



## **Data Modeling for Healthcare Systems Integration: Use of the MetaModel**

Jack E. Myers, BSME, MS, MBA  
President, The Metadata® Company

"Problems can not be solved with the same knowledge with which they were created."  
Albert Einstein

### **ABSTRACT**

Presented is a new approach to solving the critical healthcare systems integration problem. The premise is that any significant level of healthcare systems integration requires the development and use of a common data model. And, this requires the use of a new type of data modeling technology. The degree of healthcare systems integration possible is dependent on the capabilities of the data model employed. Experience has shown that traditional data modeling technology, such as that based on relational database theory and Entity-Relationship (ER) diagrams, does not possess the necessary functionality, nor is it robust enough to define and structure complex data environments, such as healthcare, in an unambiguous and meaningful way.

The need is for a data model based on concepts like the Metamodel developed by The Metadata Company. The practical innovation of the Metamodel is in its ability to represent data structure in a simple and concise way. Because the Metamodel is based on a data language, it can relate units of meaning as well as the more traditional database records. Importantly, it is not tied to any one computer or database technology like other data models. It is believed, the Metamodel, or something like it, must be used to bring together knowledge of industry data before substantial healthcare systems integration can occur. The Metamodel can be easily combined with other system development tools and industry standards to achieve a level of healthcare systems integration not previously possible.

The Metamodel has been rigorously tested in the "real-world" and has adequately handled all application environments and all levels of data complexity encountered. It has been used for the development of numerous large database applications in various industries. The Metamodel has particular relevance to the integration of healthcare systems and work is currently underway using it to create an industry-wide data model. The "medication" view of this data model, which contains more than 500 different data elements, has been completed and now serves as the underlying data structure for the Meta-Med System, Metadata's medication management system.

### **BACKGROUND**

Perhaps no other sector of the U.S. economy is undergoing greater change in its basic structure and mode of operation than is the healthcare industry. The most radical change - managed care - significantly alters the amount and types of data healthcare professionals need. The traditional organization of healthcare services along independent professional lines is giving way to multidisciplinary management organizations with patient foci. This places the primary care physician (PCP), as coordinator of patient care, at a distinct disadvantage without the availability of comprehensive patient data. The full computerization of patient health records has been proposed as a solution, but still is a long way off. Even if accomplished, this data must be shared and integrated with other healthcare provider data in order to realize needed benefits.

The Metadata® Company  
444 W. Ocean Blvd., Ste. 1600  
Long Beach, CA 90902



The healthcare industry is characterized by large numbers of providers involved in the independent provision of care. Historically, healthcare delivery has been segregated into many different specialized components. Although linked in an intricate network of relationships, most healthcare services are rendered in a vacuum. The industry largely ignores the close, natural interrelationships that exist between specialty areas and focuses on the individual treatment of disease, rather than overall patient health. Moreover, the fragmentation of services and healthcare systems has helped to polarize individual providers against managed care objectives and hindered the quality of care through uncoordinated treatment.

## **NEED**

It is generally agreed that the development and use of integrated healthcare systems are becoming an absolute necessity to support the delivery of low cost, high-quality care. To achieve this goal, however, requires cooperation and coordination not previously experienced. Providers have generally maintained their own independent data and the incompatibility of healthcare systems largely prohibits its cross-institutional use. There currently is nothing that ties healthcare data together in a coherent and uniform way, which is expensive, redundant and insufficient.

The need for healthcare systems integration is multidimensional and is being driven by a number of medical, technical, organizational and political factors. Benefits resulting from an integrated healthcare environment are numerous and substantial. Integrated data can greatly assist the various caregivers in making correct assessments and administering the proper treatments, as well as facilitating the optimization of operations across the enterprise. The collaboration and widespread sharing of data can help identify and pursue opportunities for improving outcomes and lowering costs. There is little question that the answer to many of today's global healthcare delivery problems lies in the integration of data generated by the various medical services. Integrated data is quickly becoming the desired characteristic of new and replacement healthcare systems.

Healthcare systems integration must occur in several different ways: at the provider, patient, software, computer and data levels. Users must not only share systems, but interact with each other in a seamless way based on business, political and procedural considerations. Computer software developers must not only cooperate but share a common data model and design products that can interact with each other. Computer integration is continuously being addressed by the computer industry, however, the use of different operating systems, i.e., MVS, UNIX, Pick, MS-DOS, Microsoft Windows, OS/2, etc., still impedes progress. Data integration is fundamental and a necessary prerequisite to systems integration. It requires the development of a healthcare data model that can serve as the basis for meaningful development.

