

Whitemarsh
Information Systems Corporation

Data Management Program:
Metadata Architecture
for
Data Sharing

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

Table of Contents

Acknowledgments	vi
1.0 The Enterprise Environment	1
1.1 The Data Environment	1
1.2 The Data Sharing Challenge	2
1.3 Metadata	3
1.4 A Very Key Point	4
1.5 Remainder of this Paper	6
1.6 The Bottom Line	7
2.0 Metadata Environment Success Factors	8
2.1 Metadata Management	8
2.3 "8320.1" DDS Data Standardization	12
2.3.1 Flawed Data Standardization Model	13
2.3.2 No Accommodation for Enterprise Wide Data Architectures	14
2.3.3 No Accommodation for Multiple Implementation Technologies	14
2.3.4 Having a Centralized Data Standardization Authority	15
2.3.5 "8320.1" Data Standardization Summary	15
2.3.6 "8320.1" Lessons Learned	16
2.4 The "8320.2" Data Sharing Directive	18
2.5 Metadata Management Summary	20
3.0 Shared Data Architecture	21
3.1 Data Asset Products	25
3.2 Shared Metadata, The Key to net-centric Success	32
3.2.1 XML Schemas	32
3.2.1.1 IT System Centric Alternative	33
3.2.1.2 Data Centric Alternative	33
3.2.1.3 Responses Key Points	34
3.2.1.4 GAO Study Recommendations	35
3.2.2 Data Asset Catalogs	35
3.2.3 ISO 11179 Based Shared Data Elements	36
3.2.4 Shared Data Segments	47
3.2.6 Critical Concepts in this Shared Metadata Modeling Environment	55
3.2.7 Shared Metadata Environments	55
3.2.8 Federated Metadata Repository Environments	56
3.2.9 Quality Data Management in a Net-centric Environment	56
3.3 Name Management	57
3.4 Value Domain Management	62
3.5 XML Schema Management	65
3.6 Metadata Catalog Management	67
3.7 Shared Data Architecture Summary	68



4.0	Metadata Architecture for Data Sharing Summary and Way Ahead	69
4.1	Way-Ahead Actions	70
4.2	The Bottom Line Reprise	71



Figures

Figure 1. Metadata Repository Reference Model	8
Figure 2. Data instances for the levels and application pairs.	9
Figure 3. Technical, System, and Operation Architecture Views along with Data Asset Products	27
Figure 4. DoD Process of finding and creating DDDS “data elements.”	37
Figure 5. ISO 11179 essential data model for data elements.	39
Figure 6. Example of an ISO 11179 data element, Supply Item Resource Quantity	40
Figure 7. Example of an ISO 11179 data element, Person Grade Code	40
Figure 8. Comprehensive metadata model to manage data models	42
Figure 9. Logistics example from the five layers of metadata for data model management	42
Figure 10. Mapping of Shared Data Segments to Database Tables within Schemas	48
Figure 11. ISO 11179 Data Element Layer and Conceptual Data Model Layer	50
Figure 12. Characteristics of the Conceptual Data Model Layer.	51
Figure 13. Characteristics of the Logical Data Model Layer	52
Figure 14. Characteristics of the Physical Data Model Layer	53
Figure 15. Characteristics of the View Model Layer.	54
Figure 16. Metadata data model to manage business fact names.	59
Figure 17. Example of semantics associated with the attribute, Hourly Wage.	60
Figure 18. Name management in action.	62
Figure 19. Metadata model for value domain management.	62
Figure 20. Complex metadata model for value domain mappings.	64
Figure 21. Metadata model infrastructure to support automatic XLM schema construction. ..	66
Figure 22. Integration of a metadata registry and a metadata repository.	67



Tables

Table 1. Net-centric data goals, definition, and example.	6
Table 2. Net-centric goals along with data management support.	25
Table 3. Identification of Data Asset Products within DoDAF Architecture Views	30
Table 4. Interrelationship between Data Asset Products and DoDAF Views	31
Table 5. Meta attributes for context independent and dependent business facts.	58
Table 6. Semantics associated with the attribute, Hourly Wage.	61



Acknowledgments

This material is an evolution of documents that were updated during the time frame: September 2003 through December 2004. The primary contributors were Bruce Haberkamp, James Blalock, and Michael Gorman of the Office of the CIO, United States Army. The foundational components of this work has been favorably reviewed by subject matter experts within the U.S. Department of Defense.



1.0 The Enterprise Environment

Enterprises are transforming themselves into a fully digitized network-centric environments. The core of the net-centric environment is the data that enables effective decisions. By net-centric, it is meant that the network (including its infrastructure of hardware and software) is at the center of information exchange rather than one or more database or information systems. It is however well recognized that the information exchanged over the network is valuable if and only if the databases and information systems that are the network's data sources exist in a high quality, reliable data management environment. This transformation is motivated by the need to derive the maximum utility from all relevant data assets and to ensure information superiority throughout the full spectrum of valued information exchanges.

No one doubts the need for information superiority. Just "google" the phrase and see all the references from across industry and government. This superiority is only an advantage when contextualized data can become information, which then can be converted into superior knowledge leading to better decisions. During the conduct of collaborative operations, "interoperability" is a mandate, especially in terms of communications, common logistics items, and information sharing. Information systems and equipment that enable a relevant common operational picture must work from shared networks that can be accessed by any appropriately cleared participant.

1.1 The Data Environment

The challenge of an enterprise's strategic information policy is for all mission critical information systems to be interoperable. One aspect of interoperability is data interoperability. Data interoperability is the ability to reuse data from another information system without any intermediate transformation and human intervention.

Three examples from the U.S. Department of Defense illustrate the lack of data interoperability. In stark contrast to data interoperability as we defined it, these examples demonstrate the current way of doing business, which is point to point exchange of data via messages and translators. In this environment, every database configuration board considers the standards from their community of interest (COI) to be "the standard".

As a first example, one of these COIs uses a complex digital message protocol, the Collaborative Variable Message Format (JVMF). According to Edgar Dalrymple (Crosstalk, February 2002, p. 25):

The specification for the JVMF message protocol is the Technical Instruction Design Plan that is maintained by the Army's Communication and Electronics Command (CECOM). The specification is effectively maintained as a database that is known as the Variable Message Format (VMF) Integrated Database (VID). The VID defines the possible data fields and their associated parameters, structure, and the message cases and conditions. Cases and conditions are assertions



about the consistency of the fields in the messages, and the parser must implement them in order to encode and decode a valid message.

CECOM produces a new database release when either new messages have been added, or existing ones have been changed. The specification of the messages is nested up to six levels deep, and each level can have potentially thousands of data elements that may or may not be present. The current version of the VID has 121 messages, with millions of fields possibly present. The information that could be contained in the full message set if all fields were populated would cause the storage size for the messages to be in the terabytes range.

The Army Battle Command System (ABCS), a command and control system of systems, provides a second example of the lack of data interoperability. Without including Army systems, ABCS consists of over 80 Collaborative or other Service's systems. It has 226 external system interfaces (Army, Collaborative, other Services). In attempting to determine the cost of continuing to fund and maintain these interfaces, an Air Mobility Command FY95 Study reported that 80% of its software dollars went for interface maintenance (\$335,000/interface/year - 123 interfaces). One wonders how much the ABCS System of Systems is spending annually on funding and maintaining its interfaces, and one must definitely wonder whether there is not a better way to exchange data?

The planned modernization of DoD's financial operations is uncovering a third example of the lack of data interoperability. In an article (Federal Times, January 28, 2002, p. 22), Bridgette Blair reported:

As a first step, the Pentagon is taking inventory of its many current systems. Defense officials plan to have a modernization blueprint by February 2003. At last count, Defense staffers have found 674 financial and business systems and 1,489 interfaces. Those numbers could increase as the Department continues to map out its system infrastructure. The department will rely on the blueprint to decide what systems need replacing, consolidating, redesigning or upgrading.

From the above examples, it is clear that to begin to address the challenge of achieving data interoperability while migrating to a network-centric, fully distributed, synchronized, knowledge-based solution in line with "Army Transformation" and "Vision 2020", the Army must develop and implement a Net-centric data management Program.

1.2 The Data Sharing Challenge

This paper is all about achieving the data sharing vision through net-centric data management. The term, net-centric data management, consists of two phrases: net-centric and data management. Data management is shorthand for the well ordered infrastructure that results in the contextualized data that can be turned into information and finally into the superior knowledge that leads to better decisions. Net-centric is shorthand for the sharing environment within which today's data management must live. Stovepiped systems can exhibit high quality data management and never be net-centric. Net-centric environments, however, can never exist across stovepipes because the essence of net-centric is consensus on the meaning, granularity, and



precision of data. Further, net-centric environments can exist efficiently and effectively only after a foundation of quality data management has been engineered and set into place.

To achieve this vision of data sharing, one objective of a quality net-centric data management program is to transition from an environment of information stovepipes to a data sharing environment. The problem of unsynchronized information exchanges results from a fragmented data environment made up of different systems, both legacy and modern. Because these systems have not been designed with interoperability as an objective, their data objects may not share the same syntax and semantics - a necessary condition for information to have common meaning.

Another problem is the inability for systems to "talk" to each other. This can only be solved if each and every system we wish to interlink can send and receive data in compatible formats. This is especially troubling in commercial systems such as ERPs that have their own proprietary data structures, value domains, granularity, and transfer protocols.

To achieve information superiority and a common operational picture, information must flow to those that need it in real time not only to empower the commanders and decision makers, but also individual warfighters in the battlefield as well. This is the essence of "see first, understand first, act first and finish decisively."

Sharing common situational understanding is essential for collaborative operations, but building that common picture is not a simple process. Additionally, authoritative data sources and doctrine-based business rules are required to ensure that the right information reaches the right person at the right time.

As enterprises move from service specific operations to collaborative net-centric environments, achieving this vision of data sharing becomes more complex. That is why data engineering and management are essential. The issues of non-common naming conventions, definitions, and business rules impact current operations on a daily basis, and will continue to impede interoperability and operations unless addressed in architecture and system engineering.

1.3 Metadata

Throughout this paper, the term, metadata is employed. A common definition that is almost always given is: "Metadata is data about data." That definition is both too cute and too simple. First, metadata is the concatenation of two terms, meta and data. Meta, from a philosophical point of view means, "Something of a higher or second-order kind." And, within a broad context, data are "Structured and unstructured characterizations about a domain." Therefore, the general definition for metadata employed within this paper is that metadata is structured and unstructured characterizations within a domain of a higher or second-order kind.

Within the context of Information Technology (IT), metadata are the materialized artifacts that define the requirements for, the specifications of, design of, or even executing characteristics of



an IT system, or component of that system whose purpose is the creation, collection, processing, storing and sharing data with other systems. Metadata would therefore include:

- Requirements
- Functional descriptions
- Work plans
- Database designs through to schema DDL (data definition language)
- The schema information tables within SQL databases
- Application system designs possibly through to computer program source code libraries
- Technology environment designs through to actual installation artifacts

And, within the context of Information Technology, metadata would not include:

- Actual instances of the data records of employees, invoices, products, and customers
- Executing application systems
- Operating systems and other systems software such as database management systems (DBMS) and Web browsers
- Telecommunications Networks
- Computers

How metadata relates to net-centricity is that net-centric “data” is largely metadata. Virtually all the components and all the key data involved in semantic interoperability are metadata. For example, all data asset catalog “data” is metadata. These data represent the characteristics and descriptions about data assets. The detailed specifications of the data asset are also metadata. All XML schemas are metadata. XML wrapped data is, however, “real” in the sense that the values themselves are real even though the XML tags are metadata. Net-Centric data asset sharing goals are made possible only by correct and effective management of data asset metadata.

1.4 A Very Key Point

A very key point must be made with respect to interoperability: Interoperability is “the” ultimate goal of an enterprise’s net-centric data management architecture. Within interoperability, there is a critical difference between connectivity and understandability. To wit: It is simpler to be connected than to be understood. “I hear you” is not the same as “I understand you.” In some situations, that difference could mean life or death. Therefore, within the context of this paper, connectivity is not the measure of success: Understandability is.

Further, within the context of this paper, there are two characteristics that make “understandable” interoperability successful. These are: complexity and latency. Complexity refers to the time and effort to sort through similar data that is different and vice versa, and also the time and effort to synchronize the granularity, units, precision and reference data codes and meaning. Latency refers to the time to find and gather all “right” data for mission planning, execution, assessment, and reconstitution.



It is therefore far from sufficient to merely have the interoperability that comes from connectivity. Rather, the interoperability that must be available is the “understandability” that is accompanied by both a minimization of all complexity and the minimization of all latency.

It was within that definition of interoperability (i.e., understandability with minimum complexity and latency) that data exchange specifications must be developed. Table 1 contains these net-centric requirement definitions along with an logistics example.

Net-centric Data Goals		
Net-centric Data Goal	Definition	Logistics Example
Visible	Users and applications can discover the existence of data assets through catalogs, registries, and other search services. All data assets (intelligence, nonintelligence, raw, and processed) are advertised or “made visible” by providing metadata, which describes the asset.	Ability to know where and when a needed set of ammunition will be available for an operation that is to occur in the near future. The exact status of an order is available.
Accessible	Users and applications post data to a “shared space.” Posting data implies that (1) descriptive information about the asset (metadata) has been provided to a catalog that is visible to the Enterprise and (2) the data is stored such that users and applications in the Enterprise can access it. Data assets are made available to any user or application except when limited by policy, regulation, or security.	Ability to quickly and easily acquire the needed information related to the logistics supply order that was recently entered.
Institutionalize	Data approaches are incorporated into Department processes and practices. The benefits of Enterprise and community data are recognized throughout the Department.	That all the key terms and processes are defined and accomplished in a timely, reliable, efficient, and effective manner throughout the information system life cycle.
Understandable	Users and applications can comprehend the data, both structurally and semantically, and readily determine how the data may be used for their specific needs.	That everybody along the supply chain understands the critical terms and processes necessary to order, process, route, and deliver the logistics order.
Trusted	Users and applications can determine and assess the authority of the source because the pedigree, security level, and access control level of each data asset is known and available.	That the quantities, dates, and other critical logistics measures are correct and represent the truth as it is needed to be known to all persons, organizations, and processes involved.
Interoperable	Many-to-many exchanges of data occur between systems, through interfaces that are sometimes predefined or sometimes unanticipated. Metadata is available to allow mediation or translation of data between interfaces, as needed.	That key metadata and data can be used across information systems and databases without corrupting transformation, semantic disconnectivity, or requirements for reprocessing.



Net-centric Data Goals		
Net-centric Data Goal	Definition	Logistics Example
Responsive to User Needs	Perspectives of users, whether data consumers or data producers, are incorporated into data approaches via continual feedback to ensure satisfaction.	That the entire process results in the correct logistics order at the right time at the right location.

Table 1. Net-centric data goals, definition, and example.

1.5 Remainder of this Paper

The remainder of this paper presents an approach to enable understanding-based data interoperability. This paper has the following objectives:

- The identification the required environment including the most critical success measure: understanding-based interoperability
- The case for comprehensive IT specifications in the form of metadata as the exposition mechanism for both analysis and design
- The need for a metadata repository that is both comprehensive and valid that contains all analysis, design, implementation and maintenance artifacts.
- The description of an overarching reference model that ensures that all that's needed is both identified and has its proper place and role.
- A examination of the past similar efforts to gather lessons learned for the formulation of proper engineering and solution.
- The exposition of the infrastructure of data assets that have to be built and be supported by methodologies, strategies, and metrics.
- A description of a strategy to achieve an efficient and effective method in support of understanding-based interoperability.

Section 2, in support of these objectives, provides a description of the metadata environment that must be present for success including showing how common information resource management problems can be avoided. This section also describes the 1990s DoD "8320.1" data standardization effort and why it failed. Notwithstanding its failure, not only hasn't data standardization not gone away, it, as a problem that must be solved, has grown in both size and urgency.



Section 3 provides a definition of the net-centric data goals and what these goals mean within data management. Section 3 further identifies a deployment of the net-centric data strategy through a brief description of data assets, the role of de jure standards, and the requirements imposed on data management by the net-centric data goals. The section also focuses on the need for data asset products. These data asset products have always been essential for well-engineered IT architectures. The data asset product products, properly done, provide an enterprise-wide view, and also a critical integration function. Finally, this section sets out a metadata strategy and environment that results in smart, well engineered data standardization both within functional domains and across the enterprise to achieve a net-centric environment in an efficient and effective manner.

Section 4 provides a summary, a set of conclusions, and a way-ahead set of recommendations.

1.6 The Bottom Line

Achieving net-centric environments is analogous to the game of football. You sometimes get the ball on your five yard line and most always have to make good solid plays all the way down the field until the final play, which may be a 5 yard dash into the end-zone. Everybody cheers and only remembers the 5 yard dash, that is, that you've achieved understanding-based data interoperability. Data standardization is, however, the first 90 yards. Hard work, accomplished one play at a time. Achieving net-centricity is the last 5 yard run into the end-zone. Throwing that 95 yard pass from your end zone--so as to avoid the work and accomplishment of data standardization--almost always results in failure. Worse, after four such attempts, the other team, which essentially is characterized by no-understanding-based-interoperability, and data chaos, has the ball on your five yard line. Not good. The choices are thus two: Hard fought, well earned success, or "Hail Mary" pass failures. This paper addresses the first choice. Organizations operating under the second choice seldom survive.



2.0 Metadata Environment Success Factors

The concept behind the net-centric data management reference model is that just as a house has an architecture and requires many different components to make it complete (e.g., plumbing, wiring, carpentry, etc.), that achievement of understanding-based data interoperability also requires an architecture.

2.1 Metadata Management

To be successful within a net-centric data management environment, an enterprise must have quality data management that recognizes, embraces, and manages data on all its levels. A four level approach was created by the National Institute of Standards and Technology in the early 1980s. It was created to support the architecture of metadata repositories. As an academic aside, there is another strategy to present unfolding layers of metadata. It is the Meta Object Facility (MOF). While its layers are inversely numbered, it is analogous to the NIST model.

Generally, as shown in Figure 1, the left most column presents the four levels. That is, the fundamental level, the repository definition level, the repository level, and the application level. The fundamental level contains the repository definition schema. It has, as “data” the repository’s definition database. The data in the repository’s definition database then becomes the repository schema of the repository definition level. This too then has a set of data that becomes the repository database. The data in the repository database, in turn, becomes the schema for an application database, which finally has a set of data at the application level. Unlike most top-down, left-right diagrams, this is a top-down, right-left diagram.

Repository Level Pairs Types			
Fundamental Level			Repository Definition Schema
Repository Definition Level		Repository Schema	Repository Definition Database
Repository Level	Application Schema	Repository Database	
Application Level	Application Database		
	Application Level Pair (ALP)	Repository Level Pair (RLP)	Repository Definition Pair (RDP)

Figure 1. Metadata Repository Reference Model



Across the columns from this same figure, there are three sets of pairs: application level pair, repository level pair, and repository definition pair. Within the cells of these rows and columns are the metadata contents. All the levels and pairs must be properly managed to avoid the data standardization failure that occurred during the U.S. Department of Defense data element standardization effort that occurred between about 1990 and 2002. The repository and system supporting this failed effort, the Defense Data Dictionary System (DDDS), still exists but has not been maintained since late 2003. The DDDS effort is described further in Section 2.3.

Figure 2 illustrates value sets corresponding to Figure 1 to help explain these levels and pairs. A database application (Application Level Pair) consists of value sets from the application level and the database schema from the repository level that represents the metadata for those value sets. In this example, the DBMS schema has a database table with three columns. The database table (T) is Employee, and the three columns (C) are Emp_Name, Emp_ID, and Dept_Name. In this case, the value set for Employee is “P. Shaw,” with an Employee Identifier of: “525-88-2876,” and the Employee’s Department is “SQL Development.”

In this example of an Application Level Pair, the metadata names are at the Repository Level and the data values are at the Application Level. Together, these two levels comprise the Application Level Pair.

Instances Example for Repository Level Pairs			
Fundamental Level			Fundamental Types: ENTITY ATTRIBUTE RELATIONSHIP
Repository Definition Level		Meta Types: TABLE (T) COLUMN (C)	Fundamental Entity Type Instances: table (T) column (C)
Repository Level	T: Employee C:EMP_NAME C:EMP_ID C:DEPT_NAME	Meta Type Instances: T: employee C: emp_name C: emp_id C: deptname	
Application Level	DBMS Schema Instances: P. Shaw 525-88-2876 SQL Development		
	Application Level Pair (ALP)	Repository Level Pair (RLP)	Repository Definition Pair (RDP)

Figure 2. Data instances for the levels and application pairs.



The metadata names that are at the Repository Level come from the Repository Level pair. That is, there is a software system, called a metadata repository, that has its own set of metadata types ((T)able and (C)olumn), and for these types, their own instances: T=employee, C=emp_name, C=emp_Id, and C=dept name. To populate a database that resides as an application level pair, the metadata repository is queried for the tables and columns. Because the metadata repository exists, then any database that employs the employee table can retrieve its table and column set from the metadata repository. This enables a level of data standardization that would not be practically possible if no metadata repository existed.

The metadata types, that is, table (T) and column (C) come from the repository definition pair. Very sophisticated metadata repository environment can accomplish these definition with reasonable effort. In this example, the three fundamental types are Entity, Attribute, and Relationship.

