In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Commerce & Business Administration

The University of British Columbia
1956 Main Mall
Vancouver, Canada
V6T 1Y3

Date April 27, 1982
ABSTRACT

This thesis reviews data dictionary systems, approaching the subject from four directions. First, the contents of dictionaries are examined. Second, the users and their interaction and use of the data dictionary are reviewed. Third, some popular commercial data dictionary packages are reviewed. Finally, data dictionaries are examined as a package to be implemented by a data processing department.

Data dictionary entities can be grouped into four classes, each adding to the information stored by the preceding layer. They are, the data dictionary which stores data entity data, data directories, which store processing entity data and process-data relationships, data resource dictionaries, which store data about processing environment entities such as hardware, and the metadata dictionary which stores data about conceptual entities such as events and functions. Using this structure, each class of dictionary is examined to identify the entities and their attributes.

Users are examined in the following groups: the database administrator, systems analysts, programmers, the data processing operations department, and the user group.

Five commercially available dictionaries are examined in detail, DATAMANAGER, DB/DC Data Dictionary, Data Catalogue 2, UCC-10, and IDD. In addition, the thesis examines two dic-
tionaries in overview, highlighting their advanced thinking. ICL DDS, which allows the user to model both implementation and conceptual views of data, and CINCOM's DCS which includes the dictionary in a package containing a screen design aid, a programmer workstation, and security system all integrated with the dictionary.

The data dictionary is examined from a system application view point addressing the questions to which any application is subjected when evaluated. Identifying the need for a data dictionary, three areas are examined. First the control of data as an organizational resource, second the use of system development methodologies and third, the management of change in data processing systems.

The ability of the data dictionary to support a data processing environment which uses conventional file structures, a simple database or a complex environment with multiple databases, or distributed processing is addressed.

Software selection criteria are examined, defining the dictionary content, and utilities which should be investigated, and criteria for environment such as data processing hardware and software, user-vendor relationships and cost and maintenance are proposed,

Finally we propose some events in the data processing life cycle which may assist in the successful implementation of a data dictionary. Implementation in a complex environment
where no other project proposed is critical, or before a major database management system conversion may be warranted. Or implementation may be undertaken before a large systems project, or as part of a data processing standards project.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER 1 • INTRODUCTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Objectives</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Chapter Synopses</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 2 • CLASSES OF DATA DICTIONARIES</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Data Dictionary</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Data Directory</td>
<td>4</td>
</tr>
<tr>
<td>2.3 Data Resource Dictionary</td>
<td>5</td>
</tr>
<tr>
<td>2.4 Metadata Dictionary</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 3 • USERS OF THE DATA DICTIONARY</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 The Data Base Administrator</td>
<td>7</td>
</tr>
<tr>
<td>3.2 The Systems Analyst</td>
<td>8</td>
</tr>
<tr>
<td>3.3 The Programmer</td>
<td>9</td>
</tr>
<tr>
<td>3.4 The Operations Department</td>
<td>10</td>
</tr>
<tr>
<td>3.5 The User Group</td>
<td>11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 4 • SURVEY OF COMMERCIAL DATA DICTIONARIES</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Datamanager</td>
<td>12</td>
</tr>
<tr>
<td>4.2 DB/DC Data Dictionary</td>
<td>13</td>
</tr>
<tr>
<td>4.3 Data Catalogue 2</td>
<td>14</td>
</tr>
<tr>
<td>4.4 UCC-10</td>
<td>15</td>
</tr>
<tr>
<td>4.5 IDD</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 5 • A DATA DICTIONARY IN AN ENTERPRISE ENVIRONMENT</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 The Need For A Data Dictionary</td>
<td>17</td>
</tr>
<tr>
<td>5.2 Software Selection</td>
<td>18</td>
</tr>
<tr>
<td>5.3 Implementation</td>
<td>19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 6 • CONCLUSION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BIBLIOGRAPHY</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>21</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table I • Information Sources -- Data Dictionary Packages

................................................................. 58
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Data Dictionary Model</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Data Dictionary Functions</td>
<td>39</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Function Matrix</td>
<td>48</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Information Flow Diagram</td>
<td>49</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Data For Analysis -- Stored By The Data Dictionary</td>
<td>101</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Data Dictionary In A Conventional Environment</td>
<td>105</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Data Dictionary During A DBMS Installation</td>
<td>107</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENT

I would like to recognize those individuals who have assisted me in the completion of this work. Foremost, I thank my wife, Eileen, who has supported and encouraged me throughout. Without her, this work would not have been completed. I would also like to thank my committee, Robert C. Goldstein, my advisor, Albert S. Dexter, and Alvin Fowler. Their effort and assistance are very much appreciated, especially in light of the difficulties I had in completing the work. Finally, I would like to thank Arthur Andersen & Co. for allowing me the time to complete this thesis.
CHAPTER 1

INTRODUCTION

The data dictionary is a software tool used by an enterprise to control the data it uses. The control of data is important to organizations which have grown sufficiently large and complex that it is no longer easy for management to obtain information from its organizational units without formal procedures. Organizations have responded to the need for information by creating more complex data gathering mechanisms. The state of the art in such mechanisms include data entry into a computer system at the source of data creation, followed by its storage using a rigid structure format in a database management system (DBMS).

In achieving this level of data management, the need to describe all data, and attributes of the data, such as storage, location, format, uses, and sources, has necessitated developmental work in the structure, content and format of metadata dictionaries. Several commercial products are available under the generic title of data dictionaries.

1 Metadata is a term used to indicate that the data is at a higher or meta level. An organization's data can be described by its metadata definition. Metadata is therefore, data about data.
1.1 Objectives

This paper has the following objectives:

1. To define the scope of a data dictionary in terms of its content, functions performed and individuals or groups which benefit from its existence.

2. To summarize the findings of a number of publications which have evaluated the commercial data dictionary packages available.

3. To examine the role the data dictionary plays in an enterprise environment, supporting the technical complexity of today's information systems.

4. To examine the major selection criteria which are considered when choosing a data dictionary package.

5. To present those instances when the implementation of a data dictionary package is likely to meet with success.
1.2 Chapter Synopses

This paper examines the data dictionary from four views:

1. Classifying the data dictionary based on information stored and facilities available to retrieve and manipulate the data. The four classes are:
   a) data dictionaries which store basic information about data elements,
   b) data directories which identify location of data stored,
   c) data resource dictionaries which contain information about programs, systems, files and databases as well as data elements, and
   d) metadata dictionaries which contain information about conceptual data views which may or may not be implemented in computer or manual information processing systems.

2. Examining the impact of the data dictionary system on classes users of the dictionary in an enterprise:
   a) The database administrator and data processing management who supply information and ensure its correctness and completeness
   b) The transformers of data into information (e.g. system analysts, programmers and computer operators) who translate users' desires for information into
automated data manipulation processes
c) The end users for whom data is a tool in the performance of a task.

3. Reviewing data dictionaries available as packaged software. This examination will cover the most popular systems and identify their common and unique characteristics.

4. Evaluating the data dictionary as an additional application managed by the data processing department. This section will deal with the issues of:
   a) identifying the need for a data dictionary to support the information processing task,
   b) criteria used to select a suitable data dictionary system, and
   c) guidelines for the implementation of a data dictionary system.

The final chapter will examine future trends for data dictionaries. Two issues will be addressed:

1. The place of a data dictionary system in the data processing environment.

2. The impact data management systems will have on the development and use of data dictionary systems.
CHAPTER 2

CLASSES OF DATA DICTIONARIES

Overview

Packages that document data about data, or literature that describes such data, use naming conventions which leave the reader confused about the object described. This paper proposes four classifications to describe software descriptions in the literature and packages which are available. This model (see Figure 1) can be thought of as a layered cylinder where each successive layer is a more complex, more complete description of enterprise entities. Each succeeding layer contains the information of the preceding layer. These layers are:

1. A data dictionary. This layer contains the description of data entities. These data entities are described

---


3 An entity is a person, place, thing, concept or event, real or abstract, of interest to the enterprise. ANSI/SPARC/X3 Study Group on Data Base Management Systems, "Interm Report," FDT, 7, No. 2 (1975), 18.
Figure 1

Data Dictionary Model
in terms of intrinsic attributes such as format, size, and name.

2. A data directory. The data directory contains additional data entity attributes such as location, use, and structure. The data directory also describes process entities such as programs, modules and systems and establishes relationships between data and processing entities.

3. A data resource dictionary. The data resource dictionary contains information not only about data and processing entities, but also about environment entities such as hardware processors, physical data files, and distributed database systems.

4. The metadata dictionary. This level adds information concerning conceptual entities such as events and functions, organizational entities, and their attributes. The entities documented are not necessarily implemented in a computer system.

---

4 A structure is a collection of zero or more objects connected by a directed relationship. Ibid., p.17.

5 An attribute is the representation of a property of an entity. Ibid., p.16. It is the set of values which a characteristic of an entity may take on.
With the exception of the metadata dictionary, an enterprise could have one or more data resource dictionaries (which includes the information contained in the data dictionary and directory), depending on the number of functional areas which are described. Thus, if an enterprise consists of several distinct functional areas, each could have their own data resource dictionary. A metadata dictionary, however, describes the entire enterprise, and although it may be reproduced, exists uniquely.

Entry Points

Regardless of the type or amount of information stored by the data dictionary system, three entry points are required to retrieve information:

1. Through a subject-oriented keyword or keyword phrase. Such keywords are defined by the user and allow for browsing. The keyword is an attribute of the entity and the word or phrase used in inquiry are the values of the attribute.

2. Through a unique data name. This name may be either user or system assigned. Retrieval by this method should return a limited amount of information in a pre-defined format. This access method would allow for re-
trierral of information by programs or other software systems.

3. By using values of entity attributes or names of entity types. For example, the "name-synonym" as an attribute of a data entity, or a specific entity type such as "file". This type of search would allow for the description of an entity such as a program or a file in terms of the attributes stored and would identify cross-references to other entities. A search statement such as:

\[
\text{FILE = EMPLOYEE...}
\]

where all files beginning with EMPLOYEE are identified. Such information would most likely be used by the Data Base Administrator (DBA) and other data processing personnel such as system analysts and programmers who are interested in a generic information concerning the type of information stored.

2.1 Data Dictionary

Entities described by the data dictionary are concerned with data or data structures. The author proposes five distinct data entities. They are elements, data groups, physical segments, files, and databases.

The smallest independent unit of data described in a data dictionary is the element, also called a field or an item.
in the literature. The majority of the attributes maintained by the data dictionary are used to describe this unit of data. The next most complex data entity is the data group. A data group is a logically related group of elements combined for convenience. An example of a data group is TODAY'S-DATE which consists of the elements DAY, MONTH and YEAR. The group, TODAY'S-DATE, is combined purely out of convenience and is described fully by the attributes which describe the data element.

The physical segment, also known as a retrieved record, is an entity which describes the smallest unit of accessible data on a physical storage device. This entity has additional attributes not applicable to elements. These attributes, such as key, identify the physical segment as a required entity in the data classification hierarchy.

Grouping physical segments one can create data sets or files. In a file, all members need not be alike, therefore attributes such as header, trailer or "multiple occurring" need to be used to identify each member of a file.

The most complex data construct is the database. A database is a set of records, groups, or elements which have in common their use by one or more business functional entities, and which are described in a structured format. The business functional entity is a subunit of the enterprise which carries out a particular business function. Examples of a business
function are payroll calculation, and purchasing of raw materials.

Data Dictionary Contents

As we have proposed, the majority of the data dictionary documentation is information about data elements. As such, the majority of data entity attributes apply only to the element. This discussion will deal with general entity attributes and identify those attributes which are specific to one entity type. Attributes of data entities can be grouped into five classes:

1. Name information. This would include any attribute identifying the entity.

2. Content information. This includes a textual description and would distinguish the entity from any similarly named entities.

3. Structure information. This would include information concerning the ordering or organization of the element in more complex structures such as groups, physical segments, files, and databases. Examples are key, index, and sequence. This information is commonly stored at the group, segment, file, or database level, through an
attribute or named relationship "contains", identifying a lower level entity occurrence as the members contained. However, it could be represented at the lower level (including element) by a named relationship "occurs-in", which would identify those higher level entity occurrences where it would be a member.

4. Integrity information. This would include attributes which would be used to test for the correct description of the element. Such information would include length, picture values, edit rules, and classifications.

5. Privacy and security information. This data would include attributes such as creation and modification authorization, information dissemination, and allowed usage which would permit the documentation of privacy and security levels of data entities.

Name Attributes

Name attributes are important in identifying each occurrence of a data entity. Major attributes held in the dictionary are:

1. Label or identifier information. This may be a name or system identifier which provides a unique key for each
entity occurrence.

2. A designator, one or more key words which are used as a search path in identifying a particular entity occurrence.

3. Synonyms for the label, describing commonly used identifiers related to systems, programs or user functions.

4. Language synonyms such as COBOL or PL/1 names included to identify the commonly used name within a computer program.

Content Attributes

Content information consists of description attributes which may be used in documentation or user understanding of the entity occurrence. Common attributes are:

1. The textual description of the entity occurrence. This is a free format alphanumeric description which may be used in documentation.

2. Status information which would identify the status of the entity occurrence during its life. Common status values are:
a) proposed - not yet reviewed or completely defined
b) concurred - this would indicate all users and other affected persons had agreed to the definition of the occurrence
c) approved - this would identify that the proper approval for placing the entity occurrence into daily use has been given by the proper authority. At this stage all data necessary to define the entity occurrence has been entered
d) effective - this identifies that the entity occurrence is presently in effect
e) superseded - this identifies a previous occurrence of this entity type

3. Version information provides for more than one effective entity occurrence with the same identifier in the system at one time. This may be necessary when alternative or test data entity occurrences are desired, as in the design of a database.

4. In order to further describe the entity occurrence, it is sometimes useful to identify a unit of measure. This would indicate, for example, that an element named TOTAL-EARNINGS has a unit of measure, dollars. And an element TOTAL-EARNINGS-GB has a unit of measure, pounds sterling.
Structure Attributes

Structure attributes are used to group lower entities into those of a higher type (e.g. elements into groups, physical segments, files and databases), or show the membership of lower entities in higher entities. Attributes used in defining the structure include:

1. A key which identifies the element or elements which provide a key sequence for the structure. Keys may be single, composite, or alternate.

2. The sequence of elements in records or records in files. This attribute would include information to allow specific record formats to be reconstructed.

