Medical Data Dictionary for Decision Support Applications

S.M. Huff R.B. Craig B.L. Gould D.L. Castagno R.E. Smilan

AT&T Bell Laboratories
Columbus, Ohio 43213

Abstract

Building and maintaining clinically-based medical information systems is a complex task. Advances in database management technologies, including the concept of a data dictionary, have helped support this process. A medical data dictionary is described with discussions on the role of data dictionaries within a medical information system and the conceptual model supported by the AT&T CareComm (TM) data dictionary. Additionally, a high level overview of features that we deemed important to support a medical information system with integrated decision support capabilities is presented.

Introduction

To be more than a simple repository of facts, a medical information system (MIS) requires a robust and flexible data dictionary (DD). Indeed, by defining the vocabulary, grammar, structure, and location of information in the database (DB), the DD is the "heart and soul" of a MIS and the foundation of all decision support methodologies. Conversely, the structural elements included in the DD are carefully selected so that many different management and research functions can be supported. They are inter-dependent entities; the DD establishes the environment for support of all functions of the medical information system, and the desired function of the medical information system drives the structure and grammar of the DD. Thus, as a preface to a description of the AT&T CareComm(TM) data dictionary architecture, it is appropriate to outline briefly the role of the DD, the conceptual model that is the basis for information management in the system, and a high level view of features that we deemed important to support in a MIS.

The Role of the Data Dictionary

The process of creating a MIS begins with analysis of the real world. [1] What is the task domain of the system? What information is needed to do the tasks? What are the objects that are interacting in the real world, and what relationships govern the interactions? From what perspective should the objects and relationships be viewed? What operations transform the real world domain? As these questions are answered, a conceptual model of the world evolves into the blueprint of information management for the proposed system. Since the domain of medical information is complex, the conceptual model must be comprehensive and robust.

Once the conceptual model is established, implementation of the model becomes a concern. What data structures are needed to represent the data? What transactions are needed to do the required tasks? What are the performance requirements? As these questions are answered, a physical model takes shape. Again, the sophistication of the conceptual model will be reflected in the physical model. It is at this point that the role of the DD becomes essential.

The DD stores definitions of the conceptual objects and maps them to structural elements in the physical model. It knows about the relationships between objects and supplies this knowledge to processes that populate, manipulate, and transform the objects. Besides definitions, the DD contains directory information for all data in the MIS. To provide decision support capabilities it must deal with complex relationships and the rules that are necessary to make the system "smart." It is this role of linking the abstract model of the world to physical data structures and stored definitions that makes the DD the fulcrum for all medical information system operations.

Cardinal Features of a Medical Information System

As shown above, the definition of the conceptual model is driven by analyzing the functions that a system is to do. To lay a foundation for the conceptual model of the AT&T CareComm Medical Information System, essential features of the system are outlined below. Briefly, the system should be understandable and consistent, allow information sharing between unlike environments, have a comprehensive domain, be flexible and expandable, able to manage change, frugal in computer resource usage, and provide tools for decision support.

First, users and processes must be able to find and understand the information stored in a system. Finding information means support of *ad hoc* queries and decision support searches in the data base. Information becomes understandable through consistent use of a defined vocabulary and grammar for information storage. Consistent representation means that information that is semantically equivalent is always stored in the same way. For example, if blood pressure readings can be recorded in the operating room, outpatient clinics, and intensive care units, then all results should be stored in a field called "Blood Pressure." It would be poor design to confuse the issue by creating two fields, one called "Surgery Blood Pressure," and the other "Outpatient Blood Pressure." The role of the DD is to ensure consistent information representation by centrally defining and enforcing the vocabulary and grammar of the DB, and to make the DB representation available to users and processes. Consistent representation coupled with the availability of the vocabulary information makes *ad hoc* query possible.

The ability to share information within a system is a second important feature. This is especially true in a network or distributed architecture where network nodes provide dissimilar functions. Again, it is consistency in information representation that makes data exchange possible.

If all care areas use "Blood Pressure" to mean the same thing, then it is possible to transfer records between outpatient clinics and inpatient units without the need to map between different structural data representations. Furthermore, use of uniform structures and vocabulary will eventually lead to the exchange of medical records and decision logic between unlike health care institutions. Translation, if needed, is an easier task because the vocabulary is structured on both sides of the interface.

The domain of the system should be comprehensive, spanning all aspects of patient care, including: laboratory, pharmacy, radiology, history, and physical exam data. Since these types of data are varied in content and format the DB and the DD must support a wide selection of data structures as well as vocabulary elements.

