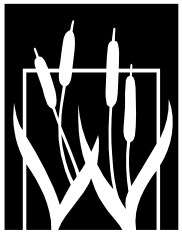

Whitemarsh
Information Systems Corporation

An Old Saw That Just Won't Cut

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

Table of Contents

1.0 Why Data Standardization Is Important

From the information technology perspective, it is commonly felt that data standardization affects only information technology. That can not be farther from the truth. While certainly data standardization affects the data element definitions and their specifications included in a myriad of data files, databases, and information systems, the domain of data standardization is much pervasive. In addition, the lack of data standardization is very expensive.

The United States Government Information Technology organizations, based on a survey within one DoD Agency is estimated to be spending about $4.175 Billion per year--every year--to overcome its lack of data standardization. These expenses are for the redundant and conflicting data definitions, the redundant and conflicting data stores and databases, and then all the information systems specifications, implementations, and maintenance efforts required to transform, re-transform, etc., all the data. When this sum is extrapolated to the United States economy, the cost is about $50 Billion per year. These are just the IT costs. Incalculable are the values of lost information, untimely decisions, and the like.

Situated within the context of the Knowledge Worker Framework, attention to data standardization must start at a high level as its effects are enterprise wide. Figure 1 presents the Knowledge Worker Framework. The cells directly affected by data standardization are shaded.

This paper addresses only one very small aspect of data standardization: data elements. Other components of data standardization are presented within other papers and books contained on the Whitemarsh Information Systems website, www.wiscorp.com

2.0 The Data Element Standardization Olde Saw

The "old saw" at issue is the traditional three part paradigm approach for data element names: prime word, modifier[s], and class word. This ***old saw just won't cut***. Never did. Never will. So, stop wasting time and money and choose one that does.

This paper identifies and describes the five main problems with the traditional approach and then presents an approach that since its introduction in 1981, has lowered cost and risk, and increased productivity and quality.

Whitemarsh Knowledge Worker Framework						
Perspective	**Mission**	**Database Object**	**Business Information System**	**Business Event**	**Business Function**	**Organ-ization**
Scope	List of business missions	List of major business resources	List of business information Systems	List of interface events	List of major business scenarios	List of organizations
Business	Mission hierarchies	Resource Life Cycles	Information sequencing and hierarchies	Event sequencing and hierarchies	Business scenario sequencing and hierarchies	Organization charts, jobs and descriptions
System	Policy hierarchies	Specified data models and Identified Database objects	Information system designs	Invocation protocols, input and output data, and messages	Best practices, quality measures and accomplishment assessments	Job roles, responsibilities, and activity schedules
Technology	Policy execution enforcement	Implemented data models and Detailed Database Objects	Information systems application designs	Presentation layer information system instigators	Activity sequences to accomplish business scenarios	Procedure manuals, task lists, quality measures and assessments
Deployment	Installed business policy and procedures	Operational data models	Implemented information systems	Client & server windows and/or batch execution mechanisms	Office policies and procedures to accomplish activities	Daily schedules, shift and personnel assignments
Operations	Operating business	View data models	Operating information systems	Start, stop, and messages	Detailed procedure based instructions	Daily activity executions, and assessments

Figure 1. Knowledge Worker Framework

3.0 Problems with the Traditional Approach

The five main mistaken notions associated with the traditional three part paradigm for data elements are:

- That data elements are synonyms for table columns, screen cells, entity attributes, or report fields.
- That data elements don't have names
- That prime word, modifer[s] and class word *are* part of the data element's name. That there is only a choice of one class word
- That modifiers are from a single homogeneous set
- That there is only a choice of one class word

3.1 That Data Elements Are Synonyms for Table Columns, Screen Cells, Entity Attributes, or Report Fields.

Whenever you review a relational table, you're likely to say, "The data elements are:..." Ditto when you review a screen, an entity or a report. So, the problem is the inference that the data element is a synonym for a table's column, a screen's cell, an entity's attributes or a report's field. In actuality, a data element is the metadata representation of the semantics for the column, for the cell, for the attribute or for the field. The data element is thus not a synonym for each, but their common, abstracted metadata representation.

For example, when you say, "Where are the First Names," you are likely to be answered, in the Person table, or the Biographic screen, the Employee entity, or on the Personnel report. So, if they all *contain* First Name, how can *Person First Name* and *Biographic First Name*, *Employee First Name*, and *Personnel Report First Name be* the same? Clearly, they can not. Rather, they all possess "containerized" representations of the First Name's semantics. Hence, data elements are not that which is contained, rather, they are abstracted common metadata representations drawn from the contained metadata representations of business fact instances.