3. The organization of structures. This would include information on physical record, file, or database organization, and would include values such as hierarchical, HISAM, and ISAM.

4. The sort sequence of the data entity. This may be the same as the key, or additional sort fields may be used. This attribute could also occur multiply depending on system requirements.
Integrity Attributes

Integrity attributes allow for validation and content checking subroutines to be used to ensure data correctness on input. These attributes apply primarily to the data element.

1. The character set used. This can have values such as alpha, numeric, or graphic character.

2. The length of the element in characters, bytes or words.

3. The mode of the element. That is, whether it is a bit string, a character, packed decimal, or in ASCII or EBCDIC.

4. The precision or number of significant digits which are carried with this element.

5. The justification, whether left or right, of this element.

6. The picture of the element. Normally a COBOL picture phrase is used (e.g. S999V99).

7. Edit rules which could apply. Typical values for this attribute are whether the element is a constant, whether it is required or optional, the usual value that the
element has, the range of values that the element may carry, and any edit masks that may be applicable.

8. The derivation algorithm used in creating this element.

9. The classification or category under which this element falls. This is sometimes useful in providing alternate search criteria for users of the dictionary. Categories are usually specific to a certain enterprise. Typical categories are name, inventory descriptor, personnel descriptor, accounting value, currency.

10. The states of the element. This includes codes which the element may have. This is different from constants or range of values in that the state defines a fixed number of values that the element occurrence may take. A typical example is a list of codes.

Security And Privacy Attributes

Security and privacy attributes identify sufficient information to ensure that unauthorized access to the entity can be prevented if such a routine is implemented in the software, and that only those who are authorized to use the entity can be identified. Typical attributes are:
1. Privacy constraints such as who may see, use, or modify the entity occurrence.

2. Specification responsibility for the entity occurrence. That is, the individual or functional area which has responsibility for creating the data dictionary definition.

3. The content responsibility or the individual or function responsible for providing the values for the actual entity occurrence.

4. Security parameters necessary to ensure only authorized users have access to this entity occurrence. Such parameters may include passwords.

5. The distribution of reports or other hard copy media which contain the entity.

Although the preceding description of a data dictionary includes only a restricted number of attributes commonly included in software which bears the name "data dictionary", this restricted subset of data can be used as a basic data dictionary. Utilities could be designed, based on this information, which would provide a large amount of extremely useful information.

This kind of system need not require a large amount of
sophisticated software to maintain. In fact, some data dictionary systems are manual. They provide standard documentation which ensures that a complete, consistent record of all data entities is maintained. Because it is at a relatively low level of detail, this record is achievable with a minimum of systems expertise. Only the most basic data gathering and documenting skills are required. In fact, this task is best carried out by personnel in the user departments, using guidelines established by the DBA to identify the level of definition for each data entity.

Once all the elements have been identified, categories of simple data elements can be recognized. Data grouped by function or use can assist in this categorization. In this task, cross-references which are based on Key Word In Context (KWIC), are frequently prepared. Data groups or structures are the first step towards creating a corporate data model.

Because all data entities are documented in a similar manner, standard routines for input or edit can be prepared. This requires only a small base of information which includes integrity attributes. Besides reducing the amount of system development or programming effort through the use of standard sub-routines or modules, it also promotes the template ap-

---

approach to systems development. A formal methodology for systems development is much easier to enforce and monitor in this environment.

A final benefit of documenting data entities is in promoting database concepts, particularly data sharing. Data interchange is facilitated through the use of common, easily readable and usable data definitions. This allows users in diverse functional areas to have a common understanding of the data definition. Through common definitions, the design of data structures which are shared is a much more realizable task.

One of the largest drawbacks to shared data files results from the parochial attitude that users have to sharing data. By identifying and assigning responsibility for entity definition to the user, so that integrity and control would be at the user level, some of the objections that users normally have to "fiddling with my data" should be removed.

A key database concept is the independence of data from single applications or systems. Key parameters necessary to provide this independence are the identification of encoding schemes, storage modes, character sets, labels or identifiers, the range of values, and the physical format of the data. Because the data dictionary can record all of these attributes, the translation into a different set of parameters is a much easier task.
2.2 Data Directory

The data dictionary is concerned with the content, structure, integrity, and security attributes of data entities; however, data exists within an environment of processes and events. The data directory is the second layer of data information software. The data directory documents the locational characteristics of data. This necessitates the documentation of process entities, and relationships between data and process entities.

Entities which are used to document locational characteristics of data identify the access or use of data. These entities include:

1. Programs (computer programs) and substructures such as modules and subroutines.

2. Users of data, including those who accumulate manipulate and store it.

3. Procedures, which includes written and verbal procedures specifying proscribed actions performed on data.

Relationships between data and processes are documented through entities which are used in information communication.

---

7 Uhrowczik, p.337.
These entities are referred to as "data interchange entities". These constructs include:

1. Transactions which identify unique occurrences of data items relating to a particular event.

2. Reports which identify data gathered to provide information to a user.

3. Documents which act as triggers or initiators for transactions and events.

Data interchange entities are important because they identify a unique relationship between data and the users. Such entities could be included in the data dictionary as a special instance of a group, however they also relate to processes, and identify the active nature of data interchange.

One of the major benefits of maintaining process entities is the ability to identify relationships between processes and data. Database management systems (DBMS) contain a structure of data entities. If one could identify the requests for data from programs which interact with the DBMS, one could achieve similar documentation to that information contained in the data directory. However, the user has no means of querying the relationships between programs and the DBMS data. That is, there is no system facility which allows the user to identify the underlying processes the programs simulate. On the
other hand, the data directory contains data entities and their attributes. It is the documentation of data stored in the DBMS, along with other data entities (transactions, documents) which are in existence in an enterprise. The database management system stores data entity information based on a single, limited view. If one stores both data and process information in the data directory, relationships which are implied in the program - DBMS interface and relationships which are not implemented through computer systems can be documented and investigated by users.

The Data Directory Front End To A DBMS

It has been suggested that the data directory can be used to replace data definition information stored in a database management system. Since all data elements and data structures are stored in the data dictionary, the DBMS should be able to use that information to format its storage area for data. This is the function of the data dictionary in the ANSI/SPARC definition. Proponents of this extension to the data dictionary claim that the information is duplicated in the DBMS and therefore, because the data dictionary is more

---


general software ideally suited for this purpose, it should be the only repository.

This format of the data directory can be achieved in two ways: free-standing and coupled. A free-standing data directory used to front end the DBMS is not part of the DBMS package. It can, therefore, store non-mechanized data entities as well as those whose values can be retrieved by the DBMS. Access to the data directory does not require that data be retrievable from the DBMS. In addition, the data directory can document a different stored data format from that which the DBMS supports. For example, if the DBMS uses a network system to format and store data, the data dictionary can allow relationships which are hierarchical. An additional argument for the free-standing data directory is that access of data in the DBMS does not require prior access of the data directory, there is no software overhead for a DBMS access. As such, DBMS retrieval performance is not affected.

The alternate method of utilizing a data directory as a front end to the DBMS is to couple them. This would mean that the data directory is extended to include data storage and retrieval mechanisms, or the DBMS is expanded to include the facilities for the storage of dictionary and directory attri-
butes. This implementation localizes the storage of the metadata. It may also allow for a more uniform method of storing both the data directory entities and the database entity values. This would reduce the number of utilities necessary to support both the DBMS and the data directory. In Sharman and Winterbottom, a data dictionary and DBMS were updated at the same time. For example, a data entity and its attributes were added to the data dictionary, and subsequently values for that entity were added to the DBMS. The primary advantage to users in this organization of data and metadata, they claim, is that access to the DBMS can be done with an entity-type search in the data directory, with all its information about the usage and relationships between processes and data, and then a search for values of that entity in the database could be done. Other advantages of integrating the functions of the data directory with those of the database management system include:

1. Security against unauthorized access is controlled at one point. All information about users who have valid clearance for data elements can be stored without duplication.

---

2. The data directory can store access path logic for repetitive DBMS accesses. This can reduce the amount of time necessary to retrieve data.

3. The data directory can store statistics concerning the access of data entities. This could be used by systems analysts in defining new applications.

4. Data validation can be integrated into the database management system using parameters supplied by the directory. This would increase the integrity of data stored in the database system. A factor which should be considered before data validation is included as part of the database system is the decrease in information retrieval performance if validation is not necessary. Very complex stored routines may be needed to ensure that all the various validation rules can be accommodated. Alternatively, stored routines could be so simple that additional validation routines written into data entry programs would result in redundancy of data validation. These considerations are tempered by the consistent application of validation criteria and ease of modification of validation rules stored.
2.3 Data Resource Dictionary

The preceding description of the data directory (which includes the data dictionary) has not explicitly named its users. This is due to the emphasis on describing classifications of metadata software. The users' requirements are described more in Chapter 3, however, the major users of the data resource dictionary will be data processing staff, the DBA, systems analysts, and programmers.

The storage of metadata about data and processing entities is a tremendous new source of information about the data processing function of the enterprise. In identifying the entities and attributes of the data resource dictionary (DRD), we can look at the five main interfaces to the data resource dictionary. Each of these interfaces describes the manner in which a user (which may be a person or an automated process) retrieves information from the dictionary.

1. Application development interface. This interface would encompass the technical design and implementation questions that systems analysts, programmers, or operators may have of data entities or processes. The data dictionary must have data to support these technical inquiries. The questions posed are often general because

of the need for a wide spectrum of information completely describing the data, process, or interface entities examined. Since such queries will begin with very basic requests, and are then honed by modifiers, a complete interface dialogue is necessary. This interface would require the user to be able to specify both the type of information desired and the format in which that information should be presented.

2. User interface. This interface requires a dialogue which will allow the user to conceptualize data and processes easily. In addition, it is necessary to present data in such a manner that the user can easily navigate through the models (represented by relationships between entities) which are stored in the data dictionary. The user will most likely identify specific data or processing entities in his/her query and, therefore, must be presented a simple, restricted view of both entities and relationships. As a result of the query the user will likely influence the data stored in the data dictionary by specifying additional relationships or entities. In this interface the relationships between data and processes is most important.

3. The information processing interface. This interface allows information processing systems such as databases, data management routines, and application programs to
extract attribute values necessary for their processing. The method of query is highly structured and formalized. The value of attributes is the information extracted.

4. The distributed process interface. This interface documents locational entities at a higher level of complexity than the data directory. In this interface the system will extract information about the distribution and location of information across physical data storage devices, which may be local or remote.

5. The control interface. This interface is used by the database administrator to record attributes that support documentation, the validity of data stored in the DBMS and the datafiles, and security that should be implemented in preventing unauthorized access to the data. Information retrieved by the database administrator using this interface will allow him/her to make decisions concerning the amount of information to be stored in the data resource dictionary.

In meeting the information requirements for the five interfaces described above, the DRD must not only store metadata concerning data, process, and information interchange entities but also:

1. Locational information at the machine address level.
2. Statistical data concerning usage frequency, usage sequence, such as access path, both at the physical database level and at the clerk/filing cabinet level.

3. Documentation of physical devices such as computers, disc files, tape drives, and people.

4. Resource characteristics such as retrieval time, transfer speed, and capacity.

5. Telecommunications attributes such as protocols, line rate, and network configuration.

Documentation of all DRD entities must include computerized and non-computerized entities and allow for input from programs, on-line monitors, database directories, as well as the capture of data through the use of manually prepared data input.

The users of the data resource dictionary - data processing staff, functional data analysts investigating user requirements, and auditors - can utilize the data stored through a multitude of application subsystems. These subsystems are used to support the interfaces described earlier.

---

Subsystems which support data processing staff in the design, development, implementation, and maintenance of systems include:

1. Impact-of-change analysis routines which allows the analyst to identify programs affected by a change in data structure, or the data files affected by a program modification. The long lead time necessary to develop systems and the average level of documentation compared to the turnover of programmers and analysts makes this a high payoff utility.

2. A program version control system to document changes to programs. This would allow packaged software to be modified yet retain the ability of applying software upgrades, with reasonable effort.

3. Test data generation. This utility allows consistent test data to be generated based on data definitions stored. This would allow systems development staff to test more comprehensively and provide a base test for program maintenance staff.

4. Program code generator. Many database definitions, database call statements, data communications routines, job control statements, and validation routines are standard. With the system and hardware information
stored in the data resource dictionary, such code could be generated automatically, or parameters for completing the development of the code could be presented in a usable manner.

Other subsystems which would be useful, and could utilize information contained in the data resource dictionary include:

1. Access control mechanisms. Because the data resource dictionary has information concerning data at the element level, access control could be implemented for both program compilation and execution. At compile time, because data definitions and programs are both defined in the DRD, unauthorized programs could not be compiled. Execution time control would ensure that the proper program and data formats were used and would permit data use statistics to be gathered. This subsystem would use an interface which would allow for ad hoc on-line access to data as well as the traditional batch program access. Implementation could be accomplished by means of program calls, similar to database management system access.

2. Data resource dictionaries, because they contain detail information concerning programs and software systems, could function as the interface between user requests, both online and batch, ad hoc and precompiled, and various database management systems (DBMS). Because the
DBMS data definitions are stored in the data dictionary, programs could call for data to the DRD which would, in turn, call each of the necessary DBMS systems to complete the retrieval. In addition, the data resource dictionary could contain sufficient information to control the restructure and reload of DBMS data.

3. Finally, the data resource dictionary should allow for reporting of the stored information. Reporting is not limited to simple lists or catalogues of attributes, entities and relationships but should include structure diagrams for programs, flow charts for the systems architecture, auditor reports which would present the required calls to extract selected data from files, data usage statistics, and physical resource allocation and usage statistics.

Armed with this type of information, the data processing staff could ultimately develop routines to simulate system environments and optimize the performance of both hardware and software.

The storage and control of database metadata would reduce or remove the problems of controlling databases which rely on specific computer hardware, and would allow for greater flexibility in using the correct database structure, based on the application requirement, while still making that information available to other applications running on other hardware.
2.4 Metadata Dictionary

The final layer of metadata software is the metadata dictionary. The preceding layers have modeled the data, processes, and the systems environment (hardware and software), the final layer models the enterprise itself. This layer contains descriptions of data types which are abstract. That is, previous descriptions have been concrete in the entities described (files, programs, databases). In the metadata dictionary, all data an enterprise uses, and ways in which that data is used, is described. Data which is not presently used, but may be used in future, or data which was used in the past can also be incorporated into the realm of data described.

