(2) For initial step
(a) Database or file access
(b) Processing steps
(c) Calculations
(d) Success reporting and verification
(e) Error reporting and verification
(i) Context
(ii) Strategies to correct
(iii) Impact on subsequent processing
(iv) Rollback and recovery as necessary
(3) For each interim step
(a) Database or file access
(b) Processing steps
(c) Calculations
(d) Success reporting and verification
(e) Error reporting and verification
(i) Context
(ii) Strategies to correct
(iii) Impact on subsequent processing
(iv) Rollback and recovery as necessary
(4) For final step
(a) Database or file access
(b) Processing steps
(c) Calculations
(d) Success reporting and verification
(e) Error reporting and verification
(i) Context
(ii) Strategies to correct
(iii) Impact on subsequent processing
(iv) Rollback and recovery as necessary
(5) Ensure that the business information system tests are appropriately mapped to the business data and process model validity, reliability, repeatability, and auditability characteristics

b. Create test cases
    i. Test data creation
        (1) Context data creation (group, class, benefit period, subscriber, dependent, etc.)
        (2) Claim and claim-line creation
    ii. Software processing environment creation
    iii. Construct test-case test data
    iv. Test test-cases to ensure proper data and software success and error reporting.
c. Validate test cases
    i. Employment of manual methods to perform calculations
    ii. Employment of alternative data extraction mechanisms
    iii. Employment of alternative data sources to verify that the same results are accomplished by as-is processing versus to-be processing.
d. Execute test cases
        (1) Execution schedules
        (2) Test cases executions
        (3) Results Review
            (a) Database verification: employ appropriate sql and other tools
            (b) Software process results verification: employ message and software processing logs
        (4) Test case findings
        (5) Test case proposed changes
            (a) Data model changes
            (b) Process model changes
            (c) Test data changes
e. Maintain and evolve test cases

6. Production environment testing
    a. Valid random set of data selections
    b. Existing value sets for these production test data cases extracts
    c. Test data cases through the business information systems processing
    d. Results of manual-based processes and automation-based processes comparisons
    e. Results and interpretations on any differences
    f. Recommendations as necessary for data and/or process model changes.
    g. Extrapolations of differences findings and reports to management.