In this specific example, the two fundamental entity types that have been defined are Table and Column. For each, there are attributes, like the table's name, the author of the table, and the definition of the table. In this example, there are only the two entity type instances: Table and Column. These instances are, in turn, read from the Repository Definition pair into the Repository Level pair and, within the Repository Level Pair become its types. In robust metadata repositories there may be hundreds of fundamental entity types such as DBMS, Business Rule, Information System, SQL View, Mission, Organization, Function, and Business event.

In summary, there are three sets of pairs--repository definition, repository level, and application level--are employed to fully define metadata repository environments.

These three pairs are critical to a well-ordered data management environment because these enable an enterprise to restrict its fundamental entity types, which, in turn, can be used to restrict the metadata that can be defined, which, in turn, can be used restrict the database application's tables, columns and all other metadata objects.

Other than the intellectual discipline of attempting to bring order out of data chaos, are there any practical applications and benefits from this strategy? The answer is emphatically, yes. The following problems, endemic to many IT environments are addressed through sophisticated metadata environments:

- Stove pipe systems, which are typified by 1) conflicting semantics, 2) non-shared data, and 3) costly, inefficient change mechanisms exist if the metadata management environment consists only of **Application Level Pairs**.
- Stove pipe systems are prevented when their application level pairs are based on **Repository Level Pairs**. Further, because of repository level pairs, application level pair systems can have integrated metadata.



- Meta data repositories can only be build through use of pre accomplished **Repository Definition Pairs**.

The repository level pairs are created through sophisticated IT environments that can employ repository definition pairs to quickly and effectively develop the metadata repository environment. This is all much more important than just elegant IT. A 1995 study and report was created by the Secretary of the Air Force on the 1995 state of its Information Resource Management. The following is a list of the USAF's information resource management's weaknesses that could have either been completely avoided or severely reduced if the USAF had a comprehensive metadata management program in place. Each item on the list points to the metadata repository level pair, which if it had existed would have prevented or severely reduced the weakness.

- Current Air Force IRM program is poorly defined with conflicting policy and procedures (Repository Level Pair)
- Technology solutions implemented prior to process improvements (Application Level Pair)
- Current IRM program does not measure customer needs well (Application Level Pair)
- Benefits from sound investments in IRM need to be articulated (Repository Level Pair)
- Lack of standards for creation, collection, accessibility, storage, retrieval, protection, and destruction of electronic information (Repository Level Pair)
- Many stove-piped and incompatible systems require costly changes to share information and to interoperate (Application Level Pair)
- Systems and communication infrastructure based on old costly and inefficient technologies which impede the decisions maker's access to information (Application Level Pair)
- Lack of data standardization which results in inconsistent data, difficult integration, and costly software development and maintenance. (Repository Level Pair)

It should be noted that only when a higher pair (e.g., Repository Level pair) exists can the lower level pair be properly managed (e.g., Application Level pair). Even though none of the IRM weaknesses are attributed to the Repository Definition Pair, this pair not only facilitates the Repository Level Pair, it also provides control and integrity. The types and kinds of remediation the USAF has accomplished in the area of metadata management since 1995 is unknown.



2.3 “8320.1” DDDS Data Standardization

The U.S. Department of Defense Data Dictionary System (DDDS) approach, defined within a set of documents referred to as the DoD 8320.1-M, Data Administration, and DoD 8320.1-M-1, Data Standardization Procedures (dated) 19 November, 1996, and then revised in 28 February, 1998, was supported by a metadata repository that was only a repository level pair (see Figures 1 and 2). Not only was the DDDS restricted in pairs, it was also severely restricted in scope to just “8320.1” data elements. Thus, the “8320.1” repository was not able to manage, for example, DBMS, Business Rule, Information System, SQL View, Mission, Organization, Function, and Business Event, which, along with many other types of metadata, are essential for a well ordered metadata environment. Further, because the “8320.1” repository only consists of a Repository Level Pair, it was unable to either manage or control instances from the Application Level Pair. Because of these severe restrictions, DDDS clients, that is, designers, developers and end-users were unable to accomplish proper metadata management. In short, the “8320.1” effort was doomed from day one. It was not the goal and objectives of the DDDS that caused it to fail, it was its engineering.

An internal MITRE study performed around 1994 showed the following problems with the “8320.1” approach:

- A fundamentally flawed data standardization model
- No accommodation for enterprise wide data architectures
- Multiple implementation technologies
- Central standardization and maintenance authority

The GAO (GAO-AIMD-94-14) also did a study of the DoD “8320.1” data standardization approach in 1994. Their overall conclusions were that:

- There are poor data administration practices
- Defense has not determined its corporate data requirements
- Data element standardization procedures are ineffective
- The defense data repository system does not support data administration goals

There was a DoD response on all these points, but fundamentally there was concurrence and a firm resolve to eliminate these problems. The DoD, however resolved none.

The sections that follow outline the key deficiencies in the “8320.1” effort from the MITRE study and concludes with the characteristics of what must exist for data standardization to be successful, that is, having the definitive set of the right data at the right time to the right person with an unambiguous and universally accepted meaning. Data standardization cannot be an



option, and further, it is the essential first ingredient towards achieving Net-Centric goals for a data sharing environment.

2.3.1 Flawed Data Standardization Model

The “8320.1” data standardization model and its procedures focused almost exclusively on data elements, which, because of the way DDDS were engineered, caused the data elements to be database table columns. The “8320.1” engineering model that caused this was its name construction process. That is, <prime word> <modifier> and <class word>. Prime word was, by “8320.1” convention the “table name;” hence, all data elements became database table columns. The simple consequence of this was that the “8320.1” data standardization would be complete only after all the DDDS data elements were standardized. But because the “8320.1” data elements were actually database table columns, that would then be, never.

The “8320.1” naming standard required the use of a single class word, for example, identifier, code, or text. As an example, if a data element were to be the unique data record selector for a database table, the class word would be “Identifier.” How, however would that square with the data element, Social Security Number? To adhere to the rules, Social Security Number would have to be changed to Social Security Identifier. But that too was a problem because the Social Security Number is really a set of three numeric codes. Because this “problem” became too complicated to deal with, an exception to the “8320.1” naming standards was granted.

As another example, consider Telephone Number. This too had to be granted a naming standard exception because it a) sometimes was not a number, b) was really an identifier, c) was really a compound data element consisting of well defined individual parts, d) the parts in common had discrete sets of valid values, and e) it was not the kind of number that could be combined with other numbers through mathematical operations.

Slowly but surely, because of the fundamental engineering problem of placing all the semantics of the data element into its name, the quantity of exceptions grew and grew. Exceptions, however, should be “engineered” out of the design from the very beginning. The overall utility of a data element, given that almost all its semantics were embedded in its name, is severely compromised. How, for example, would you find all identifiers if the data element is named Social Security Number? Or how would you know that a Social Security Number is a compound data element consisting of three codes? All of this semantic meaning was lost due to “8320.1” engineering failures. The “8320.1” engineering design also did not address derived data elements such as Final Monthly Balance, or compound data elements such as Federal Stock Numbers.

The DDDS repository only contains data element metadata. It does not contain data model metadata or any other metadata such as business rules or the DoDAF architecture products. That missing metadata, all essential for a complete environment, had to be created and managed in other CASE/repository tools. These other CASE/repository tools were not integrated with the DDDS. In short, the DDDS and all the other CASE/repository tools were just another set of



stove-pipes. Had the DoD implemented the late 1980s ANSI Information Resource Dictionary System standard through any of its several implementations, as was recommended by GAO in 1994, this single-level and single meta object (data element) problem could have been avoided.

For existing systems, to conform to the DDDS approach, a report from its Repository Level Pair would be produced and then database application builders would make sure that the database column names conformed exactly to the DDDS data element names. Since there was no automation to support or enforce this effort, consistency and completeness was almost never achieved. For new systems, a data element proposal package was created and submitted to DISA for approval. Until approved, nothing could proceed. This requirement was impossible to meet, monitor, or maintain because the need for new databases across the entire DoD greatly overwhelmed the ability of the DDDS standardization team to review and approve each and every database schema creation or change. In short, DDDS failed by its own engineering and processes.

The DDDS effort also never became a critical component in any system's development methodology because the "8320.1" data element proposal package creators had to wait months for their data elements to be approved. Many never were. Consequently, after new data element proposal packages were created and submitted, a "box" was checked, and then systems development efforts just continued. Waiting was not an option if schedules were to be met.

2.3.2 No Accommodation for Enterprise Wide Data Architectures

The "8320.1" engineering approach also did not address all the different data model architectures such as reference data, enterprise resource planning (ERP) COTS packages, data warehouses and the like. Under all these different data architecture classes, core data elements proliferate under slightly different names. For example, there could be use of specific data elements associated with an Inventory Item, such as Beginning Balance Inventory Item, Ending Balance Inventory Item, Inventory Item Reorder Point, or Inventory Item Average Quantity On Hand. All these variations are minor, but important, variations. Each, under the "8320.1" approach, had to be submitted and standardized as a separate data element, which only really addressed the data element's name and definition. Other issues such as languages, value domains, security and privacy, business rules and the like were neither addressed nor accommodated.

2.3.3 No Accommodation for Multiple Implementation Technologies

The "8320.1" approach had no way of handling the different requirements and/or capabilities of implementation environments such as database management systems, programming languages, and data types, precision and scale. Some DBMSs allow names up to 128 characters while others restrict the name to 32 characters. Are all to be restricted to the smallest name length? What about abbreviations? How are they managed? What about numeric precision? In different DBMSs, an integer number can have different precisions. How are different but equivalent value



domains handled? A very simple example is gender, that is, 1 is male, M is male, and Male is male. Are all variations stored and managed? Are they mapped one to the other? None of these problems were handled by the "8320.1" approach.

2.3.4 Having a Centralized Data Standardization Authority

The very fact that there was a centralized approach to data standardization was, in itself, a fundamental problem. There was the never ending increase in the quantity of database columns that then had to be standardized. For example, if a new database had 200 tables and each table had 15 columns, then instantly there had to be 3,000 new data elements standardized. If each data element required 15 minutes of effort then it would take about 4 staff months to standardize one database. The Army and the Navy have well over 125,000 discrete systems. If there are only 100,000 databases across all these systems then it would take about 33,000 staff years to accomplish the data standardization effort. Not possible.

2.3.5 "8320.1" Data Standardization Summary

As a consequence of DDDS failing it was decided to take an inverse approach. That is, attaching XML tags to data elements (really database table columns) within the context of automated information systems, and then posting those information system based XML tags to the DoD's newly created metadata registry. That, however, is a far worse alternative. That is because it institutionalizes all the mismatches in names, semantics, granularity, value domains and precision across all the columns of tables of the databases throughout the DoD. Every XML schema is the schematic of an information exchange. If an application system has 100 windows and each window has 2 browses, then there needs to be 200 XML schemas for just that one information system. If there are 150,000 information systems between just the Army and the Navy then there needs to be 30,000,000 XML schemas to serve their needs. In short, each XML schema then becomes an expression of a data standard. This too, like DDDS, is an impossible to create, monitor, and maintain approach.

Alternatively, if there are 100,000 databases, and each of the database's schema is standardized then there are 100 times fewer components to standardize. Further, if every column in a database is really a reuse of an ISO 11179 data element, and if industry wisdom holds true, that there are fewer than 10,000 ISO 11179 data elements in all of DoD, then through smart, well engineered data standardization, manifest through sophisticated metadata repositories, then all columns in every database can have their names, definitions, value domains and data types automatically generated. Flowing from that would then be automatic standardization of all XML tags and, through metadata integration with information systems, the automatic generation of XML schemas.

If there is to be understanding-based interoperability, then there must be agreement among the members of the community of interest as to the shared information's semantics, value domains



and meanings, granularity, precision, and any common business rules that transform data. That consensus is called data standardization.

Consequently, not only is there is no rational alternative to data standardization, the requirement for data standardization cannot be wished away or ignored. The more that it is wished away and ignored, the greater and more urgent it is to address and solve. It is a fundamental requirement for shared information environments whether they are manual, message based, through databases, or through XML exchanges. Further, the quantity of information systems, and the quantity of types, formats and meanings of data are proliferating almost without control. Every desktop with MS/Access is a source of a database. So too is every Excel generated spread-sheet. To exclude these and all other information systems and database development environments from the IT environment is not possible because such an exclusion could never be enforced. And if it could be enforced, then individual creativity and initiative would be unacceptably restricted. So, rather than a policy of exclusion, an environment of smart, well engineered data standardization must be established and co-exist with end-user development environments such that, together, they enhance and facilitate understanding-based data creation and use environments.

2.3.6 “8320.1” Lessons Learned

To be successful in understanding-based data interoperability data standardization plays an essential part. Studying the “8320.1” effort provide valuable lessons learned, which are that any data standardization effort, no matter how small must:

- Properly focus on the data concepts such that the lowest level column is easily and automatically named, defined, and abbreviated.
- Accommodate enterprise wide data architectures such that data contexts are immediately apparent without compromising data identification, selection, and melding.
- Accommodate multiple implementation technologies such that regardless of the rules, a data concept are immediately obvious and relatable to all other data that espouse the same concept.
- Allow front-line project staff to create and maintain their own names under the guidance, facilitation, and work enhancing tools and techniques provided by a data standardization organization.
- Be based on a comprehensive, valid metadata datamodel that in turn is the basis for a comprehensive metadata repository system that can distributed and federated.

As to the first success factor, focusing on data concepts versus the lowest level column, the ISO 11179 has a formal paradigm for data elements. This standard is not only critical to the execution



to smart, well engineered data standardization, it was also created well after the failure of “8320.1” became apparent, and, based on its construction, it is clear that this ISO 11179 standard incorporated many of the “8320 lessons learned.”

Key components of this standard are concepts/objects, conceptual value domains, data element concepts, and value domains. Because of the elegant engineering of this standard, most if not all of the “8320.1” engineering errors vanish. In ISO 11179, a data element concept is built from concepts and conceptual value domains. From the data element concepts, various data elements that may differ only in the precise value domain or in some other minor difference, are able to be created. Thereafter, the ISO 11179 data elements, which are essentially, independent business fact semantic templates, are able to be employed within different contexts, that is, as attributes of entities, columns of tables, or DBMS columns of DBMS tables.

After the semantics of an ISO 11179 data element has been infused into an entity’s attribute or a table’s column, each such specialization can have its own more localized name, and even more localized modifiers and value domains. While these are indeed different attributes, columns, or DBMS columns, they remain derived from their ancestor ISO 11179 data element. Thus, the essential semantic core of each attribute or column is the ISO 11179 data element.

Because of this strategy, there can be a real finite quantity of data elements, and the process of completing data element standardization can actually be accomplished in a practical quantity of time and resources. This standardization and closure process does not have to start from scratch. That is because the existing DDDS database of “8320.1” data elements can be mined to discover the ISO 11179 data elements. Once done, it is likely that about 90% of the ISO 11179 data elements will then have been identified and defined.

As to the second success factor, the ability to extend data standardization databases such as data warehouses, this approach is quite easy if there is a layered metadata management approach. Such an approach would have the ISO 11179 data element layer as the top layer, which would then provide standardization to a data model template layer. These two layers would then enable the creation of new databases from all these pre-standardized parts. It’s simply a matter of tagging the data model templates and bringing them under the domain of a database schema. Creating relationships among these preassembled parts is equally simple. The key feature of this multi-layered is that as new data models are created they are automatically mapped to their generator data model templates, which, in turn are already mapped to their ISO 11179 data elements. In short, manufacturing new databases under different architectures is equivalent to “running a production line.”

As to the third success factor, accommodating multiple technologies, this is accomplished as a consequence of being able to generate physical data models required for different DBMSs from the logical data models in the same manner as described above. Mappings are automatic.

As to the fourth success factor, allowing for the front-line project staff to create and maintain their own names without dealing with an onerous, time consuming data standardization



authority, this multi-layered approach is specifically designed to have a full set of semantics within meta attributes for every column. Because of this, and because of mapping, there is no requirement that names be severely managed.

The fifth success factor enables the multi-layered approach. The USAF's 1995 IRM weaknesses could have been prevented through the proper use of a comprehensive, distributed repository that can exist in a federated environment.

This multi-layered approach is also especially designed to handle legacy and database maintenance environments wherein reverse engineering and database modifications are the standard operating procedures rather than new development. The multi-layered approach supports automatic naming, automatic definitions, value domain management, and the like, which is ideal for top-down new developments, the bottom-up approach is fully supported.

Metadata repositories that exist in federations can support metadata export and import and then harmonization within the upper levels of pre-defined sets of 11179 data elements and data model templates. It is especially because of these upper layers that different legacy environments can be related one to the other.

Again, for a modern data management environment to be successful, that is, to have the definitive set of the right data at the right time to the right person with an unambiguous and universally accepted meaning, data standardization and value set cannot be an option. It is the essential first ingredient for data sharing environments that can then be easily adapted to support net-centric environment.

Cycling back to the USAF set of IRM weaknesses, and for the IRM weaknesses of all the other Services to be minimized and/or resolved, it is critical to have a comprehensive metadata environment that can contain, interrelate, and be definitive about IT's assets.

2.4 The "8320.2" Data Sharing Directive

The "8320.1" data standardization approach described above was canceled and replaced with the DoD Directive 8320.2, Data Sharing in a net-centric Department of Defense. The approach contained within this new directive partially addresses the problems cited in the MITRE study, but mainly avoids dealing with them altogether by pushing them down onto the data standards producer organizations to accomplish.

From above, the problems that needed to be addressed to make data standardization successful were:

- A fundamentally flawed data standardization model
- No accommodation for enterprise wide data architectures
- Multiple implementation technologies
- Central standardization and maintenance authority



The new 8320.2 resolves these in the following way. First, there is no data standardization model within 8320.2. Second, there is no strategy for enterprise wide data architectures. Third, the need for or the ability of have multiple implementation technologies is not addressed in any way. And, fourth, there is clearly no central data standardization or maintenance authority.

The directive gives scant, if any, guidance on how any of these policies are to be accomplished. Further, has been stated numerous times that zero guidance is forthcoming. Finally, other than a registry for depositing the results of any data standards producer based processes that would effect data assets that are discoverable, understandable, visible, accessible, interoperable, and are able to be trusted. Again, there is scant, if any, guidance for the repository deposits, And for quality, non-redundancy, or integration, the guidance is largely missing. Some organizations and communities of interest could do a fantastic job while others might not. In the case of data interoperability, however, the interoperability-chain is only as strong as its weakest link.

Within this directive, the definition of data asset is problematic. A careful review of its definition leads one to the conclusion that virtually anything that acquires, stores, and/or produces data could be a data asset. For example, a data collection sensor within an M1Abrams tank could be a data asset.

Under the “new” 8323.2 approach, 100% of data standardization is the responsibility of the program, the community of interest, the mission area, the data standards producer organization or the enterprise. If data standardization is not accomplished in a manner that ensures understanding-based data interoperability then the users are just plain “out of luck.” The 8320.2 directive merely provides the requirement for data to be prepared such that it can be shared. The criterion level at which data can be shared, that is because it has the right characteristics, i.e., understood, trusted, visible, etc., is completely undefined. Worse, sharable data to one person may be completely unintelligible to another. There is no real substantive guidance on these issues. In Section 5.6.1 of 8320.2, however, the heads of data standards producer organizations shall:

“Ensure implementation of net-centric data sharing, including establishing appropriate plans, programs, policies, processes, and procedures consistent with policies herein.”

Thus, the complete responsibility for creating the infrastructure through which net-centric data can exist resides within organizational components, or within communities of interest. It almost seems that the 8320.1 is total, but mis-engineered guidance, and the 8320.2 is the absence of all necessary guidance.



2.5 Metadata Management Summary

Stated at the start of this section, for an enterprise to be successful in information superiority, it must first have a metadata reference model and an approach for metadata management that ensures success in a distributed environment. To ensure success, a very careful analysis was performed on the failed DoD data standardization effort, DDS. The engineering features that caused it to fail were uncovered and analyzed. As a consequence of these lessons-learned, a data standardization approach has been created and tested that is based on multiple levels of metadata through which the necessary set of metadata can be uncovered, interrelated, and then engineered in such a way to avoid that the typical problems that plague information resource management.

Section 3 takes this foundation and builds on it the environment of data assets, metadata, and metadata architecture necessary for success.



3.0 Shared Data Architecture

The previous section identified the metadata management environment that must exist to be successful in the net-centric environment. The previous section also identified the reasons why prior data standardization efforts failed and what must be done to avoid future failure. In this section, an architecture for shared data is presented. It consists first of the identification of the set of data asset products that exist explicitly or implicitly within various IT systems and database architecture. These data asset products are then set into a metadata development environment that ensures that once the IT products are developed there will then be data management success. Underlying this development environment is shared metadata. It is essential that the data centric metadata, which permeates the IT products, be properly developed, non-redundant, integrated, and shared as they will lead to IT product integration.

The shared data architecture is deployed within the data management environment. Data, configured as data assets, include, for example, system files, databases, documents, official electronic records, images, audio files, web sites, and data access services. Every data asset must be assessed against a set of mandatory net-centric goals. That is, that it is visible, accessible, institutionalized, understood, interoperable, trusted, and responsive to user needs. Under the shared data architecture approach, every data asset consists of two fundamental forms: its content, and the envelope that surrounds its content. The collective term for all specifications about a data asset, including its content envelope, is commonly known as metadata.