The information stored in the metadata dictionary is conceptual. A conceptual view of data may not be linked to any implementation of data structures or processes. This conceptual viewpoint provides a basis for systems design and organization review. In addition, it provides a model for defining the scope of the design, allowing for better control over implementation of application systems. Additional data required in a data resource dictionary, which supports the conceptual view includes the ability to record entities such as employee, department or company. ¹³ These entities are described by attributes such as the data used, tasks performed, and output

generated. Relationships between entities are defined in terms of source documents, task completed, approved or verified, and output filed or distributed. In addition, entities are linked to functions, events or operations which are performed. These operations can be at various levels of corporate control (e.g. administration, supervision, or operations). At each level, the responsibility related to a job classification is identified. Events are documented in terms of origin, responsibility and entities affected. Several views or versions of the conceptual model may be correct. For example, an enterprise description changes over time. Where the conceptual description at one point in time may be correct, changes in the organization due to reorganization, changes in business objectives, or the takeover or merger with another enterprise will require different descriptions at those times, each one correct at that instant.

Two organizations which have discussed and supported conceptual data models within the data dictionary are the American National Standards Institute (ANSI) and the British Computer Society (BCS). ANSI identified a conceptual schema linked to an implementation view of the database, which described the database in terms of applications, and the operational view of the database, which identified the physi-

cal storage of data within the system. The ANSI conceptual schema described data at a meta level, however it required a structure since it was contained within a database system. The conceptual schema in the ANSI model could occur more than once. Each schema would be mapped to implementation and operational views.

BCS, on the other hand, defined a conceptual model which described data, events, and relationships to functions. This focus did not dwell on mappings to implementation or operational views but sought only to describe corporate entities and functions in a manner which would allow an analysis of data at the enterprise level. Because it was less precise, the BCS definition would be more difficult to implement; however it gave a broader statement of the place and function of a conceptual view.
CHAPTER 3

USERS OF THE DATA DICTIONARY

The data dictionary system supports three main classes of users:

1. Those who use data to accomplish some goal or objective. For these individuals, data is an object which is manipulated to achieve a desired result. They include the employees of an enterprise who are engaged in its day to day business dealings.

2. Those who are charged with providing data and information to users. These are individuals who provide data processing services to users making data more easily retrievable, summarizing and collecting data and designing and implementing systems which reduce the clerical effort of data manipulation. They include systems analysts, programmers, operators, and data entry personnel. For them, the data is a subject of investigation in itself and is used in aggregate or conceptual form only.

3. Those that seek to control data such as the Data Base Administrator and Data Processing Management.
Figure 2 details the interaction of different functions with the data dictionary. The matrix also includes two mechanical interfaces, with compilers and data base management systems.

We will deal with each of these groups in turn and examine, in detail, their interaction with the data dictionary system.

3.1 The Data Base Administrator

The database administrator (DBA) is the prime user of the data dictionary. In managing an enterprise's data, the database administrator must "build a data dictionary that correlates and defines the things that are important to an enterprise, as well as who uses them, where they are used, when they are used, the standards that apply, and the responsibility for accuracy and time limits". Major functions of a database administrator are:

1. Data definition documentation. To provide an accurate

---

15 Uhrowczik, p. 338.


<table>
<thead>
<tr>
<th>Function</th>
<th>User</th>
<th>auditor</th>
<th>D.P. Manager</th>
<th>Database Administrator</th>
<th>Application Programmer</th>
<th>Analyst</th>
<th>Operator</th>
<th>DBMS</th>
<th>Dialogue Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation of Entities</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Data - Process Relationships</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conceptual - Implementation Mapping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inconsistency Checks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version Control</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Planning Inquiry</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Division Source Generation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Database Schema Generation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation Module Source Generation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auditing Query Generation</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple DBMS Interface</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Flow Design Aid</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2**

Data Dictionary Functions
and complete definition of each entity used by an organization.

2. The selection and procurement of database management systems.

3. The analysis of data and processing requirements and the design of the database in an optimal manner.

4. The planning, supervision, and review of the initial loading of the DBMS with operational data.

5. The development and enforcement of database security and integrity standards.


9. Training and liaison with users.

In performing these functions the DBA requires a comprehensive system. A properly developed data resource dictionary will support each of these functions.
1. Entity documentation. The data dictionary contains data entity definitions. The use of an automated data dictionary by the DBA reduces the clerical work necessary to maintain data definitions. Once metadata has been entered into an automated system, manipulation of the metadata to answer questions can be done by applying systems routines rather than restructuring written documentation. The database administrator can distribute the gathered information and ensure that all users agree. Because data is defined only once there is central control of definitions. Users can review and suggest changes to the central definition. In addition, the formalization of definitions ensures that standards can be maintained.

2. Selection database management systems. A major task in the selection of a database management system is in characterizing the data to be stored in the DBMS. Volume, access paths, and structure must all be considered. The identification of these parameters assist in the selection of a database management system, identifying candidates that satisfy performance requirements. The data dictionary contains data definitions and performance attributes, which assist the DBA by formatting the data necessary to make the selection decision.

3. Database design. In designing the database structure
the DBA must evaluate the level of redundancy in data types versus performance considerations. In doing this, the DBA requires usage statistics which show the relative use of data. He/she can then perform an analysis of the typical access speed in arriving at a realistic database design. By using information stored in the data resource dictionary the DBA should be able to evaluate alternative designs in arriving at his/her conclusion.

4. Database loading. Once definitions have been agreed to by users, a database management system has been selected and installed, and the database has been designed, it is necessary for data to be loaded or stored in the database. The data dictionary assists the database administrator in his/her task by defining data use, the origin of the data, ultimate responsibility for its correctness, and the processes which will access the data. These processes could include ad hoc user requests, and batch programs which extract data to print on a report, or update the database. This central information source allows the database administrator to ensure that a complete definition of data is present before he/she attempts to load the database.

5. Database security and integrity standards. The data dictionary contains security and integrity parameters.
Depending on the class of metadata software implemented and utilities which may be available, it may be used as a reference by the DBA, or may assist in the implementation of execution time procedures. Execution time procedures, for example, would require data updating the database to be processed by a centralized validation program which would extract standard validation parameters from the data dictionary. Similarly, database access security could be controlled through security parameters which would be verified at compile or execution time. However, even a less automated system with only parameters for integrity or security stored in the data dictionary would enhance the database administrator's control of data by explicitly defining the level of standards enforced.

6. Database definition maintenance. Like any other software, the database definition is not static. It is therefore necessary to document the definition and to be able to modify it according to the users' needs. The data dictionary supports the DBA in doing this by allowing the documentation of more than one version of database definition. This version control would allow the DBA to identify those data definitions which are effective as opposed to those which are proposed, or superseded. Documentation of versions is an important tool in controlling the evolution of the database system, as
it provides a basis for discussion and continuity.

7. Database performance. In optimizing the performance of a database management system, the DBA requires statistics that identify the utilization of the DBMS. The data resource dictionary, assisted by utilities, can capture and store performance statistics. Because the DRD is not limited to controlling data stored on a database management system, performance statistics for alternate data structures can be captured by the appropriate utilities and stored.

8. Enforcing compliance of database standards. The data resource dictionary can assist the DBA in enforcing standards by ensuring that development methodologies for data use include the generation of database definitions and program data definitions using routines which utilize the definitions stored in the DRD. This effort can be assisted by the use of a utility which compares the data definition statements in source programs with the definitions stored in the DRD. Any differences are reported in an exception report. By using a highly visible, centralized documentation vehicle, the DBA ensures that users will be eager to maintain and use definitions stored in the data dictionary.

9. User training and liaison. Because the data resource
dictionary is a centralized, highly formalized data definition tool, the DBA has a product which can be the object of user training courses. Once a training program has been established, the DRD functions as an easily understood interface between user views of data and the data processing application of data in computer systems. Once difficulties in obtaining agreement over data definitions have been resolved, it is much easier for an analyst to complete the database definition and processing requirements design. By maintaining a data dictionary, the DBA can assist the systems analyst by providing accurate copy libraries for inclusion into new programs. In addition, the DBA can take control of data definitions used in testing new programs, keeping them distinct from production definitions. This relieves the analyst of the responsibility over data definitions for production systems and ensures that maintenance of systems will not affect production data versions, until after testing is complete.

The database administrator uses the data resource dictionary in much the same way as a buyer uses an inventory control system, to provide the information necessary to make intelligent decisions which optimize performance and minimize cost.
3.2 The Systems Analyst

The systems analyst performs two main functions. The first is the development of new systems. An analyst is responsible for identifying functional and data requirements of a user and translating that functional and data specification into an operational system.

The second function is modification or maintenance of an existing system based on changing user needs, due to a new business environment, an increase in the system's scope resulting from more sophisticated user processing, or correction of errors in the original system caused by programming or design.

In developing a new system, the analyst can utilize the metadata dictionary by documenting the conceptual view of the organization, identifying the scope of the system and its interfaces. A conceptual view of the enterprise utilizes entities such as individuals, procedures, files, and reports in completing a high level description of the enterprise. Once the analyst has identified the scope of the new system, a local conceptual model which is subset to the global view of the enterprise, can be developed.

This local conceptual model will identify the events which occur in the desired system, the entities (individuals,
data, procedures) which are affected, and the processes involved. These processes will specify relationships between entities. Definition of the local conceptual model allows for analyst-user dialogue in broad general terms. The metadata dictionary should be able to produce function matrices which map the physical organization of the enterprise to functions (see Figure 3), and information flow diagrams (see Figure 4), identifying the relationship between data, functions, and output.  

A functional description of the system will allow the system analyst to estimate the implementation effort required and to prepare an estimate of costs and benefits for the system. Supported by a high level of documentation, the system analyst can then review the justification for implementation of the system with user personnel.

Using this local conceptual model as a basis, the analyst translates the conceptual design into a systems design suitable for implementation. In doing this the analyst identifies the detailed data items and data structures necessary to support the functions. The analyst completes the design by identifying user responsibilities and work stations, data volumes and processing frequencies, security and privacy

### Function Matrix

#### Warehousing

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Warehouse Manager</th>
<th>Receiving Clerk</th>
<th>Forklift Operator</th>
<th>Shipping Clerk</th>
<th>Packer</th>
<th>Stock Clerk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warehousing</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine Inventory Requirements</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predict Demand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitor Stock Levels</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verify Inventory</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acquire Parts</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accept Shipment</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examine Contents</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store Parts</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record Parts Arrival</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dispatch Orders</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Assemble Orders</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pack Orders</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Ship Orders</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record Shipment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3**

Function Matrix
Information Flow Diagram
Dispatch Orders

Warehouse
Bin

Stock Clerk

Assemble
Orders

Out of Stock
Bin Cards

Purchase Orders,
Stock Parts

Customer
Purchase Orders

Order
Assembly File

Assemble
Desk

Pack
Orders

Packing Slips,
Containers

Completed
Order File

Record
Shipment

Purchase Order

Stock Part
Quantity

Inventory
Kardex

Accounts Receivable

Customers

Shipped
Order

Packing Slip

Shipping
Dock

Completed
Order File

Record
Shipment

Purchase Order

Stock Part
Quantity

Inventory
Kardex

Key:

- Information or material flows
- Business functions
- Information stores
- Sources or destinations of information or material outside of the business operation
- Sources or destinations of information outside of the network but inside the business operation
constraints on data and processes. The result of this implementa-
tion design is the detailed systems documentation used in the
programming and installation of the system.

The metadata database can store both the conceptual model and
the implementation design. Using cross-references and
graphical output similar to those shown, the analyst can
easily produce versions of the design which can be discussed
with the user until the optimal design is complete.

In modifying systems, the analyst is responsible for reviewing a system which is already in production. By reviewing the data resource dictionary output, the analyst can identify the impact of a change to a data structure or a program on other processes or data. With this documentation, the analyst can identify the effort required to make modifications to the existing system, and can ensure that the change is consistent with the enterprise conceptual view.

3.3 The Programmer

The programmer uses the data directory by extracting definitions necessary to prepare programs. The programmer is interested in the data entities such as elements, records, files or database structures, and programming entities such as

\[^{20}\text{Uhrowczik, p. 332.}\]
modules, programs and systems. Programmers are also interested in the physical location of data and operating characteristics of programs. In using a data directory, the programmer interfaces at compilation or execution time. If the data directory system has a compiler interface, the programmer uses the data directory independently of the program to produce output which is used during compilation. Data definitions, in the form of copy libraries and program definitions, such as validation routines, are linked to the source code a programmer has written. In addition, the programmer can use the data directory to generate tables in storage which a program can reference.

A data directory used at execution time is similar to a database in that the program calls the data directory to extract data definitions (for example, to use validation criteria stored for data elements), or to control system processes (for example the data directory, because it stores data location information, could call additional DBMS systems in retrieving data). The use of the data directory as an intermediary between the programs and multiple DBMS systems is a highly sophisticated execution time interface.

The programmer can also use the data directory to generate data for testing. By utilizing the data definitions stored in the data dictionary and a utility which generates consistent test data based on these definitions, the programmer is relieved of the burden of creating a test model. The
data resource dictionary can also generate job control language by utilizing the systems definitions stored.

The emphasis in the programmer's use of the data dictionary is in a reduction in the amount of source code written. Standard program code such as edit modules, data definitions, database definitions and calls, report headings, file definitions, and special calculation routines could be dictionary generated.

3.4 The Operations Department

The data resource dictionary system can assist the operations department in controlling production systems by providing standardized documentation. Through utilities the DRD can act as a librarian for programs and data, and assist the operator in version control. Documentation of recovery/restart for programs can also be maintained. Because the data resource dictionary records the mapping between users and reports, distribution of reports can be maintained. By documenting the physical components of a system, the data resource dictionary, aided by utilities, can store machine productivity data.

In providing this information, three additional types of

metadata are necessary: 22

1. Systems control data. This includes information which will allow an operator to make decisions concerning security, and access. Information stored should include:

a) access rights to data (databases), devices (word processors), and output (reports)
b) distribution of reports to users
c) identification of authorization for the initiation of jobs.

2. Job management data. In supporting the execution of test and production jobs, the operator requires information concerning device requirements, parameters to be input during processing to control job control language, instructions to allow recovery from system failures and to restart processing, instructions which detail the action necessary when error messages are returned from the system process, and file and program version control.