If a MIS is to stay comprehensive it must also be flexible. System flexibility includes the ability to customize input screens and reports, change file and index configurations, and allow for the addition of entirely new functions. Since the DD is the source of system functionality, system flexibility has implications for the DD. To provide new functions and to support end user configuration of the system, the DD must be able to expand and change its vocabulary. As medical knowledge, procedures, and treatments grow, new classes of data elements will be needed. Users need the facility to define new values and data items and to use these items to tailor input screens and output reports.

Since evolution of system vocabulary is inevitable, the system not only must allow change, but must also have the ability to track and make changes. Thus, it is important to determine if a proposed term already exists or is a synonym for an existing term, and to record the date that a new term became available in the system, and when an old term became obsolete. When replacements occur it is necessary to link the obsolete item to the new item. Whenever data items are referenced explicitly in program code there is a need to coordinate and link vocabulary updates to any needed program changes.

In spite of the need to be flexible, extensible, and comprehensive, a medical information system should be computer resource conscious. Since the DD is part of the system, this dictum applies to the DD vocabulary also. One way to increase expressiveness without drastically increasing size is to provide a grammar for the vocabulary that allows modifier relationships among the terms. A "modifier relationship" means that terms gain semantic meaning by juxtaposition and context of usage, just as words in a sentence do. This concept will be explained further when the DD entities are introduced.

Finally, a MIS should provide a robust set of decision support tools. Minimally, these tools must define and manage rules and do data driving. Types of rules include: data entry validation, alert specifications, diagnostic algorithms, and other contingency actions. Data driving means that the system can take actions based on a rule set that is evaluated as data is stored in the system, as described by Pryor *etal.* [2]

Decision support using rule definition and evaluation can only take place if the information domain is well defined and structured. Another way of stating this restriction is that rules can not produce reliable results when executed against data that is ambiguous. Practically, this means any data that is used in the evaluation of a rule must have its domain explicitly defined. For numeric quantities, the possible values must come from the set of integers or real numbers. For a text quantity (such as a diagnosis, physical finding, drug name, or treatment) the value must be validated against a list of specified items. For example, if the system is expected to check for drug allergies when a pharmacy order is entered, the name of the drug entered can not be an arbitrary text string, but must be a string that matches a known drug name.

An important strategy for structuring text in the system is encoding. Encoding means the use of a number or code internally to represent the text string selected. Encoding reduces the space needed to store textual data and improves performance since searches criteria use numeric comparisons instead of text string comparisons. Also, numbers or codes can be used to create relationships between terms that are not apparent in the text strings. However, as useful as encoding is, users should not be burdened with the encoding-decoding process. They should only have to deal with natural language vocabulary, and the details of encoding-decoding should be handled invisibly by the system.

Conceptual Model

Our conceptual model of the "real" world as constructed from analysis of medical care is described by three concepts: event, attribute, and attribute domain. Each concept will be defined and described in turn below.

An event is a named action or state of being that exists in relationship to an object, usually the patient. An event is uniquely identified by its name, and the time that the event occurred. The following are examples of events:

- Writing an order for a medication
- The occurrence of a pain
- The birth of a child
- A surgical procedure
- The generation of a hypothesis by a physician or computer
- The receipt of a laboratory result
- The asking of a question about a previous birth
- The recording in a computer the answer to a history question
- Creation of a discharge summary
- Posting a bill for a clinic visit

Medical events are the basis for all medical knowledge. It is from individual events that rules are formulated, that statistics are created, that experience is gained, that bills are generated, and that diagnoses are generated. Medical knowledge amounts to knowing how to categorize, synthesize, characterize, quantify, analyze, and take action on medical events. The storage, review, and analysis of medical events is the domain of medical information systems.

Events are like relations in the relational model. [3] The major difference is that the event concept is meant to emphasize the time sequence of a relationship as well as to recognize that in the medical domain most actions or states of being have the patient as the object.

An event has associated attributes. In general, attributes are qualitative and quantitative descriptors of a given event. Attributes of events are strictly analogous to attributes in the relational model. Every event has a time associated with it, namely, the time that the event occurred. Every event that is recorded also has a second time associated with it, that is, the time that the event was recorded. Distinguishing these two times and using the

information appropriately is often critical for medical diagnostic logic. If the two times are close together, it may not affect patient care to consider them as equivalent for some purposes, but conceptually they are distinct.

A second attribute of medical events that is always present is an object. Usually the object of a medical event is a patient, but for descriptions of family history, the object may be a named relative of the patient.