Data assets can also be considered to be either structured, unstructured, or some level in between. Structured data assets commonly refer to data assets that are or are derived from highly structured sources such as databases. Unstructured assets are those that appear, on the surface, to have no internal structure. However, most unstructured data assets have some level of structure; its just a matter of how much structure and how formal. An Excel spreadsheet is highly structured. Further, it may have been an extract from a database. An apparently unstructured data asset may be a letter or a recorded stream of voice. To say that is unstructured would be misleading as either has a date, target, source, and likely a reason, opening text, details, conclusion and closing text. In short, structure. These data assets would be considered lightly structured.

Envelope metadata is different from content metadata. Envelope metadata applies equally to all content and thus, would be the data asset's key words, subject, author, date, security level and the like. Given quality engineered data assets, setting these within net-centric environments merely requires posting the data assets's metadata to the appropriate catalogues, and creating the mechanisms for data asset access.

Content metadata differs in both form and extent. For a data asset that is a BLOB (binary large object) there may be very little to no content metadata. In the example of a "jpeg" BLOB data asset, there may merely be the envelope metadata and only scant content metadata. But in the case of a database, the content metadata is very significant. It consists of the database's entire set of schema information tables and could thus be 100s of millions of characters. In the case of a



data asset that is in a XML form, there is envelope metadata, XML schema metadata, and then the data asset's XML stream metadata. Additionally, to fully comprehend a XML stream, if the data was originally extracted from a database, then access to the database's metadata would also be in order.

Thus, the effort required for a data asset to be compliant to the net-centric data goals depends very significantly on the data asset's form. The effort for a "jpeg" data asset is small and quick. But for a database data asset, from which there may be a thousand or more XML schemas and corresponding XML data streams, the effort is large and time consuming.

Every data asset requires an authoritative source so that it, when employed, is known to be the definitive instance. Many data assets will also require a unique enterprise identifier so that its instance is unambiguous. Data assets will need to be configured (format, names and meaning) into a form, such that it meets the needs of communities of interest. Along with the associated business rules, the specifications of this commonly reusable form is called an information exchange standards specification. When data assets are accessed and/or exchanged, then its content may be transmitted "raw" or may be wrapped in "handles." Handles include, for example, Electronic Data Interchange (EDI) or eXtensible Markup Language (XML). Data asset content exchange format decisions are based on practicality and performance.

Data assets, then, which meet the net-centric data goals by being represented authoritatively, identified uniquely, configured within a common exchange format, and interchanged via appropriate wrappers, are interoperable across the enterprise. Data assets lacking these attributes are not interoperable, enterprise-wide.

The data asset infrastructure underlying interoperability is accomplished through the metadata management of shared data elements, built through the ISO 11179 data element metadata standard, which, in turn, are the semantic templates for the entity attributes within shared data segments of conceptual data models. The shared data segments are employed and/or mapped to database table column sets of logical data models so that the database table columns can represent standardized semantics.

The de jure standards, which are employed to make data assets technologically definable and exchangeable across the world, are ISO/ANSI SQL, WC3 for data asset XML construction, and ISO 11179 for data element standardization.

Data assets, constructed along the lines cited above, are thus able to be materialized, seen, understood, collected, interrelated, and exchanged both automatically within a weapon system's component and also within and across collections of systems that may be deployed within programs or collaborations of programs.

However, if interoperable data asset environments are not created and managed across the environment, then operations and system executions will either be severely degraded or fail



outright. Thus, “data” is an essential component to any architecture view that must be managed across the life cycle of all systems.

Table 2 presents that applicability of the net-centric data goals onto dData management.

Net-centric Data Management Support			
Net-centric Data Goal	Net-centric Definition	Data Management Support	Support Description
Visible	Users and applications can discover the existence of data assets through catalogs, registries, and other search services. All data assets (intelligence, nonintelligence, raw, and processed) are advertised or “made visible” by providing metadata, which describes the asset.	Identify data assets within natural contexts of mission, organization and function. Standardize taxonomies, ontologies and classification schemes that then enable viewing data semantics including where and how implemented through data assets.	The data management approach supports the net-centric goal by ensuring that data assets are set within their natural contexts of mission, organization, and function. Further, that they are all tagged with standardized taxonomies, etc, at a level and through a strategy that ensures consistent tagging of similar data assets. This is accomplished through use of ISO 11179 for data elements, and conceptual, logical, and physical data models. Physical data models are at the level of data assets. The Data management approach also enables the automatic redetermination that data assets need to be re-tagged and the resultant metadata re-posted.
Accessible	Users and applications post data to a “shared space.” Posting data implies that (1) descriptive information about the asset (metadata) has been provided to a catalog that is visible to the Enterprise and (2) the data is stored such that users and applications in the Enterprise can access it. Data assets are made available to any user or application except when limited by policy, regulation, or security.	Include a metadata definition in every data asset. Standardize names, definitions and structures first through templates and then incorporate these standardization artifacts, where practical, within data assets or at least map these standardization artifacts to data assets.	The Data management approach supports the net-centric goal by enabling the automatic production of a comprehensive set of metadata that would enable users to thoroughly understand the semantics of a data asset. Included would be all data elements, their encompassing concepts, conceptual value domains, value domains, the conceptual entities within the data asset, the metadata for each database table, and the operative processes and business rules that govern data integrity. For each data object in the metadata (e.g., data elements, entities, tables, and business rules), there would be definitions, descriptive material, and precise formats. For application use of the data asset, there would be both XML schemas for the various standardized interfaces and SQL Views.
Institutionalize	Data approaches are incorporated into enterprise processes and practices. The benefits of Enterprise and community data are recognized throughout.	Standardize strategies for data definition. Create multiple layers to ensure define once use many times. Use ISO standards 11179 (part 3) for shared data elements and SQL for data	The Data management approach supports the net-centric goal by ensuring that there is a comprehensive set of metadata that is consistent within a data asset, harmonized across data assets within a community of interest, and harmonized across families of communities of



Net-centric Data Management Support			
Net-centric Data Goal	Net-centric Definition	Data Management Support	Support Description
		models.	interest. This is accomplished by ensuring that all data asset metadata, that is, data elements, conceptual, logical, physical and business rule models, are both vertically and horizontally harmonized. Further, the Data management approach is supported by configuration management that enables knowledge of development, test, and iterations of production data assets.
Understandable	Users and applications can comprehend the data, both structurally and semantically, and readily determine how the data may be used for their specific needs.	Create standard vocabularies, commonly inherited semantics, commonly used data model templates, automatic names and definitions based on well defined words. Automatically create abbreviations where necessary.	The Data management approach supports the net-centric goal by ensuring that data assets are set within their natural contexts of mission, organization, and function. Further, that the metadata layer of data assets is able to be explored to ensure users that they not only understand a given data asset but also are able to browse among data assets of similar type, construction, or mission, functional, and organizational context. The syntactic and semantic metadata is engineered such that it can be readily employed by commonly available, industry best practice tool sets such as CASE tools, DBMSs, and XML readers and writers.
Trusted	Users and applications can determine and assess the authority of the source because the pedigree, security level, and access control level of each data asset is known and available.	Create consistent semantics, standard reference tables within ISO 11179 based shared data elements, and employ these to govern/map data models across multiple levels of abstraction.	The Data management approach supports the net-centric goal by standardizing both the semantics and value domains and meanings across all authoritative data sources. Additionally, the Data management approach enables the mapping of prior value domains and meanings to current value domains and meanings, and the mapping among uses of the same or semantically similar value domains and meanings. The Data management approach enables the quick determination of the processes that capture, update, and report data asset data. And finally, the Data management approach enables the allocation of various classes of security and access control.
Interoperable	Many-to-many exchanges of data occur between systems, through interfaces that are sometimes predefined or sometimes unanticipated. Metadata is available to	Standardize data structures. Establish well engineered data transactions, automatic XML wrapping of data, supported by accessible data definitions and contexts.	The Data management approach supports the net-centric goal by ensuring that regardless of localized naming and data types that the same data is commonly mapped to higher level data models and/or data elements. With the Data



Net-centric Data Management Support			
Net-centric Data Goal	Net-centric Definition	Data Management Support	Support Description
	allow mediation or translation of data between interfaces, as needed.		management process, data that is named the same but is different will be able to be discovered as easily as data that is named differently that is the same. The hierarchy of data models (data element to conceptual to logical to physical to view) enables exchanges between and among systems to be quickly specified and managed.
Responsive to User Needs	Perspectives of users, whether data consumers or data producers, are incorporated into data approaches via continual feedback to ensure satisfaction.	Support reuse of already defined data assets metadata, central knowledge of all data assets and distributed access to same.	The Data management approach supports the net-centric goal by ensuring that data assets are set within their natural contexts of mission, organization, and function. Additionally, data assets that are either universal or that cross organizational boundaries are able to be interrelated. Specified in support of system exchanges are business cycles, calendars, and data value precisions. The Data management approach causes the What, When, Where, Why, Who, and How to be both captured and interrelated in support of ensuring satisfaction among data producers and consumers.

Table 2. Net-centric goals along with data management support.

3.1 Data Asset Products

Data asset products are the data-specific inputs needed or outputs produced in data asset life cycle activities. Data asset products commonly organize and define interrelationships of data in support of each organization's missions, functions, goals, objectives, and strategies. These products give the basis for the incremental, ordered design and development of one distributed virtual database founded on successively more detailed levels of data specifications to build out the data asset product set. Metadata repository identify the set of data asset products in each data asset project.

Data asset products provide the basis for the incremental, ordered design and development of one distributed, virtual database founded on successively more detailed levels of data specifications to "build out" the Data Asset Product set.

Certain data standards are defined within data asset projects, and become the underlying foundation of other data asset project products. For example, one data asset project may fully specify reference data standards as its data asset product, which, in turn, is then used in another



data asset project as a critical component within its complete data asset specification. The functional and technical documentation about a data asset's structure, meaning, and use comprise the metadata for every data asset product. To achieve data exchange capabilities for information exchange requirements, implementing and enforcing data standards is mandatory.

To document and implement data standards, functional data proponents must be actively involved in managing information content. Functional proponents establish the business rules for data use and define the constraints governing how data are processed (e.g., referential integrity constraints for add, change, and delete transactions against records in a database). Business rule statements describe these constraints. To achieve consistency in data use, different data, obtained from different sources, must be combined (synchronized) in accordance with functional business rules defined by the functional process proponent.

The Department of Defense (DoD) has an architecture initiative known as the DoD Architecture Framework (DoDAF). While this architecture framework is unique to the U.S. DoD, there exists analogous frameworks across all complex industries. The DoDAF is not just for IT artifacts but for all classes of artifacts relating to equipment, organizations, hardware and software systems, force deployments, and the like. In addition to two overarching “views” called the “All-views” there are three specialized perspectives on an architecture: Operational, Technical or Systems. These perspectives are represented through one or more of the 26 architecture products. One of the goals of the Architecture Framework is to ensure that architecture-related information can be more readily shared and re-used among the services, especially if the architecture artifacts are maintained in databases that support all the data needed automatically to generate these products.

While the DoD Architecture Framework specifies just three types of architectures views (Operational, Systems, and Technical), there are within these views data asset products that address issues, standards, guidelines, and analysis in the area of data asset definition and exchange. Data asset products critically impact information compatibility, interoperability, and integration across all views. Operational Views (OV), System Views (SV), and Technical Views (TV) can however be developed without being specific about data asset products.

For example, the systems architecture view primarily documents hardware and software configurations and information processing/database management system platform environments, while the technical architecture view specifies standards and protocols. But, if the operational architecture views record information requirements in the form of information flows (both need lines and content) without ensuring the recorded information flow data specifications are consistent, complete, and encompass all stated data requirements, then its value is severely limited. In contrast, the data asset products provide the means for data exchange and the underlying data specifications essential to these exchanges.

Data asset products, as described above, commonly exist within each of the three DoD Framework architecture views are depicted in Figure 3. Tables 3 and 4 tabulate the multiple uses of the data asset products. The data asset product names, in **bold**, are those most readily recognized by data engineers. The not-bold product names are those that would be most readily



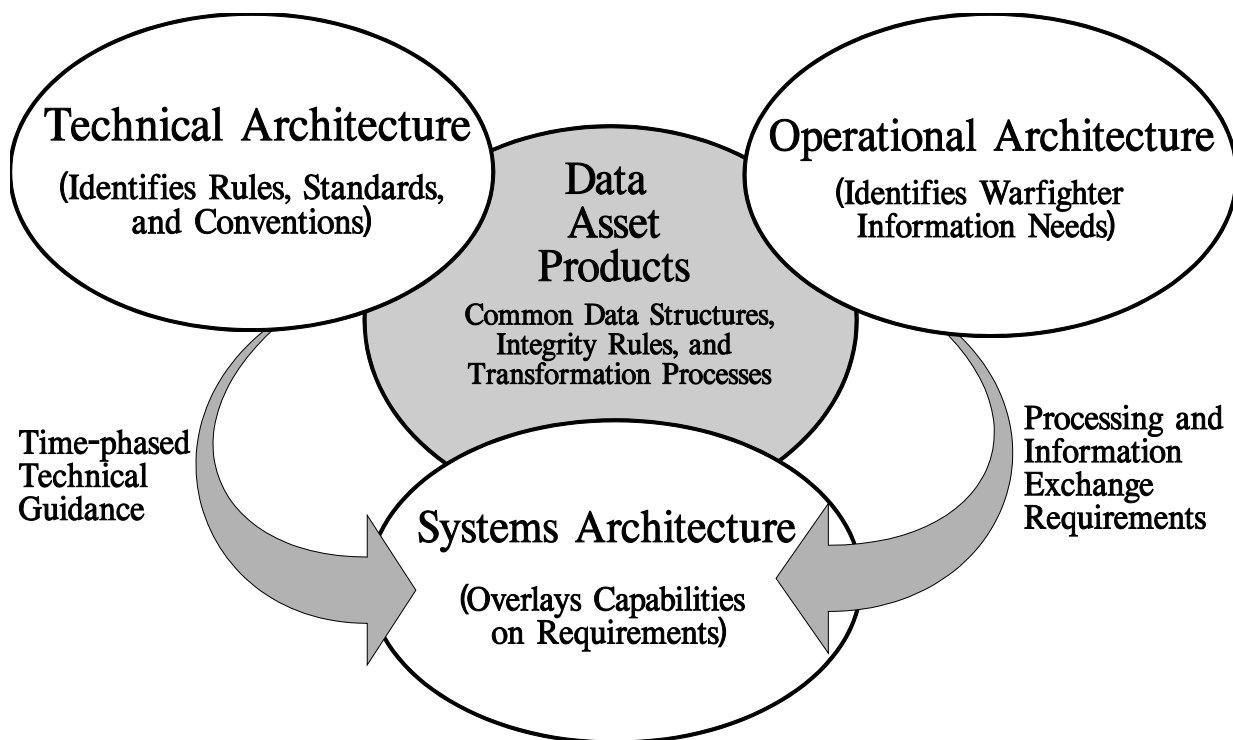


Figure 3. Technical, System, and Operation Architecture Views along with Data Asset Products.

recognized by IT system developers. Different disciplines would have their own product name sets.

Accomplishing data asset products will significantly increase the effectiveness of overall enterprise architectural efforts. In particular, information systems will share a consistent approach that addresses the definition, capture, storage, management and distribution of data as an integral component of systems design. This reduces system development and integration costs by reducing the per system (stovepipe) engineering for data. This also promotes timely identification of data issues along with operational and procedural issues impacting information interoperability.

Properly accomplished, that is, non-redundant and integrated, the data asset products act as architecture integrating mechanism not just across one architecture instance but across all architecture instances.

Data asset products are also a key aspect of ensuring the compliance and validation of architecture products. Whereas operational architecture products record information exchange requirements, and technical architecture products promote software and hardware interoperability, and systems architecture products set common links and communications



networks, no portion of the DoDAF, specifically addresses the requirements of understanding based data interoperability at the level required to permit data reuse without loss of semantic and syntactic integrity.

Data asset product initiatives are primarily focused on data exchange and data reuse among databases, where data is highly structured, and its semantics—i.e., entity and attribute definitions, enumerated domains, etc.— and physical characteristics (e.g., data length, data type, etc.), need to be tightly controlled in order to permit information transfers with minimal or no human intervention.

If data asset products are not properly accomplished within a program, that is, in an integrated, non-redundant fashion, that program cannot be truly successful in a net-centric environment. When these underdone or undone data asset products are discovered, the program will have to expend significant quantities of time and monies to not only accomplish these products, but also recast already completed products into a correct, integrated and non-redundant formulation.

The reason why data asset products need to be identified and accomplished in an integrated and non-redundant fashion is because the essential component of net-centricity is data. The data must not only be “interoperable,” it must also be understandable with a minimization of complexity and latency. Hence, the data asset products of a program must be accomplished both at their appropriate level and also from a higher perspective.

Table 3 presents an enumeration of the various products required across the Technical, Operational and System views. In this table, the “bold” components within these views are specifically data asset products. The non-bold data asset products are typical products that would have to be accomplished regardless. The point here is simple. The Technical, Operational, and System view cannot be successfully accomplished, in a net-centric manner unless and until these data asset products in bold are accomplished.

If these products are accomplished from within a community of interest, then the probability of the programs data being “instantly” net-centric dramatically increases. The other point from Tables 3 and 4 is that data asset products are employed multiple times within the different architecture views.

These products are done once, and if they are stored in a federated metadata repository which is then available to system developers then as changes in the physical data model are made then these changes appear everywhere. Additionally, if, as required by the net-centric data management approach, the [ISO 11179] data element, conceptual, logical, physical and view models are all thoroughly integrated, then changes to the physical data model are made in concert with its higher level models. Finally, if a federated metadata repository environment is employed, then changes to any program’s physical model can be immediately noticed by all other projects that may employ XML schemas, and XML wrapped data from all other physical data models across Services, Coalitions, and Communities of Interest.



DOD Architecture Framework Views, Architecture Products and Data Asset Products		
Prod Ref	View Product	Architecture Product
AV-1	Overview and Summary Information	Mission, Database Domains
AV-2	Integrated Dictionary	Business Terms, Data Element Model
OV-1	High-level Operational Concept Graphic	Mission, Database Domains , Functions, and Organizations
OV-2	Operational Node Connectivity Description	Business Event, Business Calendar, Business Cycle, Business Information Systems, Organization, and Function
OV-3	Operational Information Exchange Matrix	IESS via physical data model , with Business Information Systems, Business Event, Business Calendar, and Business Cycle
OV-4	Organizational Relationships Chart	Organization, Function, and Business Event
OV-5	Operational Activity Model	Business Function, Business Events, Physical Data Model, and View Data Model
OV-6a	Operational Rules Model	Business Functions, Business Events, Physical Data Model, Views, Data Integrity Rules , Business Information Systems, and Database Objects .
OV-6b	Operational State Transition Description	Projects (To Be and As-is) That Identify Business Functions, Business Events, Physical Data Model, Views, Data Integrity Rules , Business Information Systems, and Database Objects .
OV-6c	Operational Event-Trace Description	Mappings Regarding Business Functions, Business Events, Physical Data Model, Views, Data Integrity Rules , Business Information Systems, and Database Objects .
OV-7	Logical Data Model	Data Element Model, Conceptual Data Model, and Logical Data Model .
SV-1	Systems Interface Description	Business Information Systems, Views, and Inter-view Mappings
SV-2	Systems Communications Description	Not in the Current Domain of Data Asset Products.
SV-3	Systems-Systems Matrix	Business Information Systems, View Models, and Physical Data Models
SV-4	Systems Functionality Description	Business Information Systems, Database Objects Model, and Logical Data Model
SV-5	Operational Activity to Systems Function Traceability Matrix	Business Information Systems, Business Events, Business Functions, and Optionally to Organizations and Missions
SV-6	Systems Data Exchange Matrix	Business Information Systems, Views, and Inter-view Mappings
SV-7	Systems Performance Parameters Matrix	Not in the Current Domain of Data Asset Products.
SV-8	Systems Evolution Description	Projects Relative to the “To Be” and “As-is” That Identify Business Functions, Business Events, Physical Data Model, Views, Data Integrity Rules , Business Information Systems, and Database Objects .
SV-9	Systems Technology Forecast	Not in the Current Domain of Data Asset Products.
SV –10a	Systems Rules Model	Business Information System, Database Objects, Data Integrity Rules and Logical Data Model .
SV-10b	Systems State Transition Description	Business Events and Database Objects



DOD Architecture Framework Views, Architecture Products and Data Asset Products		
Prod Ref	View Product	Architecture Product
SV-10c	Systems Event-Trace Description	Mappings among Business Events and Database Objects
SV-11	Physical Schema	Physical Data Model and View Data Model
TV-1	Technical Standards Profile	Not in the Current Domain of Data Asset Products.
TV-2	Technical Standards Forecast	Not in the Current Domain of Data Asset Products.

Table 3. Identification of Data Asset Products within DoDAF Architecture Views

Data asset products extracted from Table 3, and uniquely listed are:

- Conceptual data model
- [ISO 11179] data element model
- Data integrity rules
- Database domains
- Database objects
- Inter-view mappings
- Logical data model
- Physical data model
- View data model

Each of these products are defined in other net-centric data management documents. The interrelationship among these data asset products and the All, System, Operational, and Technical views is provided in Table 4. A high level specification of the data asset products is contained in Attachment 1 of this paper.

