A Functionality-Oriented Criteria Set for Evaluating Information Resource Dictionary Systems

Bijoy Bordoloi, Sumit Sircar, Craig Slinkman
University of Texas at Arlington

Nitant Chakranarayan
Database Consultants, Inc.

This paper develops a detailed set of criteria with which the functional capabilities of different Information Resource Dictionary System (IRDS) products can be compared. The proposed criteria are primarily based on the ANSI and ISO standards for IRDS and related published literature in the field. The criteria set can be used as a guideline by vendors to streamline their products, or as a framework by potential adopters to compare and select an IRDS from multiple offerings.

In recent years, it has come to be recognized and accepted that data is as valuable a corporate resource as any other corporate resource such as money, material, or personnel, and that, like these resources, data must also be administered and controlled to ensure proper handling, proper access and proper utilization.

Data administration is the function that assists the organization in the management and control of data. It includes a human facility, the data administrator, and an automated facility called the Information Resource Dictionary System (IRDS), some common implementations of which are known as the Data Dictionary/Directory System, the Repository, or the Encyclopedia (Cashin, 1988). The data administrator acts as a manager of the corporate data resource. The IRDS incorporates a centralized "repository" of information about data relevant to the organization. The repository is an administrative data base that allows storage and management of all the database and related information system definitions, referred to as metadata. It contains the attributes, domains, definitions, usage and relationships of the data in an organization. The major benefits of a central repository include:

Establishing a single source of metadata to be shared and reused by various users and tools throughout the lifecycle of an application or information system

Reducing or eliminating duplication of effort in creating and managing shared data

Promoting and enforcing consistent definitions across interrelated application areas

The concept of storing data about an organization's data and its information systems originated over three decades ago, with the use of COBOL copy statements to provide consistent data declarations that could be referenced by applications (Gillenson, 1993). This was followed by a succession of products, such as of data dictionaries, and catalogs for relational DBMS. As a result of a proliferation of work in this area,

Manuscript originally submitted August 16, 1993; Revised March 14, 1994; Accepted May 4, 1994 for publication.

both the American National Standards Institute (ANSI) and what is now the National Institute of Standards and Technology (NIST) began to work on data dictionary standards in the early 1980s. They named the software "Information Resource Dictionary Systems," and formulated a set of standards (X3.138) in 1988 (ANSI, 1989). These standards were recently extended in 1992 (ANSI, 1992).

Because of its growing importance, IRDS implementation has been a major software endeavor for many software companies, large and small. For instance, IBM has announced a product called Repository for DB2, its relational DBMS. Furthermore, the concept of a repository has been most successfully implemented in ICASE (Integrated Computer Assisted Software/System Engineering) technology. In fact, the repository or encyclopedia is the cornerstone of all the well-known ICASE packages on the market. Some examples are: ORACLE corporation's CASE*Dictionary; Texas Instruments' IEF (Integrated Engineering Facility), and Knowledgeware's ADW (Application Development Workbench).

Several third party vendors have also introduced their repository products, and this market seems to be constantly expanding. The problem, however, is that there is a dearth, in the published literature, of a well-defined set of criteria to evaluate the functional capabilities of these products. This, needless to say, makes it extremely difficult for IS management to evaluate and select an IRDS product. This is the problem that we address in this paper as we develop a set of criteria with which the functional capabilities of IRDS products can be compared.

Several authors (Bruce, Fuller, & Moriarty, 1989; Narayan, 1988; Plotkin, 1992) have suggested criteria to select an IRDS, but they lean towards the business decision-making aspect of choosing an IRDS for an organization. Their criteria, for example, include marketing-oriented features such as vendor support and vendor reputation. The criteria of these authors do not focus in detail on the functional power of the IRDS and the overall functionality of the IRDS. Building on the works of these authors, we present a more detailed set of ***functionality-oriented criteria*** with which the capabilities of IRDSs can be compared (ignoring business related criteria such as cost, vendor reputation or vendor support).

Our proposed criteria are based on published literature in the area of IRDS, the ANSI and ISO standards proposed for IRDSs (primarily ANSI), and inputs from several practitioners in the field. We assume that the reader is familiar with the basic concepts and terminology of database management, particularly the concepts of database schema, metadata, the E/R model and the relational data model.

The rest of the paper is organized as follows. First we provide a summary discussion of some basic IRDS concepts and IRDS database architecture. The discussion presented in this section forms the basis for our proposed criteria which we discuss in the next section. We then provide some methodological guidelines for using the proposed criteria set, followed by concluding remarks.

IRDS Concepts and Database Architecture

Before we discuss the architecture and desirable functional features of IRDS, it is perhaps useful to review some basic IRDS concepts and the evolution of IRDSs from simple manual systems to complex automated ones.

Evolution of IRDS

In the early days of computing, data were stored in the form of small isolated islands, and were controlled by independent programs, so programmers and system developers needed a reference list for the names and definitions of the variables used in their programs. The need for sharing data amongst different applications was apparent even then. It was necessary to keep track of the data, their meaning and the relationship between the data and programs. Gillenson and Frost ([Gillenson & Frost, 1993) have traced the evolution of the concept of storing and using data about data, i.e. metadata. They found the following overlapping stages in the development of this concept: passive data dictionary, active data dictionary, relational catalog, hybrid relational data dictionary, American National Standards Institute (ANSI) Information Resource Dictionary System (IRDS), repository, and OODBMS catalog.

Passive data dictionaries were the earliest form of data dictionary. They are called passive, because they are used independently of the running of the DBMS, typically as system documentation tools. Well-known examples of such products for specific DBMS are University Computing's UCC TEN, IBM's DB/DC and Cincom's Data Dictionary. Active data dictionaries represented an enhancement of the data dictionary concept. They are called "active" because the metadata they

contain are needed for the execution-time environment of a DBMS. An example of a product in this category is the Integrated Data Dictionary (IDD) of Computer Associates' IDMS/R DBMS.

