NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A DICTIONARY/DIRECTORY SYSTEM (DDS)
FOR THE SPLICE SYSTEM

by

Vassilios Panagiaris

June 1984

Thesis Advisor: Norman F. Schneidewind

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3 RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* A Dictionary/Directory System (DDS) For the SPLICE System		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis June 1984 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Vassilios Panagiaris		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE June 1984 13. NUMBER OF PAGES 84
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Dictionary, Directory, SPLICE, Distributed Systems, Dictionary Interfaces

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

As a result of growing demands for automated data processing at the Navy Stock Points and Inventory Control Points, long range plans are being developed around the Stock Point Logistics Interface Communications Environment (SPLICE) concept. Problems and opportunities are involved with designing and using distributed systems. This thesis investigates the area of data dictionary/ directory systems with special focus on distributed systems and attempts to outline the benefits for the SPLICE system (Cont)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

ABSTRACT (Continued)

from the use of a data dictionary/directory system. Interface considerations between data dictionary/directory system (DDS) and neighboring modules are also discussed.

Approved for public release; distribution unlimited.

A Dictionary/Directory System (DDS) for the SPLICE System

by

Vassilios Panagiaris
Commander Hellenic Navy
B.S., Hellenic Naval Academy, 1966

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1984

ABSTRACT

As a result of growing demands for Automated Data Processing at the Navy Stock Points and Inventory Control Points, long range plans are being developed around the Stock Point Logistics Interface Communications Environment (SPLICE) concept. Problems and opportunities are involved with designing and using distributed systems. This thesis investigates the area of data dictionary/directory systems with special focus on distributed systems and attempts to outline the benefits for the SPLICE system from the use of a data dictionary/directory system. Interface considerations between data dictionary/directory system (DDS) and neighboring modules are also discussed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. SPLICE AND DATA DICTIONARY

The SPLICE (Stock Point Logistics Integrated Communication Environment) concept comes as a result of the always growing demands of the U.S Navy for automated data processing [Ref. 1] and inventory control at various points. A design and implementation strategy is necessary based in distributed architecture for a local area network (LAN).

SPLICE is designed to increase ADP facilities of the existing Navy stock point and inventory control point. Because the current Uniform Automated Data Processing System-Stock Points cannot support the growing requirements for automated data processing (ADP) without a total redesign, an effort has been undertaken to improve the system in the short and long term [Ref. 1]. Two major objectives are behind the SPLICE development:

1. To increase CRT display terminals so users can access interactively the system's data base.

2. To standardize the various current interfaces across the 62 supply sites.

The design approach first starts from the designing of the logical or virtual Local Area Network (LAN), by specifying all the functional modules, their characteristics, and the communication protocols without focusing on the hardware characteristics. A later phase of the SPLICE project will anticipate the mapping of the virtual LAN requirements onto a physical local network.

The following functional modules are involved in the development of the system.

- Local communications (LC)

- National communications (NC)

- Front-End processing (FEP)

- Terminal management (TM)

- Data base management (DBM)

- Session services (SS)

- Peripheral management (PM)

- Resource allocation (RA)

This IAN design provides for distributed control but does not provide for the distribution of data bases within a IAN. The data bases of the SPIICE system are geographically distributed over a wide area and for the purpose of maintaining the integrity of the system, the data base functions are centralized within each IAN. A DBMS module for the system must at least provide dictionary, integrity, recovery, query language, and security features as well as compatibility with existing COBOL programs.

The functions of the DBM module would be:

- Catalog, to maintain a catalog of file names and status (name, open or closed, size, physical address of file, physical address of index, application used in, date entered into system, expiration date if any, location of backup copy, format, access restrictions).

- Operations, under a menu selection scheme to perform various functions (retrieve and display a record, update specified fields of a record, delete a record, insert a record, print a file, print a record or specified fields of

a record, answer specified queries and display and print the results).

