



**Whitemarsh**  
Information Systems Corporation

*Data Semantics Management*  
*Volume 1*  
*Rationale, Requirements and*  
*Architecture*

Michael M. Gorman

*Whitemarsh Information Systems Corporation*  
*2008 Althea Lane*  
*Bowie, Maryland 20716*  
*Tele: 301-249-1142*  
*Email: [Whitemarsh@wiscorp.com](mailto:Whitemarsh@wiscorp.com)*  
*Web: [www.wiscorp.com](http://www.wiscorp.com)*

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where Whitemarsh Press is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

©2008 Whitemarsh Information Systems Corporation  
All rights reserved.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If legal advice or other expert assistance is required, the services of a competent professional person should be sought. FROM A DECLARATION OF PRINCIPLES JOINTLY ADOPTED BY A COMMITTEE OF THE AMERICAN BAR ASSOCIATION AND A COMMITTEE OF PUBLISHERS.

Reproduction or translation of any part of this work beyond that permitted by section 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright holder is unlawful. Requests for permission or further information should be addressed to Whitemarsh Press.

ISBN978-0-9789968-4-0

Printed in the United States of America

# Table of Contents

Preface .....	xix
Acknowledgments .....	xxiii
1 Rationale for Data Semantics Management .....	1
1.1 Rationale .....	2
1.2 The Problem .....	3
1.3 Effects of The Problem .....	3
1.4 Contents of this Book .....	4
1.4.1 Chapter 2, Why Semantics .....	6
1.4.2 Chapter 3, Data Semantics Failures .....	8
1.4.3 Chapter 4, Engineering Data Semantics .....	9
1.4.4 Chapter 5, Semantics of Names .....	10
1.4.5 Chapter 6, Semantic Hierarchies .....	11
1.4.6 Chapter 7, Value Domain Management .....	13
1.4.7 Chapter 8, Fact Specification Cases .....	14
1.4.8 Chapter 9, Data Element Model .....	15
1.4.9 Chapter 10, Specified Data Models .....	17
1.4.10 Chapter 11, Implemented Data Models .....	18
1.4.11 Chapter 12, Database Object Classes .....	20
1.4.12 Chapter 13, Operational Data Models .....	22
1.4.13 Chapter 14, Interface Data Models .....	24
1.4.14 Chapter 15, Work Plans .....	26
1.4.15 Chapter 16, Summary .....	26
1.5 Rationale Summary .....	26
1.6 Questions and Exercises .....	27
2 Why Semantics .....	31
2.1 Rationale for Data Management .....	33
2.2 Data Semantics .....	39
2.2.1 Taxonomies and Ontologies .....	42
2.2.2 Semantic Management Models .....	44
2.2.3 Data Element Models .....	47
2.2.4 Specified Data Models .....	49
2.2.5 Implemented Data Models .....	52
2.2.6 Database Object Models .....	56
2.2.7 Operational Data Models .....	61
2.2.8 View Data Models .....	64

---

*Data Semantics Management. Rationale, Requirements & Architecture*

---

2.2.9	Data Semantics Summary .....	66
2.3	Life Cycle Costs .....	67
2.4	Error Classes .....	75
2.5	Why Semantics Summary .....	77
2.6	Questions and Exercises .....	78
3	Data Semantics Management Failures .....	81
3.1	Fundamentally Flawed Model .....	83
3.1.1	The Focus of Standardization Was Too Low .....	86
3.1.2	Standardization Was Too Focused on <i>Names</i> .....	88
3.1.3	Critical Standardization Efforts Were Ignored .....	89
3.1.4	Critical Context and Subject Matter Materials Were Missing ...	90
3.1.5	The DoD Metadata Repository System Was Unacceptable ....	91
3.1.6	The Existing Procedures Excluded Critical Areas .....	92
3.2	No Accommodation for Enterprise-wide Data Architectures .....	92
3.3	Multiple Implementation Technologies .....	94
3.4	Central Standardization and Maintenance Authority .....	95
3.5	Error Classes .....	96
3.6	Data Semantics Management Failures Summary .....	97
3.7	Questions and Exercises .....	98
4	Engineering Data Semantics .....	101
4.1	Data Semantics .....	104
4.2	Data Semantics Alternatives .....	105
4.3	Effect of Complex Data Architectures .....	109
4.4	Effect of Client/Server .....	111
4.5	Effect of Internet .....	111
4.6	Effect of Enterprise Data Architecture .....	112
4.7	Effects of Binding Evolutions .....	113
4.7.1	1950s-1960s, Programs and Data Completely Self Contained .....	115
4.7.2	1960s, Program Accessing Independently Stored Data ....	116
4.7.3	1970s, Programs Accessing Data Through Database Management Systems .....	118
4.7.4	1980s and 1990s, Programs Accessing Data Through Views .....	121
4.7.5	2000s, Programs Accessing Data Constructed as XML Streams .....	124

## Table of Contents

---

4.7.6	2006 and Beyond, Programs Accessing Data Through SOA Architectures .....	127
4.8	Error Classes .....	130
4.9	Engineering Data Semantics Summary .....	131
4.10	Questions and Exercises .....	132
5	Semantics of Names .....	135
5.1	Traditional Approach Problems .....	138
5.1.1	Data Elements Are Synonyms for Table Columns, Screen Cells, Entity Attributes, or Report Fields .....	139
5.1.2	Data Elements Do not Have Names .....	141
5.1.3	Prime Word, Modifier[s] and Class Word <i>Are</i> Part of the Data Element's Name .....	142
5.1.4	Modifiers are from a Single Homogeneous Set .....	143
5.1.5	That There Is Only a Choice of One Class Word .....	144
5.2	A Strategy That Works .....	145
5.3	Single String Names .....	150
5.4	Semantic Parts .....	150
5.4.1	Prime Word Semantic Component .....	152
5.4.2	Common Business Name Semantic Component .....	154
5.4.3	Modifiers Subtype Semantic Component .....	157
5.4.4	Class Words Subtype Semantic Component .....	162
5.4.5	Full Name Construction .....	165
5.4.6	Semantics Component Summary .....	167
5.5	Error Classes .....	168
5.6	Semantics of Names Summary .....	169
5.7	Questions and Exercises .....	169
6	Semantic Hierarchies .....	173
6.1	Semantic Hierarchies Introduction and Scope .....	173
6.2	Concepts and Engineering .....	174
6.2.1	Meta Category Value Type Classes .....	176
6.2.2	Meta Category Value Types .....	177
6.2.3	Meta Category Values .....	179
6.2.3.1	Semantic Modifier Hierarchies .....	179
6.2.3.2	Data Use Modifier Hierarchies .....	183
6.2.4	Meta Category Value Types and Meta Category Value Hierarchies Summary .....	185
6.3	Application of Semantic Hierarchies .....	186

---

6.3.1	Meta Category Value Type Class	187
6.3.2	Meta Category Value Type	191
6.3.3	Meta Category Value	195
6.3.4	Meta Category Value Assignments	198
6.4	Error Classes	205
6.5	Semantic Hierarchies Summary	206
6.6	Questions and Exercises	207
7	Value Domain Management	213
7.1	Introduction and Scope	213
7.2	Concepts and Engineering	218
7.2.1	Detection of Value Sets	218
7.2.2	Process of Establishing Value Sets	218
7.3	Value Set Enforcement Environments	220
7.3.1	Direct Value Domains	221
7.3.2	Indirect Value Domains	222
7.3.3	Reference-Data Models	223
7.3.4	Value Set Enforcement Environments	226
7.4	Employing Value Domains	229
7.4.1	Value Domain Definition	232
7.4.1.1	Value Domains	232
7.4.1.2	Value Domain Structures	235
7.4.1.3	Value Domain Structure Types	241
7.4.1.4	Value Domain Data Types	241
7.4.2	Value Domain Value Definition	243
7.4.2.1	Value Domain Values	244
7.4.2.2	Value Domain Value Structures	246
7.4.2.3	Value Domain Value Structure Types	248
7.4.3	Value Domain Value Assignment	249
7.4.3.1	The ISO 11179 Data Model Ambiguity	250
7.4.3.2	Data Element Value Domain Value Assignment	252
7.5	Error Classes	256
7.6	Value Domain Management Summary	257
7.7	Questions and Exercises	258
8	Fact Specification Cases	263
8.1	Introduction and Scope	263
8.2	Concepts and Engineering	264
8.2.1	Simple Facts	265

## Table of Contents

---

8.2.2 Compound Facts .....	265
8.2.3 Group Facts .....	266
8.2.4 Pair Facts .....	269
8.2.5 Related Facts .....	270
8.2.6 Complex Facts .....	270
8.3 Data Integrity Rules .....	271
8.3.1 Single Table, Single Column .....	276
8.3.2 Single Table, Multiple Column .....	277
8.3.3 Single Table, Single Column, Multiple Row .....	278
8.3.4 Single Table, Multiple Column, Single Row Derived Data ..	279
8.3.5 Single Table, Single Column, Multiple Row Derived Data ..	280
8.3.6 Multiple Table Derived Data .....	281
8.3.7 Multiple Table, Multiple Column (Referential Integrity) ..	282
8.4 Employing Fact Specification Cases .....	284
8.4.1 Data Elements .....	284
8.4.2 Derived Data Elements .....	289
8.4.3 Compound Data Elements .....	294
8.4.4 Attributes .....	301
8.4.4.1 Attribute Creation and Maintenance .....	305
8.4.4.2 One Attribute to Multiple Entities .....	308
8.4.4.3 Multiple Attributes to One Entity. ....	309
8.4.5 Columns .....	310
8.4.5.1 Creation and Maintenance of Columns .....	312
8.4.5.2 One Column to Multiple Tables .....	321
8.4.5.3 Multiple Columns to One Table .....	321
8.4.6 DBMS Columns .....	324
8.4.6.1 Creation and Maintenance of DBMS Columns .....	327
8.4.7 View Columns .....	331
8.4.7.1 View Column Creation and Maintenance .....	334
8.4.7.2 View Column and DBMS Column Assignment .....	337
8.4.7.3 View Column Structures .....	338
8.4.7.4 View Column Structure Processes .....	341
8.4.8 Database Object Table Process Columns .....	343
8.5 Semantic Checking .....	346
8.6 Error Classes .....	347
8.7 Fact Specification Cases Summary .....	348
8.8 Questions and Exercises .....	349
Appendix 1 Database Architecture Classes .....	353

Data Semantics Management. Rationale, Requirements & Architecture

Reference Data .....	361
Original Data Capture Databases .....	362
Transaction Data Staging Area Databases .....	363
Operational Data Store Databases .....	363
Warehouse Databases .....	366
Index .....	369

## Figures

<b>Figure 1.</b> “Work” breakdown structure. . . . .	xix
<b>Figure 2.</b> Levels of abstraction required for data standardization. . . . .	38
<b>Figure 3.</b> Semantics management model components. . . . .	43
<b>Figure 4.</b> Key components of the Data Element Model. . . . .	46
<b>Figure 5.</b> Key components of the Specified Data Model. . . . .	50
<b>Figure 6.</b> Key components of the Implemented Data Model. . . . .	53
<b>Figure 7.</b> Database Object Model. . . . .	59
<b>Figure 8.</b> Key components of the Operational Data Model. . . . .	62
<b>Figure 9.</b> Key components of the View Data Model. . . . .	65
<b>Figure 10.</b> Phases for the traditional information systems development life cycle. . . . .	69
<b>Figure 11.</b> Iterative systems development life cycle. . . . .	71
<b>Figure 12.</b> Cost structure of systems development life cycle . . . . .	73
<b>Figure 13.</b> Same named semantics from data elements through to view models. . . . .	106
<b>Figure 14.</b> Differently named semantics from data elements through to view models. . . . .	108
<b>Figure 15.</b> 1950s & 1960s: Complemented embedded computing environment. . . . .	115
<b>Figure 16.</b> 1960s: Data independently stored from data. . . . .	117
<b>Figure 17.</b> 1970s: Data access through database management systems. . . . .	119
<b>Figure 18.</b> 1980s & 1990s: End user interaction with databases through DBMSs via database views. . . . .	122
<b>Figure 19.</b> 2000s Not SOA: Programs accessing data constructed as XML streams . . . . .	125
<b>Figure 20.</b> 2000s with SOA: Programs accessing data through SOA architectures . . . . .	128
<b>Figure 21.</b> Complete name construction for the business fact:, Salary. . . . .	146
<b>Figure 22.</b> Metadata architecture for data semantics management. . . . .	148
<b>Figure 23.</b> Levels of business fact specification and binding. . . . .	149
<b>Figure 24.</b> Data element semantics mapping to contained business facts and containers . . . . .	153
<b>Figure 25.</b> Examples of common business names . . . . .	155
<b>Figure 26.</b> Accuracy modifier example. . . . .	159
<b>Figure 27.</b> Geography modifier example. . . . .	160
<b>Figure 28.</b> Organizational modifier example. . . . .	160
<b>Figure 29.</b> Temporal modifier example. . . . .	161