The next stage is the relational catalog. This catalog was necessitated because of the unique requirements of relational DBMSs. The latter do not possess the direct address pointers of previous types of DBMSs, so sophisticated query optimizers had to be developed to determine the most efficient manner in which to provide data in response to a query. The relational catalog (a set of relational tables) was created to store lower-level operational data about the relational DBMS environment, typically including descriptions of databases, tables, views, field types, foreign keys and indexes. Products in this category are included for example, in IBM's DB2 and SQL/DS relational DBMSs, and the ORACLE and INGRES relational DBMSs. A logical extension of the concept, adopted by several organizations that were early consumers of relational DBMSs, was the formulation of the data dictionary as a full-blown relational database application. An example of a vendor offering of this type is IBM's Data Base Relational Application Directory (DBRAD), introduced in 1986.

Occurring parallel with these elements, from the early 1980s, has been a continuing effort by the ANSI and the ISO to formulate a set of standards for data dictionaries. The fruits of their endeavors are described in the next section as they are central to the objectives for this paper. The standards approved by the ANSI's X3.138 (ANSI, 1989) and ISO's SC21 standards (ISO, 1989) in the late 1980's were necessary because over the years the term IRDS had come to mean different things to different audiences, and there were no common rules to guide developers. Consequently, the result was incompatibility among IRDS implementations, and severe limitations on the portability of IRDS applications.

The next stage in the evolution of data dictionaries was driven by the needs of the Computer Assisted Software Engineering (CASE) approach to application development. This approach embodied more comprehensive, automated support for the complete information environment within an organization, with support for the operational aspects of that environment and storage of the relationships and locations of all the information environment components (Appleton, 1987). There is obviously a substantial overlap between the functions of the metadata for traditional data dictionaries and those supporting CASE environments, which have come to be known as "repositories". We therefore use the terms IRDS and repository interchangeably in this paper. Products in this category include Brownstone Solutions' Data Dictionary/Solution (DD/S), introduced in 1986, Reltech Products' DB EXCEL Repository, and IBM's Repository Manager/MVS, first marketed in 1990.

Finally, the emergence of OODBMSs has led to the last of the seven stages in the evolution of data dictionaries—the OODBMS catalog. This is similar to the relational catalog, but designed for the functional requirements for metadata storage in an OODBMS environment. OODBMSs are capable of handling complex data structures, so the corresponding catalog must be capable of supporting queries involving these complex objects.

IRDS Database Architecture

As indicated in the previous section, *an IRDS is a database application that manages a centralized collection of data about all the relevant information resources within an organization.* An IRDS is composed of a database component, the Information Resource Dictionary (IRD), and other components such as a query facility, and a report facility. The IRDS database (i.e., the IRD) is the ***heart*** of an IRDS. The IRD contains information about entities, data elements and their attributes such as size, type, where and how they are used and their relationships with other entities or elements. This information is represented in the form of meta-entities such as tables, records or elements. This metadata should not be confused with user data, as metadata are used to identify, declare, and describe the characteristics of user data (Dolk, 1987; March & Kim,1988-89; Van Duyn, 1982).

The ANSI IRDS is a multidimensional model, which gives a view of the data ranging from extremely conceptual to actual physical storage. It is based mainly on the Entity-Relationship (E-R) model, which specifies information in terms of entities, attributes and relationships.

The ANSI IRDS Database can be viewed as a four-level architecture in which the information specified at one level describes (and potentially controls) the information stored at the next lower level. Thus, one level defines the types of "objects" which can be described at the next lower level, and that level contains the "instances" of those types. As illustrated in Figure 1, these four levels are:

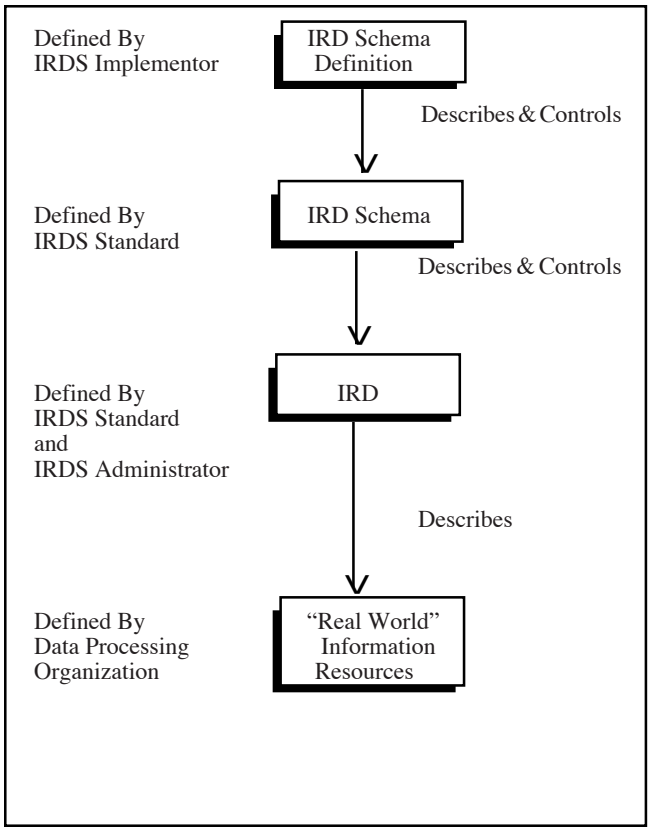

Figure 1: The Four-Level Architecture of the IRDS Database (Source: American National Standard X3.138-1988)

1) IRD Schema Definition Level
2) IRD Schema Level
3) IRD Level
4) "Real World" Information Resources (or Production Data)

As shown in Figure 1, the IRD Schema Definition level, the top level of the four-level architecture, is defined by the IRDS implementor. This level contains the types of objects which can be defined at the next lower level, the IRD Schema, the types of the relationships which can exist between these types, and certain properties of both of these types. These types are referred to as "meta-entity-types", "meta-relationship-types" and "meta-attribute-types." The prefix "meta" as seen from the previous section signifies a concept of data about data.