DoD AF Architecture Views									
	Conceptual Data Model	Data Element Model	Data Integrity Rules	Database Domains	Database Objects	Inter-View Mappings	Logical Data Model	Physical Data Model	View Data Model
AV-1				✓					



DoD AF Architecture Views									
	Conceptual Data Model	Data Element Model	Data Integrity Rules	Database Domains	Database Objects	Inter-View Mappings	Logical Data Model	Physical Data Model	View Data Model
AV-2		✓							
OV-1				✓					
OV-3								✓	
OV-5								✓	✓
OV-6a			✓		✓		✓	✓	✓
OV-6b			✓		✓		✓	✓	✓
OV-6c			✓		✓		✓	✓	✓
OV-7	✓	✓					✓		
SV-1						✓			✓
SV-3								✓	✓
SV-4					✓		✓		
SV-6						✓			✓
SV-8			✓		✓		✓	✓	
SV-10a			✓		✓		✓		
SV-10b					✓		✓		
SV-10c					✓		✓		
SV-11								✓	✓

Table 4. Interrelationship between Data Asset Products and DoDAF Views



3.2 Shared Metadata, The Key to net-centric Success

Consensus based, shared metadata is a prerequisite for data sharing in a net-centric environment, the following issues need to be understood and addressed at the data management level. The issues are:

- XML schemas
- Data asset catalogs
- ISO 11179 based shared data elements
- Shared data segments
- Levels of shared metadata

3.2.1 XML Schemas

It is the consensus of the internationally known and recognized members of the data management community that a XML schema is conceptually equivalent to a SQL View. The XML schema is the data schematic of a transaction between two or more information systems. Thus, there could likely be hundreds to thousands of XML schemas for each database. Further, there could well be XML schemas that are the result of merging the selected data extracts of many different databases. If the tags contained in the XML schemas are not founded on quality engineered, non-redundant, and shared metadata-based data models, then two problems certainly result. First, there is be virtually no chance that any of the reused XML tags will have the same meaning. Second, there is be a very large chance that different tags will in fact represent the same data element concept. Simply put, XML schemas not founded on quality engineered, non-redundant, and shared metadata-based data models will not represent any substantial progress towards meaningful shared metadata that is prerequisite for data sharing.

Section 2.3.5 presented a method to calculate the quantity of XML schemas that would need to exist if all data exchange between databases and information systems is XML based. Given that an XML schema, under this strategy of definition, is equivalent to a SQL view based exchange, then it's a fact that the quantity of XML schemas posited will in fact exist. The need therefore is to find some way of automatically generating these XML schemas. The first step in this is to determine a viable strategy for determining the XML tags. The data management industry has spoken definitely on this issue through The Data Administration Newsletter electronic journal. In an article entitled, *Where do the XML Tags Come From*, (<http://www.tDan.cOm/i021hy02.hTm>) Leading data management experts such as Peter Aiken, Terry Moriarty, and David Hay, all published and nationally recognized international experts were polled on their preference to two alternatives:

- Alternative 1: System Centric
- Alternative 2: Data Centric



The system centric and data centric alternatives along with a summary of key points from the responses from eight data management experts are presented in Sections 3.2.1.1 through 3.2.1.3. The GAO also conducted a study and made specific recommendations regarding XML as it relates to enterprise architectures and data standardization. The key points from the GAO study are presented in Section 3.2.1.4.

3.2.1.1 IT System Centric Alternative

If you have an IT System (MIS, Package, etc.) and it does not have to share data with any other Information Technology system then, you, as the Information Technology System manager, are 100% in charge of the XML Tags, and you are also in charge of all business fact value sets wherever the value sets are restricted.

Else, if the Information Technology System has to share data with another Information Technology system, then for those shared data areas, the two Information Technology System managers are 100% in charge of the XML Tags, and also in charge of all value sets where the value sets are restricted.

These XML tags are then imposed on all creators of the Information Technology system so that every data exchange must extract and transform the "raw data" into the XML Tagged sets.

3.2.1.2 Data Centric Alternative

All the data in all the Information Technology systems are to have their **names** and their restricted value sets defined and managed by the highest level functional experts in the enterprise such that when either the functionally defined names and values sets are employed, then, their XML tags and values are standard across all Information Technology systems that may employ them regardless of whether there are any plans for these systems to share data.

Names here include both container names (e.g., table or schema), or fact names (e.g., columns). All fact names are taken from ISO 11179 based shared data elements (now acting as business fact semantic templates for the fact names). Finally, that all "words" that comprise ISO 11179 based shared data element names are standardized within their domains (e.g., engineering, business, medical) via standard name-strings, definition fragments, and standard abbreviations. Finally, that all ISO 11179 based shared data element value domains are standardized.

"Highest level functional experts" are those persons designated by the enterprise to be the subject matter experts (SME) in that functional area, and thus, it should be their responsibility and authority to define the XML tags that should be deployed in what ever information systems that may employ their data.



Thus, by pushing the responsibility and authority for XML tags to enterprise-level subject matter expert, it explicitly removes XML tag development authority from the designers and builders of information systems that may employ data from multiple functional domains. This has the consequence of having the same tags across information systems regardless of whether those information systems currently are exchanging data.

These XML tags are then automatically created by application programs that are processed through SQL:2003 DBMSs that conform to the soon to be finished and ratified SQL/XML Part 14 that is under very intense implementations by the current set of SQL vendors (read: IBM, Oracle, Sybase, and Microsoft).

3.2.1.3 Responses Key Points

Across the data management experts, the following points were made:

- All eight agreed that the data centric alternative is preferred.
- Significant sums of money and time will be saved through the data centric alternative.
- The system centric alternative appears to be simpler, but will quickly become a bureaucratic swamp and quicksand.
- The system centric alternative is ideal for point-to-point interfaces, which, are to be avoided at all costs.
- The system centric alternative will most likely lead to inflexible, brittle and unintegrated solutions.
- If XML is ever to be successful it will only be after success in data management.
- The data centric alternative is essential for large, enterprise-wide environments.
- The system centric alternative is just another form of stove-pipe, point-to-point solutions.
- If XML naming exists outside a well-ordered data standardization effort, data will be more disparate than it is now.
- XML tags based on data standardization will aid search engines and change management.
- The system centric alternative always appears to be quick-and-dirty. The first is always false and the second is always true.



3.2.1.4 GAO Study Recommendations

In the GAO Report, United States General Accounting Office (GAO) Report to the Chairman, Committee on Governmental Affairs, U.S. Senate, Electronic Government: Challenges to Effective Adoption of the Extensible Markup Language, April 2002, GAO-02-327, there are three quotes that are especially instructive as they directly point to the need for enterprise architectures and data standardization prior to any XML benefits:

- XML's greatest benefits accrue when organizations, such as government agencies, use standard data exchange procedures and agree on standard data definitions and structures. Effectively using XML as a means to share data among disparate systems across the federal government will require agencies to conform to a range of technical and business standards.
- XML's larger promise of facilitating data exchange across broad domains (such as an entire agency, a group of agencies, or a set of external stakeholders and client organizations) will be difficult to realize until critical data elements and structures are identified and standardized across entire agencies and communities of interest.
- This task of identifying and standardizing critical data elements and structures is part of an agency's larger task of developing an enterprise architecture. Well-planned enterprise architectures can also promote the adoption of flexible implementations that can be modified in the future to conform to commercial standards that become established over time. Thus, agency enterprise architectures are key building blocks to effective government wide adoption of XML

3.2.2 Data Asset Catalogs

A key component of net-centricity is the existence of metadata catalogs. With these catalogs, data assets can be both visible and accessible. It is obviously critical that these metadata catalogs be founded on quality engineered, non-redundant, and shared-metadata. Else, it will be virtually impossible to discern one data asset from another. If a data asset is XML based, then there could be hundreds of millions of cataloged data assets. Finding the right data asset will be like finding one precise snow flake in a blizzard. Consequently, the metadata associated with every data asset must be automatically generated and automatically stored in the metadata catalogs.

Without high quality, automatic generation, it is unlikely that a consistent set of metadata will be created about one data asset that is all related to the discovery metadata about another data asset that is supposed to be similar. Metadata sharing can only meaningfully occur at the shared data segment (conceptual data model) and at the shared data element (via ISO Standard 11179) level.

Unless and until the physical data models, from which the metadata must be generated, are interrelated to logical data model abstractions, which in turn are interrelated to shared data



segments (conceptual data models), which in turn are related to ISO 11179 based shared data elements, then metadata catalogs cannot be usefully generated in any automated manner.

Manual generation will just not scale. If there are one hundred thousand data assets (and that would likely be a very low number) and each requires a 30 minute effort for the meaningful creation of the metadata catalogs, then that would require in excess of 25 staff YEARS for the generation. If these assets change in a way that requires metadata catalog regeneration, then the burden for metadata tagging could be unbearable.

3.2.3 ISO 11179 Based Shared Data Elements

The DDDS, because of its faulty engineering, had an infinitely expanding workload. It fell by its own weight. If the DDDS had, however, concentrated on shared data elements according to ISO Standard 11179 for data element metadata, then the workload would have not only been profoundly smaller; but it would have also reached closure. There is, however, a significant quantity of data elements in the DDDS that should be "mined" and brought into an ISO 11179 Data Element form.

For example, instead of having 27 different Supply Condition Codes, there would be just one. And, instead of having about 20 different "cargo weights", there would be only one. The value of a mined set of ISO 11179 based shared data elements is that these would form the business fact semantic templates for shared data segments within a conceptual data model. The ISO 11179 based shared data elements might also be mined for use in creating XML tags. Finally, the ISO 11179 based shared data elements might also be related to taxonomies so that the creation could be largely automated. Is this approach a return to the "bad old past?" No, it's a return to what the DDDS was intended to be, but with the engineering flaws removed that were identified in the internal MITRE and GAO studies.

The ISO standard, 11179 for data element metadata, is a comprehensive standard that has been implemented in a number of Federal Agencies, foreign countries, and is required by the Federal Enterprise Architecture's data reference model.

A shared data element created according to ISO 11179 must never be confused with an attribute of an entity or column of a table. Both an attribute and a column are deployments of the semantics of a ISO 11179 based shared data element. Thus, they are not data elements; rather, they are uses of data elements.

It is critical to understand the process that was performed by DoD in the 1990s that led to the unacceptable DDDS result. Figure 6 shows four layers of data models. These are functional area (subjects, entities, and attributes), technology dependent (schema, tables and columns), DBMS dependent (DBMS schema, DBMS table, and DBMS column), and business application system (business information systems, application views, and view columns).



It is critically important to know how to “read” this diagram. The arrows within the specific horizontal boxes mean “one-to-many” That is, for each subject there is zero, one or more entities. And for each entity there is zero, one or more attributes. This is a data modeling diagram convention. In Figure 4, the arrows at the end of the boxes that go to the DDDS “database” mean that the attributes that were extracted were sent to and stored within the DDDS database.

There is also a relationship among the four layers. That is, between Conceptual (L1), Logical (L2), Physical (L3), and View (L4). Layer 1, the functional or conceptual layer represents the high level data models about functional areas such as human resources, logistics, or finance.

Level 2 represents the data models of specific databases There are often multiple database data models about each functional area. Hence there is a one-to-many relationship between Level 1 and Level 2.

Similarly, there may be multiple physical data models for each logical data model. This depends on the power of the computers, the operating systems, and the database management system that runs the database. Thus, again there is a one-to-many relationship between Level 2 and level 3.

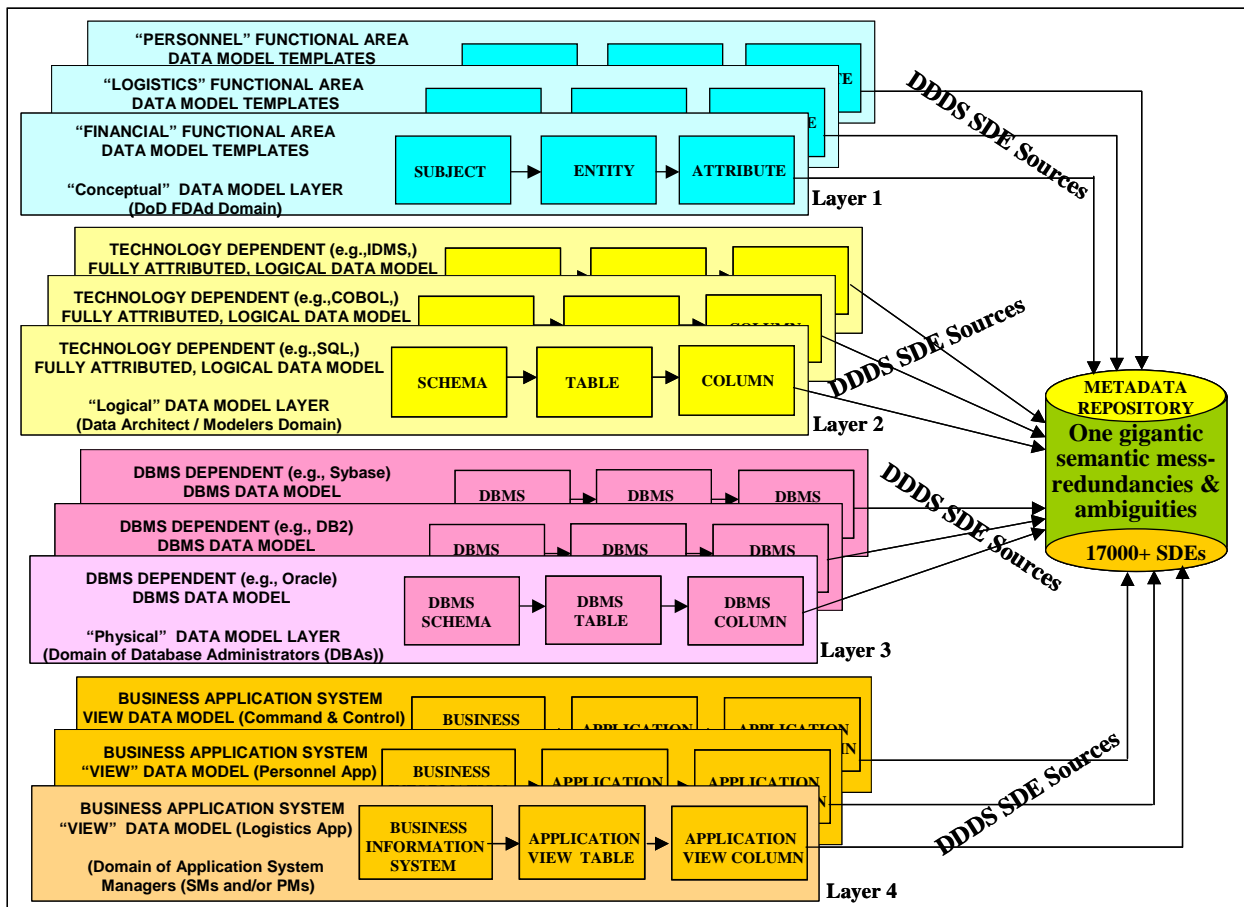


Fig 6 Type 2

Figure 4. DoD Process of finding and creating DDDS “data elements.”



Finally, Level 4 represents the view data model layer. This is the interface layer between the physical database and the information system. There are commonly many, many view data models for each physical data model. Hence a one-to-many relationship between Level 3 and Level 4.

The consequence of all this is that for each functional data model there could be multiple logical data models, each of which in turn could have multiple physical data models, and each of which could have many view data models.

At the outset of the DDDS effort there was a “data call” for data elements. Each of these four layers of abstraction had data modelers. Each responded to the data call. The L1 functional users provided an unduplicated set of attributes, having renamed these as data elements. The L2 Technology users did the same and provided their columns as data elements. The L3 DBMS dependent users also did the same and these DBMS columns were provided as data elements. Finally, the L4 modelers provided their view columns which were re-anointed as data.

This massive quantity of data elements, many of them essentially the same, had slightly different names, data types, and value domains. Since the DDDS effort essentially defined a data element as a unitary fact within a specific context, all these essentially similar business facts were not merged and duplicates dropped. Rather, all were retained. Hence the 27 different supply condition codes.

What resulted from the DDDS effort was a repository of 17,000 standard data elements and another 40,000 or so non-standard data elements with no end in sight to the data standardization effort. This infinitely expanding effort was stopped in 2003. Under the DDDS model, every column from every database that was being created across the DoD represented a standard data element. Suppose, as is the case of the U.S. Navy there are 75,000 systems, and suppose that there are 50,000 databases across these systems. Suppose further that each database contains 100 tables and 15 columns for each table. Given that the U.S. Navy is 20% of DoD, then there would have been 375,000,000 data elements to “standardize.” That quantity is however a valid count of database table columns, and it is for that very reason why names, value domains, definitions, abbreviations, and XML tags must all be automated to the maximum extent possible.

Clearly the task of creating 375,000,000 standard data elements (actually database table columns) represented a task that could not be accomplished in any practical way especially since there should be no more than 10,000 real context independent data elements in all of DoD. What was missing? It was the ISO 11179 concept of data element and its relationship to attributes, columns, DBMS columns, and view columns. This concept would have allowed a practically accomplishable layer of ISO 11179 based shared data elements across the DoD. That likely was the ultimate goal in any event.

Figure 5 depicts the missing part. It consists of an upper level of concepts, conceptual value domains, data element concepts and discrete value domains that together all lead to the definition of a data element.



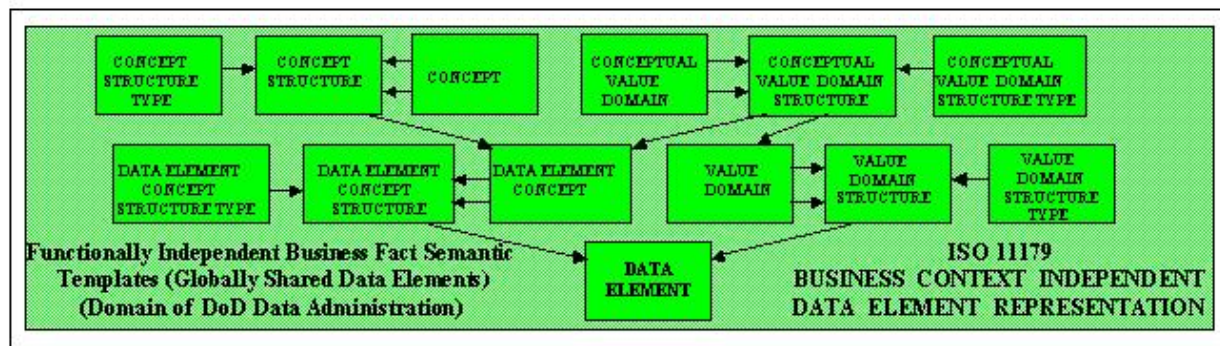


Figure 5. ISO 11179 essential data model for data elements.

In this diagram, the arrows all have a “one-to-many” meaning. Thus, for each concept structure type there can be zero, one or more concept structures. The double set of arrows from concept to concept structure are special. One of the arrows means “contains.” The other arrow means “is contained in.” This two-arrow convention is common for a type of data structure used in “parts and assemblies.”

The way to read a “parts and assemblies” data model diagram is as follows. A part (that is, a concept) can contain zero, one or more subparts (that’s the top arrow). In essence, the collection of the subparts is an assembly of these subparts. The assembly is represented by the box, Concept structure. Now, each of the assembly parts is itself a part. Hence it is represented as distinct part, that is, a different concept. Each part contained within an assembly results in a separate concept structure record. The collection of all such records constitutes the parts that are in that assembly. The bottom arrow represents the assemblies in which a part is contained. So, in this Figure 7, there are four different parts-assemblies data structures. These four data structures are an essential component of the ISO 11179 standard. Not only are these data structures in the standard, they are essential to properly model and thus standardize shared data elements in an efficient and effective manner. Using ISO 11179, as completely specified in the standard, is a critical success factor to data standardization.

Within the context of ISO 11179, a data element contains a number of contextual and then definitional parts. The contextual parts are the concepts and conceptual value domains which are the semantic sources for data element concepts. Then, when a specific value domain is specified from within a conceptual value domain, and when it is associated with a particular data element concept, then a data element is formed.

Collectively, these constructs, shared data elements, called ISO 11179 data elements, comprise business fact semantic templates for attributes that describe characteristics of entities. Entities are well bounded collections of business facts that are called attributes. Not only must all entity attributes be founded on ISO 11179 based shared data elements, the very nature of ISO 11179 based shared data elements enable them to be employed as the business fact basis of multiple attributes within one entity or across multiple entities. The value representation of ISO 11179



based shared data elements and, by inference, entity attributes are defined by the allocated value domains.

Figure 6 presents a simple example of an ISO 11179 data element, Supply Item Resource Quantity. The concept from which this data element was developed is Materiel Resource. It is about materiel resources that ISO 11179 based shared data elements, that is, unitary business facts are to be created. Materiel Resource is then combined with a conceptual value domain, for example, Logistics Measure. Together, the Materiel Resource and Logistics Measure then lead to the concept supporting the element, that is, Logistics Item Balance. Now, there may well be many other types of data element concepts like Reorder Quantity, Maximum Allowed On-hand, and the like. There may also be different conceptual value domains such as Physical Measures and Financial Measures. These are all very powerful concepts and enable contexts to be provided to data element concepts. A given data element concept, Logistics Item Balance, when associated with a specific type of value domain within a conceptual value domain, leads to the shared data element, Supply Item Resource Quantity. Figure 7 provides an additional example, but in the area of human resources.