<b>Figure 30.</b> Business data type class word subclass .....	163
<b>Figure 31.</b> Role class word subclass .....	164
<b>Figure 32.</b> Units class word subclass .....	165
<b>Figure 33.</b> Semantic hierarchy data model. ....	176
<b>Figure 34.</b> Meta category value types and meta category value hierarchies .....	182
<b>Figure 35.</b> Two methods of binding semantics into tables and columns of operational databases .....	183
<b>Figure 36.</b> Meta category value type and meta category value hierarchies .....	185
<b>Figure 37.</b> Meta category value type class. ....	187
<b>Figure 38.</b> Meta category value type class update screen. ....	188
<b>Figure 39.</b> Prefix subseted meta category value types and meta category values. ....	189
<b>Figure 40.</b> Suffix subseted meta category value types and meta category values. ....	190
<b>Figure 41.</b> Meta category value types. ....	191
<b>Figure 42.</b> Meta category value type update screen. ....	192
<b>Figure 43.</b> Meta category value type subsetting of meta category values. ..	193
<b>Figure 44.</b> Meta category value subsetting with collapsed meta category values. ....	194
<b>Figure 45.</b> Meta category value type subsetting with expanded meta category value. ....	195
<b>Figure 46.</b> Meta category values. ....	197
<b>Figure 47.</b> Meta category value update screen. ....	198
<b>Figure 48.</b> Data element browse screen. ....	199
<b>Figure 49.</b> Data element update screen. ....	200
<b>Figure 50.</b> Allocated meta category value, annual to sales data element. ..	201
<b>Figure 51.</b> Allocating the prefix meta category value, annual to the sales data element. ....	202
<b>Figure 52.</b> Complete allocation of meta category values for data element, salary. ....	203
<b>Figure 53.</b> Computer generated data element name for Salary. ....	204
<b>Figure 54.</b> Reference Data meta-data model for active and passive environments. ....	224
<b>Figure 55.</b> Value Domain management meta model. ....	231
<b>Figure 56.</b> Value domains. ....	233
<b>Figure 57.</b> Value domain update screen. ....	234
<b>Figure 58.</b> Value domain structures. ....	235

## Table of Contents

---

<b>Figure 59.</b> Value domain structure update screen. ....	236
<b>Figure 60.</b> Bill of materials data model. ....	237
<b>Figure 61.</b> Value domain structure types. ....	241
<b>Figure 62.</b> Value Domain structure type update screen. ....	242
<b>Figure 63.</b> Value domain data types. ....	242
<b>Figure 64.</b> Value domain data type update screen. ....	243
<b>Figure 65.</b> Value domain values. ....	244
<b>Figure 66.</b> Value domain values update screen. ....	245
<b>Figure 67.</b> Value domain value structures. ....	246
<b>Figure 68.</b> Value domain value structure update screen. ....	247
<b>Figure 69.</b> Value domain value structure types. ....	248
<b>Figure 70.</b> Value domain structure type update screen. ....	249
<b>Figure 71.</b> Value domain management diamond data structure. ....	250
<b>Figure 72.</b> Data element value domain assignment. ....	253
<b>Figure 73.</b> Data element value domain assignment messages. ....	254
<b>Figure 74.</b> Attribute value domain assignment–duplicate assignment error. .....	255
<b>Figure 75.</b> Attribute value domain assignment–assigning value domain subset. ....	256
<b>Figure 76.</b> Address bill of materials data model. ....	267
<b>Figure 77.</b> Data integrity rule specification meta model. ....	272
<b>Figure 78.</b> Data Integrity Rule binding meta model. ....	273
<b>Figure 79.</b> Data element meta model. ....	286
<b>Figure 80.</b> Data elements. ....	287
<b>Figure 81.</b> Data element insert screen. ....	288
<b>Figure 82.</b> Derived and compound data elements model. ....	290
<b>Figure 83.</b> Derived data elements. ....	291
<b>Figure 84.</b> Derived data element update screen. ....	292
<b>Figure 85.</b> Derived data element data element assignment screen. ....	293
<b>Figure 86.</b> Compound data element derived data element assignment screen. .....	294
<b>Figure 87.</b> Compound data element screen. ....	297
<b>Figure 88.</b> Compound data element update screen. ....	298
<b>Figure 89.</b> Compound data element structure screen. ....	299
<b>Figure 90.</b> Compound data element structure update screen. ....	300
<b>Figure 91.</b> Compound data element to data element assignment screen. ...	301
<b>Figure 92.</b> Attribute meta model. ....	304
<b>Figure 93.</b> Attributes within an entity. ....	305
<b>Figure 94.</b> Entity attribute update screen. ....	307

---

<b>Figure 95.</b> Selecting a data element for a new attribute. ....	308
<b>Figure 96.</b> Create one attribute in many entities. ....	309
<b>Figure 97.</b> Create many attributes in one entity. ....	310
<b>Figure 98.</b> Column meta model. ....	313
<b>Figure 99.</b> List of columns in a table. ....	314
<b>Figure 100.</b> Screen for creating or updating a column. ....	316
<b>Figure 101.</b> Selecting data element for column. ....	317
<b>Figure 102.</b> Selecting attribute reference for new column. ....	318
<b>Figure 103.</b> Subordinate column components to a defined column. ....	319
<b>Figure 104.</b> One column to multiple tables. ....	322
<b>Figure 105.</b> Multiple columns to one table. ....	323
<b>Figure 106.</b> DBMS Column meta model. ....	326
<b>Figure 107.</b> DBMS columns. ....	327
<b>Figure 108.</b> DBMS column update screen. ....	328
<b>Figure 109.</b> Select column screen. ....	330
<b>Figure 110.</b> View column meta model. ....	333
<b>Figure 111.</b> View columns. ....	335
<b>Figure 112.</b> View column update screen. ....	336
<b>Figure 113.</b> View column DBMS column assignment. ....	337
<b>Figure 114.</b> View column structures. ....	339
<b>Figure 115.</b> View column structures update screen. ....	340
<b>Figure 116.</b> View column structure processes. ....	341
<b>Figure 117.</b> View column structure process update screen. ....	342
<b>Figure 118.</b> Database object meta model. ....	343
<b>Figure 119.</b> Database object table process column assignment. ....	345
<b>Figure 120.</b> Data architecture classes topology. ....	354

## Tables

Table 1. Data semantics components .....	41
Table 2. Semantic management model components. ....	46
Table 3. Data Element Model components. ....	49
Table 4. Specified Data Model components. ....	52
Table 5. Implemented Data Model components. ....	55
Table 6. Database Object Model components .....	61
Table 7. Operational Data Model components. ....	64
<b>Table 8.</b> View Data Model components .....	66
Table 9. Proper location of data semantics specifications .....	105
Table 10. Component existence across the decades .....	114
Table 11. 1950s-1960s: Programs and data completely self contained .....	116
Table 12. 1960s: Program accessing independently stored data .....	118
Table 13. 1970s: Programs accessing data through database management systems .....	121
Table 14. 1980s and 1990s: Programs accessing data through independently created views .....	124
Table 15. 2000s: Programs accessing data constructed as XML streams. ...	127
Table 16. 2000s: Programs accessing data through SOA architectures .....	130
Table 17. Meta attributes appropriate for data elements and contained columns, et al. ....	141
Table 18. Examples of semantic modifiers classes. ....	144
Table 19. Examples of data user modifiers (a.k.a., class words) .....	145
Table 20. Data element standardization techniques comparison .....	147
Table 21: Repository levels and pairs .....	155
Table 22. Semantically equivalent data structures. ....	162
Table 23. Enumerated code values for a collection of healthcare tables for Yes No columns. ....	217
Table 24. Reference data specification data-model coupled with active and passive designations. ....	226
Table 25. Reference data problems and approach solution effects. ....	228
Table 26. Tables and columns for bill of materials and single file recursion .....	238
Table 27. Part table. ....	239
Table 28. Part structure table. ....	240
Table 29. Part structure type table rows. ....	240
Table 30. Address parts table rows. ....	268
Table 31. Address parts structure table rows .....	268

Table 32. Address parts structure type table. ....	269
Table 33. Data integrity rule specification meta model. ....	274
Table 34. Fact case applicability to a data element. ....	285
Table 35. Fact case applicability to a derived data element. ....	289
Table 36. Fact case applicability to a compound data element. ....	295
Table 37. Fact case applicability to an attribute. ....	303
Table 38. Fact Case applicability to a column. ....	312
Table 39. Fact case applicability to a DBMS column. ....	325
Table 40. Fact case applicability to a database object table process column. .....	332
Table 41. Fact case applicability to a database object table process column. .....	345

## Forward

This book is divided into two volumes. Both volumes contain this Forward. The Figure and table numbers. The tables of Contents for topics, figures, and tables for both volumes reside in both volumes. Finally, the Index for each volume is contained in both volumes.

The focus of Volume 1 is Rationale, Requirements, and Architecture for Data Semantics Management. In short, this book makes the case why accomplishing data semantics management is so critical to the overall Enterprise-wide success of commonly understood and shared data.

The approach embraced by these two volumes is founded on the idea of top-down and centralized architecture, engineering, policies, and procedures, but bottom-up, distributed accomplishment. If attempted only top-down, the outcomes will mirror familiar centralized czar-like failures of the past. If accomplished only bottom-up, the outcomes will be the vast forests of semantic stove pipes. This book embraces the best of both, and sets down a data semantics management metadata management architecture within the Metabase System that enables some aspects to be accomplished top-down and centralized as is appropriate while allowing individual database<sup>1</sup> efforts to be accomplished in a decentralized manner which is also appropriate.

The Metabase System is a metadata management system from Whitemarsh. It can be requested, downloaded, and installed from the Whitemarsh website, [www.wiscorp.com](http://www.wiscorp.com). All the screen shots in these two

---

<sup>1</sup>. The term, database, used in this book, can have two meanings. There is a third use, but that is a wrong use. Under the first “good” use, database is something that is constructed, that is, a Customer Database, or a Marketing Database. In that state, it is something that is real in that it contains a dictionary of all its parts (called the database’s schema), records of actual data for Customers or Sales, relationships among the data records, and indexes that speed access to individual records. In this context, a database consists of: Dictionary, Indexes, Relationships, and Data.

The second “good” meaning for database is that it is something that is accomplished such as Enterprise Database. That use implies a state of organization across large collections of data that is characterized by consistent semantics, common meanings, easy to understand data structures, integration, and non-redundancy. Thus, it is a state of success and achievement with respect to an enterprise’s data.