When you take this "giant leap of common sense," you are immediately freed from the almost impossible, thankless, and worthless job of trying to identify and define every column, cell, or field that has or will ever exist. Because, if you believe that data element is the synonym for columns, cells, or fields then the effort to define these four is exactly that: impossible, thankless, and worthless. If you believe otherwise, then you have to totally discount the confession of a senior U.S. Defense Department official who admitted to having wasted $55 million on just that effort! And that amount was just for the data standardization agency. For the

DoD as a whole, it was about $200 million. In essence this U.S. Defense Department agency built a huge landfill of discordant, unintegrated semantics of columns. After 12 years of this completely fruitless work, the program was cancelled. The agency's reaction to this colossal failure was not to fix the problem, but to completely discard the $200 Million of product and adopt the complete opposite solution: That is, change from a lock-step, highly structured data standardization process to an "everybody do your own thing" approach. From impossibility to complete anarchy. That's an improvement?

The real solution lies in a very simple change coupled with an elegant data standardization architecture that has as its apex the well regarded ISO Standard, 11179 for data element metadata. The simple change is this: a data element is the context independent semantic representations of business facts. Data Elements are thus deployed within many different contexts, e.g., tables, screens, entities, and reports. Data elements "contain," while columns, et al are "contained." Given this REAL definition, the effort to define data elements is finite, manageable, predictable, and cost effective. In general, across an enterprise wide sea of columns, cells, attributes, and fields, the ratio is at least 30 to 1. The broader the area, the larger the ratio. With this new definition, the overall definition effort is cut by 95%, or more.

The concept of PRIME WORD is appropriate for columns, cells, attributes, and fields but is inappropriate for data elements. That's because prime word represents the name of the containing context. That is, the name of the table, screen, entity, or report. To have prime word as part of the contained component's name is to state categorically that the component must be a column, cell, attribute, or field.

What is appropriate for a data element, however is the identification of the business or subject area. This is necessary when we think for example of all the many different disciplines that commonly use the data element, *vehicle*, or the data element, *track*. These are valid homonyms and must be accommodated. Figure 2 clearly sets out what metadata is appropriate for a data element and what is appropriate for a column, et al.

Meta Attributes for Data Elements and Context Dependent Business Facts		
Meta Attribute or Semantic Part	**Data Element**	**Entity Attribute, Table Column, File Field, Screen Cell etc**
Business Domain	Yes	No
Common Business Name	Yes	Yes
Prime word	No	Yes
Modifier subclasses	No	Yes
Class word subclasses	No	Yes
Generated Policy Basis Description	Yes	Yes

Meta Attributes for Data Elements and Context Dependent Business Facts		
Meta Attribute or Semantic Part	**Data Element**	**Entity Attribute, Table Column, File Field, Screen Cell etc**
Computer Data Type	No	Yes
Data Structure	Yes	Yes
Required Uniqueness	No	Yes
Relationship Function	No	Yes

Figure 2. Meta Attributes appropriate for Data Elements and Contained Columns, et al.

3.2 That Data Elements Don't Have Names

The second problem is the belief that data elements don't have names! In fact, they do. They each have a *common business name*. If however you accept the "traditional" construct for a data element, i.e., *Prime word + Modifer[s] + Class word* then you MUST believe that data elements don't have names. The reason you MUST believe they don't is because under the traditional approach, a data element's name is said to be a concatenated representation of precisely chosen word strings, each of which is chosen to represent a specific part of the overall set of semantics.

Given you believe that to be true, then how do you explain, *Employee Social Security Number*? If you examine each word, then *Employee* must be the prime word, *Number* must be the class word, and *Social* and *Security* must be two distinct and "nested" modifiers, each of which conveys semantic meaning. One thing you know for sure is that those characterizations are untrue. At the very outset, you must distinguish between the column, *Employee Social Security Number* and the data element, *Social Security Number*. This same forced, delusional analysis would also apply to *Telephone Numbers* and *Part Numbers* where *Telephone* and *Part* are prime words and *Number* is the class word.

So, either you proudly accept that exhibitions of forced, delusional, analysis are something to brag about, or you must accept that opposite: that data elements have names that are NOT necessarily characterizations of their semantics. Rather, they are their commonly known names, such as *Social Security Number* (which is a three part numeric code separated by hyphens) or *Part Number* (that commonly contains multiple embedded codes, some of which contain letters), or *Telephone Number* (which, like Social Security Number, is a multi-part code, but has either hyphens, a set of parentheses and hyphens, or under the new ISO scheme is a four part numeric sequence separated by periods). How composite data elements are treated is another issue entirely, and is not addressed in this paper.