For example, meta-entities can be entity-types, relationship-types, or attribute-types. An occurrence of an entity-type describes a particular meta-data type such as "record". All entities of a given type will have the same types of attributes and relationship types. An occurrence of a relationship-type describes a particular meta-data relationship-type such as "record-contains-field". All relationships of a given type involve the same types of entities and attributes, and they are binary in nature. An occurrence of an attribute-type describes a particular attribute type such as "length" or "date-created". A meta-relationship is a directed association between two meta-entities such as "relationship-type-contains-attribute-type". An occurrence of attribute-group-type describes a collection of related attribute types which are sometimes grouped together mainly for descriptive purposes.

The second level from the top is referred to as the Information Resource Dictionary (IRD) Schema. This level defines the types to be instantiated in the next lower level, the IRD. The IRD Schema describes the structure and contents of the IRD. It also defines various control mechanisms, including naming rules, defaults, and validation information for the IRD contents. Thus, for every entity, relationship, attribute, and attribute-group that occurs in the IRD, the IRD Schema will contain a description of the corresponding entity-types, relationship-types, attribute-types, and attribute-group-types.

The ANSI standard specifies a "Minimal IRD Schema" consisting of those "meta-entities", "meta-relationships", and "meta-attributes" necessary to establish controls over the IRD Schema and the IRD. This set includes entity-types such as "document", "file", and "record", attribute-types such as "length" and "number-of-records", and relationship-types such as "contains", "processes" and "responsible-for".

This "minimal" Schema can be used by organizations to describe most systems, but the IRDS also includes an Extensibility Facility that allows an installation to customize the IRD Schema to be responsive to its own requirements. This means that the installation can add those entity-types, relationship-types or attribute-types that will allow it to use the IRD as a repository for data that is of interest to that installation. This feature enhances the power and flexibility of the IRDS as an effective tool for managing metadata. It provides a three-dimensional enhancement to the IRDS, because the user can add new entities, enhance the description of an existing entity with new attributes, and establish new relationships between new and existing entities, between existing entities or between new entities (Allen, Loomis & Mannino, 1982). It should, however, be noted that an installation is generally not allowed to modify the Minimal IRD Schema (some modification is sometimes permitted). For this reason, the Minimal IRD Schema is

also often referred to as the Core Standard Schema.

The third level is the IRD. This level describes the environment being modeled. It describes the objects in the environment and the associations among those objects; the object descriptions are called entities, and the association descriptions are called relationships. This level also describes the properties of the objects called attributes and their associations. Thus the IRD layer describes the actual entities, relationships and attributes of a real-world database application, such as the "employee-record" or "social-security-number".

The fourth level, which is not described in the current ANSI standard, is the information resource environment; i.e., the "real world information resources", descriptions of which exist in the IRD. This production data layer deals with end-user data and is not a functional part of the IRDS. It is a conceptual storage area containing actual instances of data, such as an occurrence of the entity "customer", identified by "John Doe".

Thus, the IRD Schema Definition layer, the IRD Schema layer and the IRD layer are the three functional parts of the IRDS. The IRD layer contains the IRD data such as "payroll-record", which is an instance of the entity-type "record" in the Schema layer, which in turn, is an instance of the meta-entity "entity-type"at the Schema definition layer.

The ISO model, although using different terminology, is quite similar to the ANSI model, as it is based on the evolving ANSI model. The ISO model is also a four-layered model just like the ANSI model, and the definitions of the layers are also essentially the same (ISO, 1989; Protocols standards and Communications Inc., 1989). The ISO model, however, contains some data modeling features which may be viewed as enhancements to the ANSI model. The major differences, which bear relevance to our proposed criteria, are discussed below.

One major difference is in the area of modeling constraints. For example, consider two entity-types table and row, and a relationship type, contains. Therefore, to represent "table-contains-row", the model needs a way to specify values associated with objects, and on associations between objects, also known as cardinality. The ANSI model does not support this concept, nor does it offer a means to specify mandatory or required attributes. Therefore, all relationships have to be defined as either one-to-one or one-to-many, as there is no direct representation for many-to-many relationships. The ISO model supports both cardinality and a means to specify mandatory and optional attributes.

A second major difference is in the area of degree of relationships. The ISO model permits N-ary relationships, but the ANSI model allows for only binary relationships. Thus, in the case of a ternary relationship, the ANSI model forces the user to reclassify the third relationship as an artificial entity and relate it through binary relationships with the other two entities. This confuses the semantics of the true world model and creates unnecessary entities and relationships. Further, in general, the information represented by a ternary relationship is not necessarily equivalent to a combination of three binary relationships. If forced to do so, it may result in what is known as the 'connection trap' or the violation of fifth normal form (Date, 1990). It may, however, be noted that recently both ANSI and ISO have decided to put aside their differences and plan to work toward a uniform set of standards, which should offer a new vision to the world of IRDS (Jones, 1992).

Vendor Implementations of IRDSs

As stated previously, IRDSs can be ***passive*** or ***active***, depending upon the extent of the integration of an IRDS with the host DBMS (Leon-Hong and Plagman, 1982; Meador, 1987; Narayan, 1988; Ross, 1981; Venkatakrishnan, 1988; Weldon, 1985). Almost all modern IRDSs, however, are of the *active* type and integrated with CASE tools and/or the host DBMS. There are two types of active IRDSs: Active in development and Active in production, each of which is described below.

Active in development: In this mode the IRDS is used by users as a documentation tool and by automated development tools, such as CASE tools, during the application development process. Therefore, an analyst can store his E-R diagram in this type of IRDS, but the IRDS does not actively support application programs in the production environment.

Active in production: In this mode, the IRDS is integrated into the DBMS to the extent that the DBMS uses the definitions from the IRDS at run time. Therefore, in an active in production IRDS, every request for data is addressed to the IRDS, which has control over how the data are cataloged, where they are located and who has access to them. The IRDS, then directs the DBMS to obtain the data item and apply any operations on the item before presenting it to the user. These IRDSs tend to suffer performance problems because of the amount of activity that has to be coordinated through the

dictionary at run time.