The third meaning for database is DBMS. That is, database management system. This is bad use of the term database. A DBMS is a very large software system for defining, updating, reporting from, securing, and managing databases. Oracle is a DBMS company. It’s DBMS is called Oracle. Similarly, IBM has the DB2 DBMS; Sybase as the Sybase DBMS; and Microsoft has the DBMS SQL Server 2005.

In this book, the first two definitions are employed and are distinguishable by their context. The third definition of database is never used because that causes confusion.

volumes are examples from the Metabase System. The Metabase System user guides can be downloaded from the Metabase System's web-pages. The Metabase System brings Data Semantics Management alive.

Volume 2 is the logical follow-on volume, Deployment. It sets out a complete set of data semantics management applied examples using the single-user, single-metabase database version of the Metabase System.

Throughout both volumes of this book, the following very simple data modeling convention was employed. 1. A line from a meta-entity to itself with a one arrowhead is a recursive relationship. Ex. Entity contains entity. 2) A line between two meta-entities with a single arrow head is a one-to-many relationship. Ex. Entity has zero, one, or more attributes. 3) A first component that has two one-to-many relationships to a second meta-entity, and a third meta-entity that has a single one-to-many line between it and the second meta-entity is a bill-of-materials data structure. Ex. concept (first meta-entity), concept-structure (second meta-entity), and concept structure type (third meta-entity).

This two-volume book was published subsequent to the three Whitemarsh books:

- Data Interoperability Community of Interest Handbook.
- Enterprise Architectures.
- Strategy for Successful Development of Business Information Systems<sup>2</sup>.

The Data Interoperability Community of Interest Handbook book sets out the engineering and architecture of the Communities of Interest that engineer the databases essential for enterprise-wide shared data. This book provides the organizational, and functional engineering. This book also provides the policies, procedures, guidelines, job descriptions, duties, and responsibilities.

Collectively this book identifies the "who" and "how-organized" aspects of successful data sharing.

It is important to note that this book is based on three complementary Communities of Interest (COI). One supports 26 NATO nations in the creation

---

<sup>2</sup>. A business information system is general term for software systems that employ data to achieve some business purpose. Included are specialized packages such as Customer Management, Accounts Payable, Human Resources, and even Internet browsers. Typically business information systems access data through SQL DBMSs via ODBC or JDBC. Systems such as WinSQL could also be considered business information systems as they allow users to access and report business data. Excluded from this definition are software systems such as operating systems and database management systems.

---

## *Table of Contents*

---

of a shared-data specification for Command and Control databases; another sets out the program, organization, and management of standards for all American National Standards in the United States; and the last, arguably the most successful Community of Interest, ANSI INCITS H2 Technical Committee on Database, defines and standardizes the SQL language.

The second book, *Enterprise Architectures*, sets out and describes the critical components that comprise Enterprise Architectures. Each of these components is thoroughly defined and interrelated one to the other. Collectively these Enterprise-Architecture components identify “what” has to be engineered before any real database and/or business information system efforts can be started.

The third book, *Strategy for Successful Development of business information systems*, provides a very well-proven and successful approach to business information system development. This book contains real examples, work products, and the like. Collectively, this book is the “how-process” book.

This third book is based on highly successful business information system development efforts. In one such effort, a \$2.4 Million business information system was created for just more than \$300 thousand. How? It was accomplished via prototyping and a business information system generator. This business information system was 6600 function points and through the use of prototyping and the business information system generator it was implemented for \$50 per function point instead of the industry average, \$400.

While all these three books are important, none of them has focused in any intensive way on data semantics management. Data semantics management is important because without it, there is a very high probability of failure for shared-data efforts. There is also a very high cost to be paid if enterprise-wide business information systems are not data-based. For example, a study has shown that there is a 4.6 times quantity increase in what has to be implemented if a process-first approach is followed rather than a data-first approach.

The lessons of not having proper data semantics management are both painful and costly. The United States Department of Defense (DoD) has been trying to accomplish DoD-wide data standardization for more than 30 years. Failure has generally been the result. These failure costs exceed \$500 million.

In contrast, the business information systems of the U.S. Navy are highly successful because a unified data-centric approach that has been championed and made mandatory by the Navy Office of the CIO.

Another data-centric success has been the Multilateral Interoperability Program (MIP) that has built and mandated a shared-data model for all Command and Control data across all 26 NATO nations. This MIP effort provides the organization-engineering structure for the Data Interoperability Community of Interest book described above.

There is a multi-hundred-**billion** dollar effort within the United States Department of Defense focused on the development of Future Combat Systems (FCS). While this effort will eventually be declared successful through a sheer force of will, its cost and quality will have been dramatically affected—negatively—because the effort is hardware and process focused almost to the exclusion of any data-focus or data-engineering. It seems that there are no “data oriented” deliverable requirements in the contracts that govern the FCS effort.

This book is therefore 100% focused on data semantics management. As the picture of Captain Ahab shows in his battle against Moby Dick, successfully achieving data semantics management is a “whale” of a battle. Failing to achieving it in your enterprise is sort of like getting eaten by Moby Dick.

Essentially, we're all Captain Ahab. We're determined and we know the journey. In the book by Herman Melville, Ahab's battle against Moby Dick ultimately fails. For us, that is failing to achieve data semantics management.



Unlike Captain Ahab, however, we now have a much more sophisticated understanding of objectives. Our methodologies and approach are tested. We are supported by sophisticated tools. This book is about winning this "whale" of a battle.

## Preface

If you have ever asked the question, “What do you mean by that,” you get the whole point of this book. There are also two follow on issues: The frequency of that question during an exchange, and the frequency of any follow-up correction and re-questioning. The goal of Data Semantics Management is to both minimize the first question and to eliminate any of the follow-up questions.

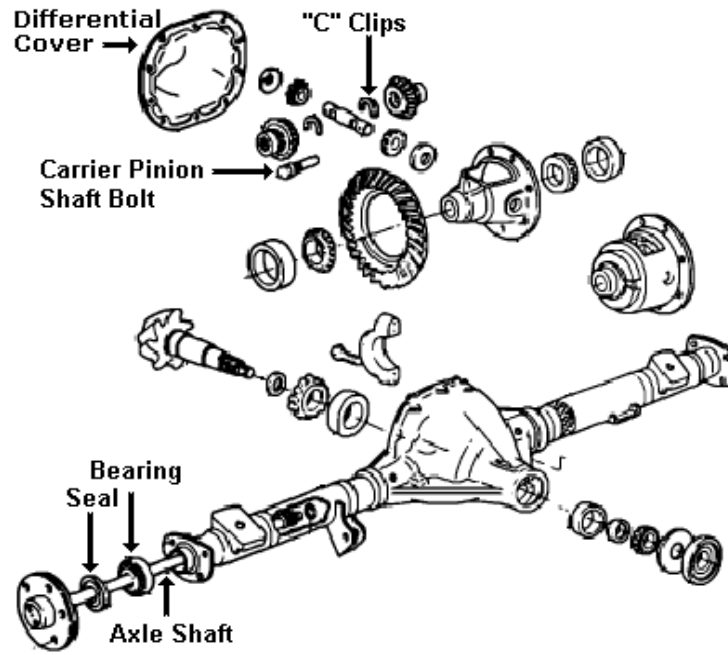
When verbal exchanges occur, there is commonly a most sophisticated computer on either end of the exchange: the brain. As the message receiver’s brain takes in the words, the brain’s semantic processor is ingesting and sending out messages such as smile, nod, frown, or “huh?” The message sender’s eyes, ears, and brain have an opportunity to stop, reverse, explain, and so forth until there is either a terminal error (e.g., “I give up, go away!”) or some other gesture of acceptance.

For example, this author and a newly met professional colleague were assigned to a project. A key element of the project was “work breakdown structure.” The colleague thought of a work breakdown structure as the physical parts’ explosion of a “work.” That it is, a noun which means for example, the rear axle of a truck. To the colleague, the work breakdown structure looked like Figure 1. In his mind, each work breakdown structure element was a noun phrase as in the Figure 1 engineering diagram of a 1994 Ford Ranger Rear Axle.

In contrast, this author thought of a work breakdown structure is a hierarchically organized set of work tasks where in each work break down structure element began with a verb. An example would be to “Write the preface of the Data Semantics Management book.” It was several days of working together before one of us reached that terminal error of “Stop, what do you mean by work breakdown structure?” Once the meaning of the commonly employed terms was synchronized, the exchanges were recalibrated and productive exchanges restarted.

Unsaid are the constantly occurring interpretations and transformations. Now, add to these exchanges, language (e.g., French to English, or Chinese to German), and cultures (E.g., European to American, or Asian to African). In either of these cases the quantity of initial questions and gestures, and of the followup questions and different gestures dramatically increases because the semantic disconnect in both areas must first be identified, analyzed, and resolved. Only then can meaningful exchanges occur.

## 1994 Ford Ranger Rear Axle Parts Explosion



**Figure 1.** “Work” breakdown structure.

For example, again, this author traveled to and from elementary school sixth grade (note, this was in the early years of the first decade of the last half of the last century of the previous millennium) with the grandson of the ambassador from Brazil. The grandson was learning English. One day he exclaimed, “English is so hard.” I inquired what he meant. He said, “I understand what is meant by ‘cutting a tree down,’ but what is meant by ‘and now you cut it up?’” Here’s another: What’s the difference between “slow up” and “show down?”

In contrast, it is often said that couples who have been married many decades can finish each other’s sentences. Why? Could it be because they have completely absorbed each other’s cultures and mannerisms? Likely. When couples meet for the first time there is commonly many long phone calls or conversations. Why? It’s probably because there’s an ingesting and synchronization of their individual cultures and mannerisms. There’s a thirst for a relationship bridge constructed of common semantics.

In all these cases, it is also common that as time goes by from days to months to years and even decades, the exchanges become fewer. It may not be because they have grown distant, Rather, it may be because there is now an acceptable environment for exchange without all the copious culture and mannerism synchronization activities that occurred when the couple first met and were dating.

With all that said, and hopefully agreed to, changing the environment of exchanges from “human” to information technology greatly complicates the whole process of exchange. From the verbal exchange scenario there are the following components:

- Commonly recognized words or phrases.
- Mutual semantic trust of inferred meanings.
- Interactive posting and receiving of “error messages” and responses.
- Previous uses of previously engineered and agreed-to semantic maps.
- Language matches.
- Culture matches.
- Context matches.
- Content matches.

It is the premise of this book that all information technology data exchanges are just surrogates for verbal exchanges. The information technology data exchanges exist merely to increase velocity, or to reduce costs. Consequently, all IT information exchanges naturally follow the same rules and requirements as do verbal exchanges. A critical and key difference between human versus computer-based exchanges is that for human exchanges, the volume and velocity are far smaller and slower than computer-based exchanges, and further, on a per-exchange basis, the human-exchanges are employed through a much larger and profoundly more capable processor (the brain) than is the computer for the computer-based exchanges.

This book is all about engineering high velocity, semantically harmonious information exchange environments which are critical because “human brains” do not exist on either end of the IT exchange. This book is not about the engineering high velocity, semantically harmonious information exchange environments that are perfect, automatically malleable, and endlessly mutable. That’s because this book is not fiction. At best, the engineered environments, just like the human environments, work well only after significant periods/efforts in which the word/phrases, meanings, language, culture, context, and content that frame the information exchanges

have all been satisfactorily addressed. Perfection and total “automation” cannot be achieved, so you must stop dreaming and get down to the real work of establishing a productive information exchange environment.

The chapters in this two-volume book are:

**Volume one**

1. Rationale for Data Semantics Management
2. Why Semantics
3. Data Semantics Failures
4. Engineering Data Semantics
5. Semantics of Names
6. Semantic Hierarchies
8. Fact Specification Cases

**Volume two**

9. Data Element Model
10. Specified Data Models
11. Implemented Data Models
12. Database Object Classes
13. Operational Data Models
14. Interface Data Models
15. Work Plans
16. Summary

Appendix 1, Data Architecture Classes appears at the end of Volume 1.