3. Resource management data. The data resource dictionary can assist operations in controlling and optimizing the

---

22 ANSI/X3/SPARC, p. 27,8.
use of physical devices by storing information which identifies system devices and their configuration, levels of utilization and load demands, the assignment of alternate devices when failure or unavailability occurs, and the accounting and charge back of device utilization to the user.

Some of this information is held in various utility systems used by operations personnel. Much of the information, however, is not documented in any place. The data resource dictionary provides an opportunity to formalize the information requirements of operations and to integrate this data with the conceptual view of operations.

3.5 The User Group

The remaining group that can make use of a data dictionary are individuals who are end users of data and management, who must plan for the continuing use and development of the data resource. The proliferation of specialized data processing systems, each using their own data base, has had the effect of creating an enterprise which is composed of a number of essentially independent functional entities.

Employees are becoming functionally specialized and may not be able to relate their tasks to the overall operation of
the business enterprise. This, in turn, makes coordination more difficult as fewer individuals know all functions which are performed and the data which is used. If an employee leaves, no guarantee can be made that other employees can meet the needs of the business. This process has been exacerbated by data capture systems which allow entry of the input at the source, reducing the access to, and sharing of, information.

The metadata dictionary resolves some of these problems by assisting in the training of new employees. The metadata dictionary can provide a model of business functions, both globally and locally, which can provide perspective for the new employee. Responsibility matrices stored within the data dictionary can identify which functions are to be performed by each employee, the output of those functions, and the data that is necessary to complete them. The metadata dictionary can identify information that is shared between functions and can document the storage of information. The data dictionary allows for a central, accessible source of information about the organization stored at a conceptual level. Changes in procedures in one functional area can be analyzed to identify other affected areas.

As planners of information usage and the development of information systems, management can use the same conceptual

---

23 British Computer Society, p. 9.

metadata dictionary to identify where the data model does not serve the needs of the organization. This allows for more efficient communication of information planning between management, data processing, and users.

In managing the data resource, three things are necessary. The first is to know where data is stored, and how it is used. This allows for the controlled dissemination of data to those individuals who require it. In essence this flow of data is the information system of an organization.

The second requirement is the ability to control the development of the information system and the change in usage of data. This control will ensure that, at a minimum, a status quo of information usage can be maintained, and the organization can respond to information needs based on changes in its external environment.

The third factor necessary for successful management of data in an organization is a plan for the change in the information system to optimally meet user requirements. This plan should focus on the application of data processing resources to provide data to users. These plans will be able to impose controls over the collection, ordering, storage and use of data and centralize the authority over data.

---

CHAPTER 4

SURVEY OF COMMERCIAL DATA DICTIONARIES

Introduction

This chapter will examine five data dictionary packages currently available commercially. These packages represent the major systems in use. Data for the review was obtained from four publications which evaluated commercial products. Manufacturers literature was also obtained for all packages reviewed. Table 1 summarizes the source of information for each of the systems reviewed. The four publications were:

1. Data Dictionary Systems by Henry C. Lefkovits. This book was published in 1977, and is the most comprehensive discussion of data dictionary systems to date. It briefly describes the definition, use and features of data dictionary systems, and then evaluates six major systems. In the evaluation, the author spends approximately one half of the text describing the data description language for the entities and attributes.

2. Technical Profile of Seven Data Element Dictionary/Directory Systems, by Belkis Leong-Hong and Beatrice Marron. This study was published in 1977.
<table>
<thead>
<tr>
<th>DATA DICTIONARIES DISCUSSED</th>
<th>CULLINANE - IDD</th>
<th>MSP - DATAMANAGER</th>
<th>SYNERGETICS DATA CATALOGUE 2</th>
<th>UNIVERSITY COMPUTING - UCC-10</th>
<th>IBM-DB/DC DATA DICTIONARY</th>
<th>ICL - DDS</th>
<th>CINCOM - DCS</th>
</tr>
</thead>
</table>

Table I

Information Sources -- Data Dictionary Packages
Three quarters of the publication described data dictionaries, defined the survey methodology, and described the feature list used to evaluate the seven systems. The profile consists of 52 features presented in tabular form.

3. Data Dictionary Systems by J.D. Lomax. This book was published in 1977, and reviewed eight systems available in the United Kingdom. Approximately 40% of the book deals with the definition, uses and implementation of a data dictionary system. A set of 24 questions is used to evaluate the packages. No examination of input data language is done, unlike in Lefkovits, where the discussion was extensive.

4. A Survey of Data Dictionaries by Robert M. Curtice and E. Martin Diekmann. This article, published in March 1981, presented a brief look at nine popular data dictionary systems which would run on IBM 360/370, 43XX, and 30XX computers. The authors allocated one page to each system, focusing on dataflow within the system, entities, inputs, and interfaces.
The author's analysis of each package will focus on four characteristics:

1. The entity types supported by each system.

2. The methods available to input data into the dictionary system.

3. The output produced by the system (e.g. reports, copy libraries, database definitions).

4. Security features available in each system.

4.1 Datamanager

Datamanager is marketed by MSP, Inc. of London, England, and is reported to have 500 installations. It is written in IBM Assembler for the IBM 360/370 environment. It will also operate on any compatible mainframes. Datamanager will support IMS, DL/1, and other DBMS systems which are supported on the IBM hardware.

Datamanager supports seven main entities as well as user defined syntax. The main entities are:

1. ITEM. The lowest level of data with attributes such as language alias, description, input and output formats,
name, length, picture, and value range.

2. GROUP. A group is a collection of items or groups, with unlimited nesting.

3. FILE, which is a collection of ITEMS and GROUPS. Attributes include CONTENT, ALIGNMENT, and FORMAT, which describe the ITEMS and GROUPS, as well as ORGANIZATION (indexed, sequential), SORT parameters, DEVICE and SIZE.

4. DATABASE, which defines the database using attributes which are particular to that database. For example, IMS-DATABASE will have attributes such as SEGMENT, SEARCH-KEY, PHYSICAL-PARENT, LOGICAL-PARENT and LOGICAL-CHILD. Special attributes are provided for each of the databases supported by Datamanager.

5. MODULE. This is a collection of program language statements performing an action. The attributes of MODULE are CALLS, INPUTS, OUTPUTS, and UPDATES.

6. PROGRAM, which is a group of modules. PROGRAMs have the same attributes as MODULES, except that they CONTAIN MODULEs rather than CALLing them.

7. SYSTEM. The SYSTEM, which is the highest level in the processing hierarchy, is composed of PROGRAMs, or other
SYSTEMS to an unlimited level of nesting. Systems and subsystems are both represented by the SYSTEM entity.

Datamanager also allows for the definition of user entities. This is done by using the basic Datamanager entities and CATALOG attributes to establish relationships. For example, DATAFLOW entities are defined as GROUPS containing ITEMS. PROCESS entities are a special case of Datamanager SYSTEM entities and have DATAFLOW as INPUT and OUTPUT attributes. Conceptual designs can be documented using regular Datamanager entities. Using search commands, and the CATALOG attributes, questions such as:

WHAT FORMS 'DATAFLOW' 'ACCOUNTING'

could identify the dataflows used by the accounting department. GROUP entities in the accounting department would be coded:

CATALOG 'DATAFLOW'

'ACCOUNTING'

Input to Datamanager can be done online or in batch mode. Utilities are provided for extracting data definitions from COBOL and PL/1 data divisions, and from IMS, IDMS, TOTAL, ADABAS, and S/2000 database management systems. These definitions can be stored and modified by the DBA before loading them into the data dictionary. The CONVERT command allows the DBA to identify inconsistencies which must be resolved before data is entered into the dictionary.
Datamanager produces reports for each of the entities stored in the system, listing all attributes. Datamanager also produces cross reference reports. This is done using the REPORT command, which traces all references lower than the specified occurrence. For example, EMPLOYEE-FILE, a FILE occurrence will list all references to GROUPS and ITEMS. Commands identifying usage (such as ITEMS in FILES, or ITEMS in PROGRAMS) are available in batch and online mode. These commands have selectivity parameters which allow the user to control the amount of information displayed.

Datamanager, through the PRODUCE command, prepares data divisions for COBOL and PL/1, BAL and MARK IV programs, as well as Data Definition Language statements for ADABAS, IDMS, IMS, DL/1-DOS-VS, MARK IV, S/2000, and TOTAL.

Datamanager allows for three levels of security:

1. CONTROLLER. The CONTROLLER generates the dictionary and assigns passwords and priority levels to users.

2. USERS. USERS are given access to the dictionary by the CONTROLLER and are assigned a priority level. The priority level permits the user to perform commands provided that their level is equal to or higher than the security level required.
3. OWNERS. OWNERS are given ownership of dictionary entity occurrences and may, in turn, define priority levels for access to their data to other USERS.

This allows data to be protected from individual users and, further, from groups of commands issued by USERS. An entity can have three security levels:

1. Access security, which controls commands used to output information from the dictionary.

2. Alter security, which controls addition and modification of occurrences in the dictionary.

3. Remove security, which controls the deletion of data from the dictionary.

4.2 DB/DC Data Dictionary

The DB/DC Data Dictionary is IBM's dictionary system intended primarily for the support of IMS and DL/1 databases. It utilizes 6 DL/1 databases and will support only IMS and DL/1, on IBM or plug-compatible mainframes. It requires a batch region of 560K of virtual storage to run.

DB/DC Data Dictionary supports 15 pre-defined subject categories (entities). It also allows for up to 200 user-de-
fined categories (entities). The IBM supplied entities can be broken down into three groups.

1. Data entities:
   a) ELEMENT
   b) SEGMENT
   c) PCB
   d) DATABASE

2. Process entities:
   a) SYSTEM
   b) JOB
   c) PROGRAM
   d) MODULE

3. Data Interface entities:
   a) TRANSACTION
   b) PSB
   c) SYSDEF
   d) DDUSER

ELEMENT can be used to identify an indivisible unit of data or a group, through the CONTAINS attribute. Attributes are included for COBOL or PL/1 specific definitions. SEGMENT can be used for both IMS segment and conventional record use. The PCB entity is used to describe the collection of ELEMENTS to which a program has access. Attributes of the PCB identify
whether it is associated with a database, a conventional file or a teleprocessing transaction. The DATABASE entity can be used to describe both DL/1 databases and non-DL/1 files (data sets).

Processing entities are all described by attributes such as Name, Description and User. Simple relationships between Process entities and data entities can be set up using the ADD-RELATIONSHIP command. PROGRAM entities have additional attributes such as program language (PGMLNG) and program type (PGMTYPE), whose allowable values are T (teleprocessing), D (database), and BATCH.

Data interface entities allow for the specification of entities describing the relationship between the data dictionary and user (DDUSER), the system and the database (SYSDEF), programs and the database (PSB), and the program and on-line transactions (TRANSACTION). DDUSER contains security information and passwords for all users. SYSDEF is used to document the IMS/V$ system definition information; PSB is used to document the database segments which are available to a particular program in an IMS environment.

User defined entities are defined by using three additional "meta-entities". CATEGORY identifies the user-defined entity. For example, the user may want to store attributes of DOCUMENTS, PROGRAMMERS or DEPARTMENTS. In defining the CATEGORY, ALIASes can be recorded, a DESCRIPTION can be main-
tained, and syntax rules can be defined. ATTRTYPE allows the user to define attributes for the category. RELTYPE allows for named relationships to be established between user-defined entities and system, or other user-defined entities.

Input to the DB/DC Data Dictionary is through online display forms, which include explanation frames, or through a batch input stream, for bulk data entry. Data definitions can be captured from COBOL source statements or PL/1 structure declarations. Data definitions can also be created from database description (DBD) and program specification block (PSB) libraries.

Output from the data dictionary includes both reports and generated source statements. DB/DC Data Dictionary produces reports through two commands:

1. REPORT. This command is used to do entity reporting. All or some attributes of a given entity occurrence, the names and definitions of entity types, or reports of generated programming code can also be printed using this command.

2. SCAN. This command allows the user to search for a specified character string in the NAME, or entity description field of an entity type or a specified related entity type.
Generated source statements can be created for database definitions (DBD-OUT), program status blocks (PSB-OUT) and for data structures in COBOL, PL/1 or Assembler language (STRUCTURE-OUT). Users can implement other utilities which interact with the data dictionary database using the EXECUTE command. These programs can be used to produce custom reports or provide customized error messages to on-line users. The interface is through the Dictionary and does not access dictionary databases directly. No update of the dictionary data is possible except through the data dictionary validation routines.

Security is provided through the use of the DDUSER subject category. Attributes such as User ID, password and status codes can be specified. Signon can be controlled at the DB/DC Data Dictionary menu level or the IMS/VS or CICS data communication interface level. The security features allow for read, or read and update access to a pre-specified number of subject categories (entities).

4.3 Data Catalogue 2

Data Catalogue 2 is a dictionary package marketed by Synergetics Corp. of Bedford, Massachusetts. It is written in ANSI COBOL and can be implemented on IBM, UNIVAC and CDC machines and their plug compatible counterparts. It features a large number of DBMS interfaces and is designed for large
organizations. Over 150 installations are reported, one half of whom are non-DBMS users.

Data Catalogue 2 supports a large number of pre-defined and user-defined entity types. The basic entities supported are:

1. Data entities such as:
   a) ELEMENT
   b) GROUP
   c) RECORD
   d) FILE
   e) DATABASE

2. Process entities such as:
   a) MODULE
   b) PROGRAM
   c) SYSTEM

3. Conceptual entities such as:
   a) EXTERNAL
   b) TASK
   c) USER

4. Information interchange entities such as:
   a) REPORT
   b) FORM
5. Database specific entities such as (for IMS):
   a) SEGMENT
   b) SSA
   c) PSB

The user can also specify other entity types, attributes for these entities and relationships to other entities.

The conceptual entity, EXTERNAL, can be used to describe an external processing resource such as a mini-computer, or procedural guide. Attributes provided for EXTERNAL include RESPONSIBILITY, NAME, and LOCATION. Forms may be referenced by MODULES, TASKS, SYSTEMS, and USERS and may reference REPORTS or data entities. Data Catalogue 2 supports HIPO (IBM's Hierarchical Input Processing Output) documentation methodology. Techniques for using entities such as PROCESS, DATAFLOW and DATASTORE are proposed by the vendor.