- Dictionary for defining and characterizing the data elements. The dictionary must be integrated with the DBMS. This will contribute to data integrity and consistency throughout the system and should also be of great assistance in designing report formats.

With this improved design it is believed that the SPLICE system will provide economical and responsive support capabilities among the 62 different geographical locations, each having a different mix of application and terminal requirements.

The SPLICE functional design approach suggests developing several functional modules, distributed in minicomputers throughout the LAN with the necessary communications to support them [Ref. 2]. This design provides for higher system availability than the centralized approach since functional modules can be moved from one physical node to another without changing their logical addresses [Ref. 3]. At the time there exist no exact methods for designing distributed systems and so an objective of the NPS research program for SPLICE is to advance knowledge about distributed systems and to increase understanding of how distributed systems must be designed in order to operate effectively.

Distributed systems have problems associated with their design that need solutions in particular areas [Ref. 4 pp 2]. The distributed system must provide the ability for the user to communicate and access information across the 62 local networks interconnected by the Defense Data Network (DDN). It must be possible for the user at Naval Supply Center (NSC) Oakland to access the Inventory Control Point (ICP) database at Mechanicsburg in the same way as the local database at Oakland [Ref. 4].

The data dictionary must provide support to the above by uniquely naming and identifying objects in the overall SPLICE system. In the case of a message which is destined to another local network, the dictionary can be used to obtain the physical destination address with the help of