## **APPROACH**

There are important reasons why existing healthcare systems are not integrated. Those that have tried know how hard it is. A certain critical mass of complexity is always reached beyond which it is virtually impossible to achieve full systems integration. To attack the total healthcare systems integration problem all at once is virtually impossible and to do so in an incremental way requires the use of software development tools not previously available. No one person, or group of people, is capable of defining and building a totally integrated healthcare system with all of the required functionality -- at least not by using traditional technology. This means that if healthcare is to be integrated, it must happen in some new and different way.

A major problem has been one of approach. Just following the path toward industry standards, i.e., ASCX12, NCPDP, HL7, ASTM, etc., is not the answer. Hundreds of healthcare standards have been established, yet



no significant level of system integration has taken place. Existing standards are fragmented, primarily addressing the electronic interchange of data within very narrow areas of use. They are more directed at establishing systems interfaces than systems integration. This is a situation precariously dependent on the adherence to common and constant industry standards, something no industry has ever achieved. Until standards address the underlying structure of healthcare data in a common and unified way, they will remain largely ineffective.

A new approach to healthcare systems development should be taken in the future. Healthcare systems are not currently designed using common data modeling constructs resulting in different database designs; there is a different database design for each application. The desired approach utilizes a language-based data model with a finite set of data relations, like the Metamodel. This results in one conceptual database design that can be shared by all systems. Databases structured in this way are more "intelligent, and can share processing routines and automatically maintain data integrity. Importantly, they eliminate the need for independent database design and greatly reduce the data administration effort.

Communication between people using databases is not the same as communication among people, which is adequately handled by natural language and mathematics. Nor is it the same as communication between people and computers, which is adequately handled by computer programming language. It requires a very different type of language, one that addresses data meaning using functional data classification and a finite number of relations. Without the use of a data language, comprehensive person to person communication via databases is extremely limited and true system integration is virtually impossible.

## **DATA MODEL**

The data model is what ties human thinking with computer processing. It provides a blueprint for database and application system design. The data model is where the various data, data relationships, rules, domains, etc., are described. Importantly, it is where the meta-data, or data about data, resides. Other related activities like establishing standards and naming conventions become hopeless tasks without an adequate data model to use as the common frame of reference.

Data models must address three distinct orders, or levels, of data structure: conceptual, logical and physical. The same conceptual data structure can be implemented using different logical data structures and the same logical data structure can be implemented using different physical data structures. A data model of sufficient functionality must be used to tie together the different data structures so that meaning is not lost as translation takes place between them.

The solution is not a set of relational database tables or Object Oriented (OO) Objects, or an ER diagram. Relational database tables are based on logical and physical data structures, not data meaning. They are formulated using dauntingly complex, open-ended normalization rules that are grossly inadequate for semantic representation. OO Objects are based on logical and physical data structures and primarily exist for OO language programming, not human communication, purposes. ER diagrams cannot adequately show the relations between attributes, mix levels of abstraction and are semantically ambiguous. None of the above technologies adequately address data meaning and structure.

People are only concerned with conceptual and logical data structure. How data is physically structured is of no concern. Computer systems, on the other hand, are only concerned with logical and physical data structure. Logical data structure is held in common and forms the nexus between human thinking and computer technology. This means that the Metamodel, which can impose the conceptual data structure on



the logical data structure, should be used. With very few exceptions, healthcare systems have been developed with only logical and physical structures in mind; they are computer- not people-oriented.

## **CONCEPTUAL DATA STRUCTURES**

Conceptual data structure, as an integral component of communication among people, focuses on and involves the use of language. Common languages, such as, natural language (English), mathematics and computer programming language, have proven to be grossly inadequate for this purpose. English is far too complex and imprecise to be used as the basis of data structure. Mathematics addresses only a small class of automated data and is not easily used for the vast majority of human communication. Computer programming languages are typically a hybrid creation of natural language and mathematics, with some added logical and physical components of data structure.

## **LOGICAL DATA STRUCTURES**

Logical data structure is concerned with how the relationships between data are maintained during computer processing. A number of different logical data structures have been used over the years by the various database management systems (DBMS's). All DBMS's, except the Metadata® - UDBMS, implement logical data structure at the record, not attribute, level. The three logical data structures in common use can be characterized as; flat, hierarchical and network. Relational DBMS's use a version of the flat record structure; most semantic and object-oriented DBMS's use a hierarchical record structure; and the CODASYL Standard DBMS's use a network record structure.

