# THE BRITISH COMPUTER SOCIETY DATA DICTIONARY SYSTEMS WORKING PARTY REPORT


### ACKNOWLEDGEMENTS

This report of the Data Dictionary Systems Working Party was prepared by the members whose names are listed below. In performing this work, these members were supported by their respective organizations. Inclusion in the following list, however, does not necessarily imply that either the individual or his organization endorses the report.

Geoffrey J. Baker, CACI Inc.—International; Mike Bibby, Standard Telephones and Cables; Tim J. Bourne, ICL; John Brereton, Kent Instruments Ltd.; David Clarke, British Rail; Harold Clarkson, British Steel, Rotherham; John R. Doughty, Honeywell Information Systems Ltd.; David Gradwell (chairman), CACI Inc.—International; Peter Hitchcock, IBM Peterlee; Leslie R. Jennison, Plessey; Ian MacDonald, CACI Inc.—International; Alan Macro, Management Systems and Programming Ltd.

Also, Ray Maskell, Arthur Anderson; Ken Meyer, British Gas; Clifford Morse, Scholl (UK) Ltd.; Robin H. Moses, MOD: Douglas Nicholls, Hoare Govett; Stephen J. Pinder, Davy Computing Ltd.; Geoff D. Plowman, MOD; Jeremy Shepherd, SCICON; Richard Struck, Management Systems and Programming Ltd.; J. Michael Sykes, ICI; Christine Warner, Open University; David J. White, Westminster City Council.

The group would like to acknowledge the contributions of those who have attended occasional meetings and made comments on the work. In particular: Pat Hall, BSRA; David Mace, Thames Polytechnic; Nigel Martin, IBM Peterlee; Peter Mason, London School of Economics; A.G. Matthews, CCA; Ray O'Connor, NCC; K. Robinson, Surrey County Council.


## PREFACE

This document is a report which sets out the provisional conclusions of the working party as of mid-1976. The purpose of publishing the report is two-fold: first, to give an account of work in progress which may be useful, especially to those implementing a data dictionary system; secondly, to stimulate discussion. We would welcome communications, from any interested person or organization, which should be addressed to:


The Secretary,
Data Dictionary Systems Working Party
c/o The British Computer Society
29, Portland Place
London, W1N 4HU
United Kingdom


The reader will find that the standpoint of the authors of one chapter may well differ from that of the authors of the next. Such differences arise in part from the working party's decision to produce a sort of sketch map of the subject while leaving certain areas for later consideration. In particular, the practical aspects of the subject have been dealt with, rather than the theoretical. (Subsequently, the report can be used to test the theory as it is developed.) A consequence of this emphasis on the practical has been that numerous underlying assumptions have not been made explicit, let alone reconciled.

As some of the important terms used throughout the report are affected by these implicit assumptions, the following observations are offered, not as definitions but by way of a guide:

A data dictionary is frequently used to provide information about data bases (or in preparation for conversion to a data base management system) and may itself be implemented on a data base management system. (This is not to exclude other cases, merely to select a central case.) Accordingly, the design of a data dictionary is subject to the same criteria as is that of any data base.

One such criterion is commonly called "data independence" and represents a family of requirements. Satisfaction of these permits and encourages the insulation of each component of a system, so that, for example, programs need no amendment when one type of disk is replaced by another. If effective insulation has been achieved, this change can be said to occur at the operational level and be transparent at the programming or implementation level.

The term "level" is used throughout the report to indicate a rough position on a scale which has the user of information at the top and the corresponding data, in the form of bits and bytes on disk, at the bottom. Different data base management systems aim at different heights on this scale and, for this, among other reasons, it is not yet clear how many levels may usefully be distinguished.

When the attention is focused on a particular level, a view of what is or should be happening can be formed. If necessary, this view can be recorded in the form of a model (preferably in the data dictionary). In the disk upgrade example cited above, two such models could be used to drive the unloading and loading of files.

The report stresses the need for a conceptual model "To record and analyze requirements independently of how they are going to be met." Clearly, the data dictionary needs to be developed, not only to record these requirements as they relate to the data of the enterprise, but also in accordance with the conceptual view formed of the data dictionary system itself.

This fundamental point (and its analogues; for example the implementation of the data dictionary depends upon a data base management system which may also be used in the implementation of other data bases) leads to the use of such phrases as "conceptual records."

Emphasis on one or another aspect of data dictionary systems has given rise to slight shifts in the usage of these

terms. These differences are more apparent than real, but nevertheless illustrate the need for a clear and consistent definition of what a data dictionary system should be. Such a definition is necessary to support this document, which is concerned largely with what a data dictionary system should do. The DDSWP is now working toward this objective.

# 1. INTRODUCTION

## *1.1 THE DATA DICTIONARY SYSTEMS WORKING PARTY*

### 1.1.1 History of the DDSWP

During 1974 the BCS Codasyl Data Base Administration Working Group (DBAWG) considered which data base administrator facilities it would be able to develop specifications for, in the short term. One of the areas that DBAWG did not intend to cover was the specification of data dictionary facilities. They wrote to the British Computer Society, suggesting that a study group be set up. Efforts were made to initiate such a group and the Data Dictionary Systems Working Party (DDSWP) has met since January 1975. Meetings have been held every month to achieve the work program described.

### 1.1.2 Aims of the DDSWP

The initial meetings established that the working party's prime objective would be "to produce a report on the need for, and the facilities which should be provided by, data dictionary systems and related data base design and operational aids."

### 1.1.3 Work Program

The following program of work was initiated:

- Consider the facilities provided by current data dictionary systems and related design aids.
- Identify the data recording and analysis needs of those involved in the design of information systems.
- Specify the requirements for aids to data base design, maintenance and operation and consider which requirements are cost effective to automate.
- Report the findings of the working party and possibly recommend to the Codasyl Data Definition Language Committee, among others, that certain basic design implementation and operational aids to be provided as an integral part of any data base management system (DBMS).

### 1.1.4 Results of the Work Program

The DDSWP has studied many commercially available and in-house data dictionary systems. It has reviewed the requirements of users for computer aids for systems analysis, data base design and operation.

The scope of these existing systems and future requirements ranges from simple data directories describing existing files to full-blown systems analysis, design, implementation and operational tools which could only be termed "systems encyclopedias."

From this review and analysis the DDSWP has produced this report. It contains a justification of the need for data dictionary systems, a description of their place in relation to the DBMS and a specification of the facilities required. These facilities would provide:

- For the definition and description of data structures and their usage found in the enterprise in terms of a "conceptual" model. This definition would include all the associated data about data required during design and implementation. It would describe the way the conceptual data structure maps to existing manual or computer files and its use by application systems.
- Incremental additions and modifications to be made to the definitions and necessary documentation produced.
- For the implementation of parts of the conceptual data structure and associated functions by mapping on to one or more DBMSs or file handling systems and application systems.
- For the documentation and control of the implementation of systems and the restructuring or reorganization from one data base version to another.

### 1.1.5 Intended Audience for the Report

The intended audience for the report is considered to be:

- Those interested in the management and control of their data resource.
- Those involved in designing and implementing data processing systems.
- Suppliers of data dictionary systems.
- DBMS and other software suppliers.
- DBMS specification or standards committees for DBMS and other software, for example DDLC, ISO, ANSI/X3/ subcommittees.

## *1.2 AN INTRODUCTION TO DATA DICTIONARY CONCEPTS*

### 1.2.1 What Is a Data Dictionary System?

A data dictionary system is a tool for recording and processing information about the structure and usage of data. Usually it will be a computerized system.

The range of information which can be held in a data dictionary system is very large. The simplest system may only hold sufficient information to document, say, COBOL file structures. The more complex may hold design requirements, designed data base structures, possible future structures, operational information and extensive details of access to data, including TP network specifications. Such complex systems, initially built around data structure recording facilities but greatly developed, are really the data processing department's own data base.

It is not useful to draw a sharp distinction between data dictionary facilities and those provided by other software tools. Indeed, some facilities may be best implemented as extensions to a DBMS. On the other hand, more DBMS independence will be gained if the data dictionary system is to stand alone.

### 1.2.2 Use of a Data Dictionary System

A DDS with a wide enough range of facilities could be used throughout the complete analysis, design and implementation cycle. It could be used during each of the following stages:

- Data analysis, to determine the fundamental structure of the data of the enterprise.
- Functional analysis, to determine the way in which events and functions use data.
- Data base or conventional file design.
- Storage structure design, where this is a further refinement of the initial data base or file design.
- Operational running of application systems.
- Collection and evaluation of performance statistics.
- Data base tuning to improve performance.
- Application maintenance and data base restructuring.

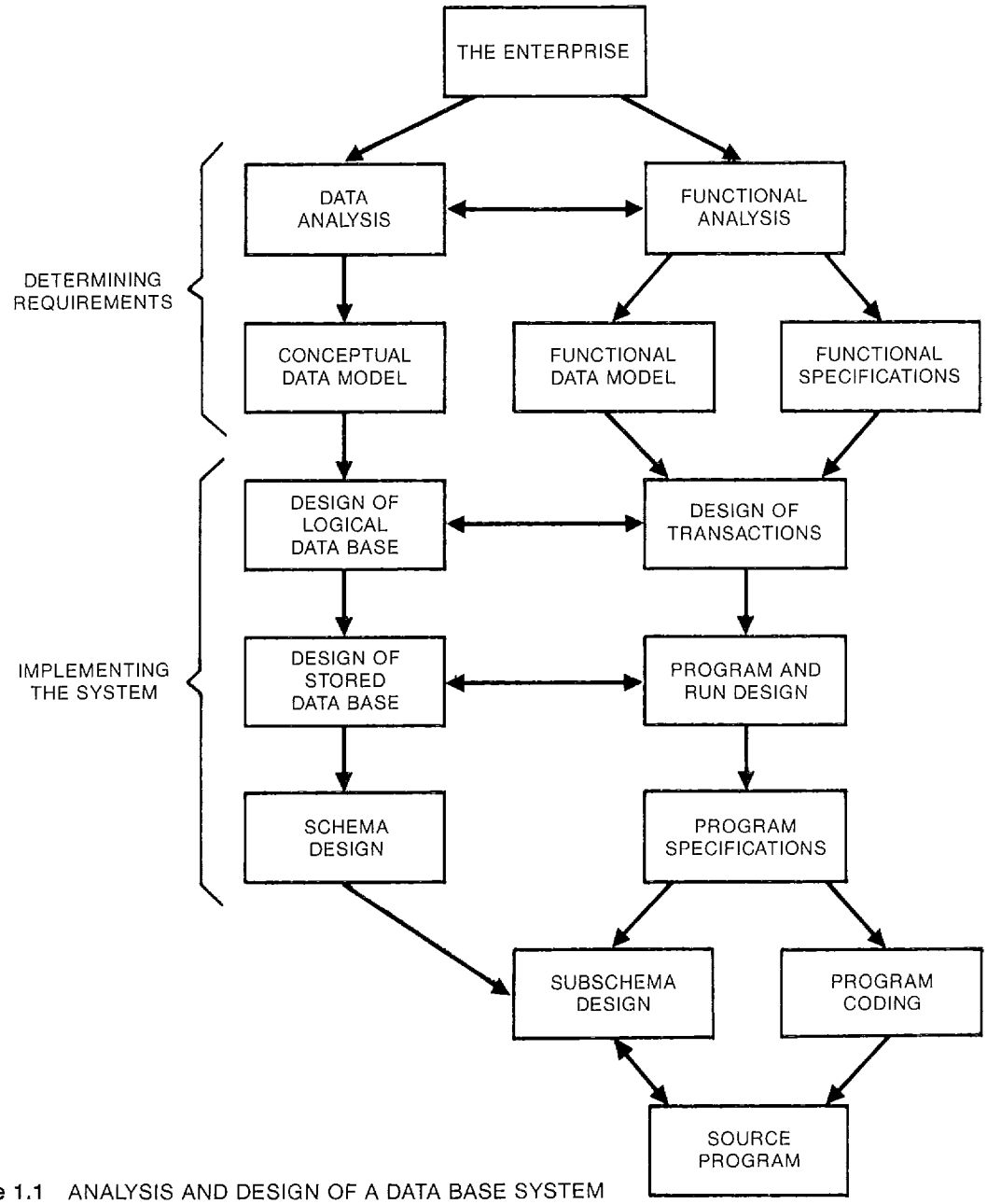


**Figure 1.1** ANALYSIS AND DESIGN OF A DATA BASE SYSTEM

### 1.2.3 Information in a Data Dictionary System

Like any other data processing system, a data dictionary system must provide the facilities needed by its users. The facilities required will depend upon the methods used during systems analysis, design and implementation. Growing awareness of the need to control the data resource and the advent of data base systems have led to changes in systems analysis techniques. It is not the intent of this report to discuss different systems analysis techniques, but rather to concentrate on the information requirements of those involved in data and data base administration. For illustrative purposes Figure 1.1 shows a typical analysis and implementation sequence.

The DDS should provide two sets of facilities:

- To record and analyze requirements independently of how they are going to be met.
- To record design decisions in terms of the data base or file structures implemented and the programs that access them.

The statement of data requirements independent of implementation considerations is termed by the DDSWP the "conceptual data model."