ISO 11179, however, is not the complete answer to the data standardization effort. That is because the problem of having an almost “infinite” quantity of ISO 11179 based shared data elements would still continue, but only a smaller “infinite” number if an ISO 11179 data element

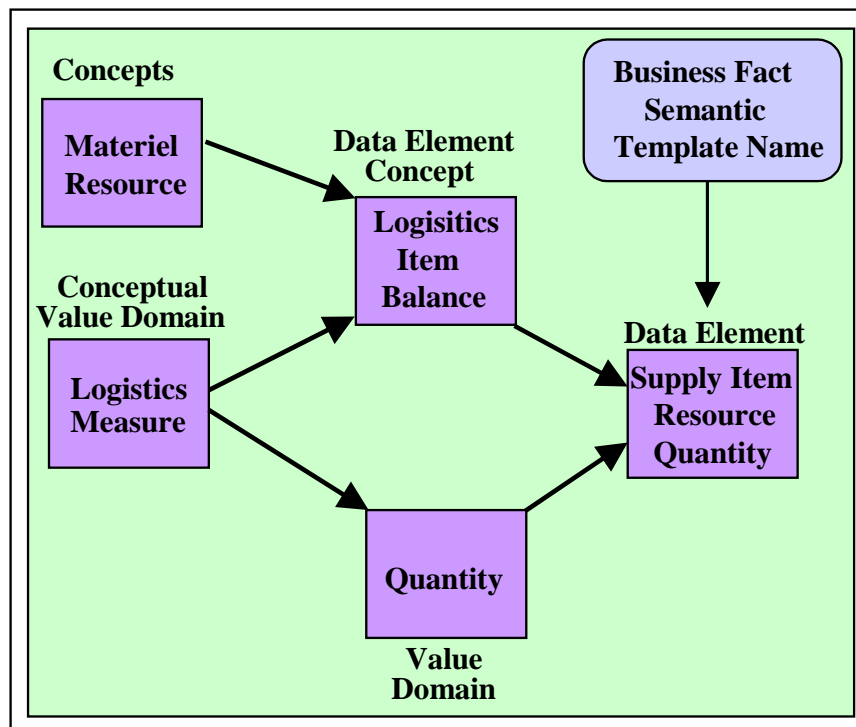


Figure 6. Example of an ISO 11179 data element, Supply Item Resource Quantity



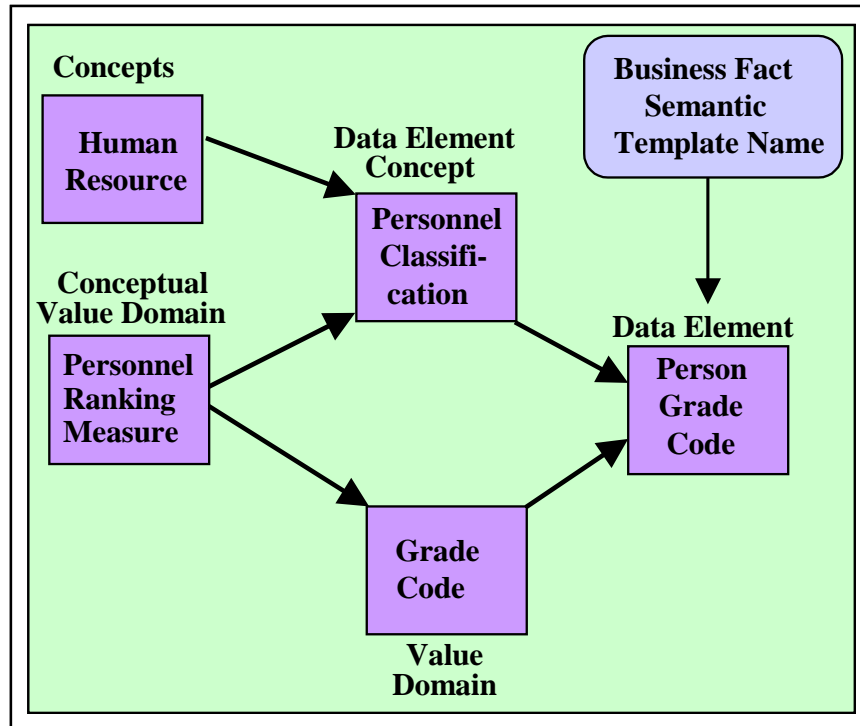


Figure 7. Example of an ISO 11179 data element, Person Grade Code

were perceived to be just an “elegant” attribute, column, DBMS column, or view column. The real solution revolves around combining Figure 6 and 7 as illustrated in Figure 8. This figure shows that the ISO 11179 data element metadata is the top layer of a five layer set of metadata to manage data models. ISO 11179 based shared data elements are semantic templates for attributes, which, in turn, along with subjects and entities are data model templates for database schemas (schema, table, column), which are then the logical data model layers that support one or more physical data models. Together, these five layers provide the ability to manage data models within communities of interest and across communities of interest.

Figure 9 shows a complete example of the five layers for logistics. A number of very critical points need to be conveyed. In this particular example, the ISO 11179 based shared data element, Supply Item Resource Quantity, has been defined and is available as a unitary fact template for use in creating data models.

The first layer of data model is the conceptual data model and within that data model there is some entity that has supply item resource quantity as the template for the two attributes, material inventory quantity, and material item inventory quantity. Note that the names are different not only with respect to the ISO 11179 based shared data element but also from each other. That’s perfectly acceptable because what these two attributes really are is Supply Item Resource Quantity.



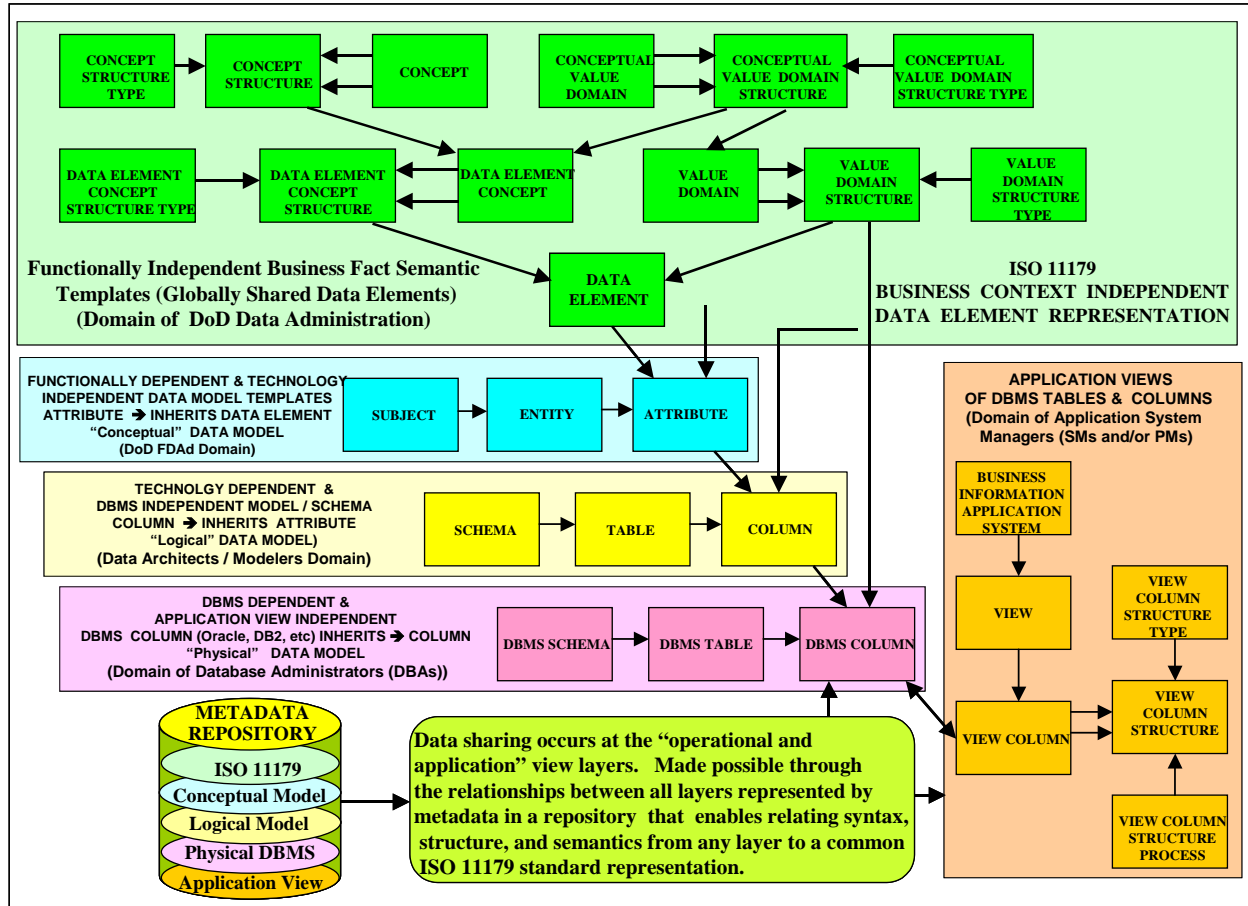


Figure 8. Comprehensive metadata model to manage data models

Further, note that the systems employing this ISO 11179 based shared data element example are from the Navy and the Army. The significance of this is that the Army and Navy are each free to employ their own particular business area languages/dialect terms in naming their specified, implemented, operational, and view deployments of ISO 11179 data elements. And, since legacy systems have already done that, this metadata strategy will accommodate those existing differences and not require those legacy schemas to be re-engineered at most likely great expense and disruption to business operations

The next level of data model, the logical data model for the Army has a table within which the column is Mat_Inv_Qty. For the Navy, the column is Mat_Itm_Inv-Q. Not only are these names different one from the other, they are also different from their “parent” attribute names. Again this is perfectly acceptable because these names are all mapped, thus traceable back to their source. The physical data model level also shows a similar pattern as does the View Model layer that supports the business information system.



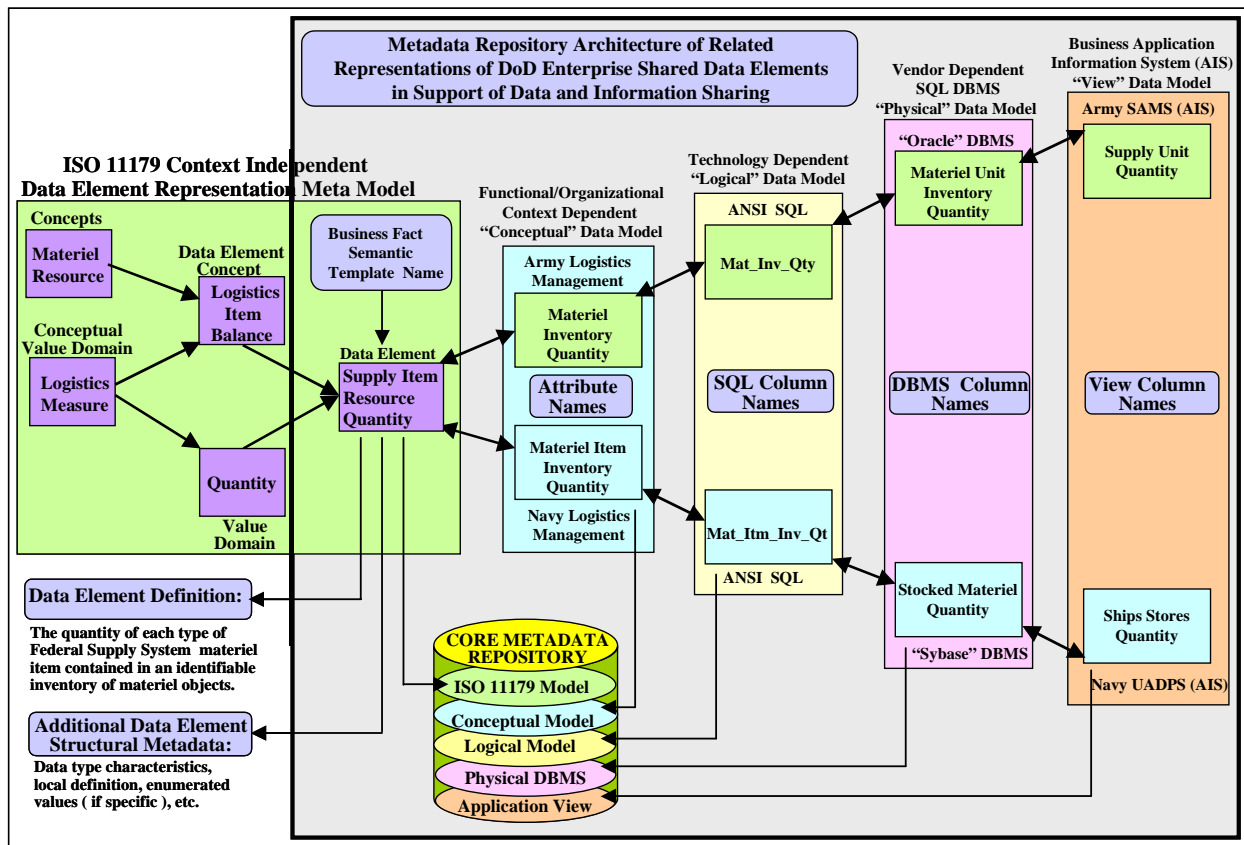


Figure 9. Logistics example from the five layers of metadata for data model management

This approach works quite acceptably for legacy systems. Their database schemas already exist at the view and physical data model levels. These schemas and views are imported into the metadata repository. Given that the ISO 11179 and the conceptual data model layer of shared data segments have been populated, then the task that remains is the construction of the logical data model layer that maps one to the other.

Once these are built and mapped, then there is semantic mapping across legacy systems, between legacy systems of different physical data models and their common logical data model, and from conceptual data model templates of standard data structures across all logical and physical data models. Finally, there exists mapping from concepts and conceptual value domains to all data element concepts, and from data element concepts to all ISO 11179 based shared data elements, and finally, from all ISO 11179 based shared data elements to all uses of that shared data element across all uses within information systems within the DoD inventory of information systems. Given that this ultimately was the intended goal of the "8320.1" standardization in the early 1990s, it can now be fulfilled. First, because of the ISO 11179 data element standard and second, because of the integration of the ISO 11179 data element standard with layers of data models. In short, there can now be smart, well engineered data standardization. Clearly this is what was intended.



The following is another example of the use of these interrelated sets of data models (data element, conceptual, logical, physical, and view). Suppose there is a 11179 data element named, Schedule Date. This is a data element which represents the association of the data element concept, Schedule, and a specific value domain, Date. Date has been assigned to identify the class of its values. Schedule in turn is particular business fact characteristic of the concept, Event. Similarly, Date is a particular form of a conceptual value domain sequence.

The concept, Event, then can have various data element concepts such as Schedule's name, starting and ending times, and description. Similarly, there are a number of different conceptual value domains such as geography, temporal constructs, precision indicators, and physical characteristics. The conceptual value domain for a temporal construct may be times and dates. Thus, the data element concept, Schedule, might obtain its semantics both from event concept and the temporal conceptual value domain. Conceptual value domains, in turn have more precise value domains such as dates, start times, end times, and duration times. In this specific example, the data element concept Schedule is associated with the specific value representation, date. Together then, they comprise the Schedule Date. Schedule Date is thus, the data element. Its common business name is Schedule Date. Other common business names, such as Telephone Numbers, or Reservation Numbers, or even Social Security Numbers all exist even though their value domains may not be numbers. It is not uncommon, for example, for Reservation Numbers to be a set letters and numbers. Notwithstanding, Reservation Number is the common business name.

While this may seem like a long and tortured path to the simple fact, Schedule Date, it is precisely because of the upper layers of ISO standard 11179 that searches for all things that are associated with event concepts, or associated with temporal conceptual value domains, can then be found. Simply put, this is all about classification schemes, taxonomies, and/or ontologies that are necessary to bring order and discipline to what might first appear to be an almost uncountable quantity of facts.

Given that the definition of Schedule Date is, "the date when a list of things arranged in a specified ordered sequence is created," it can be associated with particular classes of objects, for example, aircraft flights or surgical operations. In these cases, when the data element is associated with the database table, Flight, the data element becomes the Flight Schedule Date column. Similarly, when the data element is associated with the database table, Surgery, it becomes the Surgery Schedule Date.

The data element has now been bound into a particular context. The indicators of that binding are the modifier terms, Flight and Surgery. In this particular example, the modifiers are the table names. There may also be other modifiers such as "estimated" or "actual." There may also be a more localized definition that created for the column by associating the definition of Surgery or Flight, and estimated or actual with the data element, Schedule Date.

In this particular example, the following columns are all possible from this one data element.



- Flight Estimated Schedule Date
- Flight Actual Schedule Date
- Surgery Estimated Schedule Date
- Surgery Actual Schedule Date

The process is simply the making of assemblies of semantics from pre-defined semantic parts. And in this particular case, the one data element is Schedule Date, the two tables are Flight, and Surgery, and the two modifiers are estimated and actual. By having precise understandings, names, abbreviations, and value domains, and the like for each of these parts, names, abbreviations, and definitions can be automatically generated. So too can be metadata catalogs and XML data element or attribute tags. Importantly too, elementary facts associated with:

- Estimated,
- Actual,
- Date,
- Surgery,
- Flight,
- Events, and
- Temporal concepts

can all be searched for and found. But most importantly, if legacy system column names are associated with these ISO 11179 enterprise-level names and name parts then legacy system columns can be associated with each other regardless of their legacy names and (within certain boundaries) data types.

The Defense Data Dictionary System (DDDS) was a main component of the 8320 Data Standardization program. This repository of DoD standard data elements represents hundreds of millions of dollars of research and analysis. However, the current set of standardized data elements in the DDDS is at the level of database columns, not ISO 11179 data elements. Thus, what would be found in the DDDS would not be schedule date, but:

- Flight Estimated Schedule Date
- Flight Actual Schedule Date
- Surgery Estimated Schedule Date
- Surgery Actual Schedule Date

This is the principal reason why there are so many redundancies in the DDDS. For example, the 27 distinct Supply Condition Codes, or the 20 Cargo Weights. These redundancies were created when entities and/or attributes were independently created in different business functions that actually represented common business concepts and facts that were relevant across multiple business contexts. Because all these DDDS data elements were created independently of each other, the structure and semantics of the DDDS data elements while conceptually similar were almost always different from each other, and thus, incompatible with each other for sharing data.



What was missing from the “8320.1” effort was the ISO 11179 standard for data element metadata. It would have brought these upper level constructs through which these independently created columns from the different functional business areas could have been seen to have been essentially the same. If the ISO 11179 superstructure had been present, then even though columns were named differently, and even though their physical characteristics and value domains may not be exactly the same, they would have been known to have been derived from the same ISO 11179 data elements.

The fact that the “8320.1” effort was not perfect does not mean that the concept of pursuing enterprise information interoperability through the use of standardized data elements should be discarded. Rather, it just means that the process must be reconfigured into one that is ISO 11179 based. That step will enable the shared data element discovery effort to reach closure and be complete.

The vast majority of the body of data element metadata represented by the 17,000+ standardized data elements and their descriptive metadata in the current DDDS is good metadata that has embedded within it the information required to identify the context independent data elements, semantic modifiers, and value domains that should be the subject of the DDDS. This very valuable metadata merely needs to be “mined” and transformed into a ISO 11179 data element format.

If shared data elements are created via the ISO 11179 approach then, there will be smart, well engineered data standardization across the enterprise. That is because these business fact semantic templates, that is, the ISO 11179 shared data elements, will then be available for the construction and/or mapping of data models that are, in turn, the source of semantics for the logical data models, that are in turn, the source for physical data models, views, and XML schemas.

This smart, well engineered data standardization effort will be different from the “old” 8320 data standardization approach in the following five ways:

First, the “old” 8320 approach to data standardization mandated exact conformance in all logical and physical data models to the “8320.1” imposed data standard. Not only did that approach fail, it was doomed from its very first day. This ISO 11179 based shared data element approach, in stark contrast, only requires ISO 11179 data elements along with shared data segment mapping. Thus, data names can be different, and within best practice guidelines, both data types and value domains can be different.

Second, the “old” 8320 approach to data standardization was engineered to have only one real level, the physical level. This ISO 11179 based shared data element approach, again, in stark contrast, has four application independent levels: ISO 11179 Data Elements, Conceptual, Logical, and Physical Levels. Thus, the ISO 11179 Data Element Level supports “where-used” analyses across all conceptual logical, and physical levels.



Third, the “old” 8320 approach to data standardization enabled understanding-based data interoperability if and only if there was strict conformance to data standardization at the physical level. Once, more, in stark contrast, this ISO 11179 based shared data element approach enables automatic mapping across physical data models if they are drawn from a common logical data model, or if the physical data models employ (through logical data models) conceptual data model shared data segments, or finally if the physical data model table columns are drawn from the ISO 11179 data elements.

Fourth, the “old” 8320 approach to data standardization enabled a reasonable-effort development of logical data models if and only if there was strict conformance to data standardization at the physical level. This ISO 11179 based shared data element approach, if supported by CASE-like repositories that enable importing of multiple physical data models, then physical data model (or physical table, or physical column) “promotion” to a logical level along with automatic mapping back to the various physical data models. This enables an easier development of logical data models in case there had been no conformance to the “old” 8320. If there was conformance, then an almost automatic generation of logical data models. In addition this approach is then enhanced by the four previous differences.