## Acknowledgments

My journey to an understanding of data semantics management started almost 40 years ago. Over this 40-year span six individuals stand out. I would like to acknowledge them here. I would also like to acknowledge that this book represents materials presented to many companies through the Whitemarsh Data Interoperability Seminars. Questionnaires were issued after every day long (225 slide) presentation. The results' tabulation is provided at the end of this section.

At the very start of this trip, I was a database analyst for the System Development Corporation (SDC). SDC was one of the original inventors of database, data management, and database management systems. SDC, along with Lincoln Labs and the MITRE Corporation invented this entire discipline in the late 1950s and early 1960s. SDC created and deployed a database management system (DBMS), TDMS (Time-shared data management system), for the U.S. Air Force in the early 1960s.

As an employee of SDC in the Washington, D.C. area in 1969, I was introduced to the "programmer replacement system." That is, a DBMS called Commercial Data Management System (CDMS). It was a commercial version of TDMS. With CDMS you could design hierarchical data structures, write data conversion scripts, load extremely large databases, and create and execute large collections of ad hoc reports to prove or disprove assertions about bodies of knowledge. In 1969, a 3,000,000 character database was extremely large.

My boss was Alec Bumstead. He was the first to question whether I had any cogitative abilities at all. I read the CDMS manuals, and designed the database. He read the design and started to ask questions. For every question I had only "ah, er, I hadn't thought of that" for answers. "Think, think, be disciplined, think, think," Alec would say. So I did. After much interaction, we managed to develop a database design, that when it was loaded with data and exercised, disproved several of the major conclusions of the "Coleman Report." What the DBMS, via a good database design and sophisticated queries, was able to prove and disprove in a matter of minutes took researchers weeks. CDMS's capabilities were mind boggling. Database, DBMS, and data management has been my life's work ever since. So, here's to the memory of Alec Bumstead.

In 1971, I started to work with Ted Ziehe. Ted was the application developer manager for MRI Systems Corporation of Austin, Texas. MRI had developed System 2000, which was a "grandson" of TDMS. At the time I was the PMIS architect. PMIS, a Planning and Management Information System, a

demonstration prototype funded by the U.S. Office of Education, took weekly data extracts from operational files and configured this data into a database of statistics about Students, Teachers, Schools, and Educational Programs. This data was used by Dallas Texas educators to measure the progress of students, teachers and programs against yearly educational objectives. Clearly this was a data warehouse. During this contract, we also built a database about PMIS. This “about-PMIS” database stored all the design artifacts, query specifications, the database designs, and the like. Clearly this was a metadata database. PMIS was very successful as a prototype and demonstration project. All during the PMIS database design phase, Ted Ziehe worked with us for days to get the database design just right. His strategy was to take 3x5 cards and put a data-subject title on each, and list the data facts about each subject. We would write the “short story” about the 3x5 card on the back. Once we had all the cards done, we put them onto a wall, and with string, make relationships between them in a one-to-many fashion. Ted would pepper us with all sorts of questions that addressed granularity, precision, and time synchronization. Eliminating redundancy was Ted’s big objective. To Ted, Database meant never having the fact in more than one place. To Ted, everything had to have a clear, unambiguous definition. It had to be clear, crisp, and once only. So, thanks Ted for being my first real data modeler and data semantics management teacher.

From 1979 through 1981 I had the privilege of working with Matt Flavin. Matt, the inventor of database object classes, worked for Ed Yourdon of New York City. Matt’s course and workshop, Fundamentals of Information Modeling, started with the premise that data is executed policy. Matt didn’t say it like that, but that’s what he meant. Matt was very critical of the relational model as he felt that in the name of table-based simplicity and rigor, the relational model had done away with the overall greater objective of database. That is, coherent, state-based expressions of enterprise policy. My contribution to Matt’s effort was end-to-end database project methodology, and database administration. Matt’s workshop showed that a rigorously engineered database design methodology would almost always result in the same database design, regardless of the team applying the methodology. Matt provided science and engineering to database designs, while at the same time restoring a critical and lost component: database object classes. So, here’s to the memory and contribution of a database pioneer and legend, Matt Flavin.

In 2001, I met Hank Lavender. Hank (actually Col Henry C. Lavender, USAF Retired) was the functional data administrator of the United States Defense Logistics Agency. Hank got interested in “correct and quality” based

logistics because he could never seem to get enough aluminum patches for all the bullet holes that “mysteriously appeared” while flying around the Vietnam countryside in his Douglas A-1 Skyraider. Hank was a leader of the Hobo Squadron. After the war, Hank retired and focused all his energies on logistics management. Hank and I worked together on the development of a strategy and methodology to achieve interoperability through shared-data. Hank and I presented this strategy over a two-year period to the DAMA International Conferences. We employed the Metabase System as the metadata management system. During this time there were many changes to the Metabase System so that it could accomplish shared-data specifications with maximum effect and minimum effort. So, thanks to Hank for all his contributions and constant reminders that “if it doesn’t make common sense, it probably just makes non-sense.”

During the last eight years, and specifically from 2001 through 2005, Bruce Haberkamp and Jim Blalock of the Office of the CIO of the United States Army and I have worked very hard in the development of the Army’s Data Interoperability Program. This program has resulted in Army policy, and a large series of documents that included seminars, workshops, methodology, and the employment of the Metabase System. The Data Interoperability Community of Interest Handbook, this Data Semantics Management book, modifications to the Metabase System, and a series of Data Interoperability workshops and seminars are the result. So, thanks to both Bruce and Jim.

From the above, you would be correct to understand that much of the data semantics management work has been focused on U.S. Federal Government efforts, especially the Department of Defense. That is certainly true. But during 2006 and 2007, I took the Data Interoperability Seminar on the road and have delivered it to a large quantity of “commercial” organizations. At the end of each very long day, the attendees were invited to fill out a questionnaire. These questions represent their report card of me on this whole approach. What follows are the questions and a tally of the questions. Following that is the list of companies that attended the seminar and filled out the questionnaire.

<b>Results of the Questionnaire Given to Attendees of the Data Interoperability Strategy Seminar</b>				
<b>Question</b>	<b>Strongly Agree</b>	<b>Agree</b>	<b>Disagree</b>	<b>Strongly Disagree</b>
Organizations would benefit from a comprehensive data management program including courses, methodologies, workshops, etc.	66	31	3	0
Organizations would benefit if databases (not DBMS) had consistent semantics for names for table columns across schemas.	80	17	0	3
Organizations would benefit from automatic data element, attribute and column naming and abbreviations built with consensus-based word lists and phrases (taxonomies).	65	35	0	0
Organizations would benefit from an automatic data element, attribute and column definitions derived from contexts, valued domains, etc.	69	39	2	0
Organizations would benefit if databases (not DBMS) had consistent semantics for value domains.	75	22	3	0
Semantics should be inheritable from iso 11179 data elements to attributes, and from attributes to columns.	42	53	5	0
There is value in having data model templates that can be used in defining column collections within tables.	61	39	0	0
XML schemas should be generated from a foundation of well designed and integrated data models.	63	33	4	0

<b>Results of the Questionnaire Given to Attendees of the Data Interoperability Strategy Seminar</b>				
<b>Question</b>	<b>Strongly Agree</b>	<b>Agree</b>	<b>Disagree</b>	<b>Strongly Disagree</b>
Organizations would benefit from metadata repositories that support metadata integration and non-redundancy across projects, departments, etc. that is, define-once and use many-times.	87	13	0	0
Organizations would benefit from metadata repositories that are part of the everyday work process rather than a post implementation documentation effort.	90	10	3	0
End-users would benefit if they had access to metadata repositories for names, definitions, and the like.	83	14	0	0
Organizations would benefit from metadata repositories if the repositories could capture, report, and support updates of analysis and design work products in an integrated non-redundant manner.	70	30	0	0
Project quality and work-speed would likely benefit if supported by comprehensive metadata repositories.	75	25	0	0
Organizations would benefit from metadata management strategies supported by both bottom-up reverse engineering, and top-down new database (not DBMS) manufacturing.	53	47	0	0

**Data Interoperability Seminar Attendees who filled out questionnaires.**

American Airlines	Hudson Bay Company
American Family Insurance	IESO (Independent Electric System Operator, Ontario Canada)
Bank of Montreal	Kohls Department Stores
Blue Cross	Landsend
BTE Corporation	RBC Capital Markets
Burlington Northern Santa Fe	Royal Bank of Canada
Canadian Tire	Science Applications International Corporation
Cap Gemini Energy	Sigma Systems
ChartWell	Sprint
Chevron Texaco	U.S. Department of Homeland Security Customs and Border Protection
CIBC	Wall-Street Consulting
Cingular Wireless	Washington [State] Courts
Circuit City	Wisconsin Physicians Service Insurance Corporation
CitiFinancial	Wisconsin Education Association Trust
CitiGroup	Wisconsin DOT
CUNA Mutual Group	
DOFASCO (Canadian Steel Corporation)	
Economical Insurance	
Essential Futures	
Federal Express	
Great Lakes Higher Education Loan Services	

It is on the shoulders of all the individuals and organizations I have acknowledged that I commend this book to your reading. What is more important, I commend this book to your adoption and implementation. Go forth and do data semantics management. You will have only increased productivity, increased quality, decreased costs and decreased risk to show for your efforts. Not too bad an accomplishment, I would suggest.

Michael M. Gorman  
September 2008

# 1

## Rationale for Data Semantics Management

---

The objective of this chapter is to set out the rationale for data semantics management. The rationale is presented by first citing the clear state of the world, that is, that we live in a world economy. With the Internet, there are no borders.

Commerce and communications of virtually all abstract goods are exchanged at the speed of the Internet. Good examples are all the Internet-based jokes that travel around from person to person and organization to organization almost instantaneously. Commerce too is similarly engineered. The computer on which this manuscript was created was built in Asia. The operating system is from Microsoft in Washington, the word processor from Canada, and the Metabase System that is the demonstration vehicle for all this book from a company in Florida with supplementary products from Arizona, Australia, Germany, South Africa, and England.

Simply stated, the most common languages throughout the world are not English, French, or Spanish. Rather, they are SQL, C, C++, because all communication throughout the world are first founded upon these universally accepted and unambiguously understood computer-based languages.

This chapter presents a clear statement of the problem that must be solved and several examples to illustrate it. Thereafter are examples of the effects of not having sufficiently engineered data interoperability.

This chapter sets out the remaining chapters in the book and conveys how each chapter contributes to addressing the two classes of problems that characterize the achievement of data interoperability through data semantics management.

This chapter, as does all chapters, concludes with both a summary and a set of questions and exercises that should be able to be accomplished by practitioners of the book. This book is not just to be read and thought about. Rather, it is to be acted upon. The former will provide intriguing thoughts and

---

concepts to consider. The later will provide real action plans coupled with a down-loadable production class metadata management system, the Metabase System, to manage data semantics in support of creating the interoperable data environments critical to the world economy.

## 1.1 Rationale

The rationale for data semantics management is simple. We live in a world economy. Operational aspects of commerce are so large and the requirements for commerce processing are so high that the world's economy cannot operate without a very high degree of the semantic trust that is essential for large volume and high velocity information exchanges.

Humans are natural semantic translators. Whenever an exchange fails, error messages, e.g., "huh?" are posted. The speaker almost always adjusts in real time and corrects the understanding errors in the exchange.

Computers, however, almost certainly do not have the ability to dynamically switch the appropriate words/phrase, meanings, language, culture, context, and content, that is, recalibrate the exchanges so as to make it semantically harmonious. Attempts to create computer-based semantic processors have existed for 50 years or more without much success. An early experiment was "Eliza" in which a human questioner post a question, for example, "Why do you hate me?" The computer responded with, "Why do you feel that you are hated?" It was "off to the races," with questions that were merely turned into syntactically transformed responses. In this environment, the transactions moved only as fast as the human respondent allowed. Information-technology-based information-exchanges however move at the "speed of light" without any reasonable manner of resolving semantic error messages. At best, the information exchange transaction is set aside, a message is posted, and it is dealt with by a human at some later time.