Data Catalogue 2 supports both on-line and batch data input through a keyword based input language. A facility for fixed format batch input based on positional parameters can be specified in a CICS environment. On-line entry features tutorial prompted entry for inexperienced users. Data input is facilitated by a utility which extracts data definitions from COBOL programs. An additional feature is the RENAME command which allows an entity occurrence similar to one already stored in the dictionary to be generated.

Output from Data Catalogue 2 can be produced both on-line
and in batch, through five output interfaces:

1. On-line or batch ad hoc queries. These queries use three basic commands:

   a) COUNT - to identify the number of occurrences which satisfy a query.
   b) LIST - which returns the names of occurrences.
   c) SHOW - which returns the attributes of occurrences.

The query language has a number of modifiers to allow a sophisticated search requests.

2. Reports. There are four types of reports:

   a) CATALOG reports which return all attributes stored for a specific entity occurrence or for the entire entity type

   b) HIERARCHY reports which show all components of an entity occurrence

   c) USAGE reports which identify the references to an item

   d) RELATIONAL reports which produce a cross-reference between all entities. The resulting report can use
one of a number of names or aliases stored for each entity.

3. Data definition language and program control blocks are generated for a number of database management systems including IMS, DL/1-DOS/VS, TOTAL, ADABAS, MARK IV, IDMS, DMS1100, and S/2000.

4. Program data divisions are generated for COBOL, PL/1 and Assembler.

5. Programs can call the data dictionary during execution time to extract data.

Security is provided by password validation for users, and level of authority for command execution and data access. Users can be assigned ownership to specific entries.

4.4 UCC-10

UCC-10 is marketed by University Computing Company of Dallas, Texas. It is intended as a supporting tool to those installations using IBM's IMS or DL/1. It is written in COBOL (90%) and Assembler language.

UCC-10 has eight basic entities to describe data and processes. There are also 13 additional entities used in an IMS
data communications environment to describe communications and terminal entities. The communications entities allow relationships to be established between communications, data, and processing entities and are used by UCC-10 utilities to generate IMS/DC systems operations code. The eight basic entities used by UCC-10 are:

1. Field. UCC-10 uses field to denote the indivisible unit of data. A field is described by attributes such as FLN (name), PIC (COBOL-type picture), JUST (justification), and CBLN (COBOL name). Relationships to segments (SGN), and program (PGN) are maintained.

2. List. UCC-10 uses this attribute to identify secondary indexes in the database. Attributes of lists are LIST (which is the name) and TTL (a description of the list). Relationships are maintained with segments (SGN) and fields (FLD).

3. Segment. The segment entity can be used as an IMS segment or as a record in a conventional file environment. Attributes include SGN (segment name), MAXB (the length of the segment in bytes), MINB (the minimum size if it is a variable length record), and IMS-related attributes. Some of these attributes define the position of the segment within the IMS hierarchy. For example, segments can be logical parents and define logical children
(LCN) and the database within which they are logical parents (LDB).

4. Data Set Group. This entity is used by UCC-10 to define a data set group in the IMS environment or a file in the conventional data storage environment. Attributes include name (DSN), input or output blocksize (BLK1 or BLK2), device type (DEV), and other OS or IMS attributes.

5. Database. This entity is used to define IMS databases as well as OS files. Its attributes, such as DL/1 (which identifies it as a DL/1 database), and RMOD (which identifies the name of a randomizing routine for a HDAM database), identify additional parameters for the generation of a DL/1 or IMS database. The relationships to programs and data sets position this entity as a logical connection. There are no relationships between programs and data sets.

6. Module. The module is the lowest level of process entity. Attributes of the module are MDN (the module name), LANG (the source language of the module), PGMR (the programmer or analyst responsible for the module), and TTL (a 30 character description). Modules can be linked to programs.
7. Transaction. This entity is used to identify an on-line conversation input record. Attributes deal with IMS/DC handling of the transaction input. They include CLAS, the IMS message region class; IQRY, whether or not the transaction will modify an IMS database; SPAL, the scratch pad area location; and TYPE, whether the transaction is processed in real time and generates an immediate response. Transactions can be linked to a program.

8. Program. UCC-10 supports program entities with attributes such as PGN (the name of the program), LANG (the source language), and TYPE. A program can be one of four types: a DL/1 program which can process messages in batch, an on-line mode, a DL/1 program which does not process messages and can only run in batch, and a program which does not interface with DL/1. UCC-10 stores relationships to other data entities, such as FIELD, Segments and Databases in the PROGRAM entity.

Input to UCC-10 can be done in on-line or batch mode. Input statements are a mixture of free and fixed format statements, where keywords are used to identify the data input. On-line data entry transactions include queries and can be processed through batch update or executed immediately. UCC-10 has utilities to allow data definitions to be extracted from existing IMS or DL/1 database definitions, as well as
from COBOL and PL/1 I/O areas.

UCC-10 has a complete set of reports for each entity supported. Parameters in the REP command allow the user to specify the name of the entity occurrence, the type of attributes and relationships (called CONNECTIONS in UCC-10), and the underlying structure (which may be where-used or contains relationships) to be output. Glossary reports output general descriptions stored in the TEXT attribute for selected occurrences in the dictionary. Keyword in context reports (KWIC) are provided for words stored in the element name, title, COBOL, or PL/1 name fields or in the description text.

UCC-10 has utilities to generate IMS statements for database generation, program I/O areas, program database calls (SSA) and to support the data communications facilities of IMS. UCC-10 also generates COBOL, PL/1 and Assembler I/O areas, and Terminal Security maintenance transactions.

UCC-10 is designed to support IMS or DL/1 in an OS or DOS environment. As is obvious from the entity types, this data dictionary would not function as well in a non-IMS or non-DL/1 environment. Security features are implemented through the use of IMS terminal security codes. Terminals can be designated as read only (HFR) or read and write (HFD).
4.5 IDD

IDD is a data dictionary developed by Cullinane Database Systems, as a front end to their IDMS database. IDD utilizes IDMS (which is a CODASYL format database) to store its information. IDMS automatically updates the dictionary when users define the database schema. IDD is reported to have over 200 installations.

Entity support in IDMS is extensive. There are eight basic data processing entities:

1. SYSTEM
2. PROGRAM
3. MODULE
4. ENTRY POINT
5. FILE
6. RECORD
7. ELEMENT
8. USER

There are also nine teleprocessing entities which allow documentation of on-line systems. These entities may only be used if IDMS-DC is installed. As many of the relationships
and attribute values are automatically generated by IDMS-DC compilers. These entities are:

1. PANEL
2. MAP
3. LINE
4. PHYSICAL TERMINAL
5. LOGICAL TERMINAL
6. DESTINATION
7. QUEUE
8. TASK
9. MESSAGE

Finally, IDD allows for two "meta-entities" which allow users to define attributes of entities and classes or types of attributes. They are:

1. CLASS
2. ATTRIBUTE

All standard entities except for entry point can have relationships with entities of the same type. SYSTEMS, for example, can be related to other SYSTEMS, ELEMENTS to other ELEMENTS and so on. This allows for structures such as programs, subprograms or systems and subsystems to be recorded.
In addition, IDD provides for attributes which identify relationships between entities. Examples are:

<table>
<thead>
<tr>
<th>SOURCE ENTITY</th>
<th>TARGET ENTITY</th>
<th>RELATIONSHIP ATTRIBUTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER</td>
<td>SYSTEM</td>
<td>OF SYSTEM</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>USER</td>
<td>RESPONSIBLE</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>USER</td>
<td>PREPARED, RESPONSIBLE</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>SYSTEM</td>
<td>WITHIN SYSTEM</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>MODULE</td>
<td>MODULE USED</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>RECORD</td>
<td>RECORD COPIES</td>
</tr>
<tr>
<td>FILE</td>
<td>USER</td>
<td>PREPARED, USER-IS</td>
</tr>
<tr>
<td>RECORD</td>
<td>USER</td>
<td>PREPARED, USER-IS</td>
</tr>
<tr>
<td>RECORD</td>
<td>FILE</td>
<td>WITHIN FILE, includes KEY</td>
</tr>
</tbody>
</table>

Relationships may be defined by users through the "Relational Key IS" statement. Relationships between records and elements are provided by the RECORD ELEMENT substatement which allows elements or structures of elements to be related to a Record. The RECORD ELEMENT substatement also allows the ELEMENT to be redefined, i.e. PICTURE, VALUE, INDEX KEY and other attributes can be changed. The ELEMENT definition, however, will not change in the dictionary. IDD also allows the user to define ELEMENT-RECORD relationships by using a COBOL-
type substatement. This allows regular COBOL syntax to be used to set up RECORDS and RECORD-ELEMENT relationships. Any ELEMENT referred to which is not on the dictionary will also be set up.

The teleprocessing entities, PANEL and MAP, allow for the documentation of on-line screens and relationships between systems and screens. The Data Dictionary Definition Language compiler does not build relationships between MAPs and RECORDS or ELEMENTS. This is done by the IDMS-DC mapping compiler through the USING RECORDS statement. IDD only records the existence of PANELS and MAPS and relationships from PROGRAMS or SYSTEMS to MAPS, and from MAPS to PANELS.

The IDD MESSAGE entity allows for messages to be stored within the system. Messages can be error messages, or operator control messages. The MESSAGE entities supported by IDMS-DC are error messages. Message text, destination, and severity code can be stored. IDMS-DC uses MESSAGE entities at execution time.

The remaining teleprocessing entities, LINE, PHYSICAL TERMINAL, LOGICAL TERMINAL, DESTINATION, QUEUE and TASK are used to document an on-line environment.

TASKS are teleprocessing work units. Tasks INVOKE programs WITHIN systems. TASKS can be assigned a SECURITY CLASS and a TASK PRIORITY.
QUEUES documents a group of tasks handled by the teleprocessing system. This instructs the IDMS-DC system to allow a number (THRESHOLD) to be reached before a task (THRESHOLD TASK) is invoked within a SYSTEM.

A DESTINATION documents a cluster of terminals or users, to whom messages, from systems or the operator, would be sent. Relationships to systems (WITHIN SYSTEM), users (USER IS) and terminals (LOGICAL TERMINAL) are maintained.

A LOGICAL TERMINAL establishes a relationship between SYSTEMS and PHYSICAL TERMINALS or devices. The logical terminal relationship can create links between more than one system, to a unique physical terminal.

The PHYSICAL TERMINAL entity documents the LINE each SYSTEM will use to access the terminal. It also establishes the device type.

The LINE entity defines the type of telecommunication lines which exist in a data communications environment. The LINE entity has the ability to document LINE-SYSTEM relationships (WITHIN SYSTEM).

CLASSES and ATTRIBUTE entities can be used for two main purposes. First to more completely define an entity which already exists. For example, a CLASS "Department" can be added with ATTRIBUTES "Accounting, Marketing, Payroll". These can then be used in further defining the entity USER. Second,
the CLASS and ATTRIBUTE can be used to simulate other entities. This is possible because ATTRIBUTES can be nested. That is, ATTRIBUTES can contain classes which also have ATTRIBUTES. For example we can define this structure:

ADD CLASS NAME IS DEPARTMENT
   ATTRIBUTES ARE AUTOMATIC

ADD CLASS NAME IS DIVISION
   ATTRIBUTES ARE AUTOMATIC

ADD CLASS NAME IS CORPORATION
   ATTRIBUTES ARE AUTOMATIC

ADD ATTRIBUTE RESEARCH WITHIN DEPARTMENT
ADD ATTRIBUTE ADMINISTRATION WITHIN DEPARTMENT
ADD ATTRIBUTE CUSTOMER RECORDS WITHIN DEPARTMENT

ADD ATTRIBUTE BELL NORTHERN WITHIN DIVISION
   DEPARTMENT IS RESEARCH
   DEPARTMENT IS ADMINISTRATION

ADD ATTRIBUTE BELL CANADA WITHIN CORPORATION
   DIVISION IS BELL NORTHERN
   DEPARTMENT IS HEAD OFFICE
   DEPARTMENT IS CUSTOMER RECORDS

This structure defines the hypothetical organizational
hierarchy of Bell Canada. In this example, Bell Canada is a corporation who has one division, Bell Northern (which is a research arm). This division contains two departments, RESEARCH and ADMINISTRATION. The ATTRIBUTES ARE AUTOMATIC phrase allows the user to identify new attribute names and values by keying them in during definition. This is the case with DEPARTMENT IS HEAD OFFICE.

Input to IDD is through a comprehensive user language, and only in batch mode. The data dictionary has interfaces to COBOL, PL/1, RPG and BAC utilities to allow database usage (IDMS only) to be captured. In addition, schemas and subschemas added to the IDMS database automatically update the data dictionary. Programs which use definitions stored in the data dictionary automatically create dictionary occurrences when processed by the DML (data manipulation language) interface.

Output from IDD can be through the Dictionary/Directory Reporter (DDR), through user-defined reports which use the CULPRIT report language or OLQ (On Line Query), or through user-written report programs accessing the IDMS based IDD data dictionary through regular database access routines. The DDR is a set of predefined CULPRIT report parameters stored in the data dictionary. There are 66 standard reports which:

1. List all attributes of each standard entity type.

2. List, in summary, all occurrences in the entity type
3. Lists all attributes for those entity occurrences which have been requested by a "key 'name'" statement. Key usually refers to the name of the entity type.

4. Report cross references between:

a) files and records
b) file names and synonyms
c) record names and schemas
d) element - synonyms and groups
e) element names and element descriptions
f) element designator classes and element names
g) files and programs
h) IDMS sets and programs
i) records and programs
j) areas and programs
k) elements and programs.

5. Output source code onto cards or into a disk file.

There are also 32 reports which are available to users of IDMS/DC. These reports describe:

1. The IDMS/DC network which consists of lines and physical and logical terminals. Three reports are provided:
a) Network Description by Line

b) Network Description by Physical Terminal

c) Network Description by Logical Terminal.

2. The programs, tasks, queues, destinations, users, and systems within an operational IDMS/DC system.

3. Messages and devices in the system.

4. Screens (called PANELS), MAPS, which are relationships between PANELS and systems, and RECORDS or ELEMENTS which appear on the PANELS.

Because the DDR uses pre-defined CULPRIT parameter input to generate standard reports, the user can modify these parameters which are stored as MODULE occurrences in the data dictionary. In this way, the user can restrict the extraction of entity occurrences to those he/she desires, modify a sort sequence, or add additional entities or attributes to a report. The user can also write his/her own CULPRIT reports. The data dictionary is an IDMS database, therefore the data dictionary's schema is stored within the data dictionary and can be used by CULPRIT to extract information.