Fifth, the “old” 8320 approach to data standardization was not engineered to support automatic generation of metadata catalog entries, nor the automatic generation of XML Schemas, nor the automatic generation of XML wrapped data, nor the automatic generation of XSLTs, that is, XML Stylesheet Language Transformations, all of which are critical to understanding-based data interoperability. This approach, if properly configured within a federation of CASE-like metadata repositories, supports all the understanding-based data interoperability success factors.

3.2.4 Shared Data Segments

The section above has concentrated mainly on the configuration of ISO 11179 based shared data elements into a form that makes them reusable in an effective and efficient manner. This is only the first part of what has to be accomplished. The second part is shared data segments.

Shared data segments are deployments of predefined collections of ISO 11179 based shared data elements within the context of an entity within a specific subject. For example, a shared data segment might be Person Name with the subject, Human Resources. It would consist of Honorific, First Name, Middle Name, Last Name, and Suffix. This shared data segment would thus be a data model template that could be used within a database schema table.

Shared data segments can make database design largely a matter of assembling predefined collections of predetermined parts (i.e., the shared data segments). Because shared data segments are composed of ISO 11179 based shared data elements, data standardization is automatic. So too are definitions, value domains, and abbreviations. This is an especially powerful capability/feature for sharing authoritative reference table value sets and descriptions for maintaining synchronized implementation of enterprise level code sets.



Shared data segments are shown as conceptual data models in Figure 10. Figure 12 shows the cross reference between 9 different logical data model schemas (C-03, ... , PR-31) and conceptual data model subject areas (e.g. Logistics management) and entities contained in that subject area (e.g. organization). Clearly, these data model templates are employed over and over throughout all the data models. While they essentially represent the same data they have different table names, different column names, and sometimes different value domains and data types. Notwithstanding these differences, they the same essential data segments are represented. Identifying and mapping from shared data segments (Country within Logistics Operations) to the five different tables within the different logical data model schemas just represents good data management.

Conceptual Data Model		Logical Data Model Schemas & Tables								
Subject Area	Entity	C-03	C3-12	ES-07	ES-08	LG-06	LG-23	LG-28	PR-22	PR-31
Logistics Management	Organization	X	X	X	X	X		X		X
Personnel Management	Person	X			X		X	X	X	X
Logistics Operations	Country	X	X	X	X	X				
Logistics Operations	Location	X			X					
Management Administration	Task	X							X	
Property Management	Facility			X	X					
Logistics Planning	Plan		X							X
Environmental Management	Guidance		X		X					
Property Management	Geolocation		X		X	X				

Figure 10. Mapping of Shared Data Segments to Database Tables within Schemas



3.2.5 Levels of Shared Metadata

A key reason why the DDDS was not successful was because there seemed to be a requirement to have consensus across the entirety of the DoD. That was, and is, clearly not possible. What, therefore, should be achieved is consensus within domains of metadata sharing. Thus, if there is no need to share data outside a program manager (PM), then that is the maximum level of consensus that must be achieved for shared metadata. This is also the case for a community of interest, or a domain, or a mission area. Not coincidentally, this strategy is an essential component of net-centricity. That is, to implement data standardization at the COI level but not at the enterprise level. Then, when or if there is a need to share data across COI boundaries data standards can be created for just that subset of shared data.

If there is data standardization at the COI level, it does not necessarily mean that every information system and database that is part of the COI must redesign their information systems and databases to conform. It's just that these information systems must map their data semantics to the COI consensus semantics. Mapping might be a little as "renames" clauses or possibly the creation of a suite of extract-transform-load programs. If the later, then the processes would be just to and from the COI consensus data model and common data system.

Another alternative is that any given information system build a shared data portal that is continuously refreshed. This way other information systems would capture data from these shared data portals. This last approach would be suitable for major COTS software vendors.

As a consequence, shared metadata will most likely consist of a federation of metadata repositories, rather than a single monolithic metadata repository, that are available to various levels of communities of interest from PM through mission areas and through cross communities of interest. As the extent of sharing grows, then so too must the span of consensus for the shared metadata and the content of the metadata within the shared metadata repository.

Figure 11 through 15 illustrates the unfolding of data model layers to maximize the quantity of reuse.



ISO 11179 based shared data element modeling, depicted in Figure 11, involves determining the "facts" and the semantics of the facts that are to be represented within conceptual, logical, physical, and view models. A shared data element should be defined to meet the metadata requirements of the part 3 of the ISO Standard 11179 for data element metadata. The main purpose of an enterprise's collection of ISO 11179 based shared data elements is the ability to standardize the semantics of fact definition and/or instantiation within the conceptual, logical, physical, or view data models.

Each ISO 11179 data element is a shared data element that will be employed, many times, to infuse semantics within the attributes of entities within conceptual data models.

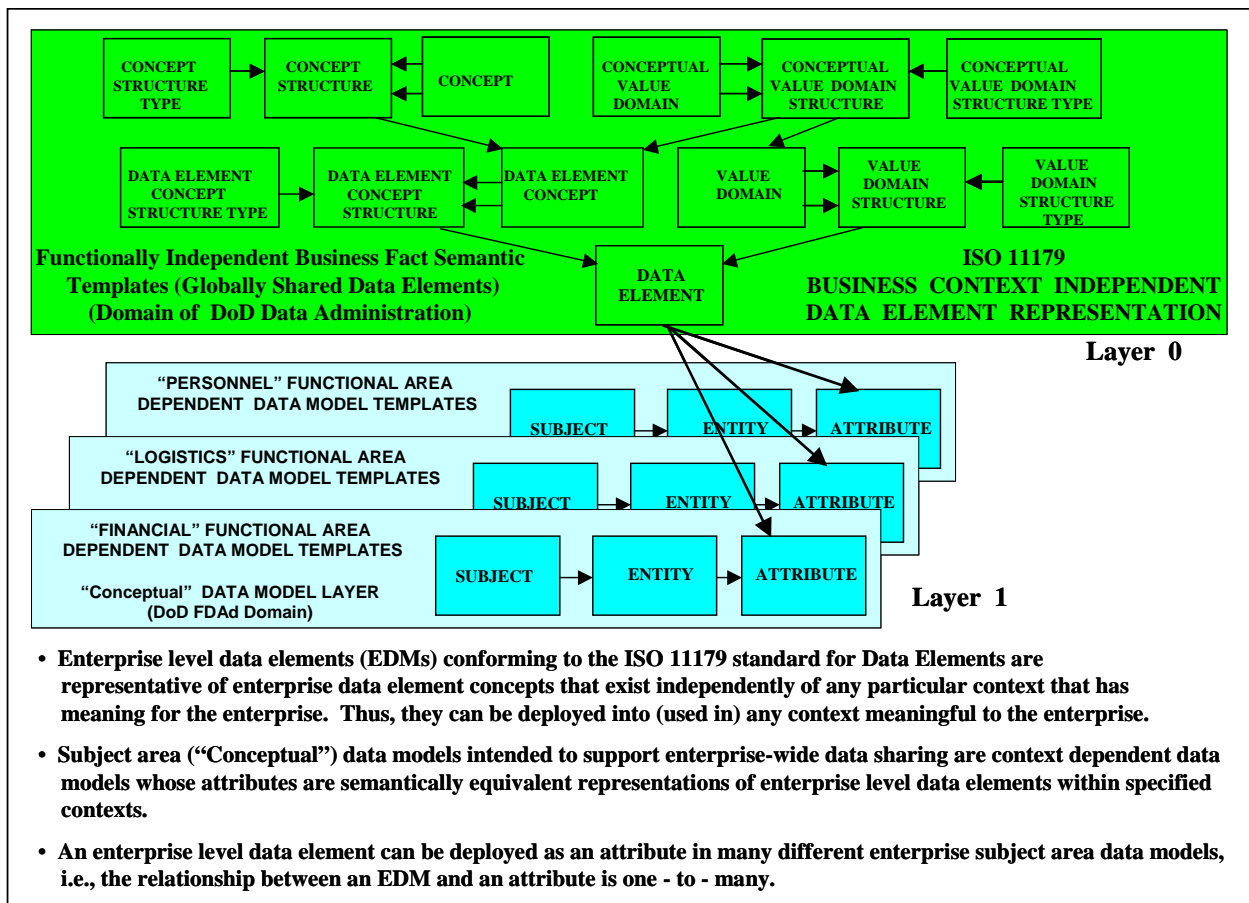


Figure 11. ISO 11179 Data Element Layer and Conceptual Data Model Layer



Conceptual data modeling, depicted in Figure 12, involves the creation of collections of attributes within entities. Each collection should represent the necessary and sufficient data in support the entity, which, in turn, exists within the domain of a subject. Entities contain attributes, which should represent single valued atomic facts and which should be mapped to the ISO 11179 based shared data elements. Each entity should also contain one or more attributes, which represent the entity's primary key, that, if instantiated would result in a unique instance of the entity. Entities within the collection may be related to each other. The most common form of relationship is the one-to-many but there are other kinds of relationships: one to one, many to many, and inferential.

Entities within the conceptual data models are the shared data segments. These are in turn, employed within logical database tables to standardize logical database data structures.

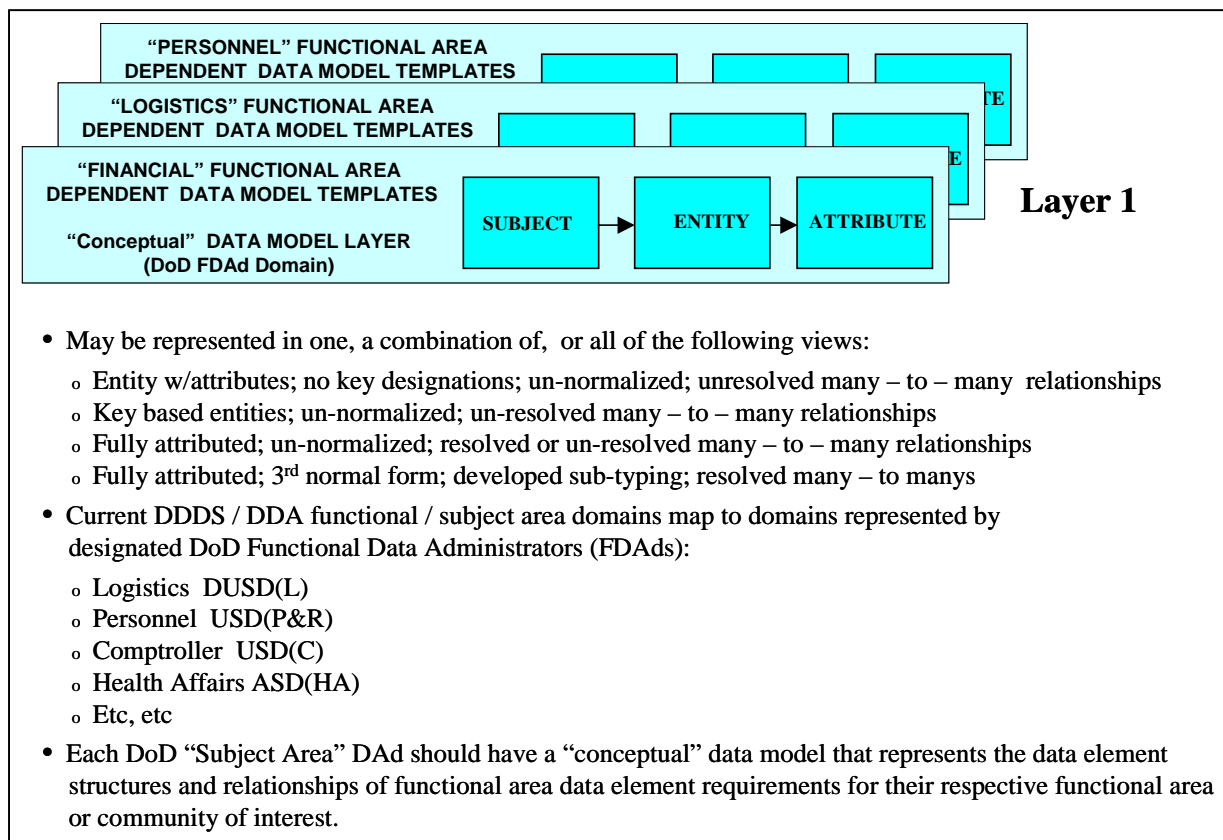


Figure 12. Characteristics of the Conceptual Data Model Layer.



Logical data modeling, depicted in Figure 13, involves creating a collection of tables, columns within the tables, and relationships among the tables. While a conceptual data model represents the data requirements of a subject, the logical data model represents the collection of tables of columns, which, in turn, map back to the attributes from the entities of one or more subjects of a functional area of an enterprise or business. A logical data model often spans multiple subject areas within a database's data model.

Thus, the logical data model revolves around the needs of the business. There are many logical databases within a business. Each logical database table should contain one or more columns, represent the table's primary key, that, if instantiated would result in a unique instance of the table's data. Tables within the logical database may be related to each other. The most common form of relationship is the one-to-many. While there are other kinds of relationships: one to one, many to many, and inferential, the only additional relationship that can be explicitly expressed in ANSI SQL is one-to-one. It is generally recommended that logical data models be normalized, usually up to 3rd Normal Form (NF). The normalization process ensures that there will not be any anomalies introduced into the database as the data is inserted, updated, and deleted.

Physical data modeling, depicted in Figure 14, almost always involves the transformation of a

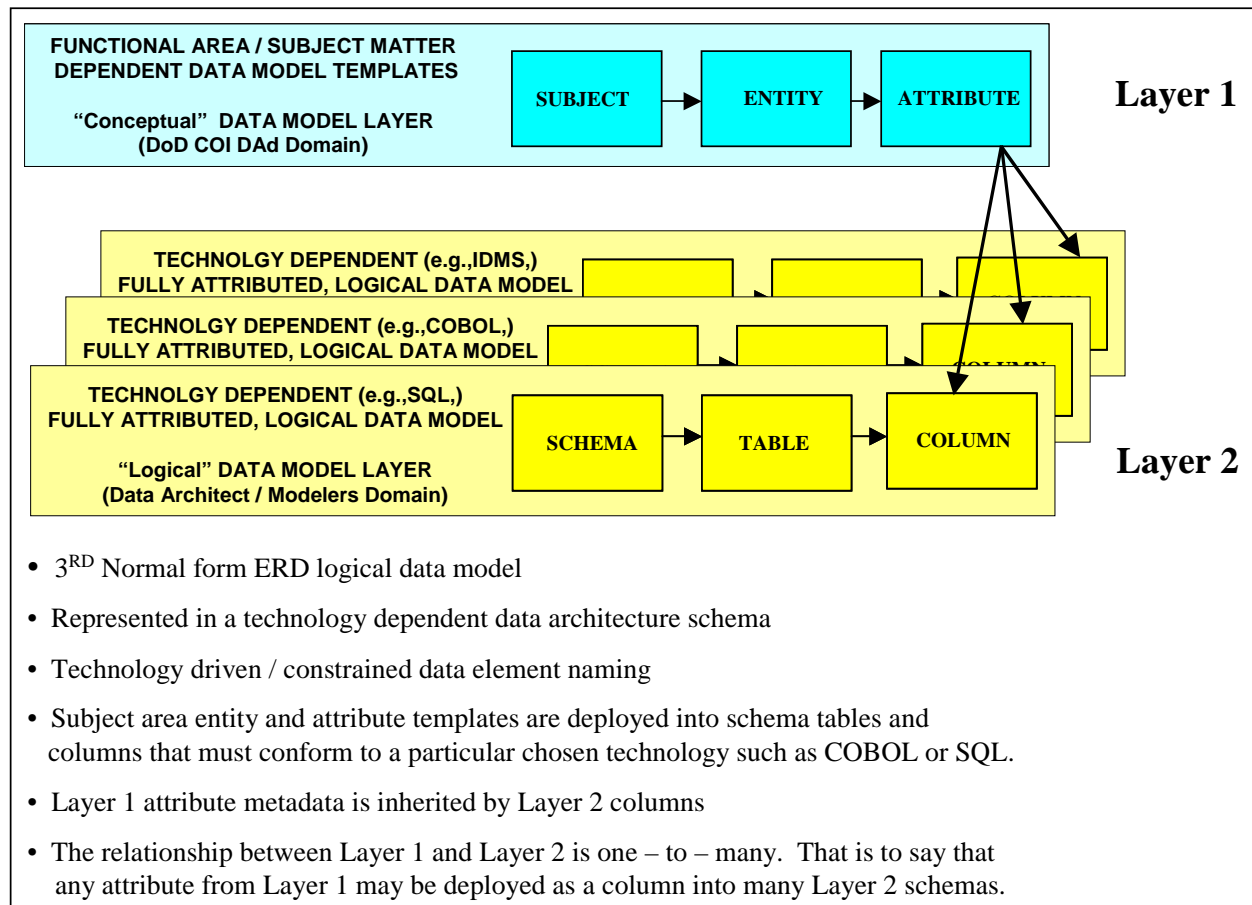


Figure 13. Characteristics of the Logical Data Model Layer



logical data model into a physical database design that is required by a specific DBMS for a specific class of applications.

There may be multiple physical data model designs for a single logical database if the database is to be deployed in a federated or distributed manner across multiple DBMSs and classes of applications. During the transformation process the data types may be more refined, column lengths changed, methods bound to the DBMS and even made more precise, and table designs may even be changed to meet the performance requirements of the DBMS, the volume of database data, and the velocity of different classes of transactions that must be supported. Performance optimization characteristics include data usage patterns (transaction update frequency) - and physical DBMS architectural considerations (distributed DBMS) are also considered during this transformation.

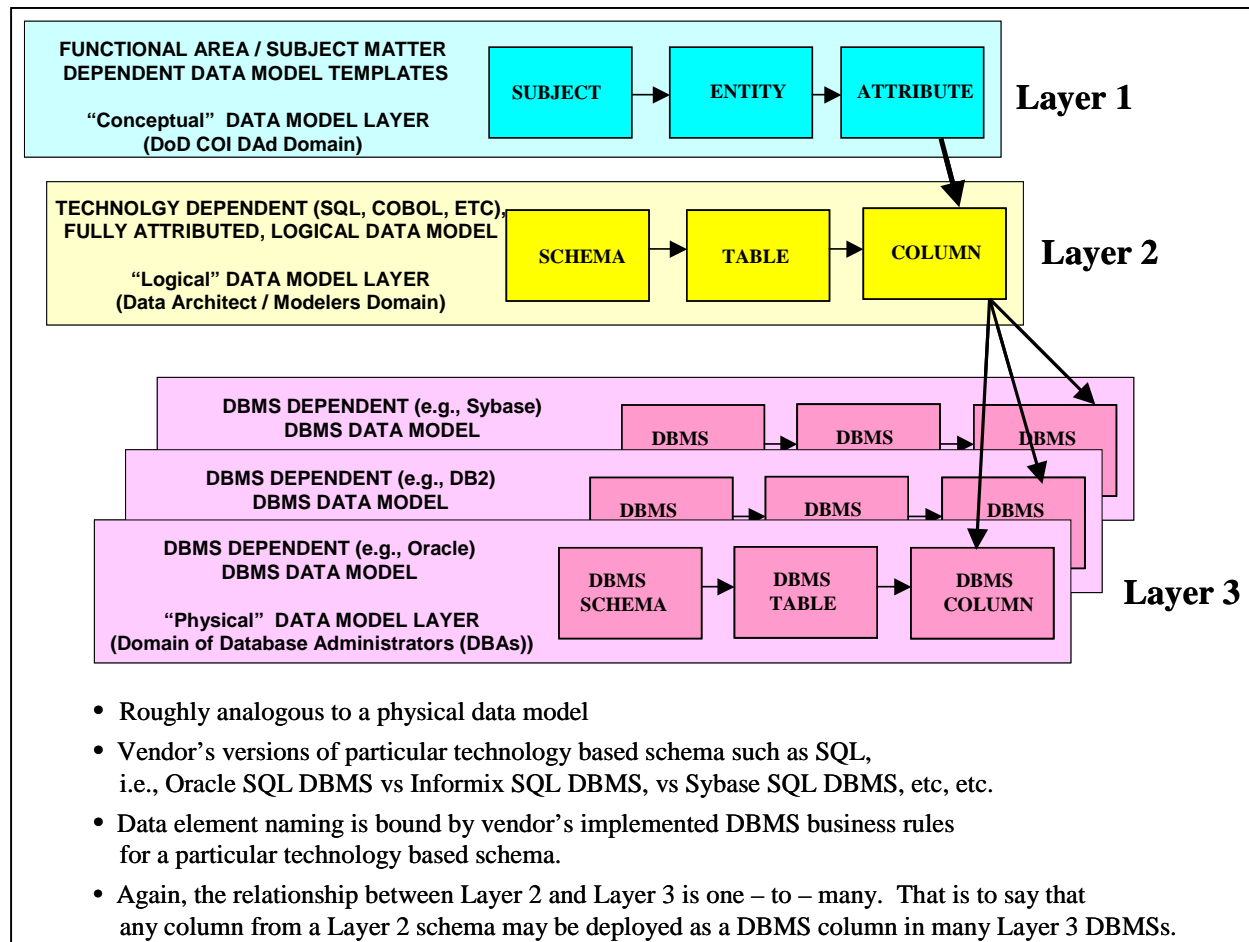


Figure 14. Characteristics of the Physical Data Model Layer



View modeling, depicted in Figure 15, is the final data model layer. It is the process of creating the data interface between an application program and a database (via the DBMS). Views may contain column re-naming clauses, the creation of derived or computed data, the expression of nested database actions, the summarization of data, data grouping, and the like. The goal of view modeling is to cause the DBMS to select, group and reformat data on behalf of the application program and to only present the application program with the consequence of these view actions.