However, active in production IRDS is desirable for several reasons. First, it minimizes data redundancy because a data element is defined only once. Second, it provides data integrity because the IRDS contains the most current data, and thus is a very valuable reference and cross-reference tool for users. Third, it enforces uniform standards and security controls throughout the system. Finally, it helps to control the costs of developing and maintaining applications because of the accurate and complete data definitions from the IRDS.

IRDS Evaluation Criteria

The IRDS database architecture discussed previously provides the basis for formulating a set of "core" criteria by which to evaluate IRDS offerrings. However, in actual organizational usage, commercial IRDS implementations would have to possess many more features and capabilities than those covered by the architectural standards. An additional set of criteria are needed to evaluate these features. As shown in Fig. 2, we propose nine "broad" criteria—three for the core, and six for the additional set. These nine criteria are then expanded to thirty-one sub-criteria to provide a detailed framework for evaluation and comparison as shown in Figure 3. The development of the core criteria and the additional criteria set are discussed below.

Core Criteria

1. Ability to Capture Core Entity Structure.
2. Ability to Capture Core Attribute Structure.
3. Ability to Capture Core Entity-Relationship Properties.

Additional Criteria

4. Extensibility Support.
5. Data Documentation and Versioning Support.
6. Security Support.
7. Integrity Support.
8. Input/Output Interface.
9. User-Friendliness.

Figure 2: Broad Criteria

1. Ability to Capture Core Entity Structure.
   - Ability to Capture Data Entities.
   - Ability to Capture System Entities.
   - Ability to Capture External Entities.
2. Ability to Capture Core Attribute Structure.
   - Ability to Capture Identification Attributes.
   - Ability to Capture Representation Attributes.
   - Ability to Capture Statistical Attributes.
   - Ability to Capture Control Attributes.
   - Ability to Capture Physical Attributes.
3. Ability to Capture Core Entity-Relationship Properties.
   - Ability to Name Relationship.
   - Ability to Represent Maximum Cardinality.
   - Ability to Represent Mandatory/Optional Relationships.
   - Ability to Represent Generalization (IS-A) Relationships.
   - Ability to Represent Mutually Exclusive Relationships.
   - Ability to Represent N-ary Relationships.
   - Ability to Represent Recursive Relationships.
4. Extensibility Support.
   - Ability to Add/Update/Delete Entity-types
   - Ability to Add/Update/Delete Attribute-types.
   - Ability to Add/Update/Delete Relationship-types.
5. Data Documentation and Versioning Support.
   - Ability to Capture current Attribute Descriptions.
   - Ability to Maintain Standard Control.
   - Ability to Maintain Version Control.
6. Security Support.
   - Ability to Control Access through Username/Password.
   - Ability to Coordinate Access through DBMS.
7. Integrity Support.
   - Provision of Edit and Validation functions.
   - Provision of Error Reporting functions.
   - Provision of Data Recovery functions.
8. Input/Output Interface.
   - Query Language Support.
   - Command Language Support.
   - Predefined Standard Reports.
9. Use-Friendliness.
   - Ability to Provide Help and Pop-up Screens.
   - Ease of Learning and Using the Product.

Figure 3: Detailed Criteria Set

Core Criteria

For lack of any other appropriate term, we call the first three of the nine broad criteria "core" criteria. These criteria consisting of 15 sub-criteria reflect the desirable characteristics of the IRDS **database** (i.e., the IRD—the heart of an IRDS). Most of the sub-criteria within these broad criteria are based on the ANSI Core Standard Schema -discussed earlier. As implied by the ANSI standards, an IRDS meeting these fundamental criteria should be able to be used by an organization to describe most of its RDBMS applications. Thus, all

IRDSs must possess some functional capabilities along each of these core criteria, discussed below.

Ability to Capture Core Entity Structure. A fundamental characteristic of IRDSs is the *breadth* (entity-types) of metadata that can be described to the IRDS (Modell, 1988). This criterion refers to the ability to capture the descriptions of all the entity-types specified in the Core System-Standard Schema. This helps to evaluate the metadata entity structure of the IRDS. The Core System-Standard Schema as per the ANSI architecture contains 15 entity-types that can be conceptually grouped into three categories: Data, System, and External.

Data entity-types: These entity-types are used to represent objects which are units or aggregates of data. An IRDS should support the following data entity-types:

1) **Database**. It is a collection of DBMS tables, rows, columns, elements, reports, forms, and views, which together act as a storage area for the organization's data.
2) **Table (File)**. It is a set of related occurrences of rows, columns, and elements usually of the same type, such as "customer" or "employee".
3) **Row (Record)**. It is an aggregation of one or more related elements or columns that is treated as a unit, such as the employee record made up of name, social security number, age, and address.
4) **Element**. It is the most basic and elementary discrete unit of data that can be identified and described in an IRDS.
5) **Form**. It is a screen in which data are both displayed and entered, such as the "order entry" screen.
6) **Report**. It is the data extracted from the database, which may be printed or displayed on the screen in a formatted manner.
7) **View**. It is a virtual table, not physically stored in the database, but derived from one or more tables.
8) **Key**. It is a group of one or more columns that uniquely identify a row.
9) **Domain**. It is a set of all possible values that a column in a database table can have.
10) **Index**. It places the database table data in a desired logical sequence regardless of its physical storage sequence, thereby reducing data access time.

System entity-types: These entity-types are used to describe or represent objects that are processes or components of the system. An IRDS should support the following system entity-types:

1) **Program**. It a set of instructional statements that specify actions in a particular programming language.
2) **Module**. It is a subset of statements of a program, which perform a specific task within a program.
3) **System**. It is a collection of related programs, modules or other systems that perform a complete set of functions.