If there is too large a volume of human-intervention required messages, the whole process of information exchange grinds to a halt. Examples of high velocity information exchanges include credit-card approval processing, insurance claims processing, payroll creation and processing, material logistics management, tax return processing stock exchanges, check, and financial instrument clearing houses, and the cellular phone system. The very existence of each of these examples depends on highly engineered semantics that are agreed to by all parties to such an extent that the volume of error messages falls within the ability of the human systems to process them.

## **1.2 The Problem**

The problem that exists is to create information exchange environments such that the semantics exchange error rates and the volumes of such errors are within the acceptable limits of human correction processing. The level of effort that needs to be expended to achieve this is inversely proportional to the quantity and velocity of semantic errors that can be tolerated. For electronic commerce, the tolerable error rates must be low, hence the investment must be high. In contrast, entry of an application for employment may both generate errors and also permit human-corrective interaction because both the volume of applications is low, and the acceptable duration for processing can be relatively long.

## **1.3 Effects of The Problem**

The effects of human information-exchange errors can be severe. The gun-battle at the "OK Corral" might have resulted from such an error. In contrast, semantic-errors as a consequence of information technology can range from having a slight effect to a severe one. A rejection of a misread credit card at a store likely has small effect, while mixing up units (Metrics versus English) caused NASA to lose a \$125 million Mars orbiter. This was because one engineering team used Metric units while another used English units for a key spacecraft operation.

Information exchange errors are just reactions to a lack of semantic mapping efforts that occurred much earlier in the analysis, design, and development of IT systems. That a credit card reader failed to correctly read a card as a consequence of a worn magnetic strip is not a semantic exchange error and thus not attributable to a poor design of some aspect of an IT system. Semantic exchange errors originate somewhere within the IT system's design. For example, the "real" data types (versus Integer or Decimal) of the fields that were employed were not checked. Such checking should not be a human controlled process. Rather, the fundamental data types (e.g, metric vs English) of the fields should have been an absolutely required entry in the IT system's metadata<sup>3</sup> support environment. Similarly, strong data typing would

---

<sup>3</sup>. Metadata is a generic term that identifies all classes of information technology specifications across the enterprise. Hence all data and process specifications are metadata. All  
(continued...)

---

prevent an object's weight to be multiplied by an hour value. While such a multiplication would be mathematically possible, it would make no semantic sense. Consequently, such mistakes must be automatically prevented.

As stated above, the ability to tolerate and/or recover from semantic errors must be assessed and employed in the determination of the scope and volume of work that must be employed to prevent information exchange errors. In the case of the credit-card machine, manually cleaning the reading head, or manually entering the credit card number that is embossed is certainly within acceptable limits. The Mars orbiter, however, is very much the opposite. There was no opportunity for error processing and recovery for two reasons: By the time the error was noticed, the orbiter would have already crashed. Second, the error wasn't understood until some weeks after the orbiter was destroyed.

## 1.4 Contents of this Book

This section of the book is much more than just an enumeration of the chapters and a one-or-two liner introduction to each chapter. Rather, it contains the overall architecture, strategy and even "battle plans" to achieve data semantics management on an enterprise-wide basis. Procrastination in accepting the dictums and strategies of this book will only result in greatly increased costs and significantly more difficult efforts in the future.

With that in mind, every chapter of this book is targeted to the reduction of the two semantic exchange errors:

- Initial semantic disconnects.
- Follow-up semantic synchronization activities.

The first error class, initial semantic disconnects, is also complicated by the two classical statistical error types:

---

<sup>3</sup>(...continued)

requirements could also be considered metadata. A metadata database is a database within which all metadata is stored. A metadata database is a metabase. Whitemarsh has employed the term, Metabase, in this context since 1982. A metadata management system is a software system that captures, stores, reports, and manages all metadata. The Whitemarsh system that manages the Metabase is the Metabase System. Sophisticated metadata management systems, like the Metabase System, are multi-user and support the capture and reporting of metadata in a non-redundant, integrated manner across the enterprise.

---

- Type I Error: An error which occurs when a true hypothesis is rejected.
- Type II Error: An error which occurs when a false hypothesis is accepted.

A Type I error would occur in an information exchange in which a Social Security Number was represented with dashes and the receiving computer system was expecting a pure integer. A Type II error might be the acceptance of a Gender = 1 as acceptable when the sending system had a reference table of 0 = Male, and 1 = Female, while the receiving system has 1 = Male, and 2 = Female. These Type I and II errors can and must be prevented through real specification and implementation engineering.

The second error class, follow-up semantic synchronization activities, are those activities that an organization must perform to reestablish productive data interoperability. In the example above, these activities would first require that there are effectively, three gender codes. That is, 0 for Male, 1 for Male and Female, and 2 for Female. That would mean that while the total count of persons is the count of all instances from the three gender codes, some of the employees associated with code 1 are male and others are female.

One set of followup synchronization activities might be to fix source the systems so that all employees are either 0 and 1, or are 1 and 2.

Another fix might be to transform the understanding of the genders by means of a views that would query one database and underlying system using values 0 and 1 and query the other database and underlying system using values 1 and 2.

Which followup synchronization activities should be followed depend entirely on what is possible and practical. If the source systems are home-grown, the first approach might be both practical and possible. Alternatively, if the source systems are all ERP or commercial vendor type packages, the second approach might be the only one possible and practical.

Where this book can help with respect to the two error classes is that this books enables the recognition of both, the ability to effect a strategy to determine the one, right enterprise approach, and then the ability to accomplish this enterprise strategy in an incremental approach rather than an all or nothing approach.

This book not only addresses these two error classes including architecture and strategies to deal with these two error classes, it is also highly engineered. Chapter 1 introduces it all; Chapter 16 summarizes it all. Every

other chapter has its own introduction, unique material, summary and next chapter introduction.

Chapters 2 through 8 provide the overall rationale, requirements and architecture for a comprehensive data semantics management solution. Chapters, 6, 7, and 8 within this set enable the construction of very key data semantics management components. Chapters 9 through 15 provide detailed strategies, examples, and screen shots of how to deploy data semantics management on a project, organization, function, and enterprise basis.

The book contains a number of “can do” chapters. These chapters, 7 through 14 contain an introduction, significant architecture and engineering material, and a detailed set of Metabase System based examples that illustrate the implementation of that data semantics management component.

Every chapter contains questions and exercises that directly bear on the accomplishment of data semantics management.

Because of the chapters’ engineering, some material that is essentially the same. That is not an accident. Rather, it is purposeful reinforcement. Additionally, this repetition enables many of the chapters to stand alone.

### **1.4.1 Chapter 2, Why Semantics**

Chapter 2, Why Semantics, addresses the fundamental need for highly engineered semantics as a way to control the error classes cited above. This need is first explained within the context for data management. Key to understanding data management is understanding the role of enterprise-level policies and procedures. Because these need to be enterprise-wide, there must be consistency in their definitions at all levels and across all enterprise organizations.

Another section of Chapter 2 identifies the need for database object classes. The specifications of database object classes are set squarely within corporate policy specifications. Data, the by-product of policy executions, are the “what” that should be shared within and across corporate organizations. Database object classes are the coherent specifications of shared data. Unique to database object classes are the definitions of the states and state transformations through which a given object proceeds from birth through all life cycle stages until the database object becomes fully mature.

A key construct in Chapter 2 is the graphical representation of the various layers that start with policy specification and proceeds down through data-based evidences of policy executions.

Chapter 2 continues with a simple definition of data semantics and illustrates this definition through several examples. Key to the explanation is the simple fact that there is no silver bullet that addresses all the needs of data interoperability across the enterprise. Because there is no silver bullet, the management of data semantics must proceed through several different, but thoroughly interconnected models of metadata. These models address: data elements, data structure templates, the employment of these data structure templates within databases, and finally as specifications of semantically complete data exchanges. Each of these metadata models is generally defined in terms of their critical components.

Chapter 2 concludes with a section on the return on investment (ROI) from effective data semantics management. The basis for the ROI is in two parts: Data and business information systems. While the ROI for data is significant, the largest ROI is on the development, evolution, and maintenance of business information systems. Simply put, the business case for data semantics management is in four parts: increased productivity, increased quality, reduced costs, and reduced risk. Not accomplishing data semantics management is not only a bad business decision in the short run, it likely is a terminal business decision in the long run.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 2, *Why Semantics*, the first overall framework for addressing these two error classes. The ultimate strategy is to design both these error classes out of existence from the very start. If there are no initial errors, there is no need for follow-up activities. To that end, first there are the Semantic Hierarchies that support the creation, meaning, and allocation of all the semantic-based words that make up critical elements of fact names.

Thereafter there is the Data Element Model, the Specified Data Model, and finally, the two database data models, that is, Implemented Data Model and Operational Data Model. Each of these provides the ability to define the data semantics that control their immediate lower contained levels. Each lower contained level provides the ability to provide a more refined set of semantic definitions and/or restrictions. The parent level of any given semantic definition, for example, concepts within which data element concepts are defined, provide the ability to posit semantic affinities due to common parentage.

If all this is accomplished top-down, there will be a great reduction in semantic errors. The problem will still exist with bottom-up development. That, however, is where most of reality exists. There is no choice then but to

have top-down connections to bottom-up developed or imported data models. As this is done, compromises is inevitable. In such cases there at least exists evidence of the mapping decisions. Over time as more and more one-off databases and business information systems are replaced with enterprise-based semantic-driven databases and business information systems, the quantity of semantic errors will be driven out of existence.

## 1.4.2 Chapter 3, Data Semantics Failures

Chapter 3. Data Semantics Failures, is a very critical chapter. That is because there has been spent 100s of millions of dollars in the attempt to achieve enterprise-wide data semantics, and in virtually all the cases, the result has been failure. A common reaction to these failures, however, has not been to understand the causes and try again. Rather it has been to abandon the efforts because it seems to be too hard or takes too long. The problem with that solution is that the next time data semantics is tried, the problem area has grown and the existing ad hoc, stove pipe solutions have become more entrenched. The right time is now. Procrastination is never the correct solution because waiting only makes it harder and more costly.

Another reaction to failure is to jump on the next “silver bullet” that whistles through. Silver bullets are nonexistent in this area, so again, waiting only makes the problem space bigger and more costly. Therefore, the time is now, not the tomorrow that never seems to come.

Chapter 3 goes to significant length to thoroughly diagnose the common data semantics failures, including some that will surprise you. The lessons learned are clearly understood and have been completely incorporated in the strategies and examples of this book. Everything that needs to be done to engineer and deploy interoperable data environments is set out in this book and in other Whitemarsh books. Additionally, every example from this book is able to be operationally demonstrated via the Metabase System.

A key ingredient to the accomplishment of enterprise-wide data semantics management is the establishment of Communities of Interest. The Whitemarsh book, *Data Interoperability Community of Interest Handbook*, sets out the architecture of these groups and the work plans, deliverables, and all necessary policies and procedures for the establishment, operation, and evolution of these groups.

Excuses of not knowing what causes failures, and once known, not knowing what to do have been removed. The time is now, not the tomorrow that never seems to come.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 3, *Data Semantics Failures*, clearly outlines the most common sources of data semantics management failures that ultimately cause instances of the two error classes. These failures must be carefully studied and solutions to these failures must be set into place. This book's dictums, strategies, and action plans represent a solution set.