Besides time and an object, an event has other attributes. The number and type of attributes depend on the particular event. For example, the birth of a child has the following attributes:

1. The sex of the child born
2. The anesthesia used, if any
3. The duration of labor before the birth
4. The mother's age at the time of birth

The final concept in the abstract model is attribute domain. Attributes can be thought of as having a *name* and a *value*. For any given attribute the possible values come from a known set or domain. Thus, some attributes have values that come from the set of positive integers, like the mother's age in years at the time of the birth of a child. Other attributes have the domain of the set of positive real numbers, as is the case for most laboratory values. Still other attributes have domains that are derived from sets of real world concepts or objects. For instance, a baby's sex at birth is either "male" or "female," while the indication for a surgical procedure could be "acute appendicitis," "fracture of the proximal femur," or any other valid diagnosis name. Expressing and categorizing domains for attributes of this last type is a challenging problem when creating a DD for a medical information system that integrates decision support analysis.

Physical Model

Alone, the conceptual model does not produce a medical information system. To be useful the abstract model must be physically implemented. The DD's purpose is to map the abstract model to a set of physical structures and terms. Thus, it is important to give a brief description of the physical model used in the AT&T CareComm Medical Information System.

The only elements necessary in the physical model are the standard items that are found in most database systems, namely: databases, files, records, and fields. These terms have generally understood meanings and will not be discussed further. The particular DB management system used by us is a transaction oriented network DB management system. The data base management system is written in "C" and operates in the UNIX(TM) operating system environment. These facts are important when considering our particular DD implementation since the network model implies certain physical relationships that must be present in the DD for the system to function. However, because the only physical elements needed to support the conceptual model are databases, files, records, and fields, our information model could be created in any DB system that supports these primary data structures.

Data Dictionary Architecture

Information is entered and edited in the DD through a form-based, menu-driven interface. "Forms" are definitions of screens that are presented to the user for doing DD editing and management functions. Several examples of forms are presented in Figures 1 through 6.

Entity types and entity definitions are the basis for the DD's structure. An entity type is a collection of fields that define the relationships between the components of the conceptual and physical models. By defining the fields that appear on a form used to edit information about an entity, the properties and attributes of the the specific entity type are defined. Information entered through a form is saved in a record in the DD and it represents an instance of an entity type definition.

Referring to Figure 1, there are nine entity types currently supported in the DD: text values, fields, events, hierarchical groups, alerts, files, secondary keys, and indexes. Of these nine, events, fields, text values, groups, and alerts will be examined

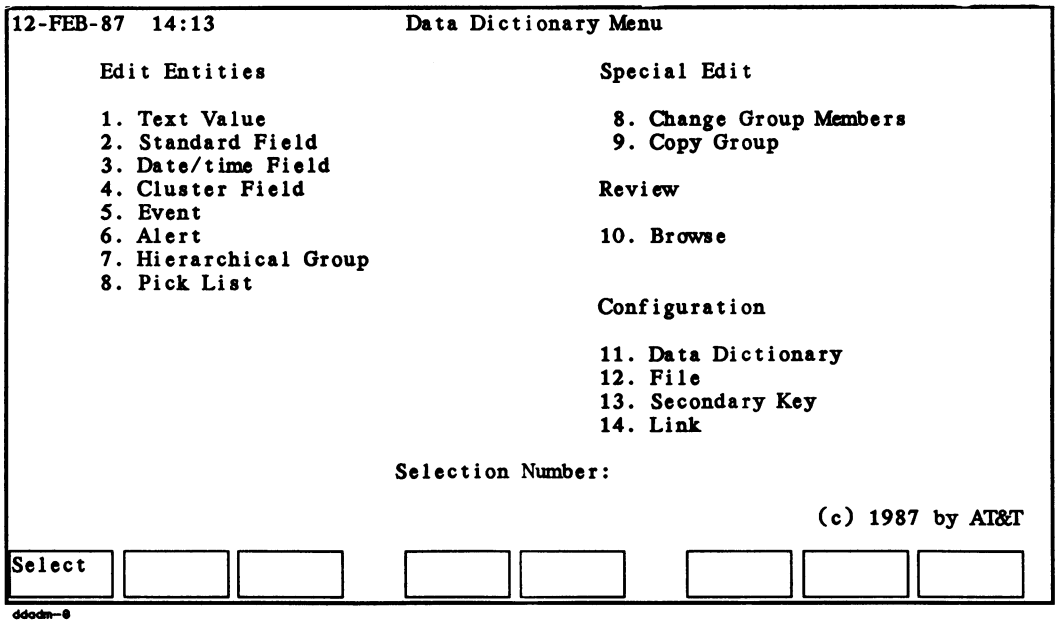

Figure 1: Data Dictionary main menu form.