## **PHYSICAL DATA STRUCTURES**

Physical data structure is concerned with how data records are physically organized on the various computer storage media. The same logical data structure may be organized in a number of different physical ways depending on the volume of data, access requirements, storage media and vendor products involved. Commonly used physical data structures include: unit-record (card), sequential (tape), indexed and direct. Most database records are now stored using some type of indexing scheme.

## **METAMODEL**

The Metamodel was developed in response to the need for a new, more pragmatic approach to database system development, one based on a unified data structure. Principles of the Metamodel were taken from English, mathematics, general semantics and computer programming language. The concepts of parts-of-speech and grammar were taken from natural language; the use of logical and boolean operators were taken from mathematics; the "structural differential" and non-Aristotelian approach were taken from general semantics. In addition, the needs for data typing, domain specification and integrity maintenance, were taken from computer programming language. The Metamodel is an amalgam of language capabilities selected specifically for data modeling use.

The Metamodel addresses three interdependent components of conceptual structure: identification, classification and relationship.



## Data Identification

Data identification is the representation of individual units of meaning -- data elements. The modeling of units of meaning (data elements) instead of units of notation is a key feature of the Metamodel. It is what distinguishes the conceptual data structure from the logical and physical data structures.

## Data Element

Each data element is assigned a non-significant data element number (DEN) for notational purposes. The meaning designated by the DEN is described using English, mathematics and/or computer language terms. Any number of different names may be associated with a DEN, both for human and computer use. Importantly, these names do not have to conform to any format, or be unique, freeing the Metamodel from dependence on naming conventions. In addition, the various notational schemes and domain rules associated with the data are identified using non-significant data element version numbers.

In Figure 1, the meaning designated by data element number, "0003", is shown along with the various names ("date", "REF\_DATE", etc.) used when referring to it. Also shown, are the different data element versions that identify the different data domain(s) and forms of notation used for recording data. Data element versions share the same meaning.

<b>DEN</b>	<b>Description</b>
0003	"A given point in time as identified by a calendar day."
Names	"Date" "Refill Date" "Claim Date" "Test Date" "Ref_Date"
<b>Versions</b>	<b>Format</b>
1	"6-1-91"
2	"June 1, 1991"
3	"01/06/91"
-----	-----

Figure 1 - Example Data Element

## DATA CLASSIFICATION

Data classification is as fundamental to data modeling as word classification (nouns, verbs, etc.) is to natural language. Far too many data relationships exist to treat them on an individual basis. Without data classification, intelligent communication is all but impossible. Placing data into functional categories grossly reduces semantic complexity and improves the conveyance of meaning. The Metamodel uses three broad functional classes to classify all data elements: identifiers, modifiers and descriptors. The same data element can function in any class.

### Identifier (I)

Identifiers serve as the primary identification terms, or keys, necessary to uniquely identify things and events, or classes of them. They symbolically represent things and events (entities) and provide for the



necessary identification of conceptual objects. Importantly, identifiers provide the skeletal structure upon which all other types of data depends. They also provide a means for explicitly defining the relationships between things and events (entities), which enables data sharing among users. Typical identifiers include: patient, account, part and purchase order numbers.

#### Modifier (M)

Modifiers serve as sub-entity identifiers and expand upon or refine primary identification (identifiers). As variant forms of identification, they cannot stand alone. Modifiers must be used in conjunction with identifiers to form fully qualified identification terms. They primarily are used to identify such things as: time, occurrence, use, type, sequence, etc. Modifiers have a unique data element value for each variation of the identifier addressed and can exist with one-to-one (1:1) or one-to-many (1:m) cardinality. Typical modifiers include: dates, type codes, serial numbers and revisions.

#### Descriptor (D)

Descriptors are non-identification data elements used to characterize entities and relationships of them. There are no logical dependencies between descriptors and they can only exist when associated with an identifier or identifier-modifier combination. Descriptors comprise the majority of all data elements and frequently are textual or codified data element values -- data that must be further interpreted by the user to have meaning. Some descriptors are numbers capable of being mathematically manipulated, while others are numerals that do not follow strict mathematical rules. Typical descriptors include: dates, names, descriptions, codes, numeric values and images.