### **1.4.3 Chapter 4, Engineering Data Semantics**

Chapter 4, *Engineering Data Semantics*, is another key preparatory chapter. Data semantics management efforts cannot be slapped together and set into motion. The right environment must exist. Ironically, even if all the factors were completely known 20 to 30 years ago, enterprise-wide interoperable data environments were not possible due to immature technologies. If, however, the dictums and strategies of this book and those from the *Data Interoperability Community of Interest Handbook* were executed, there would have been integrated data across all the stove pipes of database and business information systems. Had this been done in advance of technology being available, it would have all fallen into place.

It was no accident that one division of a world-wide corporation had a Y2K bill of virtually zero while another division of the same corporation spent many millions repairing and redeploying their ad hoc databases and business information system implementations.

A key set of material in Chapter 4 is the exposition of the evolution of bindings that existed between data, programs, database management systems, and users. There has been a long and gradual trek in the evolution of bindings from the 1950s through to the present day that has enabled, for example, service oriented architectures. It is critical to know, however, that success in service oriented architectures require careful attention and deployment of the fundamentals of this book. Disregarding the dictums, strategies, and products set out in this book is to court failure.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 4, *Engineering Data*

Semantics, sets out the overall domain within which the solution must reside. Again, the ultimate goal is to define both these error classes out of existence.

The very first line of defense is of course the operational databases and information systems that are represented by the Operational Data Models of Chapter 13. Those if defined in isolation form islands or stove pipes of semantic consistency. While these are valuable, what is even more valuable are broader areas across whole business functions and organizations. These would be guided by the higher levels of semantic specifications and would exist within the Implemented Data Models of Chapter 11. Backing those up would be the Specified Data Models of Chapter 10 and finally the Data Element Models set out in Chapter 9. If these models are implemented, not only will individual databases and business information systems be more semantically acceptable, but so will all other related databases and business information systems because they are drawn from common higher semantic levels.

#### **1.4.4 Chapter 5, Semantics of Names**

Chapter 5, Semantics of Names, squarely addresses one of the key reasons for data semantics management failure: Names. An example, the name string, Social Security Number, is all that's needed to understand the problem. According to the traditional naming strategy, Social is a modifier, Number is a class word, and therefore Security is the "real part." If the complete semantics are not instantly obvious, the point's been made. Further Number, in this case is a lousy choice for a class word. What's the meaning of an average Social Security Number. The minimum or maximum. What are those dashes, subtraction signs? Finally, Social Security Numbers are often used as identifiers. Thus, the real class word should be Identifier. That would be OK except a Social Security Number is really a set of three codes. Now, onto Social. If there are so many law firms advertising their services to "fight" the Social Security Administration, they must not be very "Social." And as far as security is concerned, the Social Security Trust funds (which really do not even exist) rushing headlong into insolvency. So much for Security.

Similar examples can be made for Reservation Numbers that seldom contain only numbers or even numbers at all. Telephone numbers are another example. Are they really the serial numbers of the telephone instruments? Of course not. The point to all this is that there is a general tendency to overload the semantics of names.

There also seems to be a unyielding demand for single definitions for given name strings. At one conference during the early 1990s, a speaker was questioned about the meaning for a given name string. The questioner indicated that a different definition had been used for the previous 2000 years. The speaker stated that they would just have to change. Oh yeah, right. The point to Chapter 5 is to set out a formal strategy and infrastructure for the construction of names that does not fall apart under even the slightest of scrutiny.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 5, *Semantics of Names*, is very important. That's because there's a commonly accepted but mistaken notion that names are the alpha and the omega of data semantics management. Far from it. Names, while clearly the most visible semantic component, can be very misleading and can give rise to both Type I and Type II errors within the first error class, and can also lead to delays and confusion as the second error class attempts to be sorted out. One U. S. DoD agency spends many millions every year just sorting out errors in logistics labels due to reference data errors. These are clearly a second error class activity caused by Type II errors such as sending M1 Abrams battle tanks to Bimini versus Bosnia.

This is not to say that names are unimportant. What is critical is that there is almost automatic strategy for name construction that ensures that when names are the same they have the same meaning, and vice versa. If and when this is done, there will be a great reduction in both error classes.

### **1.4.5 Chapter 6, Semantic Hierarchies**

Chapter 6, *Semantic Hierarchies*, is a can-do implementation chapter. That is, it defines and illustrates the actual establishment of data semantics management. This chapter identifies and provides real screen-shot examples of how to define data semantic hierarchies and how to deploy them in real-world examples.

The real world examples come from the Metabase System and examples from the Metabase System metadata database, *Movies*.

The only restriction on the use of this downloadable "Free" Metabase System is that it only allows one metadata database beyond the example metadata database, *Movies*, and only allows one concurrent user. The Metabase System, unencumbered from the "Free version" use restrictions, is

an enterprise class system capable of supporting a very large enterprise system of integrated database information systems for interoperable data sharing, and is available for purchase at a very modest price compared to other popular enterprise class metadata management system systems

Semantic hierarchies are a critical element in the development of enterprise-wide semantics because they form the building blocks for name construction. What is very important about these semantic hierarchies is that they can be allocated to a number of different components in the data semantics management space including: data element concepts, data elements, attributes, and columns. Not only can these be allocated, each semantic name string can be defined singly or in context with its hierarchy.

Further, the Metabase System software ensures that semantic nesting is enforced. Users are not able to allocate to a lower level data component, for example, a column of a table, a semantic that is the same as the one already assigned to the attribute of an entity. Nor are users able to assign to the column a semantic that is at a higher level of generalization as was assigned to the attribute of an entity. Attributes are semantically superior to columns. Thus, assigning the same or a lower level semantic to an attribute than is assigned to a column would simply be a mistake. The Metabase System prevents such mistakes.

Finally, not only are the assigned semantics employed to “automagically” (not really by magic of course, the strategy is set out in Chapter 7) construct names, but they are also employed to “automagically” define data element concepts, data elements, attributes, and columns. Both these capabilities are critical to the efficient and effective management of enterprise-wide data semantics.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 6, *Semantic Hierarchies*, is a very critical component. That is because the words that comprise fact names within an interoperable data environment can be constructed through the allocations of specific strings from these hierarchies.

Simply put, words, within their context have single meanings. When definitions are automagically constructed they will always be generated the same way. Hence, they will be reliable and repeatable regardless of the person or organization, or function within which they are generated. What is the same is the meant to be the same, and vice versa. This strategy, if followed will greatly reduce the first error quantity and it naturally follows that the second error class is reduced as well. All these same words and same meanings are carried down through the entire set of Metabase System data

models, that is, Data Element, Specified, Implemented, Operational, and View.

A very important additional benefit from these semantic hierarchies is the ability of the Metabase System to do automatic name construction and automatic definitions of all business facts from data element concepts within the Data Element Model down through to DBMS columns within the Operational Data Model. This enables all uses of databases constructed through these strategies to know the enterprise-wide and consistent definitions of all facts. Clearly this addresses both classes of errors.

#### **1.4.6 Chapter 7, Value Domain Management**

Chapter 7, Value Domain Management, is a “can do” chapter. Increasingly, value domains are important to data semantics management. The Type II error cited above, that is, the ability to specify Gender = 1 as meaning Female in one environment while it means Male in another environment can be a critical source of errors that must be prevented. The first step in prevention is having the ability to know about these possibilities, and being able to control and manage value domain value collections. The strategies set out in this Chapter enable this to take place.

Demanding that there are completely congruent value domain collections is just not possible, however. Nice, yes. Recommended, absolutely. But absolutely possible? No. This chapter sets out strategies for setting down value domains, associated value collections, and the ability to map one value collection with another. There is, however, no silver bullet here. If one collection has three values for a concept and another five values for different instances of that concept, there can only be three mappings, and none of these might be exact. This is illustrated in the case of mapping “one-third’s based values” to a set of “one-fifths-based values,” or academic “letter” grades to precise “numeric” grades. Miracles are not possible. Some value domain mappings are going to be unsatisfactory whenever square pegs are shoved into round holes. At least this chapter provides a strategy to define the square to round mappings, and a Metabase System solution to know when and where the mappings are occurring.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 7, value domain Management, is also a very important chapter because it provides a very

explicit first line of defense in stopping invalid values from entering any database.

This kind of error is the root cause of the Type I and Type II example in Section 1.4. While solving the “word” and name problems are important, so too are the problems associated with restricted value domains. When this is done, a great deal of the hidden value-based error class errors will go away. The key problem with value domain errors is that they can go undetected for a long time. They must be both prevented at the outset and ruthlessly sought out on a continual basis. All is lost if the data cannot be trusted.

Value domains and their accompanying values can take on both type and value forms. For example, in a data warehouse database, the column Region might take on the value, New England. While in another database, the column name might be New England Region. It all depends on the data modeling strategy. In either case, the value-domains component of the Data Element Model has the values and their definitions from which either the names are constructed or the values that would become part of reference data are produced. Thus, there would be a good effort to prevent the first class of errors and to therefore prevent the second error class all together.

### **1.4.7 Chapter 8, Fact Specification Cases**

Chapter 8, Fact Specification Cases, is a chapter focused on understanding all the different cases associated with fact specification. This is also a “can do” chapter from the point of view of knowing the meta model specifications of data integrity rules, how these rules are bound into the Metabase System, and finally, how and where business facts are implemented across the Metabase System.

Some of the fact specification cases are simple, for example, single values for fields like Birth Date. Other cases require groups of fact collections such as a Full Name (First Name, Middle Name, and Last Name). Still others required facts like Life Span which would be from Birth to Death. Many of these fact cases result in virtual values such as a person’s age ((Today - Birthdate)/365.25). Still others require sophisticated processes. Examples of all these data integrity rules that govern the fact cases are set out.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 8, Fact Specification Cases, enables the specification of the rules under which facts are know singly or in combinations. If facts exist in combinations, then, given that DBMSs are

employed as the “protectors” of data, processes can be created and embedded in the DBMS schema language that enforce many different classes of fact based rules.

SQL:1986 through SQL:1992 were simple. Each column represented a single atomic value. SQL:1999 and more recent SQL versions started to allow more sophisticated data structures. Returned were the “good old days.” Some, however say those days were not all good. For example, in SQL:1999 an array has persistent cell value ordering. Thus, there might be semantics inferred by being the “first” or the “last” value in an array. If so, that semantic has to be defined, stored, and retrievable somewhere. Each of the fact cases, if properly specified and enforced can reduce error quantities. For example, by never allowing a U.S. Address to be entered without a valid combination of City-State-Zip.

Fact specification classes beyond the simple, single value facts, are best specified in a Business Rules component of the Metabase System. Currently (as of this book’s publication), these rules are set out within the database object classes model that is the subject of Chapter 12. Properly engineered and embedded in sophisticated DBMSs, both classes of errors can be comprehensively addressed.

### **1.4.8 Chapter 9, Data Element Model**

Chapter 9, Data Element Model, is a “can do” chapter. Data elements are not facts that are “valued.” Rather, data elements are the specifications of fact-based semantics that are, in turn, employed to infuse these semantics into the attributes of entities, and columns of database tables. Data elements are also not simple. Data elements can be set within derived and compound data elements. They are set within data element concepts value domains. Finally, data element concepts are set within concepts and conceptual value domains. This is quite an extensive context for each data element.

Semantics, as represented by the Semantic Hierarchies set out in Chapter 7 can be allocated to data element concepts and data elements. When they are allocated to one level of abstraction, the semantics are presumed inherited at all lower levels. If the allocation to a data element concept is sufficient for a data element, the data element’s semantics are “automagically” bound by those allocated to the data element concept. If however a data element’s semantics are to be further restricted, allocation can additionally be to the data element. In this case the only allocation allowed must be a

semantic subset. Prevented as well are same-semantics allocation or recursive semantics allocation. Finally, if semantics are already allocated to attributes from within entities of a Specified Data Model, or to columns of tables of an Implemented Data Model, the same rules apply. In this case, only super-sets would be allowed to be allocated.

Finally, value domains can also be allocated to data elements. These too must be super sets if there are value domains already allocated to attributes and/or columns.