This same schema can be used to allow programs written in COBOL, PL/1, FORTRAN and Assembler languages, which have interfaces to IDMS, to extract dictionary data for reporting or any other purpose.
Security in IDD is provided by:

1. The specification of USERS who are assigned passwords.

2. The specification of a USER who is given the power to turn the global SECURITY attribute ON or OFF.

3. The specification of a RESPONSIBLE code for the occurrence. The possible alternatives are:
   a) DEFINITION - This allows the user named to modify or delete the entity occurrence
   b) RESPONSIBLE without any parameters, which will default to DEFINITION.

Other responsibility codes, CREATION, UPDATE and DELETION, are not used by IDD, but are available for documentation.

4. The specification of IDMS and/or DC authority (AUTHORITY IS IDMS/DC), which permits the user to modify and/or delete schemas and subschemas, or other IDMS functions if IDMS is specified, and modify or delete the teleprocessing entities if DC is specified.

There are a number of other systems on the market including some which have been recently introduced. Documentation,
beyond promotional literature, was not available for these systems. However, the author would like to mention two systems, CINCOM's Data Control System and ICL's DDS because of their state-of-the-art design. Features reported are taken directly from the manufacturer's literature.

CINCOM DATA CONTROL SYSTEM

This system is divided into 4 sub-systems:

1. The Interactive Data Dictionary.
2. The System Design Facility.
3. The Programmer Work Station.
4. The Data Security System.

The data dictionary allows input from on-line screens which may be applied to the data dictionary immediately or batched. Maintenance of data definitions can also be done on-line. The system also allows for extraction of data definitions from programs and databases.

Output is through on-line query and batch reporting. Queries may be catalogued and are expressed in a "relational language". On-line query results can be routed to the printer. Changes to data divisions of programs which are held by the dictionary, automatically modify copy libraries so that programs need only be recompiled to execute.
Data Control System supports data processing and user environment entities such as systems, programs, modules, databases, files, elements, source documents, terminal screens, reports, users, departments, and divisions.

The System Design Facility allows the analyst to use data dictionary information to create data-flow diagrams and logical data structures and translate them into a physical database design.

The Programmer Work Station provides programmers with full screen edit capability, library access (DCS libraries, PANVALET and LIBRARIAN), and an interface to the data dictionary to extract data division formats and generate code for database access. Programs can be compiled and executed online with results routed to the terminal or a printer.

DCS Data Security provides execute time security checking at both the file and element level. All users of DCS are screened by password interrogation at signon, to ensure that the user's profile matches the tasks performed. DCS is part of the TIS system marketed by CINCOM, which espouses a consistent, dictionary based view of data processing, database management and communications management.
ICL DDS

The ICL DDS system is important because of the formalization of a model which includes both physical data and processes as well as logical data and processes, and relationships between them.

Data input and output are on-line or in batch. Input is via keywords and the DDL has commands such as INSERT and REPLACE, output commands such as DISPLAY, SELECT, LIST and PRINT, along with selection statements FOR, WITH and ALL promise a complete and simple to use system.

Entities are broken into 4 main types:

1. COMPUTER DATA:

   a) V.FILE
   b) FILE
   c) RECORD
   d) GROUP
   e) ITEM
   f) AREA
   g) SCHEMA
   h) SUBSCHEMA
   i) SET

DDS uses the IDMS database which is a CODASYL system. This
accounts for entities f through i.

2. COMPUTER PROCESSES:

a) SYSTEM
b) PROGRAM
c) MODULE

3. REAL WORLD DATA:

a) ATTRIBUTE
b) ENTITY
c) RELATIONSHIP

4. REAL WORLD PROCESS MODEL:

a) EVENT
b) OPERATION.

Special entities allow for links between these entity types. These entities, DPROC, DMAP, PMAP, DUSE, allow for the explicit mapping of conceptual entities to implementation entities.

DPROC maps between conceptual processes and conceptual data entities using attributes such as CREATES, DELETES, UPDATES, which describe the effects of OPERATIONS on ENTITIES.
DUSE maps between implementation data and implementation processes. Attributes such as PROCESS, CREATES, DELETES, READS and UPDATES are documented for PROGRAM-FILE, PROGRAM-RECORD and SYSTEM-FILE relationships, among others.

DMAP maps between conceptual data and implementation data. Through this entity, ENTITY-FILE, ENTITY-GROUP and RELATIONSHIP-ITEM links can be made. Attributes used include CONCEPTUAL-DATA and IMPLEMENTATION-DATA. This allows, for example, an entity such as EMPLOYEE to be linked to the EMPLOYEE record or the AGE attribute to be linked to the EMP-AGE item.

Finally, PMAP maps operations to programs and systems, linking business functions to systems and programs which implement them. Attributes maintained include CONCEPTUAL-PROCESS and IMPLEMENTATION-PROCESS.

The author believes that the direction taken by these and other similar systems in implementing data dictionaries that address both the conceptual view of data and processes and the need to integrate the data dictionary into a comprehensive tool for all users of data, is an indication of the maturing of data dictionary systems.
Overview

The previous chapters have examined data dictionaries from a structural perspective. We have identified a model for looking at data dictionaries which classify the contents. In this classification, a data dictionary itself contains only data descriptions. Data element location information is contained in the data directory, which also documents process data and provides relationships between data elements and processes. Data processing information, the physical attributes of data storage and program execution, are contained in the data resource dictionary. The final level of information about data and processes is that which concerns the conceptual view of the enterprise. This level we have called the metadata dictionary.

In addition to developing a structure for dictionary software, we have identified those individuals in an organization who would most likely utilize the data dictionary, and the functions performed which would utilize the metadata dictionary.

In examining software packages which are offered by
various vendors, we have identified functions and features which are commercially available in data dictionary packages. The majority of data descriptions, data location and some metadata information is available in commercial packages. The availability of interfaces between data dictionaries, programs, and database management systems represent some of the functions performed by the data resource dictionaries.

This chapter will focus on management topics which will most certainly arise when an organization considers the implementation of a metadata dictionary. This chapter will review some of the major questions an organization will have and will provide some answers to those questions. In particular we will focus on the following questions:

1. Why does an enterprise require a data dictionary?

2. What are reasons for not implementing a data dictionary?

3. What are some of the criteria one should consider when choosing a data dictionary?

4. What factors should be considered when implementing a data dictionary?
5.1 The Need For A Data Dictionary

In answering the question an organization may have concerning its need for a data dictionary, we will focus on three needs that organizations have in the proper allocation of resources in data processing. 26 First, there is a need to control the use of data which is shared to ensure that it is correct, and that data is available for systems which identify new uses of data sharing.

Second, there is a need within data processing organizations to adopt a formal methodology for systems development to allow for development of complex systems within a small time period. Applications developed are no longer the simple payroll or accounts receivable systems, but are considerably more complex, addressing the entire needs of a functional area. As systems become more complex, the time necessary to develop them becomes longer. This is especially true for systems which use on-line data entry in real time or utilize a database management system. Advances in hardware and turnover in staff require the system payback to be relatively short. Systems were once evaluated on a cost benefit period of five years. Today the cost of hardware is decreasing so quickly, and technological advancements obsolete a system in such a

small time period, that paybacks are now calculated on a basis of one or two years.

Third, there is a need to control the system development process to ensure an optimal allocation of funds. Because of the large outlay for computer resources (such as hardware) and the large expenditures for maintenance of current systems, a small budget is available for the development of new systems.

**Evolution Of Data Use**

Traditionally, before systems were automated, users developed methodologies to handle day-to-day business problems. These methodologies were local in nature utilizing information that the user identified as necessary to do the task. The user prepared the reports and data files needed to maintain these systems. When data was shared, it was by means of a verbal or written communication which identified to both individuals where the data came from and where it went to. Depending on the nature of the data, correctness was either assumed or procedures were set up to ensure that correctness could be manually verified.

With the advent of early computer systems, this local

---

development and use of data continued. Data was held in local files for a single application and data sharing was rare. When data was shared, such as accounting data, that data was likely manually summarized on a report. The availability of data, among other factors, soon dictated that more and more data be passed, and a need was identified for computer systems which would automatically summarize and pass the data.

This sharing of data (e.g. a billing system passing accounts receivable information to an accounts receivable package), was done by means of intermediate transaction files. This mimicked the earlier sharing of data between two users where a verbal or written communication was passed from one to the other. Systems grew more complex (e.g. multiple sources of data input, many systems sharing information), and the control of multiple data files became increasingly unwieldy.

Database management systems eliminated much of the duplication of data, in particular multiple transaction files. However, since each user had been accustomed to a unique view of data, the use of a database management system identified the need for users to agree on standards for data definition, accessibility, and standard edit criteria.

Users also became aware of the intrinsic value of data in those instances when data was not available from another system. One had only to speak to the controller who, when trying to do his month end, realized that the accounts payable system
had not yet been run or that the system had run with errors to realize the value of data. Elaborate backup and recovery routines became necessary as data was required not only for the particular functional area but in other areas as well. It was soon realized that data was no longer the possession of any one user, but was a corporate resource to be shared by all. In the creation of databases, users were forced to make concessions in data description and storage format so that overall objectives concerning ease of use and accessibility were met.

The Corporate Data Resource

In managing this corporate resource, and maintaining standard definitions agreed to by users, a new position or function was created. The database administrator (DBA) was put in charge of capturing and controlling the definition of data. As the DBA had only a passing interest in the data itself, this function was perfectly placed for mediating the requirements of users and dictating the definition of the database. The DBA function provided the user with a feedback mechanism to ensure that optimal use of data was enforced and that standards were appropriate. In performing this task, the DBA required two software systems. The first was the database management system (DBMS) which would store the data controlled. The second was a system which would allow the DBA to
store and centrally control the standards for the data items identified. This system we have called the "data dictionary". The data dictionary system was not necessarily automated however it contained basic elements of data description.

System Development Methodologies

System development methodologies closely parallel the development of information systems. The earliest information systems developed were data driven and usually involved a single analyst-programmer in their development. The methodology that this individual used included the initial identification of reports that a user wanted, such as invoices or customer statements. Following this was the identification of data necessary to produce those reports and the specification of files to be used. Once the output had been defined and its storage was identified, the analyst usually identified the source documents or data input necessary to load the files.

Once all the data identification had been done, the analyst was ready to flow chart the validation, and update, extract, and report programs that were necessary to complete the system. Although this methodology may not have been formally documented, nevertheless, it was the methodology used intuitively by the analyst. Intermediate files and reports were identified after the fact, and as programming the system continued.
This methodology, which is still used, is prone to a number of serious problems including the inability of the user to review and accept the design before the programming has been completed. Problems with this design methodology become particularly acute when the output data must serve several users in different formats, and when the system is sufficiently large and complex, as is the case when the number of data elements becomes extreme and the functions that the system must perform are complex and interrelated. The methodology is also strained when a database management system is used to store data. In the design of the database storage format, the organization of data must be optimized, and one or more of the functions which use the data base may require a specific format, creating less than optimal designs for other functions. In making these decisions the detailed analysis of alternate data storage strategies is necessary.

Finally, the model suggests that where complex data or processing systems are developed, larger groups of people will be involved in the design. Communication between these groups of programmers and analysts is critical because of the inter-relation of the task. Therefore, there will not only be communication problems between analyst and user, but communication between analysts within the project will require a common base of definition and an easy accessible store of information.

As systems have become more complex, the methodologies
used to design and implement those systems have also become more refined. Many modern methodologies emphasize two concepts. The first is a functional analysis where events and functions of each user area are identified and documented. This proceeds in a top down manner where each individual function is decomposed until it can be understood, usually in terms of simple actions. User involvement in the functional analysis ensures that the system will model the user's actions. The second concept is that of model building. This involves the identification of data entities and the relationships between data entities used to perform each function. In model building, the analyst can review each data model and ensure that it is complete and provides enough information to support the function. At the end of this exercise the analyst is able to complete the conceptual design of the system; the functions and data models are described in user terms and are easily reviewed by the user. This documentation of systems development functions and data models becomes unwieldy if it is done manually. What is needed is a tool which can record and analyze user requirements, and decisions made in translating the user design into an implementation design.

Figure 5 is an illustration of the different information which must be collected by an analyst during his user design (labeled Conceptual View), and the information necessary to
### ABOUT TYPES OF DATA

**Design Requirements**
- Entities
- Relationships
- Attributes

**Design Decisions**
- Records
- Data Items
- Data Item Groups
- Sets
- Areas
- Files
- Devices
- Data Bases
- Schemas, etc.

### ABOUT THE USE OF DATA

**Access Requirements**

**Design Process**
- Events
- Functions
- Systems
- Programs
- Transactions
- Modules
- Computer Processes

---

**Figure 5**

Data For Analysis -- Stored by the Data Dictionary
install the automated system (labeled Implementation View).

In the conceptual view, the data types identified include entities, relationships, and attributes. These must be translated into an implementation view which includes data types such as data elements, records, files, databases, processes, transactions, modules, programs, and systems.

Using a data dictionary, an analyst can document each of these entities and functions and produce reports for review. Using cross-references, the analyst can determine the impact of a design change and ensure completeness of documentation. The data dictionary provides a method of monitoring development and documents the completed system.

Managing Change

The data dictionary can also assist management in controlling changes which occur in an information system. These changes can be due to three reasons. First, normal development and growth of a system may cause a change. A system is normally not completely implemented at one time. A phased approach is often used, where the most critical functions are addressed first, followed by other functions as necessary. As

---

28 British Computer Society, p. 5.

29 Ibid., p. 8.
new phases are automated, additional information may be captured which will require the modification of data storage, and programs.

Second, an enterprise itself may change due to reorganization, merger, or a change in management. Government may modify regulations which dictate the information to be stored and reported, or the enterprise may change its market requiring the capture and storage of new information.

Finally, change may occur due to the acquisition of new computer hardware or systems software. As hardware becomes faster and less expensive, and more powerful systems software is available, and/or support for old systems is withdrawn by vendors, new applications software may need to be implemented.

To control this change a tool is necessary which will allow the analyst to quickly and easily identify the location and extent of the change, and the effort necessary to make the change. The data dictionary provides a baseline which records the present and past systems definitions and gives some indication of the direction in which the system is moving. A data dictionary can provide information necessary to do a detailed analysis of the change. It can model a proposed system and be used to identify missing information. It can identify system interfaces and can be used to review the system in the overall organizational structure. The data dictionary system can also assist in costing the change by identifying the impact of mo-
edifications, the sequence of modifications, and the number of interfaces which might require change.