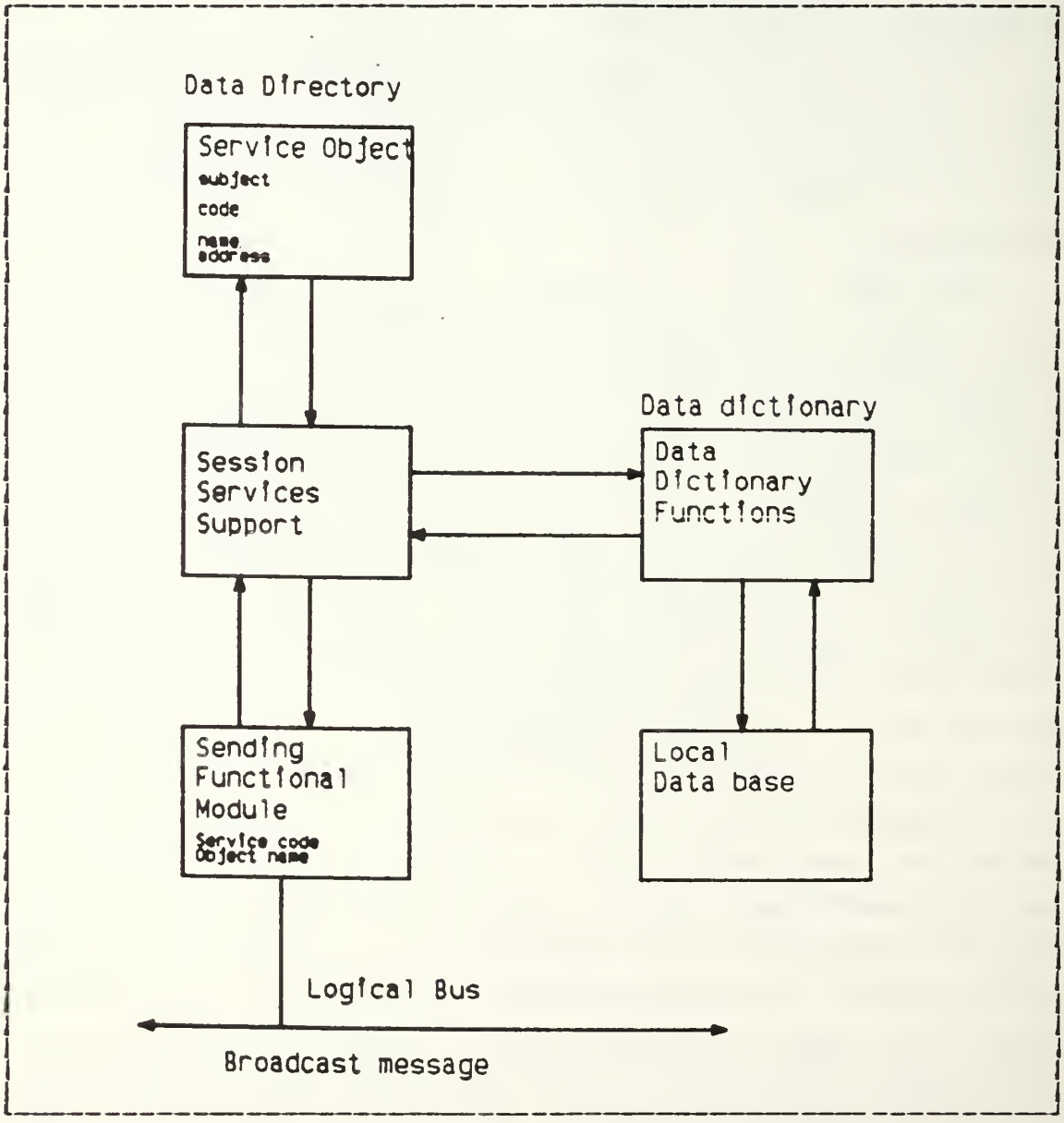

Figure 1.1 Network Services Directory and Dictionary.

Session Services module (Figure 1.1) . For object naming and addressing and software maintenance, the data dictionary can help by storing all the name-to-address mapping and routing information. The data dictionary can also be used to specify task requirements for the user terminal processes. The data dictionary in a distributed environment will cooperate closely with the session services module which provides assistance to the user terminal processes in carrying out their tasks. Thus a distributed operating system must provide, in addition to other functions, the ability to access effectively the dictionary/directory system (Figure 1.2 from Ref. 4) .

Major systems of the SPIICE application environment are the Integrated Disbursement and Accounting (IDA), Automated Procurement and Date Entry (APADE), Uniform Automated Data Processing System-Stock Points (UADPS-SP), and Logistics Data System Trident IDS. Each of the above systems has its own elements, files, programs, transactions, users and reports [Ref. 4].

It is vital for the system to manage all the resources efficiently and the distributed environment makes this job more difficult. A data dictionary/directory system (DDS) seems to be one approach to data design and managing problem solution. For the centralized database environment three aspects are emphasized [Ref. 5].

-The software interfaces between the D/D system and other software packages

-The convert functions of the D/D system

-The environmental dependency between the D/D system and a database management system (DBMS).

For the distributed database environment, as in the case of SPIICE, there must be extensions to the centralized D/D,