External entity-types: These entity-types are used to describe objects that are connected with the physical external environment in a system. An IRDS should support the following external entity-types:

1) **User**. It represents a person, department or functional group that can be identified to the IRDS in terms of its responsibility and authority.
2) **Hardware Device**. It is used to describe physical devices such as CRTs, terminals, printers or keyboards.

Ability to Capture Core Attribute Structure. Another fundamental requirement of an IRDS is the *depth* (attribute-types) of metadata that can be described to an IRDS. This criterion refers to the ability of an IRDS to capture the descriptions of all the attribute-types specified in the ANSI Core Standard Schema. Attribute-types are used to describe the characteristics of the entity-types described above. An IRDS should support the following attribute-types.

Identification Attributes: They are used to name, describe and identify an entity-type to the IRDS, for example: name, synonym, alias, and description.

Representation Attributes: They are used to describe the properties of an entity-type as represented in the environment, for example: data type, length.

Statistical Attributes: They are used to indicate how the entity-type is used in the overall environment, for example: response-time, and access-information.

Control Attributes: These attributes give information about object ownership, status, security, and access control, for example: authority level, security level, owner, password, and version.

Physical Attributes: They indicate the physical characteristics of all the entity-types, for example: operating system, storage size, etc..

Ability to Capture Core Entity-Relationship Properties

Another fundamentally important desirable fea-

ture of an IRDS is its ability to capture relationships and support multiple relationship-types. Relationship properties enhance the semantics of the associations between entity-types because they capture a precise definition of how entity-types relate to each other. The following are some salient relationship properties that should be supported by an IRDS.

Name: The first property of any relationship-type is its name. The name provides identification for a specific relationship-type, gives it more meaning, and helps in understanding the relationship between two entity-types.

Maximum Cardinality: Another property of a relationship is its maximum cardinality. Maximum cardinality refers to the maximum number of instances in one entity that ***can be*** related to a given instance in the related entity, and vice-versa. Usually, maximum cardinality is specified as one-to-one, one-to-many, or many-to-many. In many cases, however, assignment of a **specific** value to maximum cardinality makes the scenario more meaningful and helps toward better understanding of the relationship. For example, one table has to have a number of rows, and it may not be meaningful enough to just say that one table has *many* rows. Thus, the ability to represent specific values for maximum cardinality should be a desirable characteristic of an IRDS.

Minimum Cardinality: Minimum cardinality is another property of a relationship. Minimum cardinality refers to the minimum number of instances in one entity that ***must be*** related to a given instance in the related entity, and vice-versa. Minimum cardinality is specified as zero (mandatory) or one (optional). For example one table cannot exist without at least one element or data field. This is a "table-to-element" mandatory relationship. Usually, minimum cardinality of a relationship is specified as optional-to-optional, optional-to-mandatory, mandatory-to-optional, and mandatory-to-mandatory.

IS-A Relationship: Sometimes there are entity-types that are composed of a subset of another entity-type. The entity-type that forms the subset is known as a *subtype*, and the-entity-type that contains all the subsets is known as the *supertype*. A generalization relationship, or IS-A relationship, relates a supertype with a subtype. It may be noted that the maximum cardinality of an IS-A relationship between a subentity and a superentity is always one-to-one and the minimum cardinality of such a relationship is always zero-to-one.

Mutually Exclusive Relationships: In some cases an entity may be mutually exclusively related to two (or more) entities. If an entity-type A is mutually exclusively related to two other entity-types, then at any given point in time a given occurrence of the entity-type A can be related to some occurrence of only one of the two related entity-types, not both.

Recursive Relationships: A recursive relationship or loop relationship occurs when an entity-type is related to itself. For example, a "module" entity-type could be "composed-of" other "modules".

N-ary Relationships: Most of the relationships between entity-types are binary in nature. However, in some cases three or more entity-types are simultaneously related. These types of relationships are called N-ary relationships. As mentioned earlier, although this relationship property is not incorporated in the ANSI IRDS Database architecture, it is an important desirable property and is supported by the ISO architecture.

Additional Criteria

The ANSI IRDS standards committee report (ANSI, 1989) and several authors (Glenwright, 1988; Kull, 1987; Leon-Hong, 1982; Narayan, 1988; Ross, 1981) have suggested a number of desirable functional features that should be considered in IRDS evaluation. These features can appropriately be viewed as providing the following types of functionality for IRDS implementations:

1. Support for the Systems Development Life Cycle
2. Support for System Documentation and Standards
3. Support for Data Security and Integrity
4. End user Support

Although this taxonomy may not be exhaustive, it allows us to formulate a useful set of criteria from the perspective of organizational usage of an IRDS implementation. As shown in Figure 2, we propose six additional criteria (consisting of sixteen sub-criteria which are listed in Figure 3), each of which is discussed below.

Extensibility Support

The IRDS should support all the phases of the SDLC such as Analysis, Design, Implementation, Testing, Operation, and Maintenance. For the Analysis and Design phases, the IRDS should have the capability to store and record entity, attribute, and relationship metadata related to these phases. Additionally, the IRDS

should provide a means for maintaining control over the design specifications and ensure consistency between the requirements and implementation. The IRDS should also catalog an organization's data resources and produce various reports of the data definitions, logical groups, process-to-system and program-to-data relationships, which can be very useful to database design.

Similarly, the metadata from the IRDS can be used to support the other phases of the SDLC. During the Implementation phase, the programmer can take the specifications from the Design phase and couple it with metadata to produce program code for the Testing phase and also to be stored as metadata and source data in the IRDS. In the Testing phase the programmer can test the program code from the Implementation phase using reliable test data code to obtain operational metadata that can also be stored in the IRDS.

During the Operation and Maintenance phase, operational metadata from the IRDS can be combined with the tested program code for the final production system. These results can also be stored in the IRDS as a final document that may be used for maintenance of the system. During the Maintenance phase, the final document, along with the input to the system, can be used to maintain the production system. This phase can also oversee and report any updates to the requirement specifications to the users.