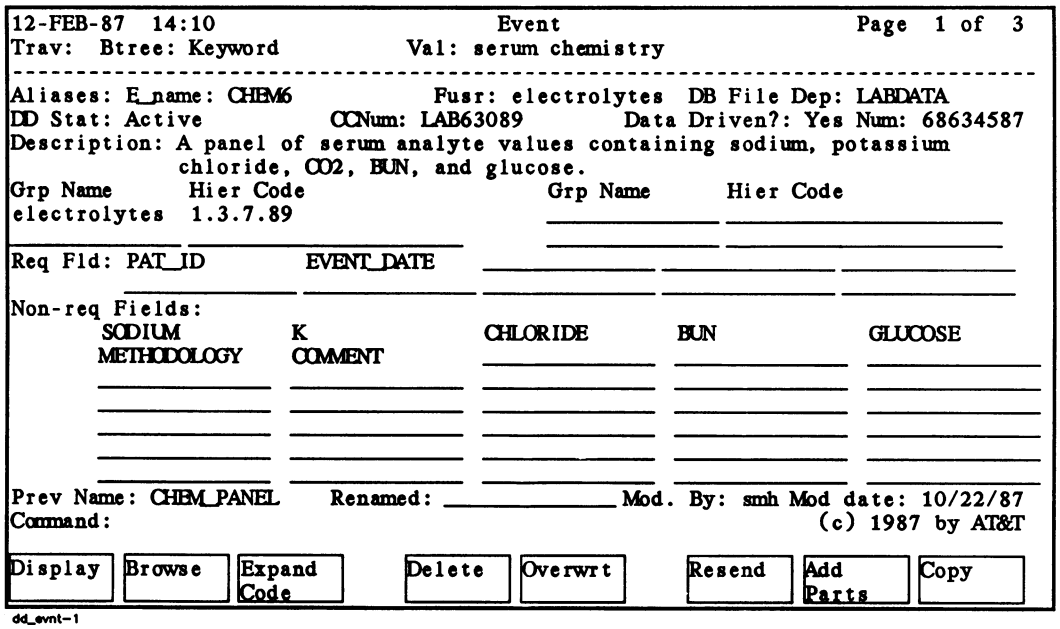

Figure 2: First page of the event definition form.

below. The number of entity types that exist in the DD is open ended based on the functionality that is needed in the system. Entity types that have been recognized but not yet created include: reports, queries, formulary items, and laboratory tests. Other entity types will be added as intelligent system functions require more information for their support.

Event

The cornerstone entity type of the DD is an event definition. Events are defined by using the event form as shown in Figure 2. This figure shows an event form that defines a CHEM6 event. The attributes of a CHEM6 are described by the fields that appear on the form. The layout of the form and the fields shown are described below.

The first two lines of the form are used for general DD information and are not part of the event definition. The first line shows the current date and time to the user as well as the the name of the form and the page number of the form being displayed. The second line of the form is used for traversing records in the DD to search for a specific item. Any secondary index maintained on the DD items can be traversed by indicating the name of the index in the Btree field, and placing the desired index value in the "Val" field. Indexes are maintained on entity names, keywords, entity numbers, and hierarchical relationships.

Several fields on the event definition form are illustrative of fields that apply generally to all entity types recorded in the DD. These include: entity name, friendly user name, DD status, change control number, entity number, description, hierarchical group name, previous name, new name, last modified by initials, and last modified date. On the second page of the form, shown in Figure 3, the "Keyword" field is shown. Keywords are used to make "flat" or arbitrary associations among DD items and are useful for finding related entries during an editing session.

Referring again to Figure 2, several fields on the event form represent attributes that apply to event definitions only. The "DB File Dep" field defines the logical file "LABDATA" where CHEM6 records are stored. The "Data Driven" field shows whether there are rules that need to be evaluated when a CHEM6 record is saved. That a patient identifier and an event date are required for a CHEM6 to be stored is shown by the values PAT_ID and EVENT_DATE in the required fields section of the form. Some optional attributes of a CHEM6 are listed in the non-required fields section, including the name of the analytes that are part of a CHEM6 and the methodology used to perform the analysis.

An important concept that allows conciseness in the DD vocabulary is the modifier relationship that exists between an event definition and its attributes. For example, the field "SODIUM" exists as a single item in the DD, but may participate in many events. Sodium is found in CHEM7 events, CHEM20 events, and URINE_ELECTROLYTE events. This means that when a record of an instance of a CHEM6 or CHEM7 is stored in the system, the same SODIUM field is used to hold the value of a sodium analysis. That a particular sodium is a urine sodium or a serum sodium is apparent because of the context of usage. The name of the event modifies or provides context for its fields. Thus, when combined in various records SODIUM really represents a specific type of sodium like CHEM6 SODIUM, or CHEM7 SODIUM. The modifier concept allows remarkable breadth of expression while substantially reducing the number of field definitions needed in the DD. It is also a concept that can be extended to relationships between other entity types in the DD, as will be shown later.