The Data Element Model set out by this book is a valid and correct implementation of the ISO Standard, 11179 for data element metadata and registries. The data element module of the Metabase System is a data element registry. While having an ISO 11179 standard's conforming registry is important, it is only really valuable when it is interconnected and integrated with the ability to employ those data element semantics within data structure templates (Chapter 9), or database data models (Chapter 10 and 11). The ISO standard, 11179 is a conceptual specification. That's unfortunate because this is not a "rubber meets the road" standard. Only real and valid implementations of ISO 11179 are "rubber meets the road" uses. Finally, Chapter 9 sets out examples of valid and invalid implementations of ISO 11179.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 9, Data Element Model, is a very important component. Data elements provide the fundamental and overarching semantics for all the business facts that are ultimately captured within the Operational Data Models.

While this overarching data element architecture can be specialized via attributes and columns, all these lower level specifications are still just variants of the fundamentally defined data element.

If data elements are seen as a define-once use many-times component, the work will be able to be completed and will be of real and lasting value.

This one-to-many approach enables a greater ability to deal with both classes of errors starting with first class because a DBMS column from an Operational Data Model can be researched to find its column, attribute, or data element parent. Similarly, a different DBMS column from a different Operational Data Model can also be researched. The sooner these two DBMS columns have a common parent the greater the ability to say that the two DBMS columns fundamentally represent the same data. Affinity lessens the need for the second class of error research.

### **1.4.9 Chapter 10, Specified Data Models**

Chapter 10, Specified Data Models, is a “can do” chapter. Just having standardized data elements is not sufficient to have data semantics management. Included in the management is also granularity, precision, and temporal aspects. By having standardized data structure templates, enterprises can specify the granularity, precision and temporal aspects of the data that will ultimately be specified within the various database data models and that finally represents standard data that is exchanged either through SQL view-based exchanges or, for example, XML-based exchanges.

The data models set out in Chapter 10 are data models of concepts. A data model of a concept is not a conceptual data model. The former is a fully defined data model, and the later represents a "fuzzy" form of a data model that, through data model "baking" becomes more well-formed, that is, a logical data model, and thereafter, through more "baking," becomes a physical data model. This is a critical difference. In Whitemarsh, Specified Data Models are data models of concepts. This is a difference with a real distinction.

There can be relationships among the various data structures participating in these subject-based data models. Each subject-based data model consists of entities, attributes, and relationships. The relationships are expressed as primary and foreign keys. Candidate primary keys can also be defined. Because every attribute is related to a “parent” data element there is a walk-back to enterprise semantics across the entire set of concept data models.

Relationships can cross subjects just as it does in real situations. By having data structure templates, there can be standardized uses of these structures. For example, if a structure exists for Address, wherever an address is employed throughout all the different database data models, they can be mapped back to the template. This provides the ability to know that data really is the same even when named differently, and really is different when named the same. This is very critical when assessing whether different database models have the same granularity and temporal aspects.

For example, one database data structure might have a real-time characteristic for Sales data while another has a daily close-of-business characteristic. In both cases the majority of the database table columns would be named the same and might exist within similarly named tables. These data cannot really be shared because in one case, the database records are real-time while in the other case, the collection of records is daily close-of-business based. If sharing must occur notwithstanding the significant differences in

granularity and temporal aspects, there would have to be some level of summarization and transformation of sets of the real-time data records into a “week-COB” record.

Semantics and value domains can be allocated to attributes but only if they are subsets of those allocated to data element concepts or data elements. The Metabase System, as it should, ensures correct allocation.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 10, Specified Data Models, provides two major assists in reducing the quantity of errors. The first is the standardization of data structure templates that will dramatically accelerate the creation of database data models. The second is the bringing forward a great deal of standard semantics from the data element to the attribute, and then from the attribute through to the DBMS column. All this work, done in advance, will dramatically reduce semantic error quantities.

This will greatly improve the ability to better know whether a given DBMS column, as a semantic implementation of an attribute has the same precision and granularity as does another DBMS column as it is related possibly to the same attribute.

As with this same analogy above, the sooner two DBMS columns share a common parent, the greater the ability to say that the two DBMS columns fundamentally represent the same data. Affinity lessens the need for the second class of error research.

#### **1.4.10 Chapter 11, Implemented Data Models**

Chapter 11, Implemented Data Models, is a database data model “can do” chapter. Implemented Data Models are very different from Specified Data Models because Implemented Data Models are schema-based. Thus, relationships are not allowed from one table to the next between different schemas. Implemented Data Models through down-stream Operational Data Models are generally designed to serve a restricted set of business information systems. There are generally five distinct data architecture classes:

- Original Data Capture.
- Transaction Data Staging Area.
- Operational Data Stores.

- Data Warehouses (wholesale and retail (sometimes called data marts)).
- Reference Data.

Each of these data architecture classes have certain characteristics, styles of design, and specific audiences and users. A recent popular class of database, master data, is a class of database that is to represent enterprise-wide definitive value sets for certain data structures such as customers, products, and the like. Thus, master data can be considered to be reference data given an expanded definition of the word, reference. The data architecture classes are described in Appendix 1, Data Architecture Classes.

Each schema-based data model consists of tables, columns, and relationships. The relationships are expressed as primary and foreign keys. Candidate primary keys can also be defined. Because every column is related to a “parent” attribute, there is a walk-back to enterprise-based data models of concepts. If these models do not exist, columns can be mapped to data elements. Again, this ensures there is a walk-back to enterprise semantics across the entire set of schema-based data models.

There can be multiple data structure templates represented in a single table as would be the case of person-data and address-data. There could also be multiple sets of person-data as would be needed if there were multiple person-roles contained in one table. Of course, there can be multiple uses of data structure templates in different tables of the same schema and within different schemas. Collectively, all this define-once use many-times enables a comprehensive mapping from enterprise data model templates of concepts to various aspects of database schemas across all the different data architecture classes.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 11, Implemented Data Models, is a great leap forward. Every column in every table can be related to enterprise-wide data elements, and the granularity and precision of the table can be related to the enterprise’s standard for granularity and precision that is set out in the Specified Data Models of concepts. These standardized walk-backs to enterprise semantics greatly reduce the first class of error because of standardized semantic mapping. With that class of error reduced, the second class of follow-up semantic synchronization activities is also greatly reduced.

### **1.4.11 Chapter 12, Database Object Classes**

Chapter 12, Database Object Classes, is a “can do” chapter, and is a key chapter with respect to data semantics management. When database first began in the late 1950s and when DBMSs started to appear in the early 1960s, the concept of a “database record” was more comprehensive than what would be considered appropriate for a database table. The database record of “yesteryear” had complex structures, and some DBMSs had embedded processes. This was because a database record was to represent a complete and coherent definition of a corporate policy. For example, all the data structures associated with a person, or a contract, customer, company, invoice, shipment, or order.

With the advent of the relational model, the concept of a table as a collection of single-valued columns took on prominence. Tables, in the relational model are very simple. Properly constructed, they consist of single valued columns with only one column type in each table. For example, if there are four telephone numbers, there should not be TelephoneNumber-1, TelephoneNumber-2, TelephoneNumber-3, and TelephoneNumber-4. There should, instead be a separate Telephone Number table. This type of data modeling is classified as Third Normal Form (3NF) and is written about extensively. The problem with 3NF tables is they break up single policy-based database records into multiple tables. Thus, there is data model fracturing. Generally, this is not an ideal situation.

The SQL data model, contained in the ISO/ANSI standard SQL for 1986, 1989, and 1992 all conformed to 3NF tables. Starting in 1990 and continuing until 1997 market pressures were brought to bear to restore database records back into DBMS standards.

During this time frame, this author created a concept called database objects. Every database object is represented by its definition, i.e., its class, and by its instances, that is, the database objects themselves. Created is therefor the complete definitions for constructs like person, contract, customer, company, invoice, shipment, or order.

Every database object class consists of four parts: database object data structure, database object table processes, database object states, and database object information systems. The database object data structure can consist of nested structures, which in relational data model terms would be nested tables. The database object table process is that set of constraints and processing rules to ensure absolute integrity of a database object table row’s insert, modify, or deletion.

Database object states represent the value-based states across a complete database object data structure that conforms to a corporate policy. For example, there might be Employee Requisition, Employee Candidate, Employee New Hire, Employee Assigned, and the like. Each name represents a state that is unambiguously achieved in terms of values across the database object class's data structure.

A database object state is achieved through the successful execution of database object information system. Each of these database object information systems, in turn, execute insert, delete, or modify database object table processes.

The ISO/ANSI standard for SQL incorporated much of the concepts set out in database object classes in the SQL:1999 standard. It was however accomplished in a unique way. Instead of creating a whole new schema object called Object, the committee decided to expand the data structures contained in tables. Column data types were freed from just being single value types such as integers. Added were arrays, groups (e.g., Address that contains Street, City, State, etc.), and nested groups such as Company contain Employees which contain Benefits, Assignments, and the like. Processes were able to be established within the confines of these newly created SQL:1999 constructs. Essentially, after an absence of about 25 years, the database record rose from the dead like the proverbial phoenix from Phoenicians Legends and became part of the SQL:1999 standard. It has since been expanded in SQL:2003. No significant extensions are part of SQL:2008.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 12, Database Object Classes, serve two key roles. First, they become coherent objects of shared data. That is, a whole customer, order, shipment, invoice, and the like. Contained in each is a precise specification of integrity, state, precision, and granularity. Second, every database object table contains columns from traditional 3NF tables, which in turn are mapped onto attributes and data elements. There is, therefore, enterprise-wide integrity across the database object data structure. Database object classes represent wholly contained structures within Implemented Data Model databases. The effect on the two classes of errors is that if data interoperability is based on these database object structures there is an automatic acceptance of precision, granularity and temporal aspects.

If the SQL DBMS does not support SQL:1999 or later data structures, the database object classes and all their components provide a template for constructing the Implemented Data Model design within the Implemented

Database. Faithfully implemented and followed either through embedded SQL DBMS processes or through outer rings of data integrity subsystems that all business information systems must proceed through, both classes of errors will be greatly minimized.

### **1.4.12 Chapter 13, Operational Data Models**

Chapter 13, Operational Data Models, a “can do” chapter, serves to represent the database models that are closest to the reality of actual data. Each Operational Data Model schema, implemented via the DBMS’ SQL data modeling language represents the actual structures, integrity, data types, rules and processes through which data is allowed into databases, is queried from databases, and is maintained within databases. Every Operational Data Model schema cannot be semantically more rich than what is allowed by the SQL DBMS through which it is defined. It is, therefore, not unheard of nor unacceptable to have different SQL DBMSs of different capabilities implementing different databases. Neither shoes nor DBMSs come in one size to fit all.

The data structures contained in Operational Data Models serve two masters. The first is the Implemented Data Model from which they are derived. There may be multiple Implemented Data Model “parents” as would be the case for a Data Warehouse database, or an even more refined data mart database. The other master would be the one or more business information systems that are the main users of the database.

If legacy business information systems and databases already exist, it may not be practical or politically possible to make massive changes to an existing database. In this situation, the only practical solution is to provide mappings to the one or more Implemented Data Model databases in terms of columns, data types, and value domains to the maximum extent possible. Then, over time, slowly transform the Operational Data Model database from stove-pipe semantics to one of enterprise-semantics.

A very practical approach in accomplishing Operational Data Model transformations is to transform the business information systems from stove-pipe engineering to enterprise-process engineering. Transforming these legacy systems through business information system generators is often much cheaper and faster than trying to retrofit an existing legacy, stove-pipe engineered business information system.

Enterprises have been procuring Enterprise Resource Planning (ERP) software packages in the hope that they will gain enterprise interoperability, industry best-practices, and enterprise-wide standardized semantics. Seldom, however has this been true. Different packages from even the same vendor can have conflicting semantics.