Thus an IRDS must permit the user to make essentially unlimited extensions to any IRD schema (ANSI, 1992). This capability would provide the user with the power and flexibility to design a schema to fit the particular needs of the organization or life cycle phase that the IRDS is being used to support. The user would have the freedom to extend the schema in any manner that will be useful. This means that the IRD can be structured to capture any type of metadata that the user can define. Thus an IRDS should allow a user to add/delete/modify the entity-types, attribute-types, and relationship-types in the IRDS.

Data Documentation and Versioning Support

Traditionally, documentation has been the last task to be completed, as it is often not begun until the system is just ready for delivery. It is often considered a dull and routine task, resulting in outdated, inaccurate and incomplete documentation. But documentation should be recognized as a very crucial activity and the IRDS can be used to automatically produce documentation from the database and the system (Durell, 1983). The IRDS can store information about the application system, programs and detailed description of each data element. All this stored information can assist in decreasing the monotony of the task of documentation and producing a well documented end product.

A well documented system establishes a common link between the system and the user. It tells the user the characteristics and capabilities of the system, makes the system easier to use and helps in data resource sharing (Cunningham, 1986). Well enforced standards in the system promote sharing of resources in a controlled environment. There are two general types of data-related standards, data definition standards and data format conformance, and IRDS should be able to support both these standards.

A data definition standard refers to a standard way of describing data such as the naming of the data. The naming standard may be in the form of rigid rules or established conventions for assigning names to data entities. Thus a specific data element will have only one accepted meaning throughout the system and different users may not use it to mean different things in different contexts.

Data format conformance means a data element must conform to a common set of accepted format rules for the data element to retain the same meaning. Thus a data element will have the same format or uniformly accepted code through out the system. The IRDSs should be used to store and record only accepted standards for data elements, and it must be able to monitor and enforce these standards using its edit and validation facilities and screen out non-standard or non-conforming data elements (Brathwaite, 1988; Cunningham, 1986).

An IRDS should be able to provide the most current documentation from its metadatabase (Modell, 1988). This depends on the completeness of the attribute descriptions required for documenting the organization's information environment, as currently stored in the IRDS. This criterion addresses the ease with which documentation can be obtained from the IRDS. There should also be a provision to specify the status of the meta-entities, that is "test", "production", or "history" and allow for more than one version of the structure to exist in the IRDS (Cunningham, 1986).

Security Support

Data security and integrity deals with protection of data from unauthorized access and the absolute completeness and correctness of these data. The main thrust

of the IRDS is to provide control over all the data resources. It is the centralized repository of information relating to systems, programs, data, users and the relationships among them. Thus one should be able to use IRDS to ensure that the appropriate levels of user access controls are in place to provide a certain level of security assurance (Allen, Loomis & Mannino, 1982; Kahn, 1985; Kahn & Lumsden, 1983).

Security support can be evaluated by checking whether an IRDS has password control and functional access control to its metadata, whether these features are provided by the IRDS itself or the DBMS, and whether the security facilities are coordinated with the DBMS to effectively control and restrict access to legal users only.

Integrity Support

Integrity support is closely related to security and is concerned with the integrity and reliability of the metadata. This can be evaluated by ensuring that the edit and validation functions allow checking for completeness, consistency, currency, accuracy, and validity of metadata, and whether error reporting, recovery functions are provided.

Input/Output Interface

This criterion directly reflects how an IRDS will be used and is a highly influential factor toward selecting an IRDS. If the users find the outputs from this interface to be understandable and useful, then from their point of view the IRDS is a valuable tool. As per ANSI and ISO specifications, an IRDS should offer an SQL interface as a command language to modify and maintain the IRD, and as a query language for the user to be able to perform online queries against it. Additionally, an IRDS should also offer a standard set of reports that will give detailed information about the contents of the IRD, such as reports describing all the entity-types or attribute-types, and the relationship-types between entity-types.

User-Friendliness

End-users are the ultimate consumers of information produced from data. They interface with the system environment through a variety of mechanisms including report generators, query languages and preformatted interfaces supported by application programs. They make use of information provided through these interfaces to do their jobs more productively and efficiently. Thus, an IRDS should be able to be used as a tool and support mechanism to enable end-users understand the available information, its meaning and constraints on its usage. Metadata and ad-hoc queries can be used to further enhance the utility of information provided to end users. Thus, built-in provisions such as pull-down menus, on-line tutorial meaningful system prompts and help screens are desirable features of any modern software system and IRDS should be no exception.

Using the Proposed Criteria Set

The proposed criteria set shown in Figure 3 can be used to evaluate IRDS products in two ways. One can be the check-list approach, where users can check a criterion if some given IRDS implements it. The user may first check the broad criteria and then the detailed criteria within each broad criterion. It is perhaps worthwhile to point out that if a product does not have some functional capabilities along each of the three broad core criteria, the user should take special precautions in adopting the product. Further, if an organization decides to adopt multiple products, even if they are from the same vendor, it should ensure that the products are similar in terms of the core criteria/subcriteria. Otherwise, it may lead to unshareable and incompatible dictionary systems. The problem may become even more acute in a multi-vendor distributed environment.

The check-list approach, however, may prove inadequate if multiple products meet all the criteria or meet the same set of criteria/subcriteria. For example, two products both may generally meet the subcriteria relating to the Metadata Entity Structure criteria, but they may differ in terms of the breadth of the Data, System or External entities that they are able to support. In such a scenario, a better approach would be to use a quantitative comparison scheme such as the one described below.