**Data Processing Environment**

In identifying the need for a data dictionary in an organization, opponents may argue that the data processing environment which exists may not be able to support a data dictionary, or may not even require one. A data dictionary can support the activities of a variety of data processing environments:

1. The conventional environment which uses sequential, or index sequential, data files.
2. An environment which is converting to a database management system.
3. One that has a single database management system.
4. A complex environment utilizing one or more database management systems, one or more central processing units, a distributed database, or distributed processing.

In a conventional environment (see Figure 6), application programs are designed to access a distinct set of files. In this environment the data dictionary is a free-standing set of files. The utilities which support it run in batch mode and extract data to support data validation and basic data
Figure 6
Data Dictionary in a Conventional Environment
definitions which are placed into system libraries. A majority of the data dictionaries in use are of this type. They provide an implementation view of data through file definitions, and a job library through file and task relationships. The system can also assist in preparing device layouts using cross-references between data files and their volumes.

During a conversion to a database management system, the data dictionary can perform two tasks (identified in Figure 7). In the first task, definition, the data dictionary can assist in locating the necessary data elements and groups. Data dictionary output can then either automatically generate the schema for the database management system or output sufficient information so that the schema can be manually prepared. If a database management system has already been designed, then there should be an interface between the database management system and the data dictionary to allow loading of data dictionary definitions. A utility should exist to allow application programs to be searched for program data definitions not contained in the data dictionary or the database management system. The output of such an analysis would be a discrepancy report which could be used in the second phase.

The second conversion task is to modify existing programs to interface with the database management system. This is done using the discrepancy report identified in the definition step. Changes are manually or automatically made to the program data definition to produce a new source program which is
Figure 7

Data Dictionary During a DBMS Installation
consistent with the requirements of the database management system. Using the data dictionary to document the processes that occur within a program, it can identify the necessary calls to retrieve data from the DBMS.

In supporting the single database management system, the data dictionary can either be a free-standing software package, which does not rely on a database management system for its operation, or it can be implemented using a database management system.

When implemented as part of a database management system, the data dictionary can either be implemented as a separate application or it can be integrated. An integrated DBMS-data dictionary would use data dictionary information as the directory to the DBMS. Although this implementation is more complex, it allows the user to access a single software system in order to identify and extract the information that he requires. 30

In a complex environment the organization may have one or more central processing units located centrally, or distributed. Database management systems and sequential and indexed files are used, and application systems extract information from the databases, or conventional files accessing one or more CPUs.

30 Sharman and Winterbottom, p.187.
In this environment, the time spent by data processing staff maintaining operational control over data and processes is a major task. New systems may have to interface with a number of different database management systems, each with its own protocols. Independence from physical storage, which is one of the aims of the database management system, once again becomes a problem. Programmers and analysts are once again forced to consider the storage methodology and data file design in preparing system definitions. A data dictionary can help by providing an interface between systems and multiple storage structures. In addition, the database can receive and process data retrieval requests even though they may reside on different machines and in different databases.  

The fact that data may be distributed over more than one database or machine, raises the problem of having a central or distributed data dictionary.  In a central data dictionary, all definitions are held centrally and every one must access the same data dictionary. Although this is the the most consistent design, there will be redundant data communications. The other extreme is to have definitions distributed to each


of the remote processing centres. In this case no data communications are necessary except to extract and update the data dictionaries in each of the locations. If data is located in a remote location, the data dictionary will execute the proper instructions to retrieve that data. The optimal design may be a hybrid of these two: some definitions held only at the local data dictionary site and some definitions held centrally. In deciding the location of information, a statistical analysis of retrieval requests would be necessary. A data dictionary can assist in this task.

One of the advantages of using a data dictionary in this environment is the standard interface format between application programs and the data resource. The data dictionary can store individual routines for interfacing to database management systems or conventional files.

In installing a data dictionary, other arguments are often raised. One is that a data dictionary is difficult to justify in terms of expenditure because of the difficulty in identifying tangible benefits that accrue out of it. We have shown that there are utilities which are available with modern data dictionaries to reduce the amount of programmer time necessary to complete programs (through data division generation, validation routine generation) and data base definitions. Although the calculation of such time saved would require some effort, it is a tangible benefit.
Another is that the level of effort necessary to gather information for the data dictionary; that is, to identify, gather, standardize and obtain consensus from users on data definitions, process definitions, and conceptual entity definitions, is a task that consumes a vast amount of resources. This was true for users of data dictionary packages which did not have utilities to extract data definitions from programs or data base directories. It is also true when users attempt to load all data, process and data interchange entities at one time. As we will discuss later, there are opportunities where the collection and input of data dictionary data is best done as the benefits are easily seen. These opportunities are frequently instances where the data dictionary can reduce the risk of failure (as when installing a large application system) because the task is better bounded and communication among the team members can be enhanced.

Still another is that the standardization of data definitions is not within the realm of a centralized corporate function, but that the data definition should remain the property of the user. The user should identify the name, validation standards, and other parameters of the definition as it relates to his own environment. The user might argue that no justification can be found for standardizing definitions which are local. This argument is also partially true. The effort of capturing all locally used data definitions simply to exercise control is not what we consider a wise use of the data
dictionary system. Standardization is justified when the sharing of data is enhanced. It is difficult for the local user to argue that there is no chance of the data ever being shared. It is a question of priorities. The DBA should insist that if the data definitions are to be used in the development of an application, then the definitions should be entered into a data dictionary, otherwise the effort cannot be cost justified.

The data dictionary is an overhead item. Many sources claim that when budget cuts are made the data dictionary and the database administrator functions will likely be the first to feel the pressure. Data processing resources will be diverted to those systems in which it is easy to show a positive payback in terms of staff reduction or tangible reduction in inventory or receivables balances. We have seen that data dictionary utilities can deliver benefits in the reduction of design and programming time, and the enhancement of user-analyst communication. These are tangible benefits, not easily quantified, and may be overlooked when evaluating effects on the bottom line.

5.2 Software Selection

Once management has made a commitment to the installation of a data dictionary, the implementation plan for the system must be prepared. Two key questions which will be asked are:
1. Should the data dictionary system be developed in-house or a commercially available package be purchased?

2. What types of features should the data dictionary contain?

In-house Development

In deciding whether to purchase a package or custom program the system internally, the analysis is similar to any other application system development evaluation. The decision is guided by the availability of software packages which contain the desired features at a reasonable cost. The cost of computing manpower and the complexity of even the most basic data dictionary, with all of its attendant interfaces and utilities, lean the decision heavily in favour of purchasing a software package. It is unlikely, considering the number of good software packages available, that it would be more costly to implement a standard package and add specialized interfaces to fulfill the unique needs of a particular enterprise's requirements, rather than writing the entire data dictionary system from first principles.
Selection Criteria

In selecting a suitable software package, a set of criteria is required on which candidate systems can be ranked. This ranking can be divided into features which are necessary and those which are desirable. For the purposes of discussion we have broken evaluation criteria into three categories:

1. Dictionary contents.
2. Dictionary utilities.

Dictionary Contents

The contents of a data dictionary system can be broken down into four categories. First, basic entity data such as its name, definition, where and how it is used, relationships to other data, labels, and key word designators. The basic data dictionary should also have facilities to record comments dealing with the entity derivation, and the individuals responsible for its input, update and usage. The system should also record dates of input and keep track of its last change.

Second, the data dictionary should record technical information. For data entities this would include their length, precision, value, range, and backup routines. For process entities it would include language, processor, special calcu-
lations, run time, and recovery routines. Similar attributes should be maintained for equipment entities such as processing units and storage devices.

The third type of information stored by a data dictionary is relationship information, including links between lower level and high level structures. For example, there should be a means of relating data elements to data files, or data elements to database structures. There should also be a facility for identifying sequence of elements within records and sequences of records within files. Similarly, for processing entities, there should be relationships which identify modules within programs and programs within systems. Links should also be maintained between data entities and processing entities, for example, to relate data elements to programs.

The fourth category is the realm of entities supported by the data dictionary. Some entities may be specific to a particular processing environment such as the ability to store IMS data structures to support an IMS database management system. In a CODASYL environment, it would be necessary to store schema or sub-schema information. The data dictionary system should record information about basic data entities such as elements, files, records, and basic processing entities such as transactions, modules, and systems. The data dictionary system should store information about operating system entities (run control language), transactions, data groups, reports, documents, and non-standard entities which are related
to the conceptual system definition. DATAMANAGER allows entity types such as process, sub-process, data store, data flow, and data structure which assist in supporting structured analysis.

**Dictionary Utilities**

The data dictionary system will only benefit the enterprise if it is supported by a comprehensive set of easy-to-use utilities. The basic set should include data input utilities, utilities to produce reports which document the state of the data dictionary, and utilities which support programming tasks.

Input utilities include programs to load bulk data into the data dictionary, and aids for extracting data entity definitions from programs and existing databases, or on-line data input to the data dictionary. The data dictionary requires an extensive validation system to ensure that only valid data is accepted. There should be facilities to automatically generate relationships between different entities, including references both ways (e.g. to identify programs when looking at element entities, and elements when looking at program entities). On-line input should be user-friendly, allowing the user to request help in completing all input. It should be organized so that both the needs of the casual and confident user are served. The data dictionary should allow
more than one status of entry so that incomplete input can be recorded. A comprehensive backup/restore facility, sufficient for the level of on-line data entry, including transaction logging, or checkpointing, should be supplied with the online interface.

Reporting utilities supported by the data dictionary should include lists for each type of entity, with the ability to select the output based on keywords or specific ranges of attribute values. Standard reports should allow for user exits that modify standard reports for user requirements. Cross-references should be provided. Systems should supply KWIC indexes on keywords and should be able to construct structure diagrams of entities. There should also be a facility to generate documentation for systems based on a standard list. This would allow the user to identify the types of reports necessary to document a program or a system. In this way many mundane system documentation tasks could be automated.

Program utilities can be broken down into two types. The first and most useful are those utilities which automatically generate program code based on information contained in the data dictionary. For example, the generation of data divisions into system-maintained copy libraries for subsequent copying into source programs. In specifying the data necessary to be included using the data dictionary, one can ensure that the definition of the copy library will be consistent
with other definitions of the same data. This relieves the
programmer and the analyst from the task of ensuring that data
definitions are consistent in the different file definitions
that they may appear. Other utilities which generate program
code to validate input data, call a database to retrieve infor-
mation, or handle input-output errors with standard error
messages would also fall under this category.

The second type of program utility is one which inter-
faces programs with the data dictionary. These utilities read
program data definitions and verify that the definitions are
consistent with information stored in the data dictionary. If
no information is stored in the data dictionary, proper input
transactions are generated to update the data dictionary. For
example, DATAMANAGER has an interface to the LIBRARIAN program
library system, which allows the data dictionary and the
source program library to synchronize their contents without
clerical updating. This ensures that the data dictionary and
the library management system have consistent views of the
program data definition.

Other program utilities which can be implemented are the
output of job control language for systems operation, and the
generation of routines for the automatic recovery/restart of
systems when they fail.

Other utilities which should be supported by the data
dictionary system include a query language, which allows on-
line query on an attribute, relationship and entity, or by keyword. The query language should be easy to use and should allow for hard copy reporting of the output. Utilities which may be supported by a data dictionary include the ability to interface with one or more database management systems, including production of database definitions using a logical design defined in the data dictionary system, and the generation of control language to automatically load the database. Likewise, an interface with the ability to read the database management system may also exist, so that definitions could be loaded into the data dictionary from the database management system directory. The data dictionary should also allow for cross-reference updates to be done automatically when data fields are maintained.

In such a system it is necessary to ensure that security is provided. The data dictionary should document the access to itself in a variety of ways. There should be a classification scheme identifying the access a user should have. It should be possible to specify, for example, that a user has update access to items bearing his function as responsibility and be able to enquire into related definitions. In addition, the system should automatically produce audit trails for all transactions processed against it. In setting security levels, it will be necessary for the database administrator to analyse the risk of fraudulent data use and to set security levels to ensure the level of security desired can be main-
Data Dictionary Environment

Environment criteria considered in evaluating software include programming language, file structures used, hardware supported and vendor support.

There are two types of data dictionary systems in existence. The first is a system which is meant to function in an environment which may or may not include a database management system. This type of system may have interface utilities for one or more database management systems. The second type of system is one which has been designed to support a particular DBMS. An example of this type of data dictionary is UCC-10 which supports IMS. Unless the enterprise can ensure that only a single database management system will be used, the free-standing data dictionary offers more flexibility. In a single DBMS environment, however, there is no need to support the extra complexity involved in a free-standing data dictionary system.

The data dictionary system can be written in one of three formats. First, in basic assembler language (BAL), second, in a high level language such as PL/1 or COBOL, and third as an

---

33 British Computer Society, pp. 11-4.
application using a DBMS. As for any software packages, the implementation language is usually evaluated on criteria such as maintainability and speed. Although BAL can be optimized to execute more quickly, its maintainability is fairly low.

A data dictionary based on a DBMS has both pro and con arguments. In favour of implementation of the data dictionary in a DBMS are the flexibility of design that the DBMS allows, the efficiencies in random processing which are already built into the DBMS access methods, the availability of on-line enquiry and update, the ease of producing ad hoc reports, and the ability to closely couple the data dictionary and database management systems.

Some of the arguments against implementing the data dictionary in DBMS are that the data dictionary will be less portable (such that if a hardware change is made, one would require hardware where the database management system operates), the requirement to purchase a particular database management system, and the development of expertise in the maintenance of the database management system as well as the data dictionary. If the data dictionary is offered by a vendor of a DBMS, the manufacturer will likely not offer interfaces to other database management systems. This is true for all systems we have

reviewed.

If the system is written in a high level programming language, investigations should be made to ensure that the design of the programs is well structured. This will indicate the ease of program correction due to errors in the code. Source code availability is also important as it allows for the diagnosis and correction of errors that are found during execution. This may be a drawback in that user modifications could make vendor update difficult, or void maintenance agreements.

The software should use standard file structures. This allows for program access of the data dictionary files, and is less likely to be made obsolete by modifications to the operating system. Finally, standard job control language statements should be included for both initial loading and production execution of the data dictionary.