Event definitions are the pivotal entity type of the DD because they provide context and structure for all other entity types. Besides the modifier relationship event definitions have with fields, they are the focal point for enforcing uniformity of data representation in the DB. Recording the structure and location of events makes the DB understandable to both users and processes. The concept of "sodium" can be examined by a user who is creating a query or report, since all events that the "SODIUM" field participates in can be displayed for examination. Alerts are created by referring to the event definitions as well, and the data driving flag associated with each event definition serves as a central control point for rule evaluation.

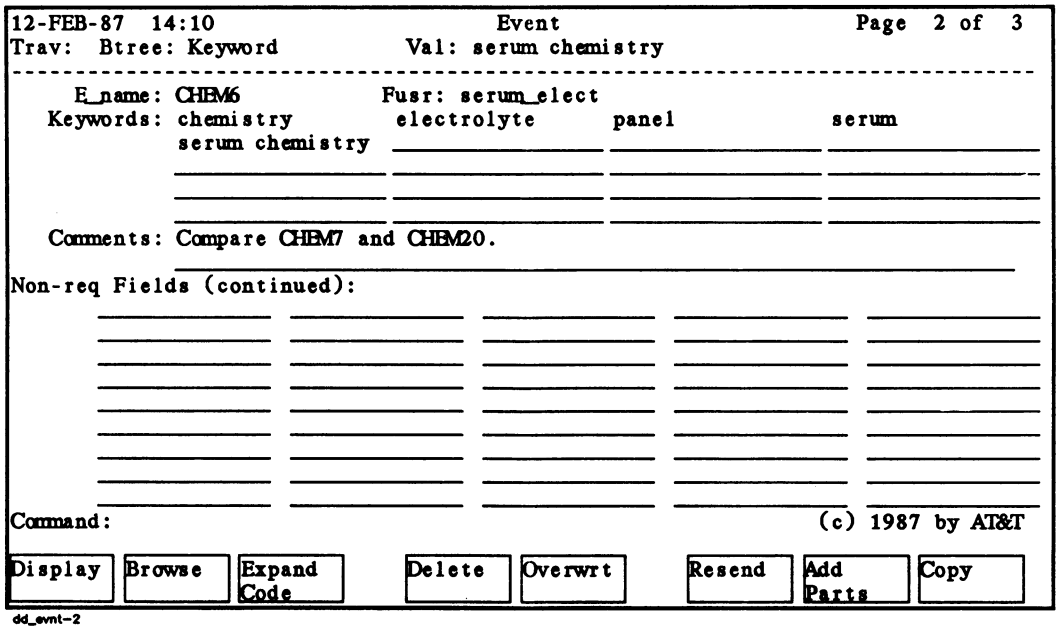

Figure 3: Second page of the event definition form.

Fields

Event definitions in the DD define record types in the DB. Records in turn contain fields. Fields in the DB are name-value pairs. Fields are defined in the DD by their own set of fields as shown in Figure 4. Some important fields include: long prompt, short prompt, display width, number of display lines, display format, default value, validation expression, and a column heading for reports.

The value part of a field can be of several different kinds. The standard field types include numeric, time, and arbitrary text strings. There are three special field types that are worthy of individual comment: date/time, code, cluster.

The date/time field type is a structure that is used to record ordinal time, and includes specification of both date and time in a single field. Additionally, the granularity, uncertainty, and offset to Greenwich Mean Time are recorded.

Coded fields are stored in the DB with a numeric quantity as their value, with the number representing a text string. The DD provides the mapping between the stored codes and the text strings that they represent. The text strings that are represented by codes are called "Text Values" and are an independent entity type in the DD that will be described later. The range of values that a coded field can take are defined by the domain group name. The domain group name identifies a collection of text values that are valid values of the field. For instance, for the field "DRUG_NAME" the domain group name is "DRUGS," where the group "DRUGS" might contain the text values "Penicillin," "Digoxin," and "Morphine." The use of coded field types provides structure to what is normally an arbitrary text association and also reduces substantially the amount of storage space needed. Since the codes are numeric there is also a processing advantage when a large number or records are examined to satisfy a query.

Again, the concept of a modifier relationship exists between text values and fields. Consider a group of text values representing the names of red blood cell antigens. If a member of the group, say "KELL," is taken as the value of "RED_CELL_ANTIGEN," it refers to the antigen found on a red cell during blood bank testing. However, if the term "KELL" is taken as the value of the field "ANTIBODY," it refers to the specificity of an antibody identified in a serum sample. Just as the number of fields needed to express concepts is decreased by the modifier relationship between fields and events, the number of text values needed to express relationships is decreased by the modifier relationship between text values and fields.