The specification of the record and file structures and the schemas (or whatever the structure definitions are called by a particular DBMS), has been termed the "implementation data structure." This is because a part of the

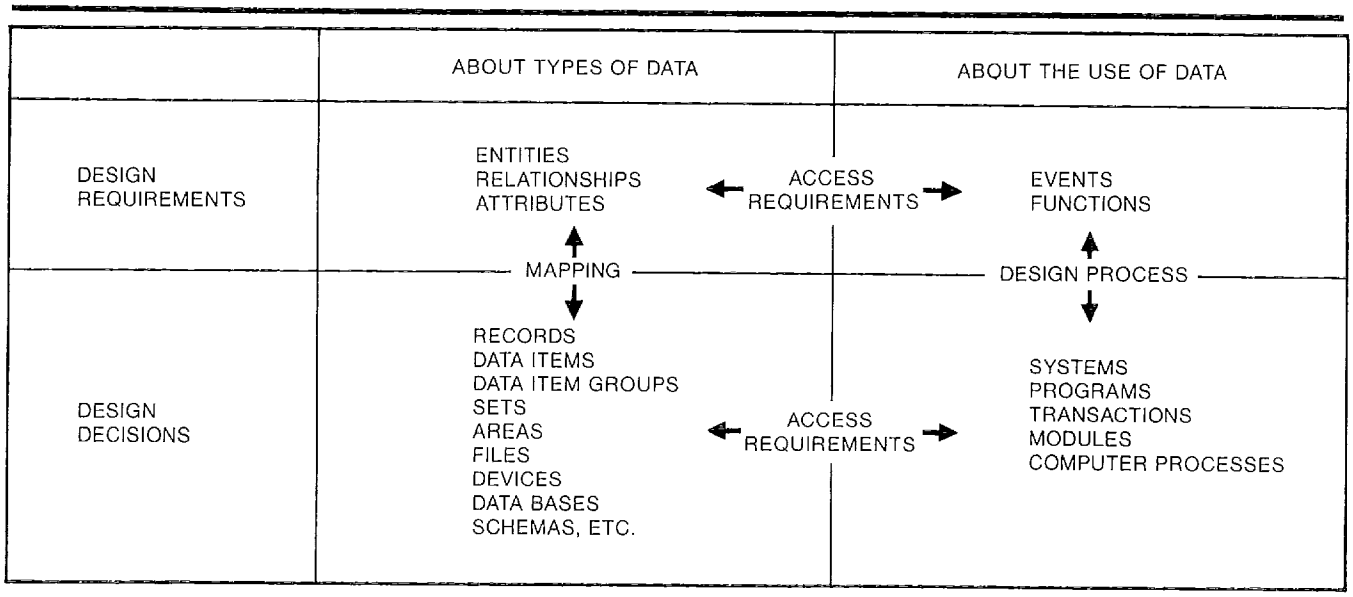


**Figure 1.2** THE INFORMATION IN A DATA DICTIONARY

enterprise's conceptual data model is implemented using a particular DBMS.

For both the conceptual data model and the implemented data structure, we need to record data usage as well as data structure. Hence four areas of information can be identified. Some of the types of object, physical or abstract, about which information needs to be recorded in a DDS are illustrated in Figure 1.2. The next two sections discuss the conceptual data model and the implementation data structure in more detail.

### 1.2.4 The Conceptual Data Model

In the previous section, the conceptual data model was defined as a description of the data requirements of an enterprise. It is a model of the data and its usage by the enterprise as it really is, before the problems of implementation constrain the data structures desired. The components of the conceptual data model include:

- Entities, that is anything or concept that is of interest to the enterprise.
- Relationships, that is associations between two or more entities of interest to the enterprise.
- Attributes, that is those properties of an entity in its own right which are not due to relationships with other entities.

To the conceptual data model should be added a description of the events that happen in the enterprise and the functions that must be carried out as a result. The way in which each function makes use of each entity and its attributes and relationships needs to be recorded. This forms a functional description of access requirements independent of any DBMS or file handler.

Chapter 5 describes the data dictionary facilities needed to record the conceptual data model.

### 1.2.5 The Implementation Data Structure

The implementation data structure is a description of the data as it is seen by the file handling system or the DBMS. Such things as descriptions of records, data items, sets, areas, files, data bases, schemas and programs form part of this description of data and its usage. It is difficult to generalize about the components of the implementation data structure because of the great diversity of DBMSs and file handling systems. Some current data dictionary systems cater for more than one type of implementation. Data structures tend to have some features in common, such as data item definition, while other features are DBMS specific.

### 1.2.6 The Mapping Between the Conceptual Data Model and the Implementation.

The data dictionary system should relate definitions of, for example, records and items to the entities they de-

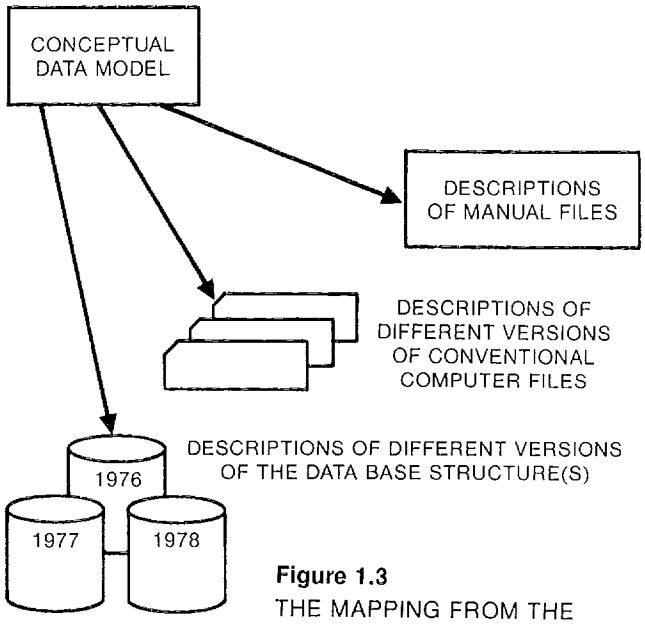


**Figure 1.3**
THE MAPPING FROM THE CONCEPTUAL DATA MODEL TO THE IMPLEMENTED DATA STRUCTURES

scribe. The process of designing efficient data bases causes many deviations from the ideal implemented data structure. Recording the mapping documents these design decisions and helps to clarify what has been decided.

The reader should also be aware of the possibility that information about a particular entity type (person, for example), or even a particular entity occurrence (John A. Smith, for example), may be stored under several file handling or DBMS systems. For example, payroll data may be on a tape, while skills data may be held under IDMS and activity scheduling data under IMS.

Data structures evolve. There should be one conceptual data model for the enterprise, which will contain new definitions and updated versions of old ones. However, there will be several versions of each data base structure since these are less flexible (at present) and operationally must have clearly defined changeover times. Figure 1.3 expresses this. During a conversion, parallel runs will maintain identical information on two different files or data bases.

### 1.2.7 Levels of Data Description in an Implementation

The distionctions between the conceptual, implementation and operational levels are clear in principle, but difficult to define in relation to much of the software that is currently available. In particular, the boundary between the implementation and operational levels varies according to the DBMS or file handling system. Thus it may be necessary for one data dictionary system to document several data structures:

- Known (or indeed not known) to one or more DBMSs or file handlers.
- Of varying levels of storage structure independence.

Thus, it is necessary to establish a framework which relates the data model to implemented data structures with different levels of data independence.

Before the advent of complex file handlers, programs were tied very closely to the storage structure they accessed. Program code was written to describe access to bits and blocks. A program had no data structure independence at all. This situation could be viewed as shown in Figure 1.4.

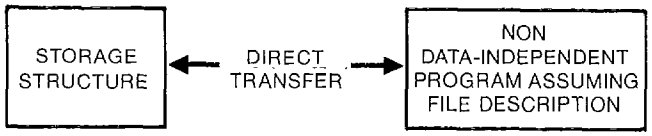


**Figure 1.4** THE DIRECT MAPPING FROM PROGRAM TO STORAGE GIVING NO DATA STRUCTURE INDEPENDENCE

The insertion of a level of mapping between the program and physical storage gives varying degrees of storage structure independence depending upon the power of the mappings. This can be represented by Figure 1.5.

Conventional COBOL file structures provide an example of this type of system. However, the power of the mapping between the logical data structure and the storage structure is very limited. Also, only the data dictionary system contains a complete description of all the files. The file handler is unaware of the whole.

When a DBMS is used, the logical data structure is

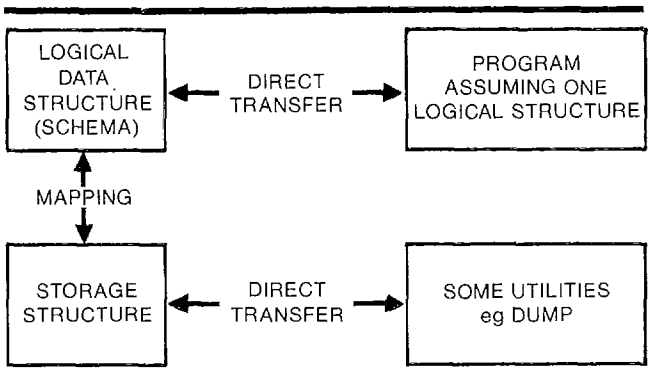


**Figure 1.5** THE INSERTION OF AN INTERFACE GIVES SOME DEGREE OF STORAGE STRUCTURE INDEPENDENCE

centralized and shared. More powerful mappings are available between the logical data structure (schema) and the storage structure.

Nevertheless, most DBMSs do not allow any mapping between the schema and the program.

Some systems are now becoming available that allow the program to have its own view of data (its own subschema). Mappings are provided between the central logical data structure and the program. This gives a further level of flexibility, illustrated in Figure 1.6.

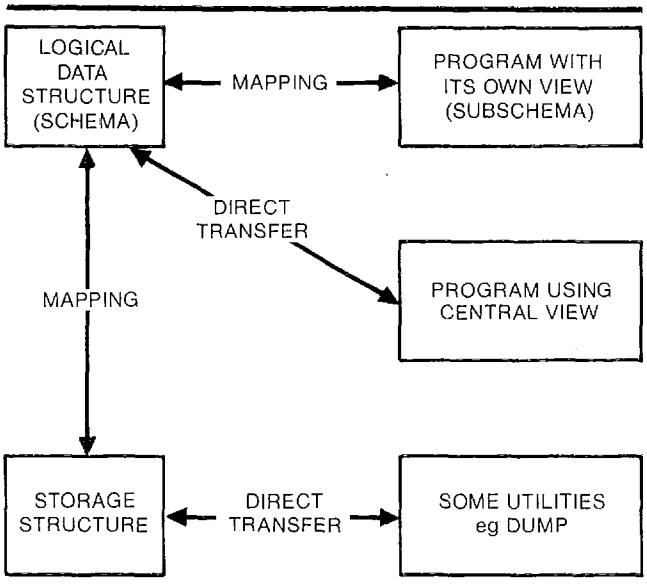


**Figure 1.6** THE SHARING OF A CENTRALIZED LOGICAL DATA STRUCTURE

The proposals of the ANSI/X3/SPARC DBMS Study Group put forward a conceptual schema which is totally storage structure independent. In this case no mapping between the conceptual schema and the data model should be necessary. Only definition of the subset of the data model to be implemented in the data base is required.

Figure 1.7 shows the relationships between the different implementation strategies that have been discussed and the conceptual data model.

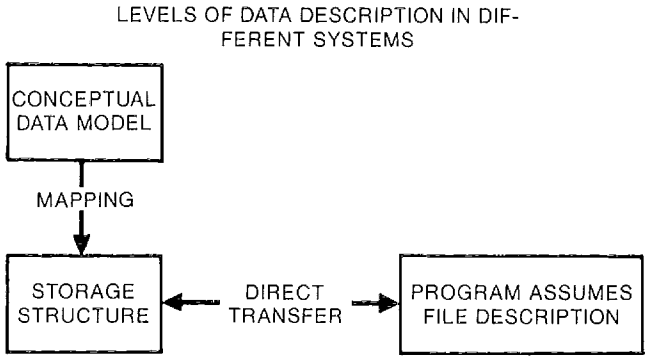


**Figure 1.7 (i)** PROGRAM WITH NO DATA INDEPENDENCE

The structure and purpose of the DDSWP conceptual data model is compared with the conceptual schema of the ANSI Study Group in section 5.5 of this report.

## 2.0 THE NEED FOR A DATA DICTIONARY SYSTEM

### *2.1 INTRODUCTION*

The purpose of this chapter is to show why a data dictionary system is needed by an enterprise and to discuss this need in the context of current trends in the management and use of data and information systems. Chapter 1 described a data dictionary system as a tool for recording and processing information about the usage of data. It is our belief that there is a growing need for formal documentation of data about data, brought about by a recognition that the developing information systems in enterprises deserve proper management.

Management is rapidly becoming aware that the data of an enterprise is a valuable and important corporate resource. As such, it should be managed and controlled with as much care as are other, long recognized resources, such as stock, cash, equipment, buildings and personnel.