Often the vendors of these packages just view DBMS and database as operating system access methods, which has the unfortunate result of embedding all data and process semantics into the application layer. Even though that kind of embedding was discredited more than 20 years ago, it seems to live on in ERP packages.

The effect of this is that there are many tens of thousands of database tables, no viable database schema, very expensive maintenance, an almost impossible task of efficient evolution, and finally, an almost impossible ability to employ ERP data in everyday business class report writers such as Crystal Reports.

One possible solution to all this self-inflicted chaos is to create a set of shadow databases that read and write the ERP databases. These shadow databases conform to enterprise semantics and common-sense-based data structures. Interfaces are built between the shadow databases and the ERP databases such that any data going to or coming from the ERP package goes through the shadow database. The practical effect of this strategy is that the ERP package is fire-walled off from the rest of enterprise applications such that the shadow databases are able to be included in enterprise-wide databases just as if the ERP package's databases did not even exist. The alternative to this is to attempt to integrate the often semantically discordant and conflicting semantics from many different ERP packages into a set of enterprise-wide semantics. This effort can range from very difficult to totally impossible.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 13, Operational Data Models, the effort to prevent these errors with respect to enterprise-wide semantics can be very difficult if the role of these databases is to serve the needs of ERP packages. In this case, all the rules that would prevent these errors are embedded inside the ERP software package itself. So either the prevention rules are there, or the package has to be modified to execute them.

If the Data Element, Specified, and Implemented Data Models are constructed from an enterprise-wide mission perspective, at least when the Operational Data Models are imported, the semantic mismatches will be obvious. Compromises will have to be made to accommodate these

Operational Data Models. Likely while databases and business information systems will have to be built to transform nonstandard data to what is required by the enterprise. While the way ahead will not be easy, at least it can be known.

Otherwise, if the Operational Databases are built in a top-down fashion from data elements through Specified Data Models and Implemented Data Models including the assignment of layers of value domains, the probability of preventing these error classes is quite high.

### **1.4.13 Chapter 14, Interface Data Models**

Chapter 14, Interface Data Models, a “can do” chapter, is focused on the interfacing of database-based data to business information systems. There are many levels of sophistication that can exist in interfaces starting with just simple one-to-one relationships between database tables and the business information systems, to whole interface specification processes that cause DBMS selection, combination, and calculations prior to the business information system receiving the database data. In either of these two cases, there is a tight binding between the DBMS and the business information system. Each must know of the other, and protocols must exist to ensure that they work well together.

Service Oriented Architectures, are entirely different. The business information system does not know when, where, or how the DBMS will get the data-based transaction. Similarly, the DBMS does not know when, where, or how the business information system will get the database data. How are the interconnections made? Magic? For sure not. There must exist what would effectively be a Cellular Phone Network.

The key to making such an environment work is highly engineered transactions wherein every transaction at least knows the ultimate comes-from and goes-to address for the message that is being transmitted. Today’s Internet is essentially an SOA environment because every transaction is sent with sufficient “goes-to” information to get to the necessary location, be processed, construct a response, and sent that response back with sufficient return “goes-to” information for the waiting business information system to process the response.

Since virtually all SOA transactions are sent in just plain ASCII, using XML-based transactions is a clear and obvious choice. These transactions can

be greatly compressed with existing technologies such as those found in WinZip or other software applications.

Key to the SOA environment are well-known semantics wrapped up in highly engineered transactions. In short, nothing's really changed except that the transaction, instead of being hardwired from sending receiving location must be enhanced to have that information inside it. If the transaction arrives and is misunderstood or not understood at all, the only possible response back is "Huh?"

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 14, Interface Data Models, the ability to catch and/or prevent errors depends greatly on whether the Operational Data Model is interfacing with business information systems perceive that database and the DBMS to be merely as a set of files and a file access method, or as a part of the semantic fabric of the enterprise. If the former, the options range from limited to none.

The Interface Data Models enable the forward deployment of enterprise-wide semantics despite Operational Data Models that might not conform. It's not that the Operational Data Models are avoided, it's that views can be constructed to some extent to transform the Operational Data Model's granularity, precision, names, data types, and value domains to be that of the enterprise.

Most ERP packages, however, have installation steps that require the specification of all the legal values, so catching errors is entirely under the control of the ERP package. The advice, therefore, is to evaluate these packages very carefully. What can be of great value is to have accomplished all the enterprise semantic steps of creating the Data Element Models, Specified Data Models, and Implemented Data Models and use these as requirements specifications for evaluation the ability of proposed software packages to trap and/or prevent errors.

If, however, the business information systems enable the creation and effective use of SQL Views, many of the first class of error can be programmed directly into views. Incorporated can be valid value selector lists, column renames, value transformations, and the ability to create new values from values that have been provided in the columns. When these are comprehensively addressed, the quantity of the second class of errors drops dramatically.

### **1.4.14 Chapter 15, Work Plans**

Chapter 15, Work Plans, consists of work breakdown structures specially designed to fit the data models that are set out in this book. That is, those for Data Element Models, Specified Data Models, Implemented Data Models, Operational Data Models, and View Data Models. Supplementing these data model structuring work plans are specialized work plans for discovering value domains, reference data, and performing overall enterprise-wide reference and master data development.

With respect to the two error classes, initial semantic disconnects, and follow-up semantic synchronization activities, Chapter 15, Work Plans, form a key component that ensures that database and business information systems projects are on the right path and accomplish the right steps in the right sequence. The work plans focus on discovering the right metadata and placing that metadata into the Metabase System so that it can be brought to bear and at the right time.

### **1.4.15 Chapter 16, Summary**

Chapter 16, Summary provides a summary section for every chapter. It also provides a set of sections that addresses how every chapter addresses the two error classes that are described in every chapter. Finally, this chapter contains a set of concluding statements and a challenge to the reader.

## **1.5 Rationale Summary**

This chapter set out the rationale for data semantics management. The rationale is presented by first citing the clear state of the world, that is, that we live in a world economy. With the Internet, there are no borders.

Commerce and communications of virtually all abstract goods are exchanged at the speed of the Internet. A good example is all the Internet-based jokes that travel around from person to person and organization to organization almost instantaneously. Commerce too is similarly engineered. The computer on which this manuscript was created was built in Asia. The operating system is from Microsoft in Washington, the word processor from Canada, and the Metabase System that is the demonstration vehicle for all this

book evidenced is from a company in Florida with supplementary products from Arizona, Australia, Germany, South Africa, and England.

Simply stated, the most common languages throughout the world are not English, French, or Spanish. Rather, they are SQL, C, C++, because all communication throughout the world is first founded from these universally accepted and unambiguously understood computer-based languages.

This chapter presented a clear statement of the problem that must be solved and several examples to illustrate it. Thereafter were examples of the effects of not having sufficiently engineered data interoperability.

This chapter set out the remaining chapters in the book and conveyed how each chapter contributes to addressing the two classes of problems that characterize the achievement of data interoperability through data semantics management.

This chapter, as does all chapters, concludes with this summary and a set of questions and exercises that should be able to be accomplished by practitioners of the book.

This book is not just to be read and thought about. Rather, it is to be acted upon. The former will provide intriguing thoughts and concepts to consider. The later will provide real action plans coupled with a downloadable production class metadata management system, the Metabase System, to manage data semantics in support of creating the interoperable data environments critical to the world economy.

## **1.6 Questions and Exercises**

1. In your review of the Chapter Sections (1.4.1 through 1.4.15) are there any missing chapters that are important to data semantics management? Which ones? How will these missing chapters affect your data semantics?
2. What are examples in your organization of dysfunctional semantics? How would you quantify the cost and lost opportunity? What factors constitute cost? How much does each factor cost? What would be these costs be determined at a project level? At an organizational level? At the enterprise level? How would you present this to management so that your efforts are funded and supported?

3. How different would these costs be determined at each different level? Are there additional costs as you proceed through the different levels? Why do these additional costs exist? How can you place a value on them? What are the benefits from incurring these extra costs? How can you present these additional costs to management so they can see the benefit from proceeding from a stove pipe semantics management environment to an enterprise semantics management environment?
4. What are examples in your organization of good semantics management? How would you quantify the benefits and gained opportunities? How much did each factor save? Extended to a complete functional area and the enterprise, how much would good semantics management have saved? How would you present this to management so that your efforts are funded and supported?
5. Why should there be semantics management? What's the business case? How can you make the business case in your organization? What are the political and cultural inhibitors to data semantics management? What strategies can be put into place to overcome these inhibitors? How would you present this to management?
6. After a review of the semantics effort failures, have any occurred at your organization? What was the consequence? Were the failures swept under the rug? How were the failures explained? What remediation was put into place? How can you make the case that all lessons are not free?
7. In regard to the semantics of names, how should names be engineered? Is it possible to have multiple levels of names from the most generalized to the most specialized? What is the consequence of forcing a single set of names onto all database columns in the enterprise? Discover examples from within your project, organization, function, and enterprise of badly constructed names and show what the costs have been from these bad constructions. How would you present this to management so that your efforts are funded and supported?
8. What is the value of semantic hierarchies? Have these been tried in our organization? How have they been constructed and used? Have these

semantic hierarchies been integrated into the processes that construct names? If so, how? What was the value? If not, why not? Discover from within your project, organization, function, and even the enterprise how there have been the same words with different meanings, and different words for the same meaning. Quantify the costs and the confusion. Determine how these might have happened and can now be prevented.

9. What is the role of value domains? Are they not just semantics with a different name? How can values from restricted value domains be used in name construction? In data warehouse dimension construction? Have value domains been standardized in your project, organization, function, and enterprise? What has been the rewards when this has happened? The costs when this has failed. What has been the effects on database design, program and business information system construction. Compute the costs at the project, organization, function, and enterprise level. How would you present this to management so that your efforts are funded and supported?
10. In an examination of the various fact specification cases, are there any that are missing? Which ones? How would you define them? How often do you find these fact specification cases in your enterprise? Have you considered identifying and managing collections of these cases? What would be the benefit from such management? What has been the costs from not managing collections of facts? How would you present this to management?
11. What is the role of the Data Element Model? Its value? Do you have such a model? If so, why? What has been the value? Has your Data Element model suffered any of the data semantics failure above? If so, what has been the consequence? How does your organization distinguish a data element? How many data elements versus columns does your enterprise have? Why? Compute the cost of column-based versus data element based semantics models. How would you present this to management so that your efforts are funded and supported?
12. What is the main role of the Specified Data Model? Has your Enterprise attempted to standardize data structures of concepts? Does it make sense and have value? What's the difference, and why is this

- important? What value could be gained by having standardized data models of concepts during the construction of databases?
13. What is the main role of Implemented Data Models? Is there value from having a DBMS independent version of a database model? What role does it play? How can and should it be related to Specified Data Models? What are the possible relationships between a Specified Data Model component and an Implemented Data Model component?
  14. What is the main role of Operational Data Models? How can and should these be related to Implemented Data Models? What are the possible relationships between an Implemented Data Model component and an Operational Data Model component?
  15. Compare and contrast the engineering, architecture and interrelationships among Specified, Implemented, and Operational data models versus the same characteristics of conceptual, logical and physical data models. Which alternative leads to integrated and nonredundant enterprise data semantics management and which leads to a large volume of semantic stove pipes? Compare and contrast the cost of each alternative. How would you present this to management so that your efforts are funded and supported?
  15. The Interface Data Models are defined as containing three components. SQL Views, XML schemas and data, and SOA. How are these the same? Different? Are they alternatives, one to the other or "three peas" in a pod? When do you need each? More than one? Is there value in having these defined in terms of database models that are supported by all the other upper levels of metadata? If so, why? If not, why not? How would you quantify the costs and the benefits?
  16. In a brief review of the work plan descriptions, are there any really important ones missing? What value is derived from standardizing work plans? How does the Metabase System of standard artifacts fit within these work plans?