The final special field type is called a "cluster," because it represents a set of related fields. That is, a clustered field represents a field whose value is segregated into sub-fields. This type of field is useful for associating fields related to a specific attribute of an event. For instance, in the CHEM6 event definition one of the fields in the event is a "SODIUM" value (see Figure 2). In reality, a sodium analysis has more than just a single parameter as its outcome. The "SODIUM" field is actually a cluster field that has sub-fields of "VALUE," "ABNORMAL_FLAG," "HIGH_NORMAL_VALUE," and "LOW_NORMAL_VALUE." It is apparent from this example that cluster fields are another instance where the modifier concept is used. "VALUE" may be a sub-field in many clusters, and the meaning of the particular value is dependent on the cluster that it is in. Again, "VALUE" in a SODIUM cluster really means "SODIUM VALUE," whereas "VALUE" in a POTASSIUM cluster means "POTASSIUM VALUE." Clusters can be thought of as parentheses within a record.

Text Values

As described above, text values are character strings that are represented by a code. The codes that are used consist of two parts, an alias number and an entity number. The entity number portion of the code is a unique identifier of a text value, while the alias portion of the code specifies a particular text string "alias" for the term. Alias in this discussion means semantically equivalent. Referring to the text value definition shown in Figure 5, the item has an entity name of "HYPERTENSION," with aliases of "Idiopathic Hypertension," "Hypertension," and "HTN."

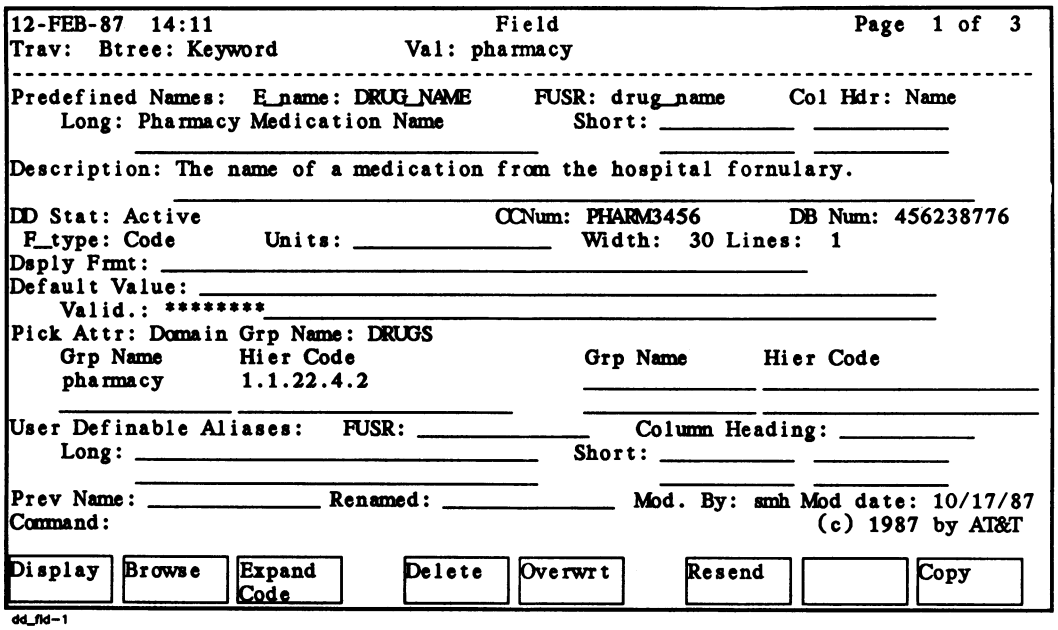

Figure 4: First page of the field definition form.