Data processing management is constantly required to

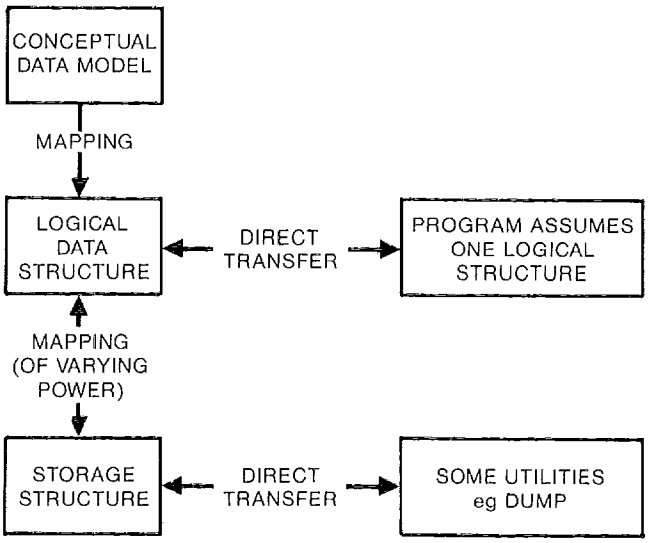


**Figure 1.7 (ii)** COBOL OR DBMS WITH NO SUBSCHEMA

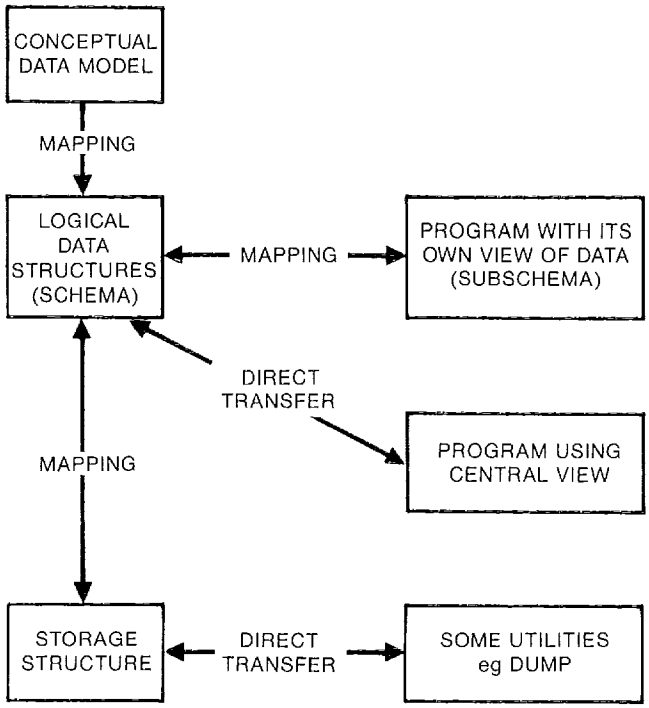


**Figure 1.7 (iii)** CODASYL WITH SUBSCHEMA

cope with quantitive and qualitative changes in the systems they serve. These changes are brought about by refinement of existing systems, addition of new applications and developments in technology. The management must ensure an orderly progression through these changes and must be able to control the effect on existing systems.

In each of these cases much of the management burden falls on data administration. This chapter will show that much of the burden may be transferred instead to a data dictionary system, thereby increasing data administration's ability to manage the data resource well. The chapter

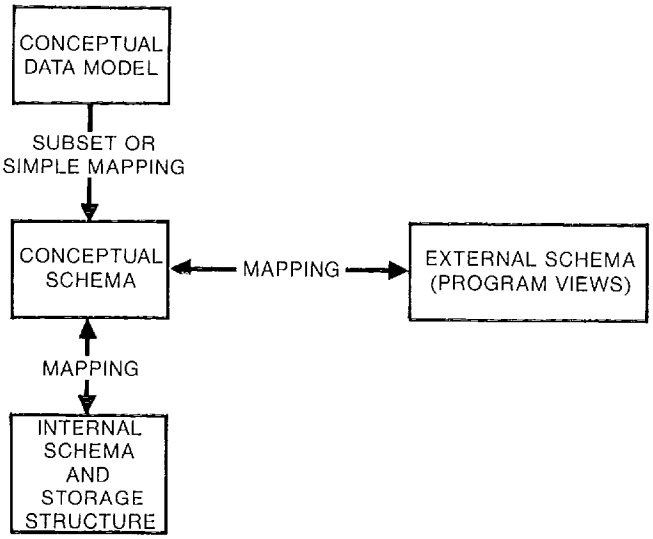


**Figure 1.7 (iv)** ANSI/SPARC ARCHITECTURE

will also show that the data dictionary system is a valuable servant of other users in the enterprise and will summarize major areas that benefit from its introduction.

## *2.2 MANAGEMENT OF THE DATA RESOURCE*

### 2.2.1 The Data Resource

An enterprise's data exists in its files, on forms passing between individuals and departments, in communications to and from other enterprises and as the result of computations it may perform on other data. The orderly flow and presentation of this data to interested parties forms the basis of the enterprise's information system. This system enables the enterprise to function in a coordinated manner and to interact with its environment.

For the enterprise to be effective it must optimize the use of its data resources to ensure that the information system is served as well as possible. Management of this process is clearly critical and for it to be successful there will need to be:

- Knowledge of what data exists and how it is used.
- Control of modifications to existing data or processes using data.
- Control over plans for new uses of data and over the acquisition of new types of data.

### 2.2.2 Management of the Data

In most enterprises, management itself does not yet have control over the data resource because such authority as does exist is scattered among the individual users of the data; there is no corporate view of the data resource.

To gain control over the data resource, management must collect, order and store information about the data. It can then begin to impose controls over the use of the data resource and develop standards for the definition of data.

In a simple computer environment, each application is a self-contained set of programs maintaining its own files, no file being used in more than one application. Information about the data may be gathered for use by new programmers or analysts simply by documenting such things as record layouts and possible code values. Control over this process is then achieved by insisting on consistent working to proven standards.

In many enterprises, the position is not so simple. There may be a large number of applications and their requirements for data may overlap a great deal. Manual documentation systems tend to be inadequate in these situations because of the effort needed to maintain the volume of information and the difficulties of retrieving relevant information. When a large proportion of the data in an installation is used by more than one application (or where DBMS software is used), it becomes necessary to establish central control of data definition and use. This is accomplished by a function known as data administration. The data administrator must ensure that all applications are using the data correctly and that it is being properly maintained. He must advise on new applications using existing data and on extensions to the data base. He may have an operational role, specifying the storage media, backup requirements and availability criteria. He must provide the information about data structures required by analysts and programmers.

To perform his function, the data administrator must have extensive central documentation of data structures within the enterprise. A data dictionary system will provide facilities to carry out this documentation to the level of detail required.

## *2.3 SYSTEM CONTROL*

### 2.3.1 Information Systems

Several levels of complexity in information systems can be recognized:

1. Very small systems where the information is available to every element and control is entirely informal.
2. Simple systems with complete connectivity between all elements but a simple, formal control over the information flow.
3. Systems large enough to be departmentalized with selective connectivity maintained by a highly formalized information flow.
4. Large systems where the information flow is such that a control function must be established to manage it properly.
5. Large systems with a complex structure requiring automation of many of the information system's control functions to deal with the variety of information management tasks now tackled.
6. Very large systems which are so complex that additional aids are required to predict and optimize the information system's performance.

### 2.3.2 Control of Information Systems

As an enterprise's information system develops in size and complexity, standards alone fail to ensure an optimum use of the data resource. This is due to the fact that as the system develops, its data becomes available to more and more functional areas which may not inform each other fully about the uses to which they put the data. In fact the amount of information about the data itself becomes large and unwieldy. This creates a need for formal methods of handling the information and a need to establish a data administration function to control them. The formal methods are embodied in the data dictionary system which provides a means for effectively recording and handling the new information.

The data dictionary system is mainly applicable at levels 4, 5 and 6 above. At level 4 it may be possible to maintain control with a manual data dictionary system; level 5 depends on a fully automated approach; level 6 requires a full battery of automated facilities including impact analysis and simulation.

### 2.3.3 Control of Change

Information systems are rarely static in form and may change for a variety of reasons:

- Normal development and growth of the enterprise leads to a progressive development of the information system through levels of complexity such as those described at 2.3.1.
- Changes in enterprise structure brought about by reorganization or merger may produce an abrupt change in the form or complexity of the information system.
- Changes in the software environment, for example by the introduction of a DBMS, may alter the form and complexity of the information system.

The effectiveness of the information system will be impaired if it is not possible to incorporate the results of these changes quickly and smoothly. Management, therefore, is in need of some device to help achieve the transi-

tion. A data dictionary system is ideally suited to this need since it can provide the following forms of support:

- It records all details of the current system and therefore provides a baseline from which to plan changes.
- It can record details of all proposed changes and can report on any inconsistencies or gaps in these proposals.
- It can report on some of the major cost factors involved in implementing specific changes and from such information a cost-effective implementation sequence can be devised.

*2.4 RELATIONS WITH USERS*

As an enterprise develops, it typically moves toward an increase in functional specialization. During this process the information system becomes more selective in its dissemination of information, supplying to each area only what is necessary to support it. Two major problems are created by these changes:

1. Users lose sight of the overall structure of the enterprise, become unaware of the significance of their part in it and fail to appreciate the effect they have on it.
2. It becomes difficult to insure that the information disseminated is the best for the purpose and is as complete as possible.

A data dictionary system can serve the enterprise's need to deal with these problems.

In the first case it serves as a central easily accessible source of information on any aspect of the enterprise's information system that the user may feel is relevant to him. It therefore increases his awareness of the system and boosts his confidence in it. The data dictionary system therefore performs a valuable public relations role.

In the second case it insures that all relevant information is provided to each functional area and automatically informs areas of changes that may affect them. It will enforce proper standards on the use of all data and insures that its quality is high. The data dictionary system therefore helps to optimize each of the functions performed in the enterprise.

*2.5 CONCLUSION*

It is clear from the foregoing discussion that there is a need for sound management of the data resource. Further, it is evident that this is unlikely to be achieved unless the data administrator is supported by a data dictionary system—the obvious device to complement his endeavors. There can be little doubt, therefore, that the majority of enterprises do need a data dictionary system and that it will help to insure the best use of the corporate data.

## 3.0 THE USE OF A DATA DICTIONARY SYSTEM

*3.1 INTRODUCTION*

This chapter outlines some of the ways in which a data dictionary system may be used.

In most cases, members of the DP department, rather than users, will require access to the data dictionary. However, users may wish to consult the dictionary directly to find out what sorts of data are available relating to entities of interest to them, such as orders, stock items and so on. Further, automatic use of the dictionary by software and utilities may be extensive (see Chapter 4).

What follows is concerned with the direct use of the data dictionary by DP staff. (The organization of the enterprise may be such as to require the inclusion of O & M staff in this category.)

*3.2 THE DP ENVIRONMENT*

There are several widely recognized factors which are characteristic of the DP environment; the use of the data dictionary is particularly connected with the following:

1. Iteration, since most "new" systems are designed to satisfy an "old" requirement more effectively.
2. Iteration in another form, since different aspects of a system are considered in turn, until the proposed design appears to fit all (or most) of the criteria.
3. Maintenance, which frequently involves some form of investigation as a prerequisite before a change may safely be implemented.
4. Reliance on human memory, especially for information about interdependencies.
5. Awareness that one or more of these factors is getting out of hand, so as to soak up resources or hold up progress; for example, as extreme cases of the operation of the factors cited:
   a. The third conversion in five years (say: disks, mainframe, programming language).
   b. The system that is being developed just fast enough to remain at the stage "70 percent complete."
   c. The system or program that has been so much patched that the next amendment will probably break it.
   d. The two people who developed the system four years ago both left last month. The new system which was linked to it just before they left has produced nonsensical results.

*3.3 DOCUMENTATION*

Clearly, good documentation helps to solve and even to prevent some of the problems such as those outlined above. However, most documentation is concerned either with projects or with systems. Project documentation should help in case 5b and system documentation in all cases. Nevertheless, after completion, project documentation is only of historical interest. System documentation, on the other hand, should be maintained throughout the life of the system, though this activity is often neglected.

Documentation is also required which can deal with the use of a program in several systems, with the use of a data item by several programs and, in general, provide cross-references between the various components of the information system.

Manual documentation of this kind is possible but unlikely to be adequate, as the relationships involved are complex and tend to change fairly frequently.

A computer system, especially one based on a DBMS, has two main advantages:

- Accessibility, both for reference and for maintenance.
- Scope, since the system required to produce the minimum benefits needs little added to record models at all levels (conceptual, implementation and operational) and to provide a variety of cross-referencing facilities.

*3.4 ANALYSIS*

The analyst should record (or verify) the basic data about the existing system in the data dictionary. The current implementation model so constructed will be used for reference until the new system is established.

If, as is desirable, a conceptual model is maintained, this, too, should be brought up to date.

The implementation of a data dictionary system is unlikely to change the work of the analyst (or of the designer), but it is likely to change the methods employed. In particular, the flow of data across functions can be recorded as a byproduct. A data dictionary is a central facility not only for the recording of data about the data and functions of the system under examination, but also for the capture of data about neighboring systems. Without such a facility, it is seldom practicable to record incidental discoveries in a permanent and accessible form.

On the other hand, the standardization of methods of investigation and recording is encouraged, since a simple inquiry of the data dictionary can show what metadata (that is, data about data and functions) remains to be collected. Further, conflicting usages can be identified and resolved and redundant data removed from a data base or procedures implemented to insure consistent update.

The data dictionary would include metadata of several different kinds, relating, for example, to:

- Data and data structures (such as files, data bases).
- Processes and processing structures (such as programs, systems).
- Relationships between data and processing (such as transactions).
- User functions and responsibilities.
- Volumes and frequencies.
- Security and privacy constraints.

Frequently, the difficulties encountered in maintaining existing systems arise from the domino effect set off by a small change. The use of a data dictionary enables the analyst to predict the impact of a proposed change and define the measures necessary to prevent unwanted side-effects.

The distribution of responsibilities in DP departments differs greatly from enterprise to enterprise, but failures of communication (whether between function and function or across time) are common, whatever the organization of the DP department. If analysis can be described briefly as the statement of a problem and design as the solution, the role of the data dictionary can be seen in these terms: It is a device for collecting all the facts necessary for the clear and complete statement of the problem and for providing data to test the solution.

In most cases the statement of the problem will cross the boundaries between the conceptual, implementation and operational levels, but preserving these distinctions will lead to a flexible solution.

The description at the beginning of this section is oriented toward the work of the business systems analyst, but it applies similarly to the technical systems analyst or data base administrator considering, for example, file or data base reorganization (centered on the operational level) or to the data administrator reviewing, such as the organization of the enterprise to build a functional model (mapping from the implementation to the conceptual level).

### *3.5 DESIGN*

The designer should have available the conceptual model to take a global view of the enterprise and a view of the new system within it. The changes or improvements required by the user or rendered possible by more powerful hardware may lead to little change, if any, to the conceptual model. Without such a model, however, confidence in the future of the system may be misplaced.