This enables the application programs to have little or no data access logic, thus freeing the program even more from having to deal with the database's design. In short, views significantly enhance the application program's data independence.

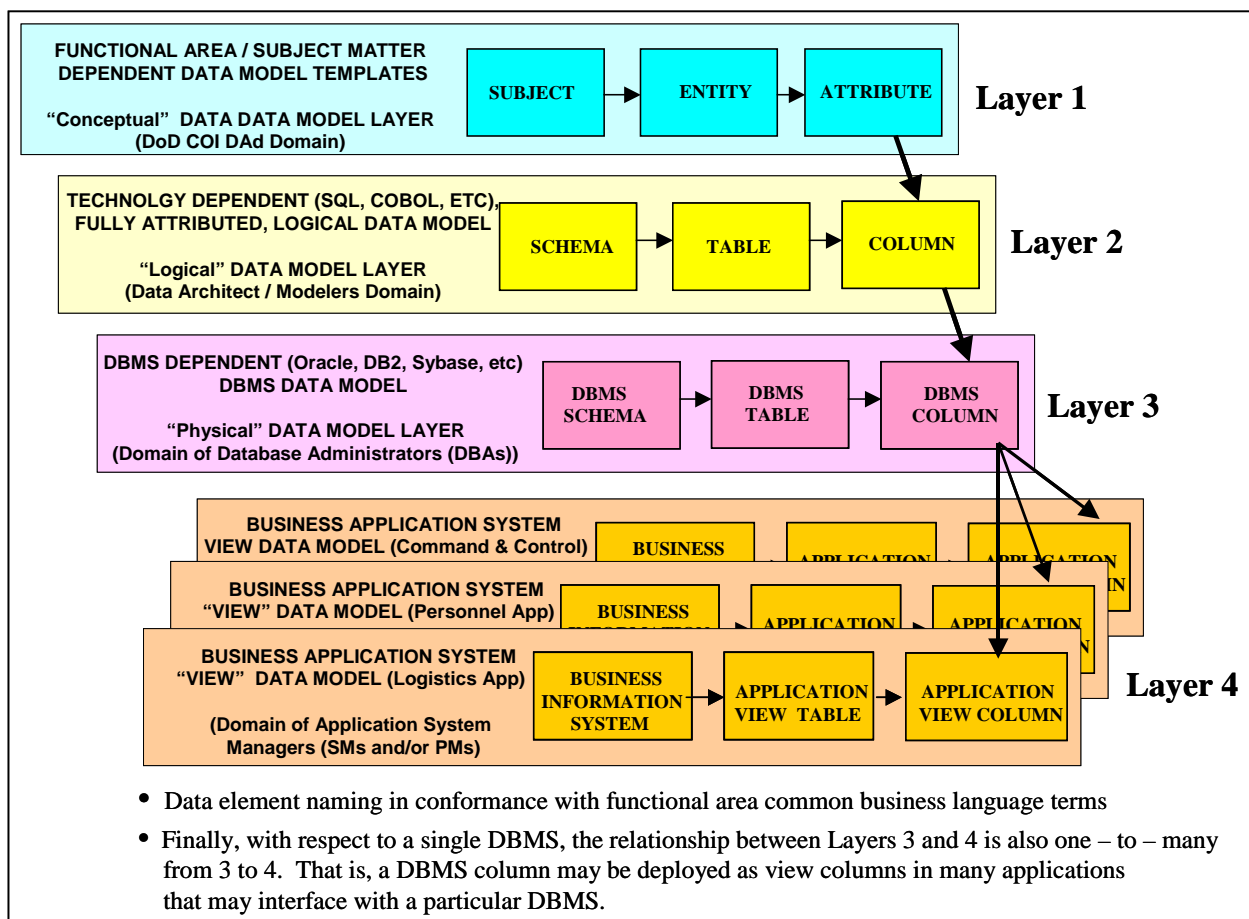


Figure 15. Characteristics of the View Model Layer.



3.2.6 Critical Concepts in this Shared Metadata Modeling Environment

Lest it be missed, an examination of the conceptual, logical, physical, and view data models in Figures 11-15 clearly show that the relationship among these models is **not** from subject to schema to DBMS schema. Nor is it from entity to table to DBMS table. Rather, it is from ISO 11179 based shared data element to attribute to column to DBMS column. This is a very critical distinction. These models, conceptual, logical, and physical are thus **not** transformations one into the other. Rather they are completely separate, distinct data models, one mapped to the other. As a result, ISO 11179 based shared data elements are templates for attributes and optionally, columns of tables. Conceptual data model entities may map to one logical data model table, or multiple conceptual data model entities may map to one table. Finally, a conceptual data model entity may map to multiple tables within the same schema. Logical data models may be the source for the more than one physical database DBMS schema as each such DBMS schema and associated DBMS tables and DBMS columns may require different physical designs to then accommodate different performance requirements.

Additionally, given that this process is employed to create IESSs, that is data models that represent shared data across or within communities, there may be a logical database that consists solely of the tables and columns from all the physical databases that share data.

These distinctions are critical to the understanding of this approach because it is due to these distinctions that an efficient and effective data management environment can be created and maintained.

3.2.7 Shared Metadata Environments

It is not enough to just have consensus based metadata. The metadata must exist within a shared environment. That is, federated set of metadata repositories. This is critical for the following two reasons:

- Programs need to know in close to real-time what other programs are creating, modifying, or deleting. If they merely know after the fact then programs may have to forego deployment because other programs may be critically affected. Time and money will have been lost because of late or incomplete notification. More critical still, is that different feeder/user programs plans may be crippled.
- Programs need to be able to “use” the metadata from other metadata development efforts. This will automatically increase productivity and quality. It will enable an almost automatic generation of XML Schemas, XML wrapped data, standard value domains, and common business rules.



3.2.8 Federated Metadata Repository Environments

There is an absolute requirement for federating metadata repositories because it would be practically impossible to have one such repository either within the enterprise, within a community of interest, or across collaborative communities of interest. The DoD Data Dictionary System (DDDS) and the Defense Data Architecture are two classic examples of failed attempts. The value from these failed attempts, however is that the essential and non-redundant data elements from the DDDS should reside as ISO 11179 based shared data elements within enterprises's metadata repository. Similarly, the DDA's shared data segments should reside within an enterprise's metadata repository as shared data segments. If there are multiple such repositories there still needs to be a strategy to intersect metadata so as to minimize semantic conflicts and uncontrolled redundance. This task is accomplished through any of the following, or combinations of the following ways:

- Having an enterprise-wide ISO 11179 based shared data element model.
- Having an enterprise-wide shared data segments.
- Having a formally declared federation-based relationship strategy among ISO 11179 data element, conceptual, logical or physical data models.

3.2.9 Quality Data Management in a Net-centric Environment

If quality data management is implemented within an environment of distributed and federated metadata repositories that are harmonized within communities of interest with their associated information systems, and across communities of interest, then, while not all the problems cited in this section are automatically eliminated, many are significantly minimized. The approach presented herein enables:

- The automatic generation and/or management of names, abbreviations, and definitions.
- The management of value domains and the mapping among value domains for current and historical data.
- The ability to pre-define allowed or disallowed combinations of data items based on data types or other semantics.
- The generation of XML schemas and all DDMS metadata.
- The ability to determine where and in what way data is employed throughout the enterprise to then support either new system development or legacy system evolution and maintenance.



When the data management environment is in place and functioning, then business rules (the processes that ensure integrity and correctness across the enterprise, communities of interest, or collaborations of communities of interest and associated information systems and databases) can be identified, configured, verified, and managed.

In short, without quality, efficient and smart data management practices in place as the key foundation stones of net-centricity, its accomplishment may fall significantly short of expectations. And XML, a key ingredient of the net-centric approach and the main-stay mechanism of non-proprietary data interchanged, can only be successful if and only if quality data management has first been institutionalized.

3.3 Name Management

Likely a key reason why there were so many different DDDS data elements was because business facts that were the same were named differently, while some same-named business facts were different. How to tell? How to sort it out? How to resolve it? And finally, how to manage it? This is a “cat that has to be belled.” Unless and until it is, data standardization, while critical, and while necessary for any enterprise-wide data sharing will likely fail. And if data standardization fails, enterprise-wide data sharing will never even start to succeed.

Name management, if successful is not the sole requirement for enterprise-wide data integration success. Successful also must be value domain management, XML schema generation and evolution management, and finally, metadata catalog generation management. None of these four can succeed in isolation. DDDS is a clear example of such a failure. It was not engineered properly and was tried in isolation. Failure was the only possible outcome. Name management is treated in this section. Value domain, XML generation, and metadata catalog generation are address in subsequent sections. Common across all four is the absolute requirement for a well engineered metadata repository that has an enterprise-wide view, not a stove-pipe database or system view.

The root cause of our name management problem began in the 1970s. That is, the three-part paradigm of *prime word*, *modifier[s]*, and *class word* to satisfy the needs of naming fields within the contained data structures of COBOL programs. While quite suitable for COBOL, once this three-part paradigm was brought into the database world, five distinct problems immediately arose. These were:

- That data elements were mistakenly seen as table columns, screen cells, entity attributes, or report fields.
- That data elements don't have names
- That prime word, modifier[s] and class word *are* part of the data element's name



- That modifiers are from a single homogeneous set
- That there is only a choice of one class word

Collectively, these errors have cost hundreds of millions of dollars in wasted data standardization efforts. The solution to these five problems is a business fact standardization strategy that is based on a meta attribute classification scheme that both supports a business domain, common business name, modifier classes and subclasses, multiple class word classes and subclasses, and also preserves these classifications within the metadata repository so that relevant semantics are immediately retrievable.

Data elements, are thus context independent business fact semantic templates. Context dependent business facts are cells in a screen, fields in a file, columns in a table, or variables in a program. While most of the meta attributes necessary to describe the context independent and dependent business facts are the same, there are some differences. Table 5 tabulates the differences.

This paper only presents the overall strategy and brief examples that addresses name management. Other papers and books from Whitemarsh describe and illustrate the solution to the name management problem in detail.

Meta Attributes for Context Independent and Dependent Business Facts		
Associated Meta Attribute or Semantic Part	Context Independent Data Element	Context Dependent: Table Column, Screen Cell, Entity Attribute, Report Field, etc
Business Domain	Yes	No (but yes by inheritance)
Common Business Name	Yes	Yes
Entity or Table	No	Yes
Semantic Modifiers	No	Yes
Data Use Modifiers	Yes	Yes
Generated Policy Basis Description	Yes	Yes
Computer Data Type	No	Yes
Data Structure	Yes	Yes
Required Uniqueness	No	Yes
Relationship Function	No	Yes

Table 5. Meta attributes for context independent and dependent business facts.



The metadata model that must minimally be present in the metadata repository to have successful business fact name management is provide in Figure 16.

In this metadata model, business fact names exist at seven distinct locations. These names all represent some part of the total semantics of the business fact. Each name, for example the name associated with a view column is in fact the sum total of the semantics associated its parent names, that is, from DBMS column all the way to the name associated with concept and conceptual value domain.

The actual name, however is really a string of words. Where do those words come from? Are they controlled? Do they have precise meanings, precise abbreviations, and the like? The answer is of course, yes. The part of the metadata model in Figure 16 that is labeled meta category value <something> contains all these word lists, relationships among the word lists, their meanings and their abbreviations. The meta entities, e.g., Data Element Concept & Meta Category Value represent the association of some meta category value word with some data element concept. Through carefully constructed software that honors hierarchy, the names for data element concept all the way through view column are able to be automatically constructed.

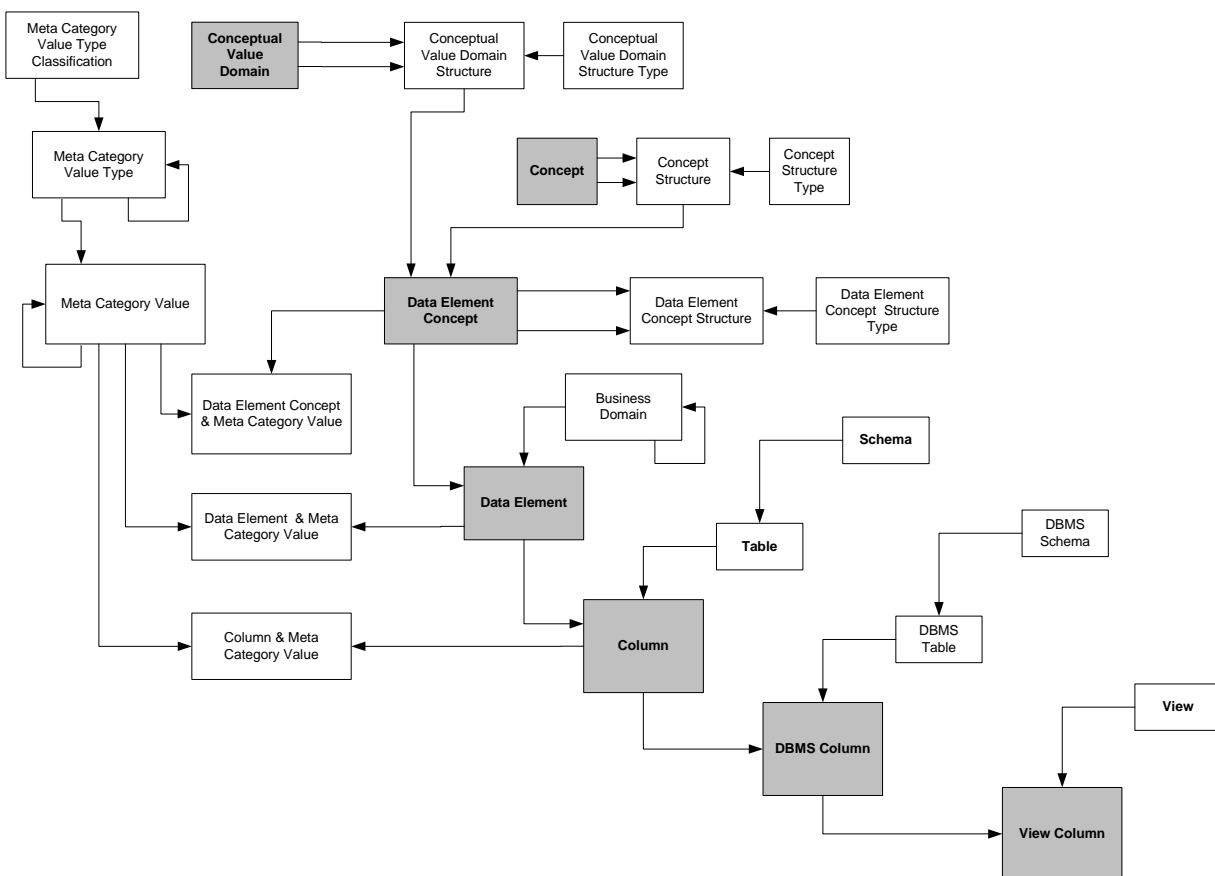


Figure 16. Metadata data model to manage business fact names.



model is almost automatically interoperable with the original data model because of all the maintained relationships. Now, what the figure precisely shows is that for the data element concept, Organizational Identifier there is a data element called Government Institution Identifier. This data element concepts maps on the the attribute social security number from within the entity employee. The entity employee and its attribute, social security number maps onto three different database columns at the logical data model (IDM) level. And the one highlighted column at the logical level maps onto a DBMS column at the physical data model (ODM) level. Even if all the names were different across all these levels, the relationships among these business facts would be maintained. In short, name management.

Does all this happen by magic and take no work? Of course not. It takes clear thinking, good analysis, and work. What is provided here is the metadata management infrastructure and ability to accomplish this work much fast with a higher degree of quality. Creating that second database, for example, took only hours versus weeks or months. Creating the names for all the database columns was automatic as was the creation of the column definitions.

Attribute: Hourly Wage		
Inherited Meta Attribute Class		Inherited Hierarchy of Values as Appropriate
Data element domain		Compensation
Business domain		Human Resources
Common Business Name		Salary
Data Element		Salary
Data use modifiers	Role	Dollars
	Data Type	Compensation
	Units	Business Fact
Semantic modifiers	Geography	North American
	Temporal	Current Year
	Accuracy	Final
	Organization	Central Office
Entity Subject area		Human Resources
Entity		Employee

Table 6. Semantics associated with the attribute, Hourly Wage.



Data Hierarchy

	Data Element:	Data Ele Concept:	Business Domain:
Name	Name Government Institution Identifier An identification mechanism that is created by an appropriate government agency that is assigned to	Data Ele Concept Organizational Identifier The identifier that is associated with a organizational instrument. Examples	Business Domain Organizations Data elements generally related to organizations and their characteristics.

Attribute	Entity	Subject	Null Allowed
Distributor_Name	Distributor	Organization	No
Distributor_Name	Store	Organization	Yes
Employee_Hire_Date	Employee	Person	No
Employee_Name	Employee	Person	No
Employee_Social_Security	Employee	Person	No

An identification mechanism that is created by an appropriate government
 An employee is a person who is employed by the enterprise. Included is
 Someone of interest to the movie rental corporation. Persons may either be

Column Name	Table	Schema	Prec	Scale	Data Type	Null
EMPLOYEE_SOCIAL_SECURITY_#	EMPLOYEE	Movies Original Data Capt	0	0	INTEGER	Yes
Employee_Social_Security	Employee	Movie Sales	11	0	CHARACTER	No
Employee_Social_Security	Movie_Rental_Record	Movie Sales	11	0	CHARACTER	No

An identification mechanism that is created
 An employee is a person who is employed by the
 The movie original data capture schema
 The NUMBER datatype stores fixed numbers.

DBMS Column Name	DBMS Table Name	DBMS	Database	Prec	Scale	Data Type	Null
EMPLOYEE_SOCIAL_SECURITY	EMPLOYEE	ORACLE	Movies Original Data		0	INTEGER	Y

An identification mechanism that is created
 An employee is a person who is
 Oracle is a SQL based, general
 The Movies Original Data
 The NUMBER datatype stores fixed numbers.

Close

Figure 18. Name management in action.

3.4 Value Domain Management

Value domain management has only recently surfaced as a independent discipline from business fact management. Value domains are almost always associated exclusively within the context of a single data element, or attribute, column, or DBMS column. The ISO standard, 11179, for data element metadata caused value domains to be seen on their own and to then be associated as appropriate to data elements, attributes, columns, or DBMS columns. Figure 19 illustrates a simplified data model for value domain management.

In this model, all the value domains are managed centrally and then are assigned to either data elements, or columns, or DBMS columns. In all the cases, if a value domain is not assigned to a column, for example, then the value domain associated with the data element is inherited. Proper software causes the value domains to be assigned in nested manner. Thus, a broader value



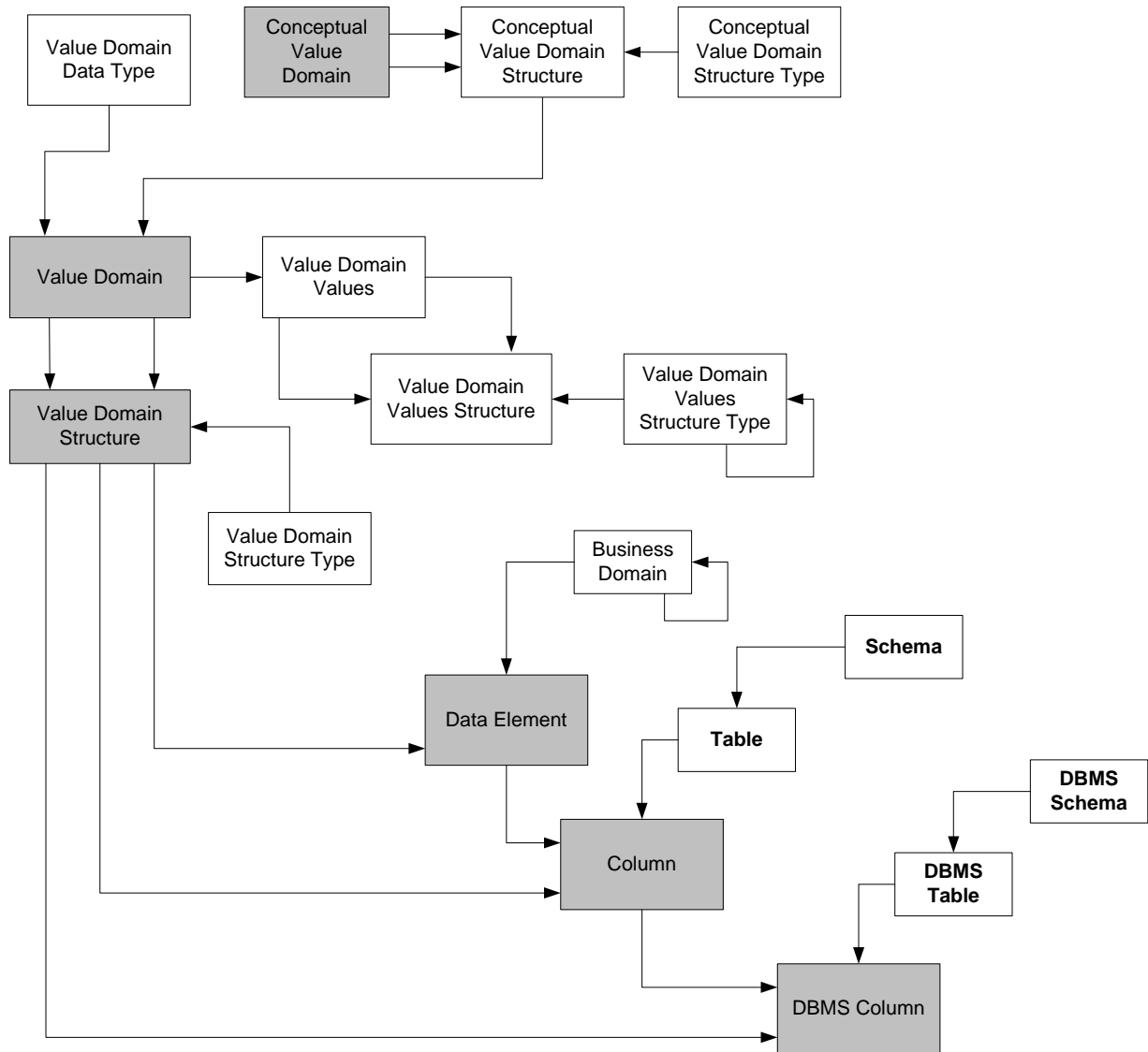


Figure 19. Metadata model for value domain management.

domain cannot be assigned to a attribute than is assigned to a data element. Value domains may also be assigned exclusively to a DBMS column but not a column. Value domains are all typed. And those data types must correspond appropriately to the data types assigned to a column or a DBMS column. Finally, value domains value sets can be complex as would be the case of say an invoice line item number which might consist of the concatenated values of Customer Number, and Invoice Number, and Line Item Number. Whether such a scheme is good data modeling is a different issue.