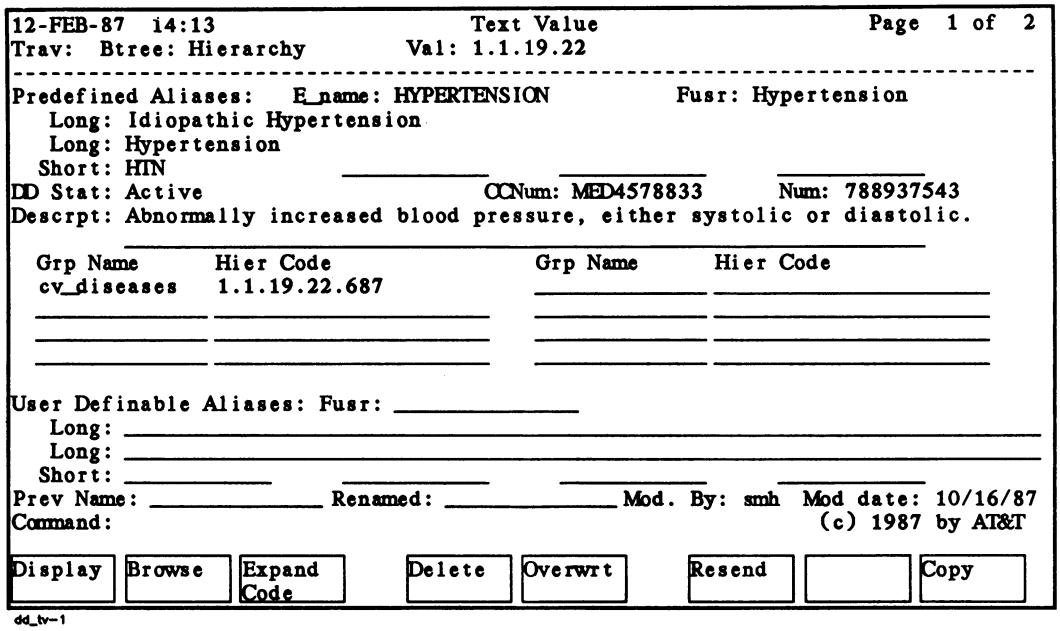

Figure 5: First page of the text value definition form.

When this text value is taken as the value of the "DIAGNOSIS" field, the code stored identifies that the diagnosis is "HYPERTENSION" and also that the alias, "Hypertension," was the particular text string entered. Aliases are useful for allowing people to refer to semantically equivalent items by the names that they prefer. Note that the meaning of the term is defined in the description field, and this definition is the final arbitrator of the term's meaning, not the names by which it is known. The use of aliases for this term allows a person to look for patients in the DB who have a diagnosis of "Hypertension," and the system will recognize patients with the diagnosis of "Idiopathic Hypertension" as also having hypertension. Terms that are related to "Hypertension" but are not identical to hypertension can be associated by keywords or by the use of hierarchical groups.

Hierarchical Groups

One special entity type supported in the DD is a "hierarchical group." A group is a named collection of other DD elements. Groups are used to define the set of text values in the domain of a coded field, to show hierarchical relationships between data elements, to provide structure and organization to DD entries for ease in maintaining a consistent and non-redundant vocabulary, and to support smart behaviors in the system such as queries by field group name and retrieval by event group name.

"Groups," in a more general sense, can be thought of as a convenient way for describing relationships between data elements in the DD. For example, the set of text values in the domain of a coded field can be thought of as defining an "is a value of"

relationship between the text values and the coded field. Groups can be used to describe any "*is a*" relationship, such as, "Amoxicillin *is a* penicillin," or "anemia *is a* blood disorder." Recording conceptual relationships in the DD provides the structure for a "group query" that could be phrased like "Is the patient on any drug that *is a* member of the penicillin drug group?" A group can also be defined to represent "all the events that are used to store laboratory data." This can be used by applications to provide automated summaries or other "smart" behaviors related to laboratory data.

Groups can be formed from elements of a single entity type such as text values, fields, or events, or groups can contain a mixed collection of entities, such as all text values and fields used by the laboratory application. Groups of fields are distinguished from "events" because event definitions and group definitions have different purposes and uses. An event definition specifies fields that can be stored as a set in a record in the DB, and it includes information about what fields are required, when the attributes were added to the event, etc. A group of fields, on the other hand, is simply a named list of fields without any further qualifying information about the elements, and it is used principally to allow organization of the "fields" that exist in the DD.

Groups in the DD are arranged hierarchically for maintenance and so that simple inheritance can take place when this is a desirable property. Several levels of the hierarchy are discussed below. The first level of the hierarchy contains only the root node. The second level of the hierarchy makes the distinction between groups that are homogeneous (only contain data items of a single entity type) and groups that are heterogeneous (contain data elements of more than one entity type). The homogeneous groups are further categorized according to the entity type of the elements that the groups contain, otherwise, beyond the second level, the structure of the hierarchy is dictated by the functional needs of the system.

Data elements become members of a group by being assigned a hierarchical code that is subordinate to the desired group. Groups become members of other groups by the same mechanism. Any new data item added to the DD must have at least one hierarchical code, which is another way of saying that every data item must be a member of at least one hierarchical group. This policy is enforced for purposes of ease in DD vocabulary maintenance. However, data items can have as many hierarchical codes as needed. Allowing a data element to reside at multiple places on a hierarchy makes possible the representation of more than one logical relationship for each data element. This capability becomes particularly useful when dealing with complex data elements such as the names of diseases or syndromes, the names of medications, and the names of anatomic sites. In general, the DD supports the concept of a "complex tangled hierarchy" as described by E. R. Gabrieli, M.D.[4] [5]