The overall design of the system should be carried out at this level. The result of this stage of development is a local conceptual model which can be used, for example, to:

- Verify that the proposed design will satisfy the user's requirements.
- Exhibit the links with related systems.
- Plan and control the project.
- Uncover the need, if any, to alter any other existing systems before proceeding.

The justification of the project can be reviewed at this stage. Even if it is decided to postpone or abandon the project, the work already done will have improved both the documentation and the understanding of the information system. Moreover, it is possible that the design could prove a useful starting point at a later date.

If work is to continue on the project, the necessary tasks can to some extent be carried out separately, using the conceptual model as a framework. These tasks include:

- The refinement of the conceptual model to incorporate the details of the processing and data structures (without regard to the implementation constraints).
- The construction of an implementation model dealing, for example, with the conventional or data base file structures.
- Analysis of the impact on, for example, existing data bases which will need to contain additional data.
- Examination of the uses of common data by neighboring systems. This is particularly important when converting to a DBMS, as existing systems may make use of similar, but not identical, data.

A feature of this stage is flexibility, since responsibilities may be allocated according to the resources available, and the sequence of tasks need not necessarily be predetermined. This flexibility is fundamental to a data dictionary system, since each individual responsible for a particular aspect of analysis or design can record his findings or decisions in the appropriate places in the dictionary; further, he can reference the other items of metadata of interest to him.

The data dictionary, in fact, can provide not only a guide to the system itself, but also to the progress of the project and to the documentation. Moreover, as it should be much easier to update than any form of manual cross-referencing system, it is likely to be reliable.

### *3.6 PROGRAMMING*

The methods used for the specification, writing and testing of programs are, naturally, more dependent upon hardware and supporting software (especially upon the DBMS if one is installed) than are the methods used in analysis and design.

Nevertheless, the use of the data dictionary should be extended if techniques such as structured programming are employed. The specification of programs can be recorded in as much detail as is necessary and the entry for each module (or smaller unit) linked to the corresponding entries for data.

If the distinction between the conceptual and implementation levels is observed in both the data model/ implementation data structure design and the function/ program design, then each will increase the value of the other. The data model and functional description may be developed in

parallel, insuring consistency. The implementation data structure and programs may then be designed, taking into account implementation constraints, with reference to the conceptual level throughout.

Clarity of structure, especially as to the conditions of data usage, is vital; the "Typhoid Mary" program which accepts invalid data from an incorrectly amended system and transmits it to another, innocent, system and brings it down is otherwise expensive to detect and cure. Further uses of the data dictionary as an aid to programming include:

- The storing of common source code (possibly in conjunction with both library facilities and data base descriptions maintained in the form appropriate to the DBMS).
- The creation and maintenance of test files based upon statistics collected in the dictionary.
- The storing of data to control interfaces between, for example, application programs using more than one DBMS or a single, less than adequate DBMS or conventional system.
- The storing of data to control general maintenance and inquiry utilities.

*3.7 OPERATION*

During the operation of live systems, most references to the dictionary will fall into one of the categories discussed in Chapter 7. However, two further uses of the dictionary should be mentioned here:

- Any errors that can be trapped at an accessible level of software could be handled by a common module. This would consult the data dictionary for details of recovery or other action required and inform the operator.
- The dictionary could support the facility to browse through a data base. This would not only provide a means of obtaining information, but would also demonstrate to users that they can access their data in a simple and flexible manner. Such demonstrations improve user confidence in computer systems as can nothing else.

*3.8 CONCLUSION*

The varying allocation of responsibilities in DP departments has already been mentioned (3.4 above). With the introduction of DBMSs, new functions have been identified, if not clearly defined viz, data administrator and data base administrator. Even where the same individual or group undertakes both tasks, these functions can be distinguished; in broad terms, the data administrator is concerned with user data and data structures at the conceptual level, and the data base administrator with data structures at the implementation and operational levels and with the implementation and operation of the DBMS package.

The descriptions of the use of the data dictionary in this and the following chapter indicate the existence of a separate function: that of the data dictionary controller. It is his task to provide the administrators and his other colleagues, particularly analysts and desigers, with a framework for their data and processing structures and to control access to it.

Some of the methods used in DP departments have relied, and will continue to rely, on intuition. However, there is an increase in complexity, which arises largely from the need for computer systems designed to operate across functional or organizational boundaries. Whether or not a DBMS is installed, central recording of the results of investigation becomes essential; major design decisions (even if arrived at intuitively) require justification in relation to a global conceptual model, and general purpose utilities of greater power and flexibility are demanded.

Finally, as user's expectations have increased, DP departments now have to provide integrated data base systems which bear little relation to the simple stand-alone systems that were implemented only a few years ago.

# 4.0 THE PLACE OF A DATA DICTIONARY SYSTEM

*4.1 INTRODUCTION*

This chapter discusses the relationship of the data dictionary system to different software environments and how it can be used in them. The environment normally discussed in accounts of data dictionary system usage is a single DBMS, but a data dictionary system is also important in conventional processing environments and has additional functions in conversion and in a multi-DBMS environment.

The ANSI/SPARC Report presents a prototypical view of the software architecture of a DBMS which includes a data dictionary that supports both the descriptive or definition functions and the internal control functions. Few if any existing data dictionary systems yet achieve complete integration with a generalized DBMS, but the conceptual, simulation and user's application functions of a data dictionary system discussed in this report cannot easily be supported by DBMS internal directories. So the chapter concludes with a brief look at the implementation of the functions and software architecture discussed.

*4.2 THE CONVENTIONAL ENVIRONMENT*

**4.2.1 Relationships**

The software relationships are shown in Figure 4.1.

The data dictionary is a free-standing file, or set of files, supported by its own maintenance and reporting programs, with a structure which is capable of recording a variety of entity types and the relationships between them. Many current data dictionary systems are of this kind.

The conventional software facilities, such as RPG, COPY Library, are not, of course, designed to feed information back to a data dictionary system and therefore do not record, at the time of actually performing the process, the fact that a relationship has been created between particular data elements and programs. Such information is input to the data dictionary system either manually or by software which analyzes the generated programs and creates data dictionary system input transactions to add the relationships to the dictionary.

The generation of data descriptions for high-level languages and reporting utilities is in the form of output from the data dictionary system to the software facilities (such as libraries and parameter validation procedures) in their appropriate language dialects.

In this way, the data dictionary system can support an implementation view of the data; in COBOL and PL/I this is the file description. It would not normally (except in an installation with sophisticated systems programming) provide physical level information for automatic file placement and control. In principle, the conceptual view could also be supported, but this does not affect the software relationships.

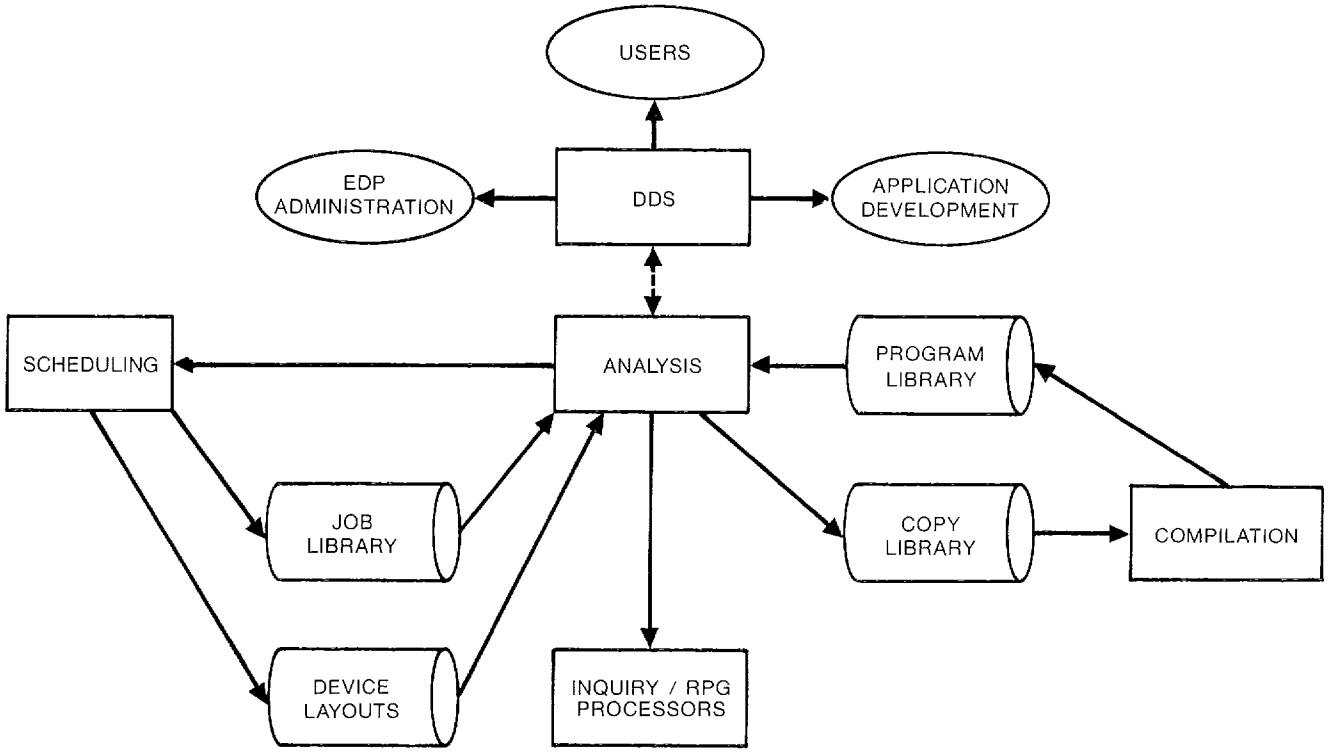


**Figure 4.1** THE DDS IN A CONVENTIONAL ENVIRONMENT

### 4.2.2 Operation and Maintenance

In applications which do not use DBMS-style control files, changes in the encoding of data are facilitated by a knowledge of where the data is used and by adopting a common translation mechanism.

Device layouts can be organized and runs scheduled from the file/task relationships and file volume data. This becomes particularly important when it is required to marshal related or redundant data in different files. Where device analysis programs already produce the information needed by the data dictionary system, an intermediate editing process assembles it into the data dictionary system input format. It is useful to identify program/data relationships in an environment where they are bound together earlier in the compile, load, access process. A JCL interface for recompilation is desirable, but not always easy to achieve.

## *4.3 CONVERSION TO DBMS*

### 4.3.1 Relationships

The ability to interface with a DBMS is important if any conversion processes are to be automated. If the choice of a DBMS has not been made, then the ability to construct a control interface with the data dictionary system should be a criterion for the choice.

If the DBMS is not chosen, or if it is impossible to run the data dictionary system on an existing installation of the DBMS, the data dictionary system can be chosen or developed with eventual conversion in mind, simulating the expected access methods and using the best available update and report package. This is not necessarily easy,

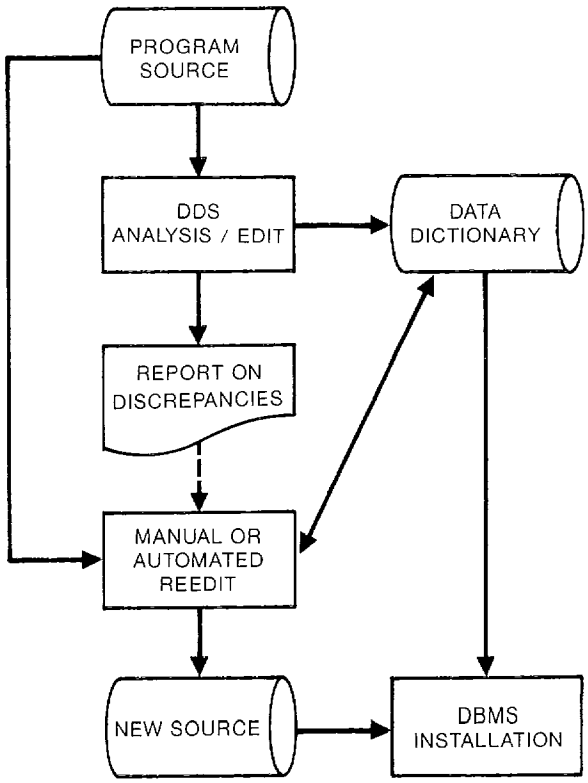


**Figure 4.2** DICTIONARY GENERATION

but it is easier than converting existing applications designed without this constraint.

### 4.3.2 Dictionary Generation

The data dictionary system can be used to prepare existing conventional application programs for use with the DBMS and to create the dictionary entries required to support them. A possible procedure is shown in Figure 4.2.

Whether this process is practicable or desirable depends on the number of programs to be converted and the quality and variety of the programming standards previously employed. The programs are analyzed by software similar to that mentioned in 4.2.1 above. It is basically the implementation view that the data dictionary system is creating. Inconsistencies within this view can be flagged for correction and then perhaps compared with a conceptual definition.

### 4.3.3 Parallel Running

The place of a data dictionary system in the parallel running stage of conversion is shown in Figure 4.3.