#### Data Relationship

In order to integrate healthcare data, there must be a known, fixed schema with a finite number of data relations, as is required of any closed relationship system. Of existing data models, only the Metamodel is structured in such a way that multiple data elements can be combined to precisely express higher-level meaning. For example, the meaning designated by a data element number used as a D or M, is intertwined with that of the data element number(s) used as an I or IM to which it is related. As such, data relations represent a connection, or affiliation, of established meanings. A considerable amount of meaning can be represented in this way, in fact only in this way. Business and processing rules, such as, cardinality, occurrence, etc., can be associated with each data relation.

Data relations are established by first isolating the primary identification term for each entity and then relating all the other data elements that further identify/characterize the entity. They are autonomous and stand alone as individual statements of meaning. This allows different people with different knowledge and at different times and places to independently contribute to a common data model. The constructs of the Metamodel automatically integrate the results of independent actions into a seamless semantic network. As more and more data relations are established involving multiple entities, ever larger data structures are created. The end result is a meaningful model of all healthcare data within the enterprise/industry.

#### Logical Operators

Use of logical operators allows data element dependencies to be expressed in precise terms, replacing the ambiguous use of "verbs". The Metamodel symbolizes the relationship between entities using the "=" (equal), ">" (greater than) and "<" (less than) logical operators. Any number of different words may be used, however, when referring to the use of logical operators. Importantly, there are no additional data relationships created by the use of these words, as with other data models. Figure 2 shows how logical operators might be referred to in a data relation.



"="	">"	"<"
also known as analogous to equivalent to the same as	owns manages uses contains	belongs to managed by used by goes into

Figure 2 - Example Logical Operators

There are two basic types of data relations, simplex and complex. Simplex data relations are composed of data elements pertaining to one entity -- only one identifier. There are only 4 such data relations; I, ID, IM and IMD. There may be any number of ID, IM and IMD data relations associated with any one I. For example, the meaning represented by the IMD data relation involves three data elements. The D has meaning as it applies to the meaning of the IM and the M has meaning as it applies to the meaning of the I.

Complex data relations result from the interdependency of two different entities or occurrence of the same entity. One identifier represents the primary entity, or subject, and the other represents the related entity, or direct object of the relationship. It is possible to have 5 different data elements along with the logical operator participating in one data relation. This creates a semantically rich data model capable of representing considerable knowledge. Because the Metamodel uses only 3 functional categories of data (I, M, and D) and 3 logical operators (=, >, <), they can be structured in a finite number of ways -- 28 in the case of the Metamodel. All data relations used by the Metamodel are shown in Figure 3. As shown, 12 of the data relations have logical opposites. For example, the opposite of I1M1>I2 is I2<I1M1.

		<b>Number</b>	<b>Data Relation</b>	<b>Opposite Number</b>	<b>Opposite Data Relation</b>
Simplex Data Relations		1	I		
		2	ID		
		3	IM		
		4	IMD		
<hr/>					
		5	IM<I	27	I>IM
		6	IM<ID		
		7	IM<IM	15	IM>IM
		8	IM<IMD		
		9	IM=I	23	I=IM
		10	IM=ID		
		11	IM=IM	11	IM=IM
		12	IM=IMD		
		13	IM>I	19	I<IM
		14	IM>ID		
		15	IM>IM	7	IM<IM
Complex Data Relations		16	IM>IMD		
		17	I<I	25	I>I
		18	I<ID		
		19	I<IM	13	IM>I
		20	I<IMD		
		21	I=I	21	I=I
		22	I=ID		
		23	I=IM	9	IM=I
		24	I=IMD		
		25	I>I	17	I<I
		26	I>ID		
		27	I>IM	5	IM<I
		28	I>IMD		



Figure 3 - Example Data Element

## EXAMPLE DATA MODEL

Figure 4 shows a hypothetical healthcare data model. Although limited in scope, it demonstrates how 5 different healthcare systems might share in the use and/or creation of the same data relations. Shown are 21 different data elements that are used by 41 different data relations and associated with 6 different entities. In reality, there may be hundreds of different data elements associated with each entity, and hundreds of different entities. As new healthcare systems are developed/analyzed, any new data elements and data relations are identified and added to those shown. The data model can grow incrementally as required without first having to know the total data environment.