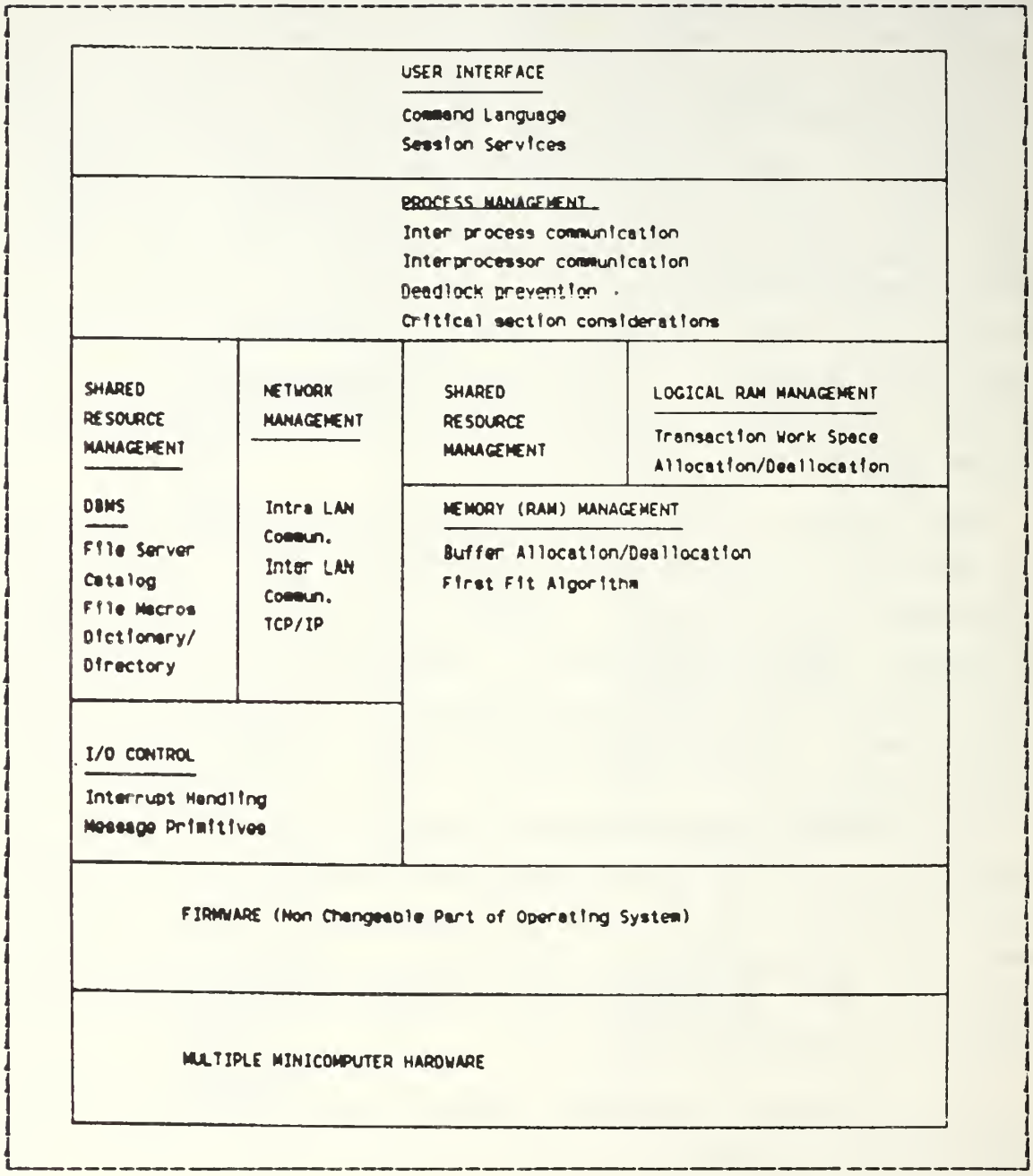

Figure 1.2 Layered Operating System Design (Ref. 4).

additional software interfaces required, and the use of the D/D as a distributed database.

E. OBJECTIVES OF THESIS

The SPLICE project at the Naval Postgraduate School (NPS) takes the approach of designing the logical or virtual local Area Network (LAN) first, specifying all the functional modules, their characteristics and the communication protocols, rather than focusing on the hardware characteristics of LAN first [Ref. 1] developing alternatives for SPLICE local Area Networks. After providing a functional specification for a distributed operating system, user interface specifications are provided, where the dictionary/directory system (DDS) constitutes a major component [Ref. 4] and its function is to provide support for naming and identifying objects in SPLICE.

The objectives of this thesis are to investigate the area of data dictionary/directory systems (DDS), to outline the advantages/disadvantages of these systems, and to present the underlying ideas. Also, to pay special attention to the distributed environment, and to introduce the benefits for the SPLICE system from using a dictionary/directory system. Finally an attempt will be made to introduce the interface requirements between a data dictionary/directory system for the SPLICE, and the neighboring modules.