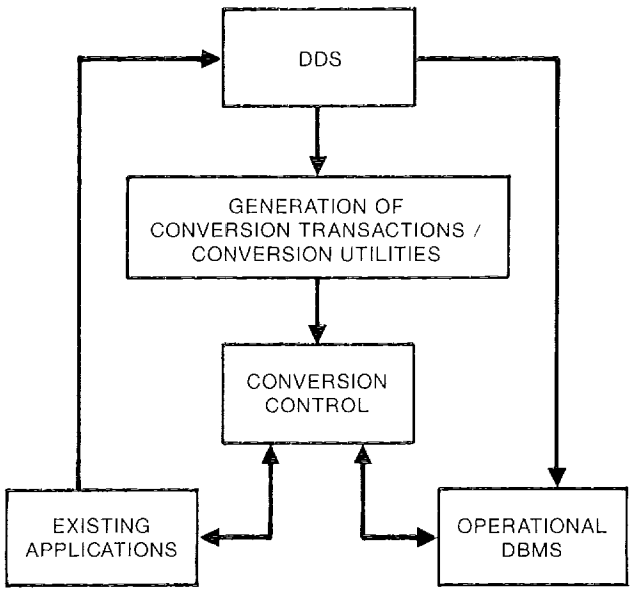


**Figure 4.3** THE DDS AND PARALLEL RUNNING

The data dictionary system provides definitions of conventional files and conversion transactions for use in generating conventional file management utilities. The detail of this process depends on the parallel running strategy adopted, which is outside the scope of this report. The driving of conversion utilities in this way is easier (if the utilities have been selected or developed for this function) than generating the commands for a DBMS to handle all the conventional data. The DBMS update and inquiry utilities are used if available.

## *4.4 SINGLE DBMS*

An architecture for a DBMS has been proposed by ANSI/SPARC (Reference 20) in which dictionary functions are integrated into the software and its manner of use. The dictionary relationships are complex, and there is some doubt whether all the software interfaces can be standardized.

If the directory functions of existing DBMSs do not provide adequate support for the dictionary functions, then the data dictionary system functions should lie between the DBMS and its users. In this way, the data dictionary system helps to overcome any control shortcomings of the DBMS, especially in recording the current state of the conceptual and implementation views of data. These are often either not represented or are not capable of being output from directories in program-readable or user-oriented formats.

### 4.4.1 Relationships

The data dictionary system may be free-standing or implemented as a DBMS application. The DBMS directories should ideally be readable by the data dictionary system to provide data on the current state of the DBMS. The directories are not updated directly by the data dictionary system. The original DBMS interfaces with data base administration and with applications development staff are removed to insure that the data dictionary view of the DMBS is the current one. Instead, the data is updated via the data dictionary system, where it is accepted only when it is also found acceptable to the DBMS.

The data dictionary system is the source for data management control language generation (for which no standard at present exists) as well as the schema and subschema generation. If the DBMS does not have full file maintenance or reporting utilities or if additional packages, such as for user inquiry, are introduced, these must also be serviced by generation facilities. One way of driving this generation is to describe the formats to be output, plus the transformation procedure for deriving the fields from the dictionary subschema data definition language entires. This capability should be present in the subschema translation of a CODASYL-compatible DBMS.

Where a DBMS supports one programming language only, the data dictionary is used as the input to alternative generation facilities. This can be done either at the source level (subschema and CML) which involves knowing DBMS internal directory formats, or at the subroutine library level. Figure 4.4 is a necessarily very simplified view of some of the above relationships.

## *4.5 COMPLEX SOFTWARE AND USER ENVIRONMENTS*

Most data base literature seems to assume a single, all-embracing DBMS. Conceptually, this may be so, but at the software and operations level the DBMS environment is usually more complex because of an operating system interface, the existence of more than one DBMS or the continued use of non-DBMS packages, applications and files.

### 4.5.1 Relationships

Figure 4.5 shows the data dictionary system in a complex environment.

This environment might exist because a large and structured enterprise has different views of some of its data (held on the same or a different DBMS). A similar case is the use, on a single machine, of a mixture of applications using a DBMS-style approach (such as control files, some transparency) which are not integrated at the software level. The mixed DBMS and conventional file environment which might exist for some time after installing a DBMS will appear similar. This would be a transition toward the ideal of realizing a single conceptual view.

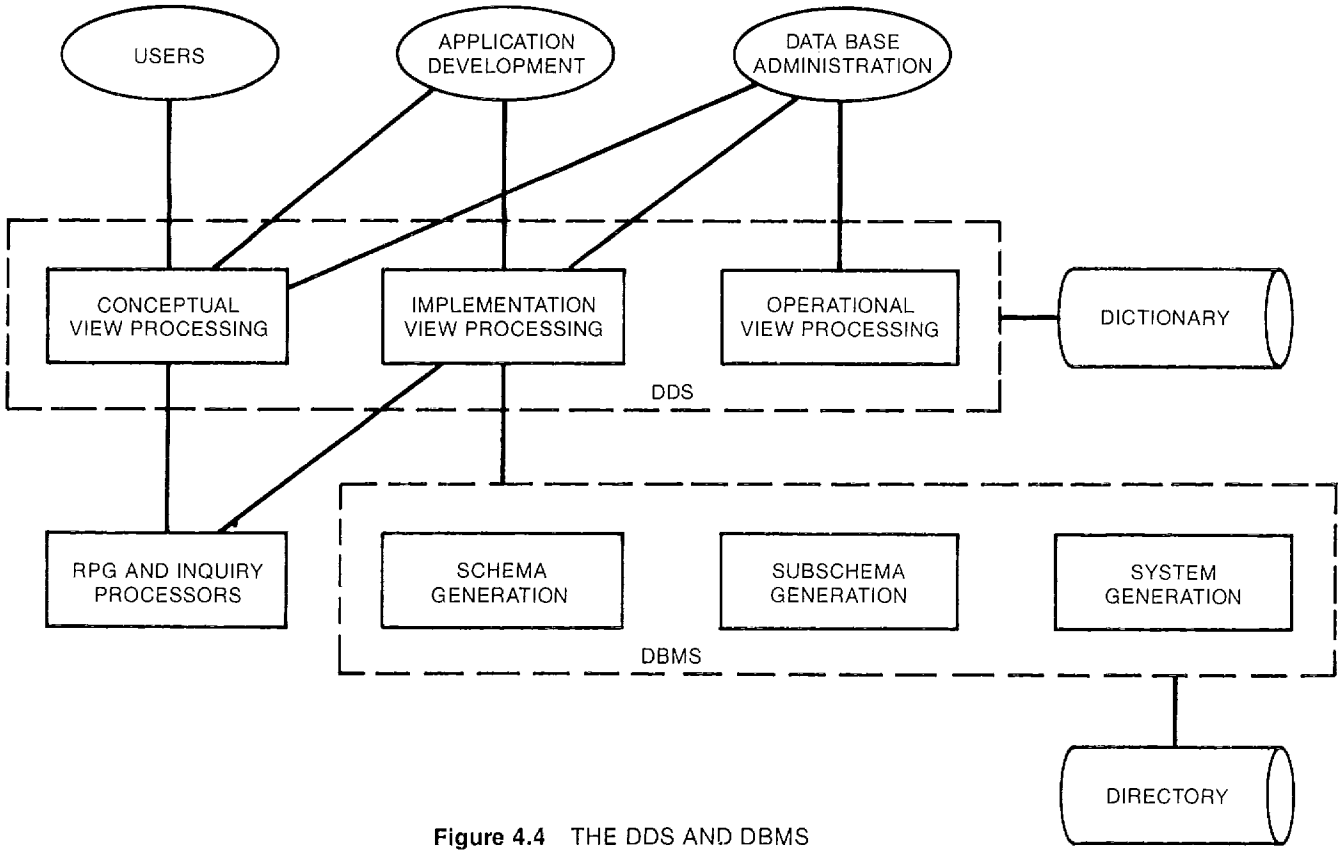


**Figure 4.4** THE DDS AND DBMS

There are alternative architectures for the users' view of their part of the conceptual data:

- A centralized approach would attempt a common definition from which each view was extracted, whereas
- A decentralized approach would extract from separate dictionaries those parts which were common and which the enterprise administration required.

As a software implementation, the data dictionary system might be independent or it might exist within one of the DBMSs.

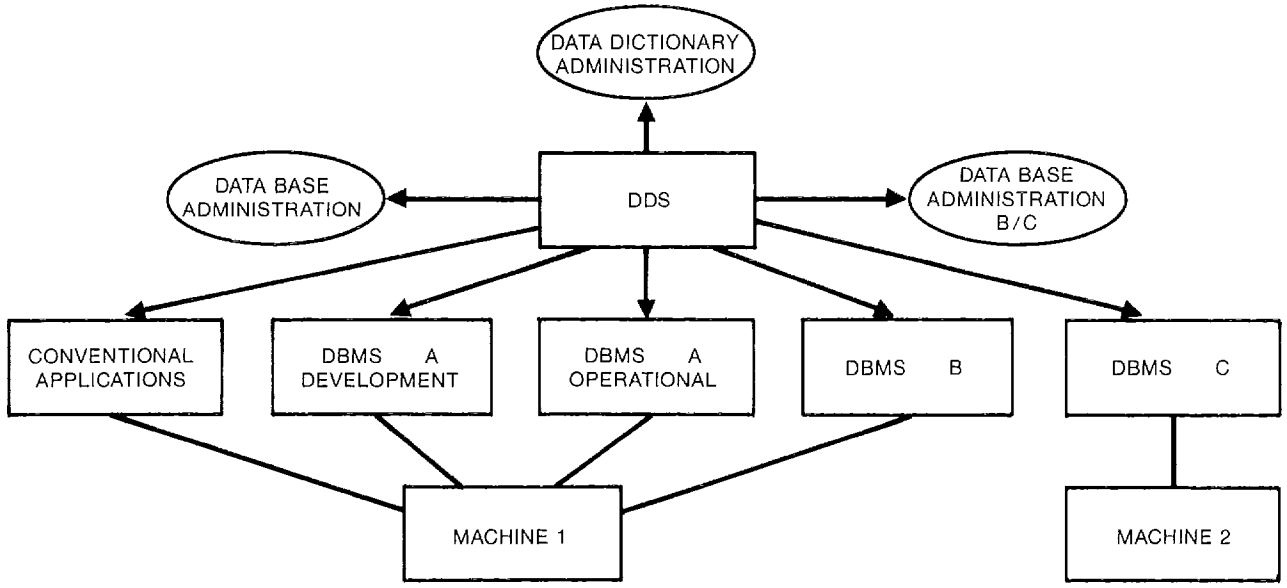


**Figure 4.5** THE DDS IN A COMPLEX ENVIRONMENT

### 4.5.2 Consistency Control

Centralized or not, a separate data dictionary administration function is required, and the consistency of the entries needs to be controlled as far as is possible in the enterprise environment.

Dictionary entry formats should be flexible, especially if several machines are in use or portability is required. The implementation of the dictionary itself by means of one of the DBMSs under control makes this easier to achieve. If the software architecture forces the generation functions to be written under the DBMS concerned, operational control may be practicable only with a separate dictionary under each DBMS.

### 4.5.3 Common Applications

Even the best controlled enterprise may have difficulty (after merger, for example) in relating data formats between distant parts of the organization. This particular requirement for data independence is sometimes overlooked: the need not only to access the same data from different applications, but to access different data from the same applications. If the application is old, or an outside purchase with no built-in transparency, a DBMS alone may not be the answer. File construction utilities similar to those required during conversion may have come from a different supplier from the DBMS, or be written as applications under the DBMS. The data dictionary system provides control of the non-DBMS files, and serves as a record of the nonstandard procedures.

### 4.5.4 Package Generation

Otherwise desirable application packages, especially those in intimate contact with users, sometimes have forbidding hieroglyphic user interfaces with their own naming conventions for data, or none at all. Since users need a view of their data which is consistent between, say, a DBMS maintaining a library of project costs, and a financial performance evaluation package, yet another generation utility will be required.

## *4.6 DATA DICTIONARY SYSTEM IMPLEMENTATION*

This section discusses the impact which the place of a data dictionary system in different environments and the different requirements of its users in the same and different organizations have on the design of the data dictionary system.

### 4.6.1 Software Architecture

The purely internal facilities designed to drive a DBMS and sometimes called a directory are not discussed here. For a conceptual description of directory functions, see Reference 20. The relationships between a dictionary and a directory in a DBMS are serviced by the various DBMS processors mentioned below in 4.6.6.

There are at present two broad groups of data dictionary systems in use:

- The free-standing package which could be used in a non-DBMS or a DBMS environment.
- The single DBMS facility designed to interface with a particular DBMS. In addition each group contains both privately developed systems, originally or still intended for the use of the one organization only, and packages intended for general marketing.

The subject of dictionary selection is discussed in reference 5. A wide variety of features and methods of use for a dictionary are identified, but their priority in selection varies from one enterprise to another.

The environments differ also. An enterprise may wish to use more than one model of DBMS simultaneously, or may not wish to be tied to a single DBMS to retain future flexibility (Reference 1).

Implementations of a DBMS do not all use the same mix of DDL, DML and DMCL statements, or the same relationships between them. It is difficult for the data dictionary system designer or implementor to foresee all the data items that may be required to interface with different DBMS implementations, and the control of non-DBMS software, if required, is even more problematic.

It is necessary to see the data dictionary system as an application in its own right and to apply ordinary application design principles to it. If this is done, its appeal will be much wider, and the private implementation will be better able to provide the flexibility which is usually an important requirement. It is possible to cope with the situation of a pre-DBMS dictionary if it is seen as an application which will itself need conversion.

The data dictionary system should be easy to extend and therefore written in a high-level language for easy modification, interface with and possible storage on a DBMS.

### 4.6.2 Software Components

The main components are shown in Figure 4.6.

From the previous sections of this chapter it can be seen that in use a typical data dictionary system consists of a data base of data about data and about procedures, which is supported by:

- Formal transactions to maintain and report on it.
- Interface software to gather input from directories and compilers in other parts of the operational environment.
- Editing software to generate input required by compilers, packages and so on, both inside and outside a DBMS.
- Complete "applications" using the dictionary to perform consistency control, change impact analysis and so on.

Some of these components are part of the data dictionary system as a package, while some will have to be provided by the individual installer. This has implications for the data dictionary system design and method of use.