	DATA REL	DEN	DATA ELEMENT NAME	SYSTEM				
				1	2	3	4	5
<b>Entity (1)</b>	I	0001	MEMBER ID	*	*	*	*	*
	ID	0002	ADDRESS	*	*	*		*
	ID	0017	COB IND		*			*
	ID	0006	GENDER CD	*	*	*	*	*
	ID	0009	MEMBER NAME	*	*	*	*	*
	ID	0019	BIRTHDATE	*	*	*	*	*
	IM	0003	DATE	*				
	IMD	0004	BLOOD PRESSURE	*				
	IMD	0007	WEIGHT	*				
	IM	0011	EFFECTIVITY	*	*	*	*	*
	IM<I	0005	PROVIDER ID	*	*		*	*
	IM<I	0008	HEALTH PLAN ID	*	*	*	*	*
	I=I	0012	SUBSCRIBER ID		*		*	*
	I=IM	0013	RELATION CD		*		*	*
	I>I	0014	RX NO		*	*		
	I>I	0021	CLAIM NO		*			
	<b>Entity (2)</b>	I	0005	PROVIDER ID	*	*	*	*
I>I		0001	MEMBER ID	*	*	*	*	*
I>IM		0011	EFFECTIVITY	*	*		*	*
I>I		0014	RX NO		*	*		
I>I		0021	CLAIM NO		*			
<b>Entity (3)</b>	I	0008	HEALTH PLAN ID	*	*	*	*	*
	ID	0010	CO-PAY AMT	*	*	*	*	*
	I>I	0001	MEMBER ID	*	*	*	*	*
	I>IM	0011	EFFECTIVITY	*	*	*	*	*
<b>Entity (4)</b>	I	0012	SUBSCRIBER ID		*		*	*
	IM	0013	RELATION CD		*		*	*
	IM<I	0001	MEMBER ID		*		*	*
<b>Entity (5)</b>	I	0014	RX NO	*	*			
	ID	0015	DRUG LABEL NAME				*	
	IM	0003	DATE	*	*			
	IMD	0016	RX REFILL				*	
	IMD	0020	RX PRICE				*	
<b>Entity (6)</b>	IM<I	0021	CLAIM NO	*	*			
	I<I	0001	MEMBER ID	*				
	I<I	0005	PROVIDER ID	*				
	I	0021	CLAIM NO	*	*			
	ID	0018	CLAIM AMT	*				
	I>I	0014	RX NO	*	*			
	I>IM	0003	DATE	*	*			
	I<I	0001	MEMBER ID	*				
	I<I	0005	PROVIDER ID	*				

Systems

1. Visit    2. Claim    3. Rx    4. Eligibility    5. Enrollment

Figure 4 - Example Healthcare Data Model



## **RELATED SOFTWARE TOOLS**

The Metamodel has been incorporated into a family of software tools, which include:

### Metadata® - Unified DBMS

The UDBMS is a full function DBMS that utilizes one database design for all applications. Its database structure, which is a direct implementation of the Metamodel, accommodates all forms of logical DBMS structure. It also accommodates all types of data: numerics, characters, text, images, blobs, etc. Relationships are maintained between data elements (attributes), not records (tables) as with other DBMS's. Each UDBMS record occurrence may contain a different set of attribute values (data). There is no need to provide for nonexistent data in the UDBMS database.

### Metadata® - Integrated Repository

The Integrated Repository (IR) is a standalone data dictionary/directory system based on the Metamodel. It is used to interactively develop the data model and to maintain all meta-data associated with the Metamodel and DBMS applications. Meta-data is defined using the Metamodel in the same way as application data. The IR can accommodate any data structure and can span computer platforms and systems. Since both the meta-data and data model are stored in the UDBMS database, they are readily available for additional use, such as input to CASE tools and DBMS dictionaries and catalogs.

### Metadata® - DataConvert

DataConvert is a data integration and migration utility based on the Metamodel. It can automatically convert any application data to/from the UDBMS database, or from one format and/or data structure to another, without programming. Once the data model is created using the Integrated Repository and mapped to the source record(s), DataConvert automatically parses the record(s) into Metamodel data relations. Source records from various applications are automatically integrated within the UDBMS database in this way. Target records can be created simply by mapping the data model to the desired record format.

### Metadata® - DataSchema

DataSchema is a utility that automatically generates database designs. Any view of the data model, or set of data relations, may be used as the basis of design. This can be done simply by selecting the desired data elements. DataSchema then transforms the specified view into the desired DBMS design, for example, relational database tables in 5th normal form. The created database design can be automatically populated using DataConvert. DataSchema is a valuable tool for re-engineering existing applications, developing new applications and integrating databases.

### Metadata® Semantic Search Engine

The Semantic Search Engine (SSE) provides user search capability without knowledge of computer technology. Searches are performed using a semantic network with weighted connections.