



DATA MODEL EVALUATION WORKPLAN

Michael Gorman - Whitemarsh Information Systems, Corp.

Introduction

In the March 1998 issue of The Data Administration Newsletter www.tdan.com a user submitted the following question:

Should the customer be able to "demand" data models from software package vendors and what can be done to convince the vendors that supplying data models is in their own best interest?

This question only arises when the RFP under which the package is procured is not well specified. All the meta data that exists within a package should have been available and been provided with the package as part of the contract. No meta data, then no contract. If you've acquired the package, done the conversion, and paid your money then you've lost all your bargaining position.

In addition to having the data model, you need to know how well the data model was done. There's a whole check list for evaluating the quality of a package's data model. If the data model is bad then so too is probably the rest of the package. If the data model is bad or not sufficient then the vendor probably had to create tens-of-thousands of extra lines of application code to compensate for the lack of schema based validations, referential integrity, referential actions, stored procedures, triggers, and the like.

Simply stated, if a vendor doesn't want to proudly show, share and give you the package's data model then the vendor is probably hiding something that will cost you a lot in the long run.

In support of that answer, the following data model evaluation work plan is offered.

Package Data Model Evaluation Workplan

Identify, acquire, extract, and store a complete data model for each of the current set of packages that are currently addressing the problem/subject area

- a. Identify package
- b. Acquire data model
- c. If data model is automated
 - i. Identify and acquire the data model
 - ii. Enter data model into reverse engineering tool

- iii. Obtain relational model report (tables and columns) from reverse engineering tool
 - iv. Create repository data loading transactions|
 - v. Store tables, columns, data types, et al, as well as all obtainable definitions
 - vi. Deduce the data elements and store data elements
- d. If data model is manual
- i. Identify and acquire the data model
 - ii. Identify tables from available documentation
 - iii. Identify columns, data types, et al, as well as all available definitions
 - iv. Create repository data loading transactions?
 - v. Store tables, columns, data types, et al, as well as all obtainable definitions

Inductively arrive at a unified data model from every software application package, that is, an operational data store (ODS) for the problem/subject area.

- a. Initially include into ODS data model all entities, attributes, et al from the largest subject area package
- b. Examine each remaining entity from each remaining package to determine whether it is semantically the same or should be included into the ODS. If added, then add/modify the appropriate set of relationships and attributes in the ODS affected entities.

Assess ODS coverage of the problem/subject area and identify any additional entities that may be required to complete coverage of problem/subject area.

- a. Examine the ODS to ensure that its entity coverage includes is at least one set of overlapping entities to adjacent systems to guarantee proper interfacing.
- b. Add missing entities and attributes to the ODS database and the repository holding ODS entities and attributes.

Map from the ODS to each package and produce ODS-to-package difference reports, and package-to-package difference reports.

- a. Identify those entities that are semantically equivalent

- b. Identify those entities that are in the ODS and missing from a package
- c. Identify those entities that in one package and missing from the other
- d. Infer functionality that is and is not possible in each package with respect to the ODS difference reports
- e. Infer the overlapping functionality between packages as possible areas of semantic conflict

Map each attribute within each package to the corresponding ODS entity attribute and produce ODS-to-package difference reports, and package-to-package difference reports.

- a. Identify those attributes that are semantically equivalent
- b. Identify those attributes that are in the ODS and missing from a package
- c. Identify those attributes that in one package and missing from the other
- d. Infer functionality that is and is not possible in each package with respect to the ODS difference reports
- e. Infer the overlapping functionality between packages as possible areas of semantic conflict
- f. With respect to each attribute, determine if its semantics can be recorded into the corporation repository meta attributes for each attribute
 - i. Evaluate the ability to map its use/data environment, that is, its world, region, country, and local sites
 - ii. Evaluate the ability to map its data types, that is, character, date, integer, decimal, and money
 - iii. Evaluate the ability to map its meta characteristics, that is, common name, disambiguators (e.g., modifiers), contexts, and class words,
 - iv. Evaluate the ability to map its business rules, that is, value ranges, valid and invalid values, and various types and classes of interdependencies.
 - v. Determine if all the value ranges, valid and

invalid values, and various types and classes of interdependencies are defined in application packages, embedded as subclauses within the schema DDL, or are fully value based through reference data tables

vi. Ensure that there is are sufficient attributes to support necessary historical data and data update audit trails

vii. Evaluate each attribute and determine how its data value class membership is handled

(1) Simple (single field and restricted value set),

(2) Compound (single data field but with multiple contained data fields),

(3) Group (single concept with multiple contained data fields such as address),

(4) Pair (two data fields that exist as code and value),

(5) Related (two fields that require a value rule to ensure consistency),

(6) Complex (a data field that contain more than one of the preceding five data value situations)

Identify, assess and report on the keys

a. Determine if each entity within the package's data model has a primary key that is either non-information bearing, or is based on a non-volatile value set

b. Determine if each entity within the package's data model has an additional unique value key comprised of business data fields that ensure that unambiguous rows of data are stored in the entity

Identify, assess, and report on Referential Integrity and Referential Action rules

a. Assess referential integrity rules and actions within each package are fully defined and are enforced by schema defined procedures.

Assess the proper specification and consistency among all data integrity

- a. Have all seven classes of data integrity rules been properly defined, incorporated, and enforced through schema defined column and table constraints, assertions, and before and after triggers.
 - i. Assess the single table, single column class
 - ii. Assess the single table, multiple column class
 - iii. Assess the single table, single column, multiple row class
 - iv. Assess the single table, multiple column, single row derived data class
 - v. Assess the single table, single column, multiple row derived data class
 - vi. Assess the multiple table derived data class
 - vii. Assess the multiple table, multiple column (referential integrity) class

Create data model assessment report

- a. Identify significant problem areas
 - i. Entity design
 - ii. Attribute specification
 - iii. Data integrity specification
 - iv. Values specification
 - v. Conflicts among application packages with respect to ODS data model
- b. Determine whether the vendor will resolve problem areas prior to package acquisition and implementation
- c. Determine whether vendor will refrain from making similar data semantic errors in future releases
- d. Determine cost of designing, building, and maintaining ancillary information systems and program remediate problems discovered in data models

- e. Add cost of initial and continuing remediation efforts to cost of package
- f. Prepare data model efficacy report and include any additional costs

Michael M. Gorman, President of Whitemarsh Information Systems Corporation, has been involved in database and DBMS for almost 30 years. Mr. Gorman has been the Secretary of the ANSI Database Languages Committee, X3H2 for 19 years. X3H2 standardizes SQL. A full list of Whitemarsh's clients and products can be found on the web site, www.wiscorp.com. The goal of the web site, WisWeb, is to make data management books, courses, methodologies, software, and metrics available to the database community through electronic publishing and downloading. WisWeb memberships are very reasonable and are designed for the individual, the ISD organization, universities/colleges, and professional training organizations.

*Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716-1518 USA
Phone: +1.301.249.1142
FAX: +1.301.249.8955
Email: mmgorman@wiscorp.com
WWWWeb: <http://www.wiscorp.com>*

[\[The Article Archive\]](#)

The Data Administration Newsletter
Robert S. Seiner - Publisher and Editor - rseiner@tdan.com