### 4.6.3 Input/Output

Where the data dictionary system is to be implemented using a data structure (DBMS or conventional) which will also be needed for application systems, modular design will allow for the use of common utilities for updating and reporting. The data dictionary system or these utilities should be capable of providing simple input to editors for the production of JCL, DMCL and so on. Some of these will use very simple file structures, some will require that the editor be capable of reading the dictionary files. The dictionary formats should be able to hold code for the less intelligent generator programs, and since this code may depend on syntax checking outside the dictionary, there should be an input method capable of associating it with existing entries.

### 4.6.4 Dictionary Data Formats

Dictionary formats need to be extensible; this will normally require the transparency supplied by a DBMS, but it is also possible to use software tools like macrogenerators to generate a dictionary system for file formats specified by the user.

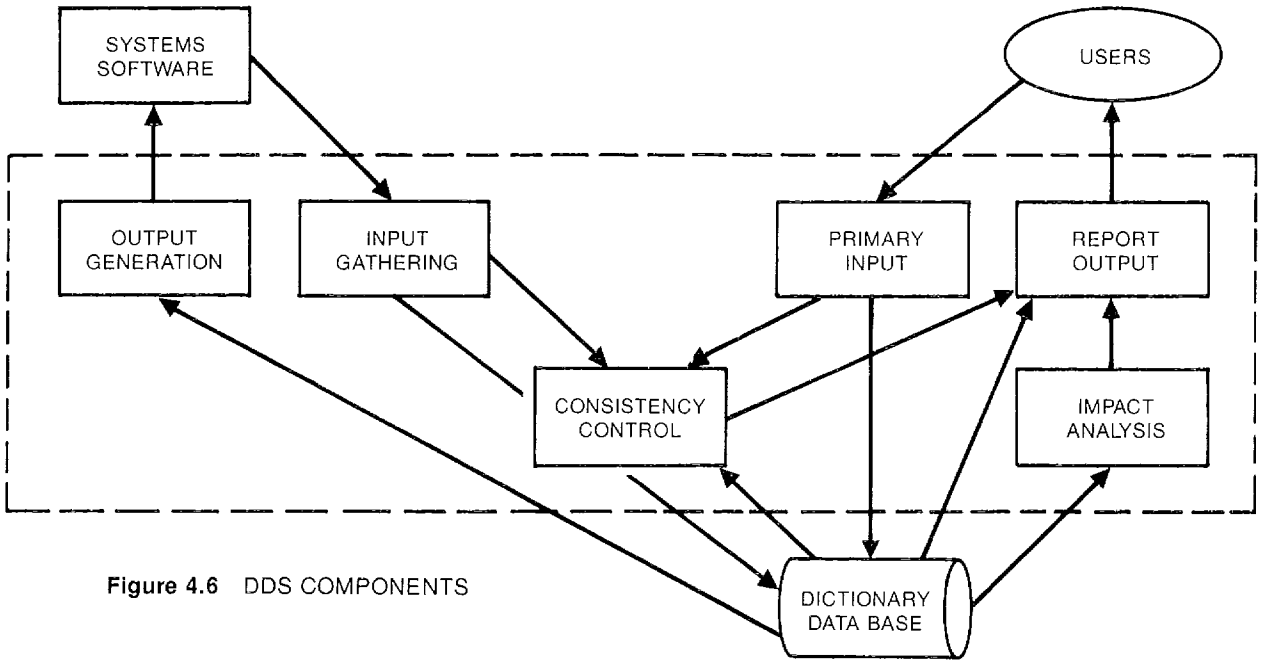


**Figure 4.6** DDS COMPONENTS

#### 4.6.5 Security and Privacy Control

Dictionary data should be placed under the same constraints as any other important part of the data base. This means that all dictionary updates should be performed under the same control and by formally defined transactions, which are also used by any extended applications operating on dictionary data, such as reorganization or simulation. In this respect the dictionary is a part of the data base. Directories are derived from the dictionary, and are not normally available for update by any other means.

Various users will require access to the dictionary, either to browse through, or to analyze the data. The data dictionary controller (who may also be the data base administrator) will need privacy facilities to restrict access, where necessary.

#### 4.6.6 Software Interfaces

The data held in the various dictionaries and directories must be kept in step. This necessitates automatic transaction generation, tight job control or integration of the software components concerned. In this way the data dictionary system can be informed that, for example:

- DDS input of a subschema is valid to the DBMS and
- that this subschema has been bound to a particular version of the DBMS and
- that the status is live.

In short, to control the DBMS, and provide better documentation than the DBMS can achieve unaided, requires an interactive dialog between the data dictionary system and DBMS functions.

## 5.0 THE CONCEPTUAL VIEW

*5.1 INTRODUCTION*

This chapter explains why a data dictionary system must support a conceptual view of the enterprise, outlines the facilities needed in supporting that view and goes on to specify the data which must be held in the system. It is assumed that the system will also support an implementation view of the enterprise, so the mapping between the two views is discussed.

The conceptual view describes the nature of the enterprise and its data in terms which are quite independent of any data processing implications. It therefore consists of a model of the enterprise containing descriptions of things of interest to it, functions it can perform and events which influence the way it performs.

The basic argument in favor of including a conceptual view can be outlined as follows: When considering information handling within an enterprise, it is useful to make a distinction between, on the one hand, the functions performed by parts of the enterprise, such as invoicing, staff recruitment or recording stock receipts, and, on the other, the supporting data processing applications which assist the enterprise in performing these functions; similarly a distinction can be made between the things and events of interest to the enterprise and the files and transactions used by its various application systems. These distinctions can thus be used to define both a conceptual view, representing the enterprise and its operation, and an implementation view, representing the manual or computer-based application systems and their data.

*5.2 THE NEED FOR THE CONCEPTUAL VIEW*

#### 5.2.1 Coordination of Overlapping Applications

There is often a substantial degree of overlap between the functions performed by different parts of an enterprise. As a result, there may be a similar overlap between their data processing applications when dealing with similar data. If such overlapping is to be recognized and effectively controlled, we must acquire descriptions of the functions performed by the enterprise which are set in the context of the overall operation of that enterprise, and not merely that of a single application. For example, it is easier to recognize potential duplication between:

- The function of recording the name, address and personal details of each new employee and

- The function of recording the name, address and contribution details of a new member of the pension fund,

if we have first identified "employee" as an entity of interest to the enterprise than it is if we have information about the entities (for example, "new employee," "fund entrant") and functions only as conceived by the corresponding application areas.

What is required is a view of the functions performed by the enterprise, and the entities of interest to it, which is not biased by the way in which particular aspects are viewed or implemented locally. This conceptual view sets a standard to which each implemented application system should conform and by which their development may be coordinated.

#### 5.2.2 Independence From Implementation

Since data management techniques are being refined rapidly at the moment, it is likely that the lifespan of many of the programs in an application is potentially greater than that of the data management software which supports them. If the content, structure and means of processing of the data have been defined only in terms of the capabilities of the support software, then it will be difficult, if not impossible, to adapt the application for new software or an alternative data organization.

If, however, the data dictionary system contains a conceptual, implementation-independent view of the data and the application (as well as details of the implementation), then the original implementation will not contravene the logic of the conceptual view. It should therefore be possible to substitute an altered implementation by working within the conceptual framework.

The conceptual view of an application can also be used as a link between parallel implementations of the same system, perhaps using different software or even different hardware and software.

#### 5.2.3 Application Analysis and Design

The conceptual view is a record of the enterprise's real world, and so provides the analyst with a description of the situation which his systems have to model. It therefore provides a context for his design and, by defining the scope, a means of controlling the design. The presence of a conceptual view also helps him when trying to assess the effectiveness of changes to his own designs and further allows him to estimate the effects that changes in the real world might have on his systems.

#### 5.2.4 Data Base Design

It is particularly important in the early stages of design that there should be a clear statement of the data requirements of the enterprise which does not presuppose any storage structures or access methods. This statement will allow the designer to take the needs of all existing and planned systems into account and will give him the freedom to design a data base structure of sufficient flexibility to meet all likely demands. The designer also needs the conceptual structure to help him in understanding the conversion of:

- Manual files to a data base.
- Conventional files to a data base.
- The restructuring of an existing data base.

#### 5.2.5 Control of Security

Unless security aspects are controlled from the outset in creating a system, problems are likely to arise which can only be solved in an inefficient and probably inadequate manner. If considered within the conceptual view, security can be covered by consistent planning throughout all applications and implementations.

### *5.3 THE FACILITIES REQUIRED*

#### 5.3.1 Recording

It should be possible to record details of:

- The entities with which the enterprise is concerned. (An entity is the basic unit about which data can exist; it is any thing, person or place of interest to the enterprise.)
- The functions of interest to the enterprise or carried out by it.
- The events which occur within the enterprise or impinge on it and which initiate or result from the operation of a function. Events act as triggers and may arise in the environment of the enterprise or may represent the interaction of functions within the enterprise or some form of scheduling constraint.
- The relationships which exist among entities, functions and events and between any of these. Privacy constraints on access to data and the stringency with which they are to be applied may be treated as particular forms of relationship.

Information of this sort is liable to change as (but only as) the enterprise and its environment change; for all of these elements of the conceptual view, therefore, the data dictionary system should be able to record details of different versions recognized as valid at different times or in different contexts.

It should, of course, also be possible to correct errors in the initial analysis of the enterprise and its data, but this should not necessitate different versions.

#### 5.3.2 Analysis

A wide variety of analyses of the data will be required, both general and specific. Some form of searching of the textual descriptions stored in the dictionary is probably essential if the full potential of the system is to be realized; so are interactive inquiry facilities if users of the system are to accept it as a real aid to their work, rather than as another source of delay and paperwork.

#### 5.3.3 Mapping to the Implementation

Much of the value of the conceptual view lies in its relationships with corresponding implementation views. It is vital that the system contains adequate facilities for defining this relationship by specifying and mapping between implementation concepts such as files and records and conceptual concepts such as entities. A lot of thought will be necessary to insure that this requirement is fully met, and in the short term it can perhaps best be satisfied by the provision of a general-purpose mapping facility rather than a set of specific links which might not be complete.

### *5.4 DATA REQUIREMENTS*

#### 5.4.1 Entities

Entities must be described in terms of their attributes (properties) and relationships. Ways of identifying uniquely an occurrence of the entity must be stated; while it appears to be possible to define all relationships between entities implicitly, through the sharing of attributes between entities, it is probably desirable that the user should be able to

define relationships explicitly if he finds this more convenient.

### 5.4.2 Functions

Minimum details of functions will include the actual functions or operations, probably at more than one level, and some data concerning the responsibility for these functions, probably in terms of the structure of the organization.

### 5.4.3 Events

The origin of the event, responsibility for it and the functions affected by it form the minimum details required.

### 5.4.4 Relationships Within the Conceptual View

Links will be needed to express the relationship between functions and the entities with which they are concerned.

### 5.4.5 Mappings to the Implementation View

It will be necessary to record mappings between the conceptual and implementation views. Because of the potential diversity of implementation views, a very general mapping scheme will be needed to insure that valuable information is not lost.

### 5.4.6 General

Many of the details which would normally be considered at the implementation stage may sensibly be considered part of the conceptual view. It should be possible to record at this stage such details as permissible ranges of values, number of occurrences of an entity type and other assertions about the data base.

## *5.5 THE DDSWP CONCEPTUAL DATA MODEL COMPARED TO THE ANSI CONCEPTUAL SCHEMA*

The purpose of this section is to comment on those areas of the interim report (February 1975) of the ANSI/X3/SPARC DBMS study group which are relevant to the work of the DDSWP. The reader of this section should be familiar with the content of at least Chapter 2 of the ANSI report.

### 5.5.1 Purpose of the Conceptual Levels

The DDSWP conceptual data model is intended as a tool to model all the data held by the enterprise, not just data held in the data base, and to do this in a storage structure independent manner, using a data model common to all sources of the data. As illustrated in Figure 1.3 in Chapter 1, descriptions of manual and conventional files, and their mappings to the data model, should be recorded, whether or not a DBMS is in use.

The purpose of the ANSI conceptual schema is somewhat different. It exists to insulate the external schemas from changes to the internal schema and vice versa; to provide the lowest common denominator between several external schemas which require different views of the same data, for example a relational, Codasyl or an IMS view; and to provide a central frame of reference for resolving security and integrity problems.

For these reasons, the conceptual schema is not view independent; there are constructs which correspond to hierarchical or network structures. To support differing external views, conceptual records may be of arbitrary complexity (a construct of conceptual records when named becomes a conceptual record). It may be convenient, though not necessary, to define simple underlying conceptual records in terms of which the others are defined. These records would correspond most closely to the DDSWP data model, although they only model that data which is held in the data base, rather than that of the whole enterprise. This comparison is illustrated in Figure 5.1.

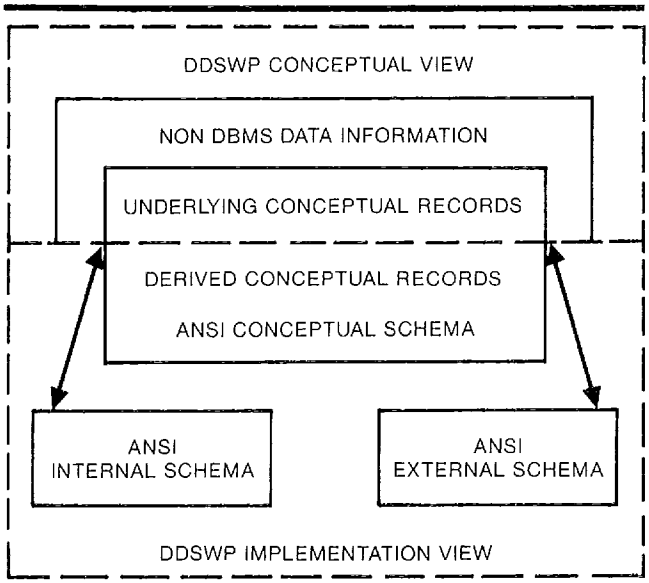


**Figure 5.1** A COMPARISON OF THE DDSWP CONCEPTUAL VIEW WITH THE ANSI/SPARC CONCEPTUAL SCHEMA

### 5.5.2 The ANSI/SPARC Data Dictionary

There is a box in the ANSI architecture labelled Data Dictionary, which has 10 identified interfaces into it, each a candidate for standardization. It is viewed as a central repository for declarations, mappings, compiled access paths, usage statistics and so on. ANSI also note that the conceptual schema can form the basis of a data dictionary.