3.3 That Prime Word, Modifer[s] and Class word *are* Part of the Data Element's Name

The third mistaken notion is that the prime word, modifer[s] and class word comprise the data element's name. Given that we have shown above that a data element has a common business name which is NOT the prime word + modifer[s] + class word combination, then what happens to Prime Word, Modifer[s], and Class word? Well, they are useful and they do describe some important semantics of data elements, contained components, or both.

An additional problem arises when the prime word, modifier[s] and class word are melded into the name string: they lose their individual semantic identity and distinguishability. That's especially so when abbreviations are required, or when wholly different words must be used to accommodate different languages and cultures. Simply put, each is a different meta attribute within the complete set of meta attributes that fully define either the data element or the context dependent business fact.

Most importantly is the distinction between that which *contains* and that which *is contained*. Data elements are the former, and columns, cells, attributes and field are the latter. All these are meta entities within an overall meta model for information technology. It's just that data elements largely stand alone while columns, cells, attributes and field are properly contained in tables, screens, entities, and reports, respectively. Collectively, these latter four are context dependent metadata about business fact value *instances*, while data elements are context independent metadata about context dependent business facts. Because of this difference, each has a slightly different set of meta-attributes. Figure 2 above presents these differences.

As can be seen, prime word is a proper meta attribute of a context dependent component but not the context independent component, data element. This same difference holds as well for modifier. Some of the other meta attributes from Figure 2 above are common. For example, data structure is now really important since SQL/1999 is no longer relational. An SQL/1999 column within a single row may represent single values, multi-values, groups, repeating groups, nested structures, BLOBS, CLOBS, REF-types to other-site objects, and abstract data types with both data structure and methods.

3.4 That Modifiers Are from a Single Homogeneous Set

It is commonly thought that modifiers are derived from a single homogenous pool. While that may have been true for a single database within a single project, it is not true for enterprise-wide suites of databases that cross countries and cultures. Figure 3 contains a minimum set of modifier classes and examples of the value sets for each.

Modifier Classes and suggested contained values			
Temporal	**Accuracy**	**Geographic**	**Organizational**
last	estimated	world	world-wide
first	projected	hemisphere	business unit
latest	revised	north American	region
earliest	initial	United States	district
current	actual	mid-Atlantic	territory
this year		Maryland	
last year		Bowie	

Figure 3. Examples of Modifier Classes

3.5 That There Is Only a Choice of One Class Word

The final key problem is that class words have always been thought to be drawn from a single choice list. That is, only one class word per component (e.g., data element or column). That, however, has never been true despite our being "forced by tradition" to believe it. For example, suppose there is a price table, where PRICE is the primary key column. Shouldn't PRICE's class word be Identifier? Yes. But, isn't price also an Amount? Yes. Why must we force ourselves to make only one class word choice? How can Social Security Number be an identifier, but not a code, and certainly not it's already existing class word choice, number?

The way out of this "absurd to be in" situation is to recognize that there are multiple classes of class word, just like there are multiple classes of modifiers. Figure 4 offers suggestions in this area. This categorization and example sets are not complete by any means. For example, the *role* class should probably be divided into a *business* role and a *container* role. A *factor* would be an example of the former and an *identifier component* an example of the latter. The key issue here is that there are multiple classes of class words and that they must be carefully categorized and selected so that the full set of semantics is properly chosen to reflect both the needs of the business and the needs of the role the column performs within the "container."

Class Word Types and Example Value Sets		
Data Type	**Role**	**Unit**
Date or date component	Identifier component	Day

Class Word Types and Example Value Sets		
Data Type	**Role**	**Unit**
Code	Factor	Case
Text	Flag	Aisle
Weight	Indicator	Pallet
Dimension	Identifier component	Transaction
Money	Rank	Percent
Integer	Business fact	Inches

Figure 4. Examples of Class Word Classes

4.0 A Strategy That Works

When all the meta attributes from Table 1 are appropriately valued for either a context independent or a context dependent component, a full set of semantics can be printed AS IF THEY WERE the name of the data element or the context dependent component. Figure 5 illustrates just such a scheme. It is critical however NOT to be misled into believing that these brought together strings ARE the component's name. The name may be contrived wholly differently and by another set of rules altogether. For example, the column's name represented in Figure 5 may just be SALARY within the Employee table.