First, assign a ***relative*** weight to each of the ten 'broad' criteria based on its importance to the organization (we suggest that core criteria be assigned greater weights than the additional criteria for reasons discussed earlier in the paper). Next, assign a ***relative*** weight to each of the sub-criteria within each broad criterion (i.e., the sub-criteria weights for each individual broad criterion should sum up to 1). Next, assign a numerical rating (on a scale of 1 to 10) to each sub-criterion within a broad criteria and use the corresponding weighting factors to determine the total score for a given broad criterion. Next, multiply this total score for each broad criterion by

Criteria & Subcriteria	Weight		Score		Total	
1.0 Ability to Capture Core Entity Structure (Weight (W) = 0.13)						
1.1 Data Entities	0.58	X	8	=	4.64	
1.2 System Entities	0.22	X	7	=	1.54	
1.3 External Entities	0.20	X	6	=	1.20	
			SUM (S)	=	7.38	
Weighted Criteria Score	0.13 (W)	X	7.38 (S)	=	**0.96** (WS)	
2.0 Ability to Capture Core Attribute Structure (Weight (W) =0.13)						
2.1 Identification Attributes	0.31	X	8	=	2.48	
2.2 Representation Attributes	0.23	X	8	=	1.84	
2.3 Statistical Attributes	0.13	X	6	=	0.78	
2.4 Control Attributes	0.22	X	8	=	1.76	
2.5 Physical Attributes	0.11	X	6	=	0.66	
			SUM (S)	=	7.52	
Weighted Criteria Score	0.13 (W)	X	7.52 (S)	=	**0.98** (WS)	
3.0 Ability to Capture Core Entity-Relationship Properties (Weight (W) = 0.13)						
3.1 Relationship Name	0.15	X	9	=	1.35	
3.2 (Specific) Maximum Cardinality	0.13	X	0	=	0.00	
3.3 Mandatory/Optional Relationships	0.17	X	9	=	1.53	
3.4 Generalization(IS-A) Relationships	0.18	X	9	=	1.62	
3.5 Mutually Exclusive Relationships	0.10	X	9	=	0.90	
3.6 N-ary Relationships	0.16	X	0	=	0.00	
3.7 Recursive Relationships	0.11	X	9	=	0.99	
			SUM (S)	=	6.39	
Weighted Criteria Score	0.13 (W)	X	6.39 (S)	=	**0.83** (WS)	
4.0 Extensibility Support (Weight (W) = 0.12)						
4.1 Add/Update/Delete Entity-types	0.34	X	9	=	3.06	
4.2 Add/Update/Delete Attribute-types	0.33	X	9	=	2.97	
4.3 Add/Update/Delete Relationship-types	0.33	X	9	=	2.97	
			SUM (S)	=	9.00	
Weighted Criteria Score	0.12 (W)	X	9 (S)	=	**1.08** (WS)	
5.0 Data Documentation and Versioning Support (Weight (W) = 0.11)						
5.1 Current Attribute Descriptions	0.4	X	8	=	3.20	
5.2 Standard Control	0.31	X	6	=	1.86	
5.3 Version Control	0.29	X	6	=	1.74	
			SUM (S)	=	6.80	
Weighted Criteria Score	0.11 (W)	X	6.80 (S)	=	**0.75** (WS)	

Figure 4: IRDS Evaluation Schema

6.0	**Security Support (Weight (W) =0.10)**				
	6.1 Control Access through Username/Password	0.62	X	8	= 4.96
	6.2 Coordinate Access through DBMS	0.38	X	8	= 3.04
				SUM (S)	= 8.00
	Weighted Criteria Score	0.10 (W)	X	8 (S)	= **0.80** (WS)
7.0	**Integrity Support (Weight (W) = 0.09)**				
	7.1 Provision of Edit and Validation functions	0.34	X	9	= 3.06
	7.2 Provision of Error Reporting functions	0.33	X	9	= 2.97
	7.3 Provision of Data Recovery functions	0.33	X	9	= 2.97
				SUM (S)	= 9.00
	Weighted Criteria Score	0.09 (W)	X	9 (S)	= **0.81** (WS)
8.0	**Input/Output Interface (Weight (W) =0.10)**				
	8.1 Query Language Support	0.34	X	8	= 2.72
	8.2 Command Language Support	0.33	X	8	= 2.64
	8.3 Predefined Standard Reports	0.33	X	8	= 2.64
				SUM (S)	= 8.00
	Weighted Criteria Score	0.10 (W)	X	8.00 (S)	= **0.80** (WS)
9.0	**Use-Friendliness (Weight (W) = 0.09)**				
	10.1 Help and Pop-up Screens	0.47	X	8	= 3.76
	10.2 Ease of Learning and Using the Product	0.53	X	8	= 4.24
				SUM (S)	= 8.00
	Weighted Criteria Score	0.09 (W)	X	8.0 (S)	= **0.72** (WS)

TOTAL SCORE (Sum of WSs) = 7.73

Figure 4 (continued)**: IRDS Evaluation Schema**

its weighing factor and add all such scores to arrive at a final total for a given IRDS. An illustrative example of this quantification scheme is presented in Figure 4. It may be noted that the proposed quantification scheme allows for the comparison of different IRDS products at the subcriteria, criteria, and overall level, and highlights the inadequacies of each specific product.

In order to illustrate the use of the proposed evaluation scheme, an independent data base consultant, with extensive experience with IRDS products, was asked to use it to evaluate a leading IRDS product. The numbers provided in Figure 4 are the results of his analysis. It is interesting to note that a total score of 7.73 out of a possible 10 for one of the most popular packages available today, still leaves plenty of room for improvement.

Conclusion

In this paper, we developed a detailed set of criteria with which the functional capabilities of different IRDS products can be compared. Our proposed criteria are based on published literature in the area of IRDS and the ANSI and ISO standards developed for IRDSs, and our own experience as researchers/educators and developers of database applications.

We hope that the proposed criteria set will be of interest to both practitioners and researchers. Practitioners should find the proposed criteria (specifically the 'IRDS Evaluation Schema' presented in Figure 4) useful in making selections among prospective IRDS offerings.

We would, however, like to point out that this paper by no means is the final authority on how to evaluate IRDSs, nor does the criteria set presented in this paper represent a fully complete set of required features of IRDS. The concept of criteria development and evaluation will become more accurate, as more detailed and diverse ideas are presented. This may be accomplished through surveys of experts or further research into the background and development of IRDS requirements. Researchers may use the proposed criteria as a stepping stone towards a more refined and/or a more extensive set of characteristics to be considered. As the area of IRDS grows and new features and functions are required of the IRDSs, new criteria and subcriteria could be added to the list presented in this paper. The quantification and weighting technique may also be improved upon.