### 5.5.3 Recommendations

The DDSWP believes that the avowed purpose of the conceptual schema, to provide the description of the conceptual model of (some portion of) the enterprise's information, is obscured by the other aims of the conceptual schema. We therefore recommend that the ANSI Data Dictionary contain a description of *all* the enterprise's information, using the conceptual data model described earlier in Chapter 5. This means that a further interface must be identified through which such a model can be entered and viewed.

The DDSWP believes that the aims of the ANSI conceptual schema will only be achieved if some discipline in its use is enforced. We suggest that underlying conceptual records should be defined which correspond directly to the objects defined in the conceptual model. Ideally this would be a straightforward subset. More complex conceptual records to support user views are then defined in terms of these underlying conceptual records. It is possible that these underlying conceptual records may be redundant, in that internal records may be mapped directly to the more complex conceptual records. However, the fact that this underlying view exists will ease future changes to the conceptual schema.

We propose that the ANSI conceptual schema be di-

vided into two parts, the underlying conceptual records (or the conceptual schema) and the derived conceptual records (or the derived schema). The derived view is analogous to, say, a Codasyl schema, and the external view is then analogous to the subschema.

The new conceptual view could then be extended so that it described all of the data in the system and not just that held in the DBMS. This is illustrated in Figure 5.2.

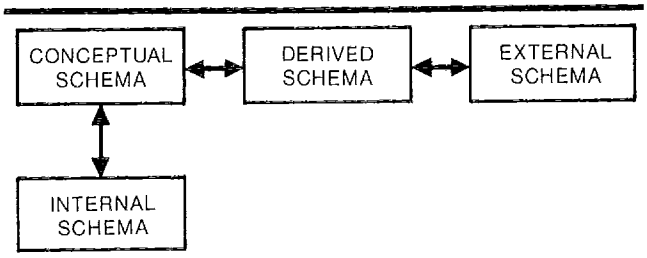


**Figure 5.2** RECOMMENDED CHANGES TO THE ANSI/SPARC ARCHITECTURE

The old conceptual schema is split into a conceptual schema and a derived schema. The new conceptual schema should model the whole enterprise.

# 6.0 THE IMPLEMENTATION VIEWS

## *6.1 INTRODUCTION*

This chapter outlines those features of a data dictionary system which should be provided to enable it to describe elements of data processing systems at the implementation level. It discusses some of the reasons for maintaining a view at this level, describes the facilities required to support the view and the facilities which it, in turn, can support and indicates the types of data elements which can be used to describe the system structures.

The implementation view covers the form of the data processing system devised to meet some or all of the requirements specified in the conceptual view. It includes descriptions of manual files, conventional computer files, data base systems and supporting processes or applications (although this document deals only with those aspects of the implementation view which involve computer processing). It also includes both complete system descriptions and local descriptions particular to processing areas (such as Codasyl schemas and subschemas). This view, in the data dictionary, is therefore the basic source of information about the DP system and supplies data for studies to establish the design of the system, to prove its correctness, to determine forms of improvement and assess their cost and to examine strategies for future development.

The implementation view must be logically consistent with the conceptual view, when defined, or with that portion of the conceptual view which it represents. It can never have a greater scope than the conceptual view. Its scope could be regarded as covering all aspects of the implementation, but we have chosen to restrict our discussion to those aspects which serve to meet only the functional requirements of the enterprise. Those aspects which are concerned with run-time efficiency and the operational requirements of the enterprise are discussed in Chapter 7 as the operational view.

## *6.2 THE NEED FOR IMPLEMENTATION VIEWS*

### 6.2.1 A Source of Information

There is a need for an adequate means of recording and disseminating information about the data processing system. The data dictionary meets this need by providing a way of maintaining a coherent, centralized library of data about all aspects of the system, thus enabling all users to have a consistent view of the system.

One of the most important features of a data dictionary system is its ability to document the use of elements of the system by other elements. For example, many installations developing toward a DBMS find it necessary to devise procedures to list items used by programs and to cross-reference programs accessing items. It is difficult to produce such cross-references manually and to keep them adequately updated, but the data dictionary system can document the use of items by programs and can, moreover, provide the essential cross-references automatically. The user thus obtains a valuable additional report at no cost to himself in terms of time spent preparing data for input.

### 6.2.2 A Design Aid

Knowledge of the frequency and sequence of use of data items is essential to the design of both data structures and accessing procedures. Furthermore, once the data structure has been determined, this will determine to some extent the structure of the application processes. The implementation view also serves the needs of system tuning, since it carries information on the frequency of use of the applications processed along with the accessing paths they follow. This data can be used to make simple predictions of the machine resources required for each process, thereby highlighting those making the greatest demands on the system which are therefore candidates for further investigation.

### 6.2.3 Control, Coordination of System Use Elements

The implementation view contains all the available information about the form and usage of data so it can be used to provide a degree of quality assurance in program production. A data dictionary system can be used to insure that the use of data by programs is both correct and allowed.

The data dictionary also contains the validation rules pertaining to each item of data in the system so it can further meet the needs of quality control by confirming the adequacy and completeness of validation procedures in the system.

It is possible to use the data dictionary to meet the need for coordination of the usage of data by a variety of application subsystems. The consistent usage of data is insured; the correct sequence of transformations for equivalent data items is insured; checks may be made to insure that the data conforms to the rules of usage under its particular representation (such as host language or DBMS); finally, forward compatibility standards can be checked to insure that one representation can be transformed to another in the future with minimum inconvenience.

### 6.2.4 Analysis of the Impact of Change

In an active installation, program modification and development is continuous and changes to the data structures are frequent. Some such changes not only need to be carried out immediately, but also form part of a series of

amendments which have to be effected in sequence. Planned maintenance depends, in part, upon easy access to information about these consequent changes. Other changes may be proposed as design modifications and the implementation view can then be used to meet the need for information on the ramifications of the proposals, thus permitting an assessment of their effect and cost.

*6.3 THE FACILITIES REQUIRED*

Many facilities can be provided by a data dictionary system, but we regard certain of them as essential. Of those that follow, some form of data capturing, consistency checking and access and reporting facilities are essential, while the others, though desirable, may be omitted at the discretion of the implementor.

### 6.3.1 Data Capture

The data dictionary system must provide facilities which enable descriptions of the implementation level structures to be set up and maintained. The data required for these descriptions may be either input directly by the dictionary users or captured from existing machine readable sources. Input may therefore be acquired in at least the following ways:

- Directly, possibly through an input language.
- From program data definitions in high-level languages such as COBOL.
- From DBMS source definitions, such as Codasyl schema and subschema definitions or IMB Data Block and Program Specification Block definitions.
- From program procedure statements.

### 6.3.2 Consistency Checking

The data dictionary system must be capable of performing consistency checks so the information it contains is complete, correctly formatted and properly cross-referenced during the development process. These checks may be carried out during input or at any later time when sufficient information is available.

A further validation of the implementation view must be achieved by confirming the correctness of the mapping between it and the conceptual view.

### 6.3.3 Access and Reporting

To insure that the data dictionary is convenient to use, facilities must be provided for formal reporting and to permit impromptu access.

Report generation giving formatted descriptions of the contents of the data dictionary and their uses must be a standard feature of the system. Reports should be produced when requested and could include the following:

- Catalogs and plain language descriptions of the data and system elements.
- Lists of elements in keyword order or by category.
- Data structure/system structure cross-reference listings.
- Data definitions for host language programs and for data base views, such as COBOL data division descriptions, schema/subschema descriptions, data block definitions.

Selective reports should be available on demand providing a subset of some full report, such as elements listed for a single keyword or the data structures associated with a particular application process.

### 6.3.4 Test Data Generation

The inclusion at the implementation level of information on permissible values of ranges of values for items can be used in conjunction with the descriptions of the items to allow test data to be produced.

### 6.3.5 Data Description Generation

The description of data structures at the implementation level can be sufficiently detailed for the data dictionary to generate data descriptions suitable for incorporation in host language or data manipulation language programs. These descriptions might be placed in the programs during a precompilation pass against the dictionary. These may be used for the generation of declarations for the DBMS and so on. Alternatively, the DBMS may access the data dictionary system directly. Automatic generation of declarations enforces accurate recording of program requirements.

### 6.3.6 Program Code Generation

The implementation level can hold information on the sequence of data use by processing units which, in conjunction with information on the data structure, is sufficient to enable it to generate access or input/output modules for the processing units. Likewise, the inclusion of validation rules at the implementation level would enable validation programs to be generated. Similarly standard code can be generated to provide installation specific facilities.

### 6.3.7 Access Control

The dictionary can hold information which indicates who may have access to particular items of data and what conditions govern that access. It can use the information in the following ways:

- In examining programs prior to compilation to check that no data is referenced in a manner that contravenes the access regulations.
- By providing an interface so that a program or utility with a wide range of users can have its user checked at run time.

If the data dictionary is to be used for access control, then it must itself be subject to access control. The number of people allowed to modify the contents of the dictionary should be restricted and privacy constraints may be applied to parts of the dictionary.

### 6.3.8 Impact Analysis

The data dictionary system should contain sufficient information to provide a complete analysis of the impact on programs of a particular change to the data structure. It should be possible to request reports describing the impact of proposed changes and including estimates of the likely cost of such changes.

### 6.3.9 Multiple Version Handling

Several versions of programs and data structures may exist at any given time. These may represent live, test or design states, slightly different forms used at different computer sites, or similar structures implemented under different software. The dictionary system should be capable of recognizing these and reporting on the differences between the versions.

### 6.3.10 Utilities Interface

The data dictionary system contains sufficient information to support various utility systems. There should therefore be a well defined interface to utilities such as:

- Interactive query processors.
- Interactive data base edit/update systems.
- Report writers.
- Data base restructuring programs.
- Other data dictionary systems.

*6.4 DATA REQUIREMENTS*

### 6.4.1 Data Description Elements

Elements are required to describe the types and structures of data used within the system, such as items, records, files, Codasyl sets, IMS segments and so on. The type of data handling software employed in the implementation (such as COBOL, IMS, Codasyl implementations) will determine to some extent the categories of elements needed, but the information required to describe the elements can be relatively independent of the software and should be of the form described at 6.5.

### 6.4.2 Process Description Elements

The dictionary requires elements such as systems, programs, transaction procedures and modules which can be used to describe the structure and nature of the processes operating within the implementation. If an arbitrary number of levels of subdivision is permitted, a few element types should be sufficient to describe all types of implementation. The information used to describe the element types should be of the form described at 6.5.

### 6.4.3 Relationships

It should be possible to describe the relationships that exist between different types and structures of data (such as file to record to item, item to record to file), and between different types of processes (such as system to program, program to system). It should also be possible to describe the relationships that exist between the data and the processes (such as program to file, file to program), thus providing a complete picture of an enterprise's data processing function.

### 6.4.4 Mappings

The data dictionary should include mapping elements which make it possible to relate each element in the implementation view to the corresponding element(s) in the conceptual view. This must be done in such a way that any inconsistencies between the two views can be detected. The mappings are not necessarily one-to-one and there are no obligatory equivalences between particular pairs of element types. It is likely, though, that a data item will correspond to an attribute, a record to an entity and a program or module to a function.

*6.5 ELEMENT DESCRIPTION*

For each element described in a data dictionary, the information held falls into the following categories:

- Naming information. This covers the variety of names that may be used at different times and in different places to identify the element, such as data dictionary name, data base name, data administration name, programming names. It may also include the status of the name, that is, current, superseded.
- Classificatory information. This covers the full range of details which go toward establishing the unique identity of the element type. It may include a description in natural language, details of ownership, privacy constraints, security classification, keywords, classification information, element status, optional/mandatory presence and so on.
- Representational information. This describes the form of the element as it appears within the data processing system and applies largely to elements of data structures. Any data definition language feature may appear in this category, such as item type, item length, index names, location mode, set order. Processing units may be described in terms of programming language, number of statements, usage mode (read, write and so on), parameters and so on.
- Usage information. This describes the use of the element quantitively, giving for each type of measure the maximum, mean and median values and the standard deviation. It includes the frequency of use and period, the number of times called and the frequency of use for specified purposes (create, read, modify, write, delete).
- Administrative information. This describes the resources used or required by the element, such as size, number of occurrences, total storage requirement, maximum memory space requirement, CPU time, maximum memory utilization (millisecond/word times), processing mode (batch, time-sharing, transaction processing).

# 7.0 THE OPERATIONAL VIEW

*7.1 INTRODUCTION*

This chapter describes those facilities of a data dictionary system that would support an operational view of an organization's data. The conceptual and implementation views are concerned with defining the data structure as seen by the enterprise and the programs. On the other hand, the operational view is concerned with the details of the physical storage of data and its use.

*7.2 THE NEED FOR THE OPERATIONAL VIEW*

### 7.2.1 A Consistent Source of Information

There is a need for an adquate means of recording and disseminating information about the data processing system. The data dictionary meets this need by providing a way of maintaining a coherent, centralized library of data about all aspects of each application, thus enabling users to have a consistent view of the system. The operational view includes the most basic form of the data processing system, the actual media upon which the system is implemented. It is therefore a basic source of information about the system and supplies data upon which to base decisions covering the actual operation of the system.