Additionally, all these value selections are really just foreign key value instances within their respective meta attributes of the meta-entity, COLUMN. Their originating meta-entities, e.g., PRIME WORD, or ORGANIZATIONAL MODIFIER, etc, not only have the real values, but also have important information about the real value such as who authorized it, when, why, different accepted abbreviations (none, medium, short, ultra-short), full definition, and definition fragment.

Of especial value are definition fragments. These can be brought together to AUTOMATICALLY provide definitions for all the context independent and context dependent business facts. That gives a real boost to the 90% reduction in the amount of work both to identify context independent and dependent components, and to define them.

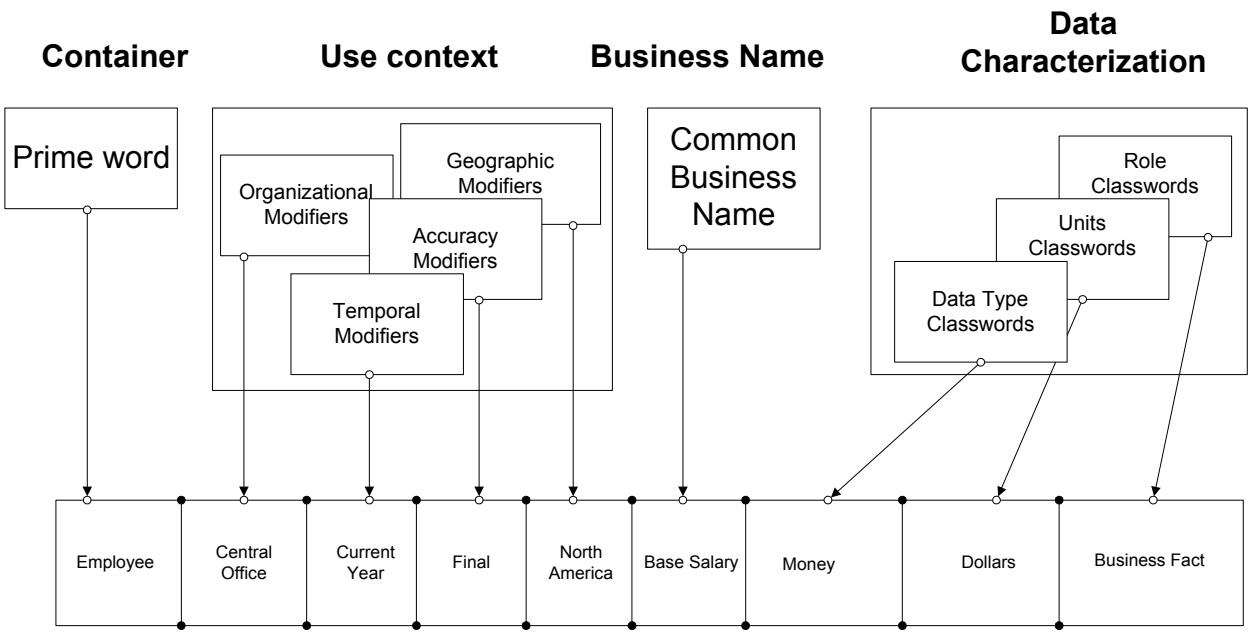

Figure 5. Fully Selected Semantics for Employee Base Salary Business Fact

5.0 The Payoff

The pay off from this approach is really compelling. In a reflective study of a very large data standardization effort, the following techniques comparison shown in Figure 6 was developed. Which OVERHEAD budget line item do you want to take to your management for support?

This strategy of having a high quality "driving" repository that is coupled with a end-user supportive CASE, code generators and database project methodology is critical if we are ever to empower the end-user database designer.

Today, it seems that we "want" everybody to go through database administration or data administration. While that may "appear" to give us feelings of importance and criticality, I believe that such feelings will be short lived and ultimately suicidal. Our "glory" can be true only when it's reflected "glory." That is, when we empower others to make maximum use of facilities we engineer or build for their success. Then, and only then will their success be our success.

Data Standardization Alternative	Final Quantity of defined components	Cost
Traditional (prime + modifer + classword) across all systems	19,000 (at the columns, cells, fields, and attributes level)	$6.75 million
Accomplished by standardizing closely named columns and fields	3,000 (still at the columns, cells, fields, and attributes)	$1.06 million
Accomplished through standardization techniques in this paper.	560 (at the data element level and then generation of automatically defined columns, fields and cells	$450,000

Figure 6. Data Element Standardization Techniques Comparison