Two major factors should be considered concerning the hardware on which the package executes. First, that the software executes on current or proposed hardware configurations. In evaluating this criteria, it is advised that the purchaser speak to users that are presently using the system on a configuration similar to that proposed. This will ensure that the system not only executes on this type of system, but it also allows for discussion to identify any problems or special mo-

35 Ewers, In Depth p. 5.
difications necessary to use the system in production. This should be done for current and proposed hardware configurations. The discussion should also include investigation of special hardware features used for speed or information density, such as microcode or cache memory. The system must also operate under present and proposed operating systems. If other users have converted from the present system to the proposed system, they may have developed conversion aids or have identified specific problems in the conversion which could aid in a more successful effort.

The second hardware criteria is the efficiency in which the software package uses the hardware resources. 36 Factors of efficient hardware use include core utilization, on-line storage necessary, system overhead, the size of the on-line monitor and executive program, and the amount of system time necessary to execute system maintenance transactions such as adds, changes, deletions and other standard system commands.

Vendor support may be more or less important depending on the size of the data processing department, and the extent to which support of the data dictionary package is done internally. 37 It may be that the quality of the software is an overriding factor. Clean software, which is structured and well

36 Adam, pp. 46-51.

37 Ewers, In Depth pp. 5-6.
documented, will allow a sophisticated data processing department to support the product. The purchased data dictionary system may be a basis for extensive internal modification. However, where the purchaser is looking for a good fit; one which satisfies all of the required functions, possesses a majority of the desirable functions and some of the functions he/she deems attractive, and does not plan to provide extensive inhouse maintenance, a high level of vendor support is necessary. Vendor support should include local support, or a hot line (problem line) which is available 24 hours a day. This indicates that basic support is present.

The purchaser should examine the documentation to ensure that the minimum documentation is present. This would include:

1. Overview of system functions. A high level flow chart of system operation and structure charts or other similar graphic exhibits documenting the processing of the system.

2. Description of the technical features of the system, including documentation of any specialized technical procedures which are incorporated into the programs.

3. Description of file structures, comprising file layouts, access methods, and any special data-related topics such
as logging of transactions, backup or recovery.

4. A summary of each program in the system, identifying inputs, outputs, a narrative of processing, any subroutines used, and a function chart identifying the flow of processing.

5. Data input instructions. This would embrace detailed descriptions of input data, documentation of input screens used, examples of forms necessary to support the system, and error messages and their resolution.

6. A description of reports. This should include examples of all reports which identify their function and contents, and a description of how the report has been produced.

The best way to verify vendor support is to discuss it with some of the users of the software. Some of the topics which should be covered include:

1. The reliability and ease of installation of fixes to the system. A system which has to be unloaded and reloaded for each fix to be installed, and which does not come with sufficient software to do such an operation, can be

---

38 Ewers, In Depth pp. 5-6; Adam, p. 51.
cumbersome to maintain.

2. The frequency of enhancements. This indicates an ongoing interest by the vendor to upgrade the software package. If these enhancements have been done based on user requests, then continued good relationships with the vendor are implied. The existence of a user group would also indicate good communications between the vendor and the users.

3. The availability and extensiveness of training. Users are best able to identify good and bad points concerning the vendor's training program. Extensive training during the installation and beyond, as new personnel are hired, should be available. Training after enhancements have been made to the software would also be desirable.

Finally, in evaluating software packages the purchaser may require a specific type of payment scheme. It should be possible to lease or purchase the software, maintenance fees and maintenance contracts should be inspected, and a review should be made of the possibility of cancellation and penalties which could be incurred.

\[39\] Adam, p. 51.
5.3 Implementation

The possibility of a successful implementation can be increased by choosing an implementation period which will show immediate benefits of a data dictionary system. There are four instances when a data dictionary system implementation can provide such benefits:

1. When the data processing operation has become complex, and the proposed application systems development projects proposed are not critical.

2. Before a major database management system conversion effort.

3. Before undertaking a large systems development project.

4. As part of an ongoing data processing standards project.

In a mature data processing environment, where the maintenance activity may occupy more than 50 percent of data processing staff time, the implementation of a data dictionary may have benefits. There are two objectives in undertaking this type of ongoing project:

\begin{itemize}
  \item Canning, p. 11; Ewers, In Depth p. 4.
  \item Nolan, p. 101; Canning p. 9.
\end{itemize}
1. To clean up the confusion in the data.

2. To aid users in identifying enhancements and systems problems.

In a mature environment, where operations oriented systems have been implemented to support standard business functions, the environment is likely to consist of a number of separate application packages each using numerous disc and tape files, unless the organization has used a comprehensive systems methodology which included systems planning and data modelling. The organization may also have acquired a database management system and some applications may use a DBMS as an access system. This will inevitably result in poor data sharing, and redundant, inconsistent data.

Once some order has been established in the specification of data and processing entities, the second objective can be pursued, that is, that the data dictionary aid users of this information. The benefit which can be cited here is the ability to look up approved data definitions and identify the manner in which they are accessed, where they are accessed, and the types of relationships between data and processing entities.

In pursuing this benefit, the database administrator should promote the desire to create descriptions which are readily and selectively retrievable through the use of data
descriptors and cross-reference reports. In providing these features the data dictionary will require security and privacy controls to ensure the data dictionary's integrity and to limit the access to authorized users.

A key point is that ongoing data dictionary maintenance is significant and the creation and staffing of a database administrator function, which has responsibility for the data dictionary, is paramount. "Clearly, an out-of-date data dictionary will be useless". 42

The second major event which can be used to promote the installation of a data dictionary system is the installation of a database management system. Before this is done, it is necessary to perform a thorough analysis of data entities and identify their relationships. In addition, the DBA will need to explore alternative database structures in developing an optimal database format.

In supporting this type of analysis, the data dictionary would require one or more database links. If the database management system has already been identified, a data dictionary which supports that DBMS is necessary. The data dictionary should be a metadata dictionary to allow the definition of data, processes and relationships at a high level, and

have facilities to allow for the automatic generation of the DBMS definition.

The problems of control and monitoring performance on large systems development projects provide the third opportunity for the use of a data dictionary system. The data dictionary system can reduce the system development effort by identifying the conceptual design and providing documentation for functions and data models at an early stage. A data dictionary can assist the designers in resolving implementation decisions, such as the system's scope or implementation database design. The data dictionary can also supply support information to allow alternate designs of data and programs to be generated.

Once the design has been completed, the data dictionary provides a baseline by which completion can be measured. In this way, development can be contained in that any changes to the implementation design can be analyzed in terms of the amount of additional programming or database definition work necessary.

The data dictionary can also reduce the amount of programming effort necessary by supplying copy libraries which contain data definitions for databases and data divisions of programs. In addition, the data dictionary can provide standard code which can be included in edit modules and provide a significant portion of the documentation necessary for a comp-
lete system.

Finally, the data dictionary can be used as a tool in assisting an ongoing standards development program. In a large organization, it is necessary to have explicit standards. These standards are defined and maintained by a standards function. A data dictionary can be one tool used by the standards officer. In supporting the standards officer, the data dictionary requires three bridges:

1. An auto load facility to allow data dictionary information to be extracted from programs, data file handlers, and database management systems. This bridge should be complete, identifying differences between standards documented in the data dictionary and actual implementation within programs. Exception reports and suggested changes to programs should be generated.

2. A utility which can supply definitions to programmers in forms of copy books, libraries, or direct definitions provided at execution time.

3. A utility to supply database management system definitions through a copy book library or a real time interface to the database management system.

With these bridges, the standards officer can ensure that the systems being developed and those already in operation
conform to standards defined by the organization.

In order to ensure that an enterprise has a successful implementation of a data dictionary, which not only achieves the objectives of loading new data definitions and information concerning other entities, but is also a viable long term function, there are five requirements:

1. Senior management must have a commitment to establishing control of data entities. In the long term, the data dictionary can provide information for utility programs which will reduce the development time for new systems, increase data sharing between organizational entities, and reduce the effort necessary to maintain existing systems. The achievement of these benefits can only be assured if the implementation of the data dictionary system is successful. This requires a senior management commitment.

2. The setup of an active database administrator role to identify and enforce standards, and to take responsibility for the integrity of the data dictionary. The database administrator is a function which is staffed by one or more individuals. The data dictionary should be the first application responsibility for the database.

\[^{43}\text{Adam, p. 47.}\]
administrator.

3. An honest evaluation of the cost and benefits should be done. In this evaluation it should be noted that the data dictionary is an additional level of management and responsibility. The effort in developing and maintaining a data dictionary will most certainly result in the DBA function and the data dictionary, as a cost centre, with little chance of direct payout. However, growth should be easier once the data dictionary is in place. For example, the installation of the database management system should be a more straightforward task.

4. Successful data dictionary implementation requires strong user involvement. The users of data in a non-data dictionary environment have the responsibility to themselves at least, of defining and maintaining the definition of data entities. User involvement will be necessary to reconcile naming, format and definition differences of data which is shared.

5. Finally, the organization as a whole requires a long-term commitment. The task of installing a data dictionary is both long and tedious and can never be said to be 100 percent complete as new data and new definitions are constantly being developed. In addition, the standards which a data dictionary imposes are restric-
tive. Management will probably want to rotate staff assignments within the database administration function and ensure that tedious tasks are shared by all.
CHAPTER 6

CONCLUSION

This thesis has examined the data dictionary from a number of viewpoints: structurally, where we have proposed a classification of data dictionary systems; organizationally, where we have reviewed the users and uses of the system; and commercially, where we have examined five packages in detail and two in overview. Finally, we have approached the dictionary from the pragmatic viewpoint, answering the questions: why do we need it?, how do we choose it?, and how do we successfully install it?

With all of the benefits of the data dictionary system, their acceptance is far from universal. CINCOM, in their literature, identify some possible reasons:

1. Installation is an odious task, where large amounts of data are entered.
2. Poor data entry facilities, usually batch, with positional parameters, and output which is presented in massive tabular reports.
3. No tangible benefit to end users.

---

We can add to these the following:

1. A lack of utilities which are useful in reducing the effort of systems design, and implementation.
2. Often no Data Base Administrator function is set up, leaving the responsibility for the dictionary unclear.
3. An over-ambitious implementation plan, often to load all of the enterprise's definitions at once.

The data dictionary systems reviewed in Chapter 4, especially those which have been modified and updated in the last year, show great advances in responding to these criticisms. Following the installation guidelines we presented in Chapter 5, a reduced scope of data input, one which will show benefits to an associated project (such as a DBMS installation), is suggested. The utilities which allow data to be extracted from existing programs also assists in solving the problems of attempting to do too much at once, without a clearly perceived benefit.

Several packages now support online data entry with user-friendly menus, prompting and tutorial features. Input is free form, relying on keywords rather than positional parameters.

Benefits to end users can be found in reduced application development and maintenance time, and assurance that a more completely defined system will result from the design effort.
These benefits are supported by utilities such as on-line query languages which allow the analyst to examine the conceptual design, identifying unanswered questions, and source code generation to assist the programmer in the development and maintenance of programs. Test data generators, data use maps and automated database generators also assist in system development and maintenance.

Data dictionaries have recently begun to evolve in a different direction. A good appreciation of the types of entities to be supported was well documented in ANSI and BCS publications. Packages available today reflect that design. What remains is the development of facilities to use this information completely. "Data Management Systems" (DMS) (not to be confused with Data Base Management Systems (DBMS)), are the data dictionary systems of tomorrow. DMS are composed of four main software systems:

1. A simplified data dictionary.
2. A screen design aid.
3. A database management system.
4. A simplified program language.

DMS were developed to assist end users in developing systems,

---

usually on mini or micro computers. Their data dictionaries contain little else except name, picture and output format. The screen design aid allows the developer to specify the data and its location on the screen. The programming language has basic input, modify, and delete logic to support the maintenance of data, a simple report generator to list and select data, again specifying the names of data stored in the data dictionary. Additional features, to subtotal and perhaps perform statistical functions, may also be available. The specifications are often translated into a procedural language such as COBOL, so that more complex procedures can be added if needed.

We believe that the data dictionary of the future will participate in this type of organized data processing system. Additional utilities and features are necessary to provide a system which addresses each user's needs. These functions have been described in Chapters 3 and 5.

Through the use of a data dictionary, the characteristics of management of an information source can be realized. First, an information system can be opportunity driven rather than technologically or functionally driven because development tradeoffs can be analyzed in the conceptual data framework. In this way information that is necessary will be deli-

---

46 Joseph Ferreira, "IRM: An Evolutionary Mosaic," Infosystems, 26, No. 10 (1979), 86.
vered before that which is easy to develop.

Secondly, the information resource can be user dominated rather than analyst dominated. Users can identify information necessary for the completion of their duties by analyzing the conceptual structure and accessing information they require. User analysts and technical analysts can then cooperate in developing the proper implementation structures necessary to prepare the information required.

Third, data can be dynamically rather than rigidly structured. Data independence can be achieved by using the data dictionary as an interface between programs and the database management system. The data dictionary can selectively retrieve and provide data in the format which the program requires, and allow for use of data with differing views. Some data independence is already provided by database management systems and the use of schemas and subschemas. However, a data dictionary can interface between a single program and several database management systems, or conventional file structures, in providing the data that a program requires.

Fourth, the system can be administratively constrained rather than technologically limited. By identifying the actual requirements of an enterprise, systems planners can identify those functions which need support. This is in marked contrast to systems which are implemented merely because of technological advance. Good examples of this type of
system are on-line banking systems which have traditionally shown that they have very little benefit in terms of labour savings and provide information at a much more timely rate than is necessary. The data dictionary can assist the enterprise in developing application systems which are result and process oriented. By identifying and documenting data and processing entities, the data dictionary can assist management in identifying those areas which require information, and providing that information, rather than allowing systems to be developed by identifying functions which are in crisis and attempting to solve that crisis.
Adam, Robert G. "Data Dictionaries: For That Mature Look." 
Data Management, 17, No. 10 (Oct. 1979), 46-51.


Chamberlain, Robert B. "DBMS and Data Dictionary: To Merge or Not?" Computerworld, 29 Oct. 1979, p. 36; p. 38.


Ferreira, Joseph. "IRM: An Evolutionary Mosaic." Infosystems, 26, No. 10 (1979), 86.


Kreitzer, Lawrence W. "Data Dictionaries - The Heart of IRM." Infosystems, 28, No. 2 (1981), 64; 66.