Alerts

The final entity type to be discussed is an alert definition. Alerts are simple rules. Rule evaluation is triggered automatically when data is entered into the system, for those events that have been specified as being "data driven." Currently the alert definition language is limited to boolean expressions, as shown in Figure 6. The boolean "or" is represented by "||" in the expression, and the boolean "and" is represented by "&&." The output from an alert evaluation is a record written in the DB, and a mail message sent to the patient's physician. By storing the alert output in the DB, the alert result again represents data to the system, and by establishing secondary rules that feed on the results of previous alerts, a forward chaining loop is established in the system. As needed the alert specification language will become more sophisticated as will the types and destinations of the evaluation results. This will mean that new DD tools and entity types will be needed to manage the knowledge base as it grows.

Conclusion

Analysis of our real world needs lead to the development of a conceptual model of information management that had three main elements: events, attributes, and domains. To support our desired set of medical information management functions we have created a DD that supports our conceptual model in an actual DB system.

The DD is organized around entities, which are types of information needed to support the conceptual model of events in the DB. The most important entity types in the DD include: events, fields, clusters, text values, hierarchical groups, and alerts. The structure of the DD allows indefinite growth in both the types of entities defined in the DD and in the number of definitions that exist for each type of entity. At the same time modifier relationships between events, fields, clusters, and text values provides a vocabulary that is expressive but concise. The use of event definitions and encoding of text insures that data in the DB has a known structure and location making the DB contents understandable by both users and intelligent processes. Alert definitions based on events and coupled with the data driving function make the system active in patient care. Future work on the DD includes expansion of the rule definition language and development of tools to manage a large knowledge base.

REFERENCES

1. Flavin, Matt, *Fundamental Concepts of Information Management*, Yourdon Press, New York, N. Y., 1981.
2. T. A. Pryor, R. M. Gardner, P. D. Clayton, and H. R. Warner, The HELP System, *Journal of Medical Systems*, 7: 87.
3. E. F. Codd, A Database Sublanguage Founded on the Relational Calculus, *ACM SIGFIDET Workshop on Data Description, Access and Control*, November 1971, 35-61.
4. E. R. Gabrieli, "Nosologic Standards for the Construction of a Medical Nomenclature," Draft Document of ASTM, October, 1986.
5. E. R. Gabrieli, "A New Nosologic System - A Proposal," Gabrieli Medical Information Systems, Inc. Statler Towers, Buffalo, NY 14202.

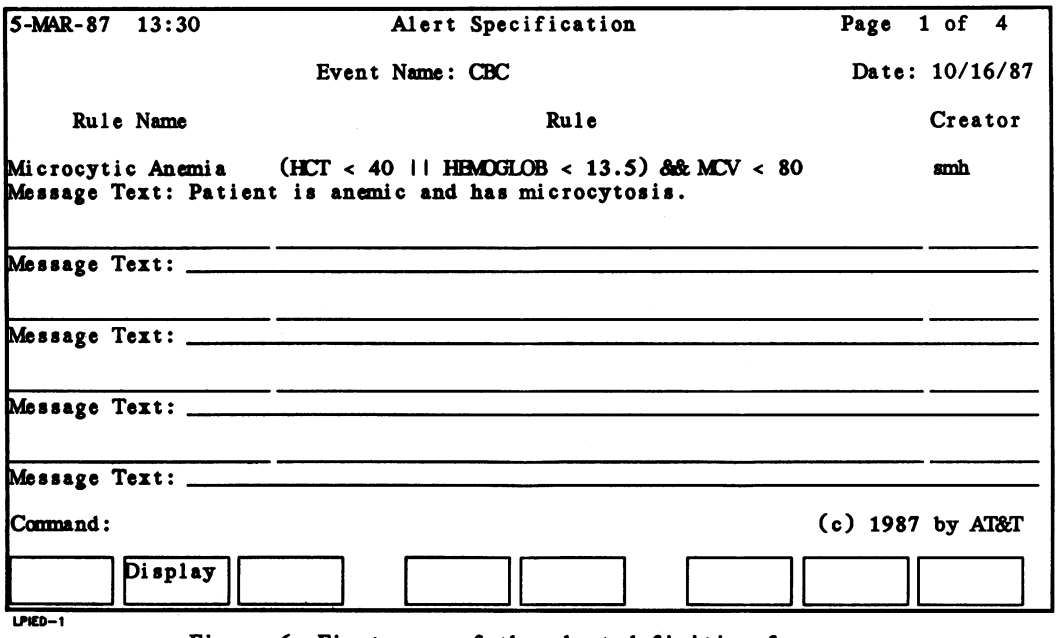

Figure 6: First page of the alert definition form.