### 7.2.2 An Aid to the Coordination of Resources

Modern data processing systems often involve complex multiple usage of common resources. Operating systems are becoming more efficient in their scheduling of queued jobs to optimize the use of these resources. However, scheduling is only performed on the jobs within the queue and within a specified sequence. There is a need to provide an overall source of information concerning the physical files involved, the jobs to be run and their interrelationships for such scheduling to be both meaningful and realistic.

### 7.2.3 Operational Control Facilities

A data dictionary system should provide facilities to minimize the manual work involved in system operation. The following are some of the functions a data dictionary system, subject to security criteria recorded on the dictionary, could provide:

- The automatic generation of job/system control language.

- The auditing and restoring of files to valid states.
- The automatic generation of file/data base utilities.

#### 7.2.4 Analysis and Control of Change

In an active data base or conventional file environment, change is inevitable. Schema changes will normally need to be implemented in accordance with project schedules; knowledge of the physical realizations of the schema provides some of the material for the planning and control of these changes.

Data base or file reorganization normally arises from the status of the physical files themselves; again the operational view may be used to provide information to assess the desirability and the cost of such action, the problems which may be encountered and their solution.

Generation of JCL and utilities from the data dictionary definitions may be used to aid the execution of reorganizations.

### *7.3 THE FACILITIES REQUIRED*

#### 7.3.1 Standardized Reporting and Interrogation

Full standardized reports should be produced on demand or after an amendment to the dictionary. The content of these reports will be highly dependent upon the software environment. Typically these could include:

- Catalogs of operational files and their attributes.
- Catalogs of programs, transactions and processing units and their attributes.
- Listings of the mappings between the operational files and the implementations.
- Listings of cross-references within the operational view, for example between operational files and processing units.

So users may find the data dictionary convenient to use, it should provide for ad hoc interrogation by authorized users.

#### 7.3.2 Mapping

If the operational view is supported as part of the implementation view, then no additional mapping is required. If the operational view is supported as a separate view, then mapping will be required between it and the implementation view. The operational level of the data dictionary must be consistent both within itself and within the implementation view.

#### 7.3.3 Validation

The data dictionary system should validate the input for syntax, consistency and completeness. These checks could include:

- That the characteristics of each physical file are described.
- That the contents (in implementation terms) of each file are described.
- That for each physical structure, all the constituents of the implementation data structure are allocated to at least one physical file (or expressly nulled).
- That each processing unit (job, program) is recorded as accessing a physical data structure which is consistent with the entire implementation data structure which it is required to process.

#### 7.3.4 Operational Scheduling

The scheduling of resources is a critical function of the operations department. It is, for example, somewhat pointless to optimize the I/O performance of one processing unit by arranging to scan the data in physical sequence, only to permit contention for resources by a concurrent run unit. Also the DBMS may provide concurrent update protection facilities, but these facilities themselves have an inherent operational cost. Thus it may be desirable to process jobs consecutively.

The operational view contains the information from which it should be possible to derive an "optimum" operational schedule. The derivation of such a schedule is obviously arduous and thus preferably performed automatically.

#### 7.3.5 Statistics Monitoring

The operational view of the data dictionary should record the allocation of physical resources and could also supplement this information with utilization and performance statistics. These statistics would be entered manually or dynamically obtained from the operating system or DBMS, depending on whether the data dictionary system is free-standing or part of a data base or file management system. Information would then be available to review and optimize these allocations.

#### 7.3.6 Simulation

An important use of a data dictionary is to provide information on which to base investigation of the performance of proposed data structures.

The inclusion of each operational definition of data volume information in addition to the physical descriptions of files themselves would provide information for the realistic simulation of data base performance.

It may not be possible to hold this information at the implementation level since there may be several physical realizations for one implementation level data structure. The simulation could be provided by the data dictionary system or by interfacing to simulation aids or facilities provided by a DBMS.

#### 7.3.7 Job/System Control Language—Command Generation

There is a need to set up and change the job control commands relating to the physical realization of the data base or conventional file environment and its relationship with programs and jobs.

The automatic generation and inclusion of this JCL into operational jobs enables changes to be made to the data base or file environment transparently to the operational control function.

Such a facility would reduce not only the possibility of error, but also the heavy manual load otherwise required during a period of major reorganization. These elements of the JCL could be copied into the total JCL either prior to or, preferably, at execution time, thus permitting a smooth transition from one operational view to another.

#### 7.3.8 File and Program Control

Program and file descriptions should be maintained for all versions to:

- Control file generations.
- Facilitate the retrieval of backup files and programs.
- Control and relate versions of programs and versions of the data base structure.

#### 7.3.9 Generation of Data Base Definitions

All data base management systems require the structure

of a data base to be defined to them; some also require physical information to be defined. Automatic generation of these definitions from the data dictionary not only eliminates manual work, but also insures that the dictionary is consistent with the data base.

*7.4 DATA REQUIREMENTS*

The data dictionary system should provide facilities, which will enable the setting up and maintenance of descriptions of the physical files used to realize any implementation used within the data processing system. These descriptions should be available for use by:

- Data base administrators.
- Data base designers.
- Operational schedulers and controllers.
- Data handling software.

The descriptions could include the physical characteristics of the files themselves (both initial and dynamic), their "operating system address" and their contents in terms of the data structures defined in the implementation view. A method of recording any characteristics of a processing unit (job, program and so on) which are specific to one physical realization of the data base is also required. The ability to describe several physical realizations of one implementation view is highly desirable.

# 8.0 THE IMPACT OF A DATA DICTIONARY SYSTEM ON AN ENTERPRISE

*8.1 INTRODUCTION*

It has been pointed out in Chapter 2 that there is increasing awareness of the importance of data as a corporate resource and a recognition that it must be controlled as carefully as other major resources such as stock, cash, equipment, buildings and personnel.

When management has gained control of the data resource by centralizing all information about it, it can then begin to impose controls over the availability of the resource and to develop standards for its use. In this way the data resource can be managed to the best advantage for the existing information system and can be effectively redeployed to meet changing information requirements.

Individuals will lose the freedom to define and use data in their own way to satisfy their needs alone. But in return for this loss of freedom, they will have access (security procedures permitting) to accurate information about the definitions and usage of all the data contained in the corporate resource.

This chapter discusses the impact of a data dictionary system (and therefore of data management) on management and on the functions performed by those individuals most directly affected.

*8.2 MANAGEMENT*

A data dictionary system improves management's control and knowledge of the data resource by centralizing all the information about it. Management is able to assess quickly the impact of any proposed change to commonly used data and to estimate the likely cost and time-scale of any such change; during a change-over, management can be assured of completeness (because the data dictionary system will show all the programs, files and reports affected) and accuracy (because the data dictionary system can generate new coding to reflect the change).

A data dictionary system enables management to enforce data definition standards; it supplies information about the creation, usage and relationships of data; it is an aid to the elimination of unwanted data redundancy and to the elimination of errors caused by inconsistent data definition and usage; it aids the securing of sensitive data definitions against unauthorized use.

*8.3 DATA BASE ADMINISTRATION*

A data base dictionary system reduces the clerical work of the data administrator, and at the same time gives him more control over the design and use of the data base. A data dictionary system enables him to control and document the formulation, meaning and usage of data structures; it enables him to evaluate existing data redundancy and to control data redundancy in the future; it enables him to provide accurate data definitions for inclusion in programs; it enables him to provide accurate data definitions for inclusion in programs; it enables him to control the availability of sensitive data to particular users; during the implementation of a change, it enables him to control the currency of test and production versions of files and programs; it enables him to insure that management's requirements for data standards are obeyed. The introduction of a data dictionary system requires a data administrator to realize the full benefits of such a system. Conversely, a data administrator requires a data dictionary system to carry out his task effectively.

*8.4 DATA AND FUNCTIONAL ANALYSIS*

A data dictionary system aids the analysis of an enterprise's data flow by providing a method of recording the arrival or initiation, movement, processing and storage or destruction of the documents which flow through an organization.

The implications of this are considerable. Possibly for the first time, a computer system can be used at the outset to examine the clerical processing of data. This examination can pave the way for:

- Simplification of clerical procedures.
- Standardization and coordination of document design.
- Translation of existing clerical procedures to planned computer procedures (preferably via a conceptual model).

In an enterprise which already has some or all of its procedures computerized, the data dictionary system will firstly be a means of documenting these systems and their interaction with clerical procedures. Any further data and functional analysis work will be able to build on the information already available in the data dictionary system, thus avoiding both the duplication of effort and the introduction of inconsistent data definitions.

*8.5 SYSTEMS DESIGN*

A data dictionary system is used in different ways in systems analysis (or data and functional analysis) and systems design. The systems designer is able to use the data dictionary system as a central source of information. It will enable him to make use of data that is already available; it will enable him to avoid duplicating data definitions, thus preventing redundancy and inconsistency; it will enable him to generate data files to be used for system testing; it will enable him to check the contents of data files produced during systems testing and, finally, it will provide documentation of the system.

*8.6 APPLICATION PROGRAM DEVELOPMENT*

A data dictionary system will impinge upon application programming to a greater or lesser extent depending upon the facilities available in the system. Certainly programming management will be able to enforce data definition standards more easily, by insisting (perhaps through job control language procedures) that all data definition coding is produced by the data dictionary system. It will be easier to control the implementation of design changes that arise during development.

For the application programmers themselves, much of the tedium of writing large amounts of coding will be removed. As data dictionary systems become more powerful, greater amounts of coding will be generated by them. Further, assistance with the generation of test data and the checking of results by the data dictionary system will reduce application development time and improve the accuracy of the finished programs.

*8.7 APPLICATION PROGRAM MAINTENANCE*

Once management has used the data dictionary system to assess the impact of a proposed change, the task of amending the applications programs can proceed with increased confidence. The data dictionary system's cross-referencing will show the maintenance programmer precisely what programs will be affected by a particular change.

According to the power of the data dictionary system, much of the program amendment may be automatic.

As programs are progressively amended, the data dictionary system can be used to record the changes, so that programming management can control the progress and completeness of the change.

Before allowing the changed programs to replace the production versions, the data dictionary system can be used to generate test data (perhaps in a different format from the current production data) and check results.

Where the data dictionary system has been used to record clerical procedures and document flow, management can insure that the change is correctly coordinated in all the affected areas of activity in the enterprise.

*8.8 OPERATIONS*

A data dictionary system can encompass the activity of the operations department.

Other departments in the enterprise will then (subject to security procedures) have access to information concerning the physical locations of data and the archives that have been stored.

The operations department will be able to maintain the privacy of data by reference to the data dictionary system to check who is allowed to use what data. The data dictionary system will aid the operations department in the creation of job control language parameters; control of different versions of program libraries and data files; distribution of (multiple) copies of output; discovering authorization to run particular jobs; discovering the source of (invalid) data.

*8.9 SUMMARY*

From the above, it is apparent that the introduction of a data dictionary system will have an effect at many levels in an enterprise. It will create new tasks, but in return it will reduce the effort required by such activities as documentation, coding of programs, creation of test data files, checking and auditing of output files.

The data dictionary system will enable management to control data processing at all levels—from the top (conceptual) level to the bottom (implementation and physical) level—and will provide an effective means of communicating data processing requirements between user departments, clerical departments and computer departments.

Data will be recognized as a resource of immense corporate importance and value, and it will be managed and controlled accordingly.

## REFERENCES AND BIBLIOGRAPHY


1. Plagman, B.K., Altshuber, G. A Data Dictionary/Directory Design Within the Context of an Integrated Corporate Data Base. March 1973.
2. Martin, G.N. Data Dictionary/Directory System. *Journal of Systems Management.* December 1973.
3. Collard, A.F. A Data Dictionary/Directory. *Journal of Systems Management.* June 1974.
4. Curtice, R.M. Some Tools for Data Base Development. *Datamation.* July 1974.
5. The Data Dictionary/Directory Function. *EDP Analyser.* November 1974.
6. Keen, Setzer. Use of Data Dictionary for IMS Change Control. UCC 10, November 1973.
7. Data Base Management. Proc IFIP Conference on Data Base Management. April 1975. North Holland.
8. Codd, E.F. Normalized Data Base Structure: A Brief Tutorial. IBM Research Report, San Jose, California.
9. Lindgreen, P. Basic Operations on Information as a Basis for Data Base Design. Information Processing 1974. North Holland.
10. Senko, M.E. Concepts of a Data Independent Accessing Model. ACM SIGFIDET Workshop, 1972.
11. CODASYL Development Committee. An Information Algebra. CACM, April 1972, p. 190–204.
12. Codd, E.F. A Relational Model of Data for Large Shared Data Banks. CACM, June 1970, p. 377–387.
13. Bubenko, J. Jr. Notes on Data Structures, Data Identification and Retrieval. CADIS Working Paper No. 20, Royal Technical University, Stockholm, March 1970.
14. Bubenko, J.JR. B-Facts, Objects and Processes. Elements of Logical Level Information System Description. CADIS Working Paper No. 57, Royal Technical University, Stockholm, March 1972.
15. Namian, P. Algebra of Management Information. IFIP-1968, North Holland.
16. Lindgreen, P. Some Fundamentals of Information Structures. First Scandinavian Workshop on Computer-Aided Analyses and Design. Student-litteratur, Lund 1971, p. 67–71.
17. Lindgreen, P. The Development of a Computerized Tool for System Design Based on the Qualitative Information Theory. Approaches to Systems Design, NCC, Manchester 1972, p. 63–84.
18. Nerad, R.A. Data Dictionaries for Data Management. Infotech State of the Art Report on Computer Economics, 1973.
19. Peck, P.P. Data Base Standardization and Documentation. On-line Conference on Data Base Design, September 1974.
20. ANSI/X3/SPARC DBMS Study Group, Interim Report, February 1975.
21. Wolfendale, C.C. A Data Description System for the Embodiment of Data Description Languages. ICL.
22. GUIDE DDWG Minutes and Report.