An even more complex issue arises when value domain values need to be mapped, one value set to another. For example, over time, the value domains of say, gender, might change. The value



domain sets would then need to be mapped one value to another. 0 = male = F, or 1 = female = F. These complexities too need to be handled in the metadata repository.

A more complex example of value domain mappings is provided in Figure 20. What is unique about this model is that the value domain values are never stored in the actual databases. Stored instead are the foreign key values to value domain value table that contains the actual values. This enables the changing of value domain values without massive database updating.

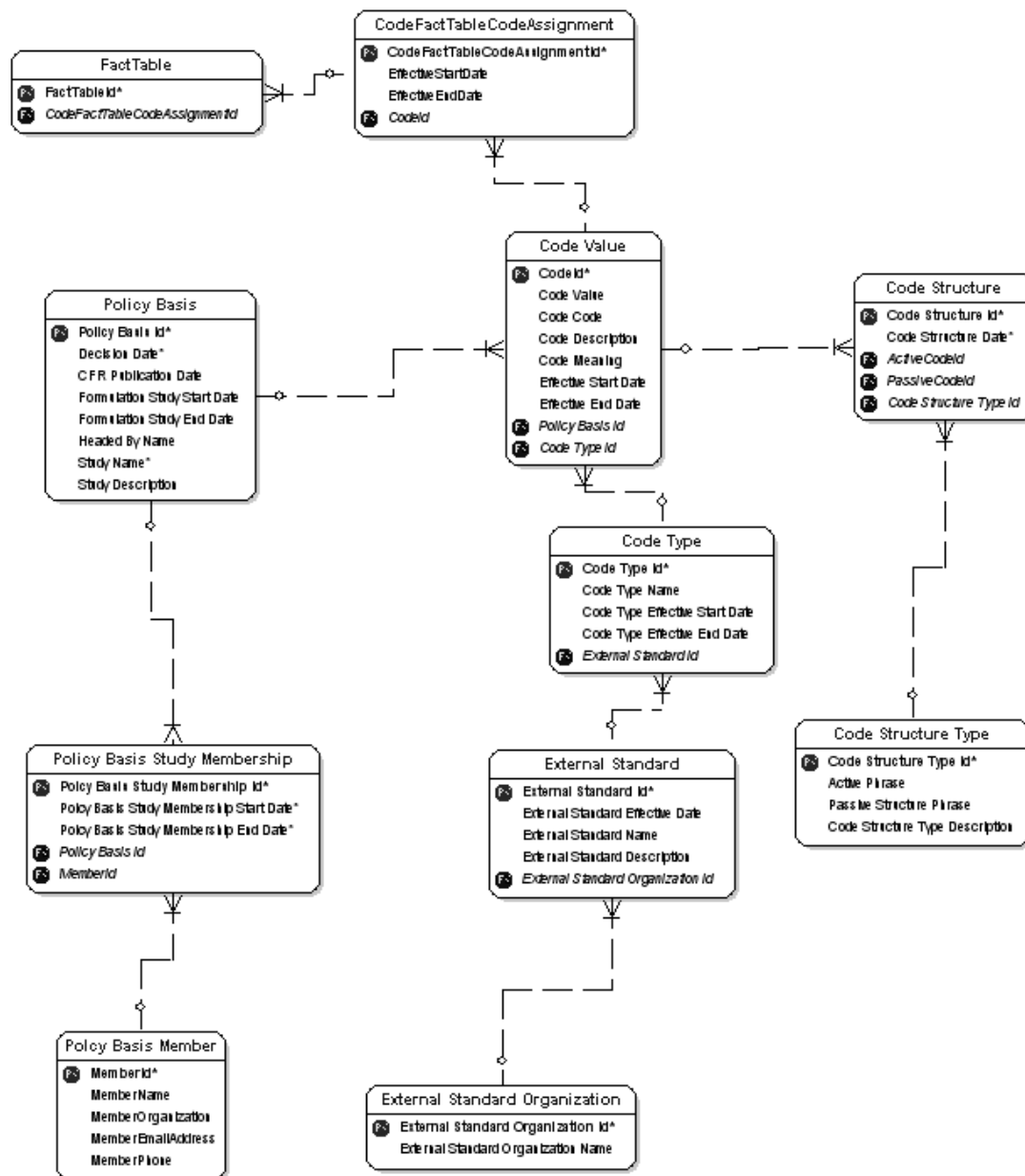


Figure 20. Complex metadata model for value domain mappings.



In this value domain mappings strategy, the actual tables that contain the value domains are similar to those in Figure 19. Different about this diagram is the administrative infrastructure for standardizing value domains and for knowing when value domain values start and end.

Regardless of which value domain management meta model is employed, the key point is that value domains must be managed on an enterprise-wide basis so that they can be employed consistently in which ever data elements, attributes, columns, or DBMS columns are appropriate. If different columns, for example represent the same value domain but have different value domain values, then the mappings should be stored in the metadata repository so that information system that access both databases can perform automatic value domain value transformations.

3.5 XML Schema Management

As stated above, an XML schema is equivalent to an SQL view. That is, the SQL schema is the data schematic of a transaction. Similarly, the SQL view is the definition of the format of the data extracted from a database via the SQL language, or received from an external source into an SQL database.

The ANSI NCITS H2 Technical Committee on Database Languages has recently completed the specification for the interface between XML and SQL. The specification is in two major parts, composing (constructing an XML data stream) and shredding (decomposing an XML transaction into a SQL table). In all cases, the XML transaction comes from the “real” database, thus, from the physical data model.

The main point here is that within a large enterprise there could well be hundreds-of-thousands to possibly millions of XML schemas. Because of the potentially vast number, the only practical way for them to be constructed, understood, and then mapped to information systems is through an extension to a well ordered metadata repository that contains data models. Figure 21 illustrates a strategy for automating the generation of XML schemas and for automatically composing the XML data streams. Conversely, this same model supports automatic shredding and mapping of XML schemas and data streams to information systems that would read the XML data streams.

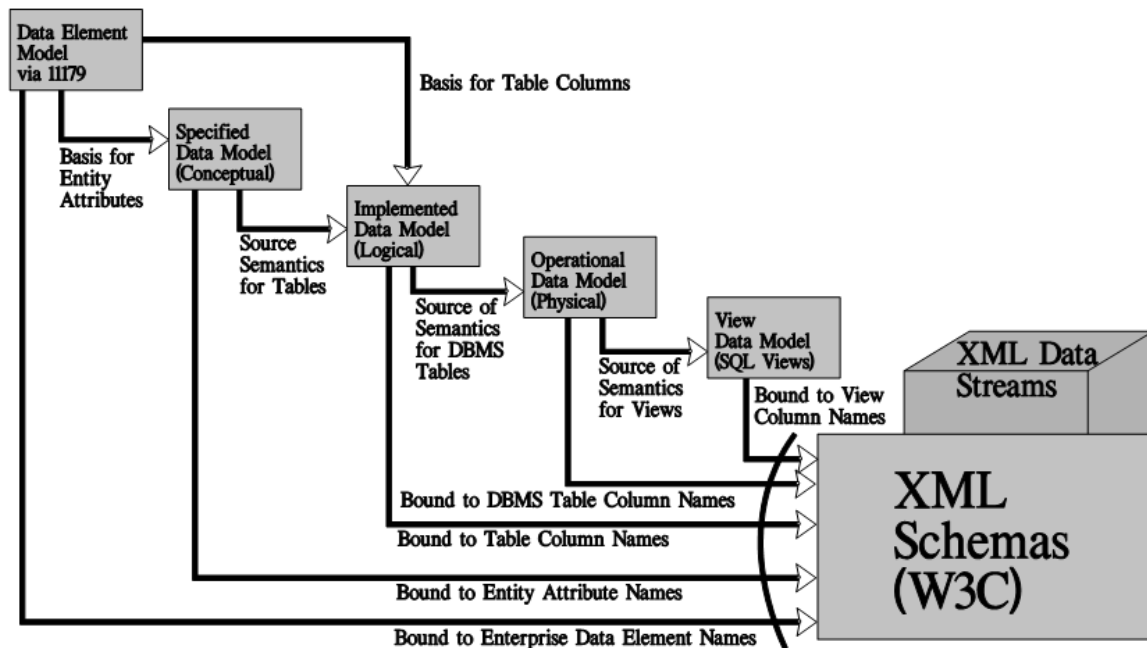
Key to this strategy is both the five abstraction layers for data modeling as shown in Figure 20. Data elements are mapped to attributes to columns, to DBMS columns, and then to either view columns or XLM Schema elements. Key also to the strategy is the inclusion of an enterprise identifier for every data element, and for every DBMS table within every DBMS column. If a data element is employed multiple times within a single DBMS table, then a sequence number would have to be employed as well. What this inclusion accomplishes is the creation of a unique identity that can then be processed by the metadata repository system to know the exact DBMS column, its containing table, and all the higher levels of data abstraction. If an XML schema is created, these unique enterprise identifiers are also included. When the XML schema is then received (along with the XLM data stream), the contained enterprise identifier leads to the



correct semantics for the data from the composing system. The receiving system can, through pre-existing mappings between physical and logical data models deduce the semantics of the XML element (or XML attribute) and make the correct transformation assumptions.

Through this infrastructure of metadata models and mappings to XML schemas, the process of generating, mapping, and understanding XML schemas can be largely automated.

In addition to automatic XML schema creation along with XML transaction composition and shredding, another key benefit is that every XML schema potentially has a vast store of metadata that can describe its every detail. That is because there would be a mapping between every XML element and a DBMS column, thus providing every XML element all the metadata information associated with the DBMS column. Table 6 is an example of that extensive, automatically



XML Schema "generalization" is directly related to the "level" of name binding

Figure 21. Metadata model infrastructure to support automatic XML schema construction.

inherited metadata.



3.6 Metadata Catalog Management

The last item that needs to be addressed in the creation of an efficient and effective shared data environment is the automatic creation of the metadata catalog. Figure 22 illustrated the current components of the U.S. Department of Defense Metadata Registry. Each ellipse represents a targeted data store of metadata within the registry. These data stores, however are intended to be independently populated without any integrating infrastructure across them. This lack of integration across these registry data stores be problematic (possibly even fatal) because there will be no method of determining the complete semantics for any component or across classes of registry products. The five layers of metadata data models, shown as rectangles within this figure act as an integrating mechanism.

Even more importantly than that, this metadata data model environment enable the automatic

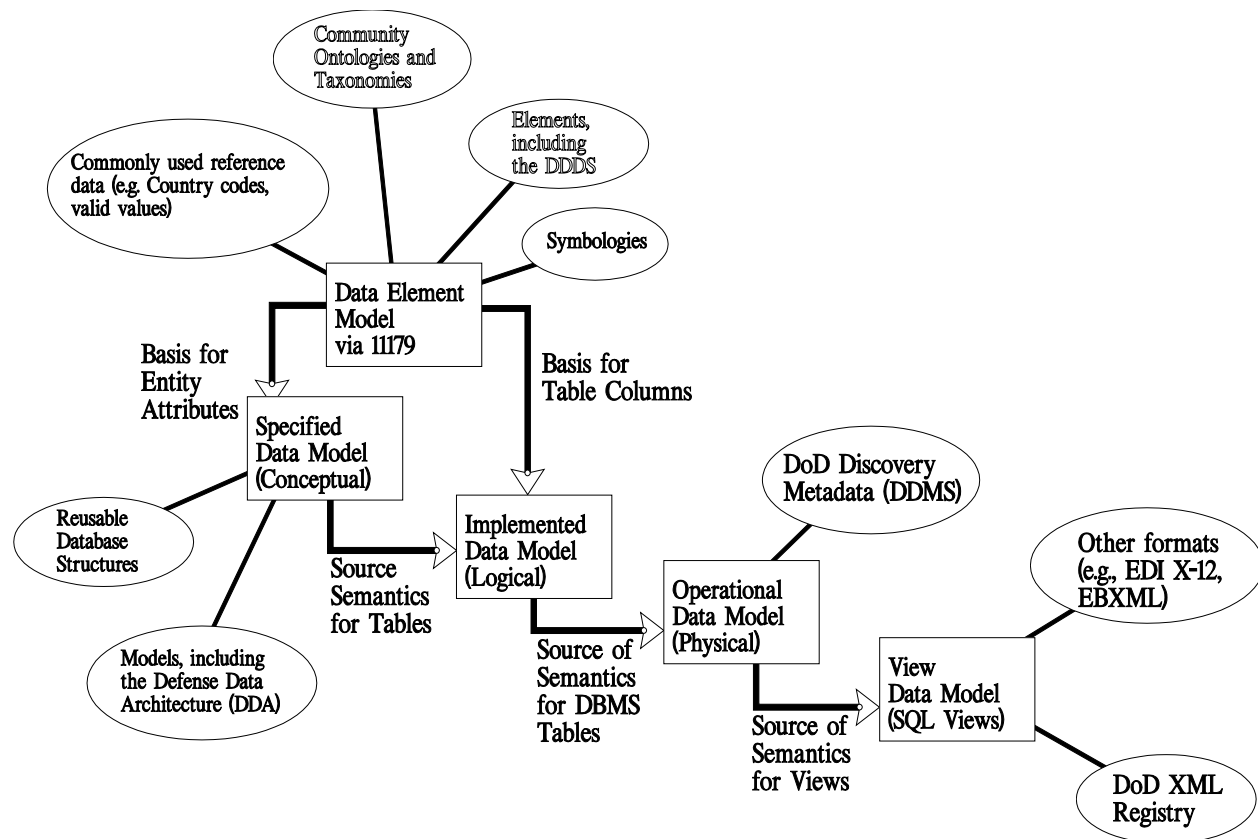


Figure 22. Integration of a metadata registry and a metadata repository.

generation of the metadata catalog entries. Entries in this catalog are identified in the figure as DoD Discovery Metadata (DDMS)). These entries are all generated from the physical (operational data model) rectangle. Further, if each metadata catalog entry is generated, then there is an automatic link back from the metadata catalog entry and the complete set of metadata supporting that entire data model that is represented by the metadata catalog. Even if the data



asset represented by the metadata catalog entry is an XML transaction, its XML schema will have link backs to the comprehensive set of supporting metadata.

3.7 Shared Data Architecture Summary

The objective of this section was to identify and set out the various data centric data asset products that exist explicitly and implicitly within the various architecture views. These data asset products were then described along with an overall strategy was presented that ensures their proper capture. Capturing and interrelating all this data centric metadata is critical because these data asset products interrelate IT products both within and across domains and mission areas.

Presented also in this section was the real problems associated with two key components of enterprise-wide data sharing, that is, XML schemas and metadata catalogs.

The set of all problems that can block a shared data architecture were then addressed and a way forward towards problem solutions was shown.

Finally addressed in this section are solutions to the problems associated with name management, value domain management, XML schema generation along with composing and shredding, and finally a real approach to the automatic generation of metadata catalog entry generation, mapping, and understanding.



4.0 Metadata Architecture for Data Sharing Summary and Way Ahead

This paper set out the following objectives:

- The identification of the data sharing environment including the most critical success measure: understanding-based interoperability. This was accomplished in Section 1.
- The case for comprehensive IT specifications in the form of metadata as the exposition mechanism for both analysis and design. This was accomplished in Section 1.
- The need for a metadata repository that is both comprehensive and valid that contains all analysis, design, implementation and maintenance artifacts. This was accomplished in Section 1.
- The description of an overarching reference model that ensures that all that's needed is both identified and has its proper place and role. This was accomplished in Section 2.
- A examination of the past similar efforts to gather lessons learned for the formulation of proper engineering and solution. This was accomplished in Section 2.
- The exposition of the infrastructure of data assets that have to be built and be supported by methodologies, strategies, and metrics. This was accomplished in Section 3.

In support of these accomplished objectives, Section 2 provided a description of a metadata reference model through which all net-centric data management program is delivered. This section also described the metadata environment that must be present for success including showing how common information resource management problems can be avoided. Finally, this section described the 1990s DoD "8320.1" data standardization effort and why it failed. Notwithstanding its failure, not only hasn't data standardization not gone away, it, as a problem that must be solved, has grown in both size and urgency.

Section 3 provided a definition of the net-centric data goals and what these goals mean within data management. Section 3 further identified the deployment of the net-centric data strategy through a brief description of data assets, the role of de jure standards, and the requirements imposed on data management by the net-centric data goals. The section also focused on the need for data asset products within architectures. The data asset products exist explicitly or implicitly within Technical, Operational, and Systems Architectures views. These data asset products have always been essential for well-engineered IT architectures. The data asset product products, properly done, provide an enterprise-wide view for the other views, and also a critical integration function within the Technical, Operational, and System view. A high level set of data asset product specifications were included as Attachment 1. Finally, this section sets out a metadata strategy and environment that results in smart, well engineered data standardization both within



war fighters domains and across the enterprise to achieve the net-centric environment in an efficient and effective manner.

4.1 Way-Ahead Actions

This paper comes to the following conclusions that include way-ahead actions that should be followed for a net-centric data management program to be successful.

- To be successful in information superiority, an enterprise first have a data management reference model and also a highly engineered approach for metadata management that operates in a distributed environment.
- The enterprise must engineer and accomplish data standardization that heeds well the lessons learned from the DDDS and DDA experiences, which were known by the programs 9 years before they were cancelled. The way ahead is known, has been demonstrated successful as early as 1985, and has been successfully presented at international data management conferences.
- The enterprise must engineer its data asset products into clear and precise specifications, and store them in a metadata repository in an integrated, non-redundant manner.
- The data asset product specifications and metadata repository must employ enterprise-identifiers to ensure that data asset product metadata can be uniquely identified, made non-redundant, be completely integrated, and can be accessed through a federation of metadata repositories.
- All XML schemas, data streams, and metadata catalog data should be automatically created from the data assets themselves and stored automatically in the metadata repository to the maximum extent possible.
- The enterprise net-centric data management program should be embraced by all IT projects so that the following capabilities are readily available:
- The enterprise should manufacture its databases by assembling pre-engineered standardized parts. When accomplished, the will then be:
 - ◆ The ability to quickly and completely determine the cost, effort and time required to accomplish IT data changes.
 - ◆ The ability to quickly and effectively respond to new IT requirements for information superiority.



- ◆ The ability to automatically generate prototypes of entire information systems just based on database designs.
- ◆ Automatic generation of XML schemas, data streams, and metadata catalogs.
- ◆ The ability to quickly and comprehensively understand the complete semantics supporting any XML schema.
- ◆ The ability to conform to the net-centric data goals, that is, visibility, accessibility, institutionalization, understanding, trust, interoperable and responsive to user needs.

4.2 The Bottom Line Reprise

The “bottom-line” from Section 1 deserves repeating.

Achieving net-centric environments is analogous to the game of football. You sometimes get the ball on your five yard line and most always have to make good solid plays all the way down the field until the final play, which may be a 5 yard dash into the end-zone. Everybody cheers and only remembers the 5 yard dash, that is, that you’ve achieved understanding-based data interoperability. Data standardization is, however, the first 90 yards. Hard work, accomplished one play at a time. Achieving net-centricity is the last 5 yard run into the end-zone. Throwing that 95 yard pass from your end zone—so as to avoid the work and accomplishment of data standardization--almost always results in failure. Worse, after four such attempts, the other team, which essentially is the no-understanding-based-interoperability, data chaos, and severely reduced C2 on the battle field, has the ball on your five yard line. Not good. The choices are thus two: Hard fought, well earned success, or “Hail Mary” pass failures. This paper addresses the first choice. Organizations operating under the second choice seldom survive.