Finally, as mentioned in the introduction, there is a rapid growth in IRDS implementations in related software technologies such as ICASE and DBMS. It may therefore be difficult at times to evaluate IRDS independently of these facilities. By the same token, the proposed framework developed in this paper could very well be used to evaluate the repository portion of these related technologies. The framework could conceivably be broadened to take into account the other functionalities provided by such technologies.

References

Allen, F.W., Loomis, M. E., Mannino, M.V. (1982). The Integrated Dictionary/Directory System. *ACM Computing Surveys*, 14(2), 246-286.

American National Standards Institute (ANSI X3.138-1989). *American National Standard Information Resource Dictionary System.*

American National Standards Institute (ANSI X3.138-1992). *American National Standard Information Resource Dictionary System.*

Appleton, D. S. (1987). The Modern Data Dictionary. *Datamation*, March 1, 66-68.

Brathwaite, K.S. (1988). *Analysis, Design, & Implementation of Data Dictionaries.* McGraw-Hill.

Bruce, T., Fuller, J. and Moriarty, T. (1989). So You Want a Repository. *Database Programming and Design*, May, 60-69.

Cashin, J. (1988). Another Megatrend: IRDS is the Future. *Software Magazine*, November, 35-39.

Cunningham, E., Davis, R. (1986). Data Dictionary Systems and Their Application in Auditing. *Data Processing.*, 28(1), Jan/Feb, 30-35.

Date, C. J. (1990). *An Introduction to Database Systems*: Volume I. 5th ed. Addison-Wesley.

Davis, J.P., Bonnell R.D. (1988). EDICT—An Enhanced Relational Data Dictionary: Architecture and Example. *IEEE Computer*, 184-191.

Dolk, D. R., Kirsch R. A. (1987). A Relational Information Dictionary System. *Comm. of the ACM*, 30, 1, January, 48-61.

Durell, W. (1983). Disorder to Discipline via the Data Dictionary. *Journal of Systems Management*, May, 12-19.

Gillenson, M. L. and Frost, R. D. (1993). The Evolution of the Meta-Data Concept: Dictionaries, Catalogs, and Repositories. *Journal of Database Management*, 4(3), 17-26.

Glenwright, J. C. (1988). Tips to Manage Your Data. *Database Programming & Design.*, November, 13-16.

Hazzah, A. (1989). Data Dictionaries: Paths to a Standard. *Database Programming & Design.*, August, 26-35.

International Standards Organization (1989). *Information processing systems: Information Resource Dictionary System (IRDS) Framework.* Draft International Standard ISO-IEC DIS 10027, March, 11.

Jones, M. R. (1992). Unveiling Repository Technology. *Database Programming & Design.*, April, 28-35.

Kahn, B. K. (1985). An Environmentally Dependent Framework for Data Dictionary Systems. *MIS Quarterly*, September, 199-220.

Kahn, B. K., Lumsden, E. W. (1983). A User-oriented Framework for Data Dictionary Systems. *DATA BASE*, Fall, 28-36.

Kull, D. (1987). Data Dictionaries Can Point the Way. *Computer & Communications Decisions*, July, 69-70,97,126.

Leon-Hong, B. W., Plagman, B. K. (1982) *Data Dictionary/Directory Systems: Administration, Implementation and Usage*. John Wiley & Sons, Inc.

March, S. T., Kim, Y. (1988-89). Information Resource Management: A Metadata Perspective. *Journal of Management Information Systems*, 5, 3, Winter, 5-18.

Meador, J. G. (1987). Evaluating Dictionary Technology: The Case for Integrated Systems. *Auerbach Publications on DBMS 22-04-12.*

Modell, M. E. (1988). Management's Perspective of Data Dictionary Systems. *Auerbach Publications on DBMS 21-20-10.*

Narayan, R. (1988). *Data Dictionary: Implementation, Use, and Maintenance*, Prentice Hall.

Plotkin, D. (1992). Selecting a Repository. *Database Programming & Design*, April, 39-46.

Ross, R. (1981). *Data Dictionary Systems and Data Administration,* Amacom.

Protocols Standards and Communications Inc. (1989). *Situation Report on the Information Resource Directory System (IRDS),* (Prepared for the National Archives of Canada), Ottawa, Canada.

Van Duyn, J. (1982). *Developing a Data Dictionary System,* Prentice-Hall.

Venkatakrishnan, V. (1988). Differences in dictionaries. *Computerworld*, March 14. S13-S16.

Weldon, J. L. (1985). The Case for Active Data Dictionaries, *Auerbach Publications on DBMS 22-04-10.*

BIJOY BORDOLOI is an Assistant Professor of Information Systems and Management Sciences at The University of Texas at Arlington. He received his Ph.D. in MIS from Indiana University, Bloomington. His current research interests include data modeling, data administration, distributed database systems, and software project management. His publications have appeared in several journals including Journal of Information Systems Management, Journal of Information Science and Technology, Journal of Microcomputer Systems Management, International Journal of Information Resource Management, and International Journal of Production and Operations Management.

SUMIT SIRCAR is Director of the Center of Information Technologies Management at the University of Texas at Arlington. He received his doctorate from The Harvard Business School. He specializes in the management of information technology and has published numerous articles in journals such as the Communications of the ACM and Information and Management.

CRAIG SLINKMAN is an Associate Professor of Information Systems and Management Sciences at The University of Texas at Arlington. He received his Ph.D. in Quanitative Analysis from University of Minnesota. His current research interests include logical and physical database design, CASE technology, and health-case information systems. His publications have appeared in several journals including MIS Quarterly, Communications in Statistics, and other journals.

NITANT CHKRANARAYAN is a senior consultant with Database Consultants, Inc., Houston, Texas. He receved his BS in Computer Science and MS in Information Systems from University of Texas at Arlington. He has been associated with the database consulting industry for the last 5 years.