II. DICTIONARY/DIRECTORY SYSTEMS

A. GENERAL REVIEW

A data dictionary is a description of data resources. It contains both machine-readable and human-readable descriptions of the database tables, their attributes, interrelationships, and semantics. It is usually not very large, but it has a very rich structure. Most systems have a data dictionary facility which stores metadata about the database aside from the database itself. The data dictionary is often built on top of the DBMS as a special application with a special data definition language.

Thus a DDS is a set of one or more databases containing data about an organization's information resources. These resources can be retrieved and analyzed using standard database management system (DBMS) capabilities. The concept of a data dictionary system has existed in the data processing industry for a number of years. Use of such a system consists, basically, of an attempt to capture and store in a central location definitions of data and other entries of interest [Ref. 6]. The principles of such a system are:

-Provide for better data control

-Provide for better documentation

-Improve the quality of the systems that are built in terms of user functionality and satisfaction and system maintainability.

The data dictionary helps to capture and document data elements, their definitions and some of their descriptive

attributes. It also provides for logical grouping of data elements during the process of gathering requirements to build a new system. The data element dictionary provides the vocabulary that can be used between the systems analyst and the end-user [Ref. 6].

Next in the spectrum of usage the DDS help is twofold. First if the data dictionary is available it can be extended to include information of how and by whom the data elements can be used. Thus a dictionary can be used to store the definitions of data elements and the definitions of other data constructs (records, files), the definitions of processes (programs or manual processes), and definitions of data users (individuals, organizations). The Second trend that contributed to this extended usage of a dictionary system was the gradual migration away from the use of traditional files toward the concept of a central, integrated database distributed across the DDN but centralized within each LAN, under the control of a database management system.

The problem of duplication of data (data redundancy) can be solved inside each LAN but another mechanism must be provided in order to solve that problem across the DDN. This problem must be examined carefully and that mechanism must provide for economy because sometimes data redundancy may be more cost-efficient than the frequent use of DDN.

The above is vital for system design because in the SPLICE environment, data are to be shared not only by different systems, but also by a wide range of users. The basic concept of a DBMS is to provide a centrally located set of definitions of data within each LAN that is to be shared in order to assure that different users will access common data with a set of consistent definitions.

The DDS acts as a repository of all definitive information about the database such as characteristics, relationships, and access authorizations. These databases, as

implied by the term 'logically', can be physically stored in diverse locations within each LAN but are logically linked via communications and the DDS.

The data dictionary system located in a node within each LAN can be used to provide the above definitions and thus the required data consistency.

Separating the data dictionary from the database raises two problems [Ref. 7].

-The dictionary and data base may disagree with one another unless one interface has control of both functions

-Having a separate data dictionary implies having a separate language for the definition and manipulation of the dictionary database.

Users who define tables and other objects (case of system-R) are encouraged to include English text to describe the meanings of the objects. Later other users can retrieve attribute tables with certain attributes or can browse among the descriptions of defined tables, if they are so authorized. A user later can modify these entries to change the attributes of an object.

E. MANAGEMENT OF INFORMATION RESOURCES

Information resourse management (IRM) is a methodology that attempts to solve a set of problems related to the system life cycle in an integrated and coordinated manner. The data dictionary system will play an important role in this area.

In the case of SPIICE the DDS can play an important role in providing a documented inventory of information resources, a control mechanism for the analysis and design of new information resources and the necessary resource independence.

A data dictionary can be used as a powerful tool (not as a solution) that can aid in the solution to various problems such as the inventory control, report production, proper routing of data, proper routing of requests, data consistency, security, etc.

Finally the dictionary system project is in fact an Information Resourse Management (IRM)[1] project. The SPLICE system possesses much valuable data that has been generated, collected, and stored in an automatic and 'formated' state. Utilization of any class of data involves one or more processes. These are [Ref. 6]

- Collection: It is a process that tends to be expensive as the cost of identification and recording (including input to an automated system, as necessary) can be high.

- Processing: The data collected is generally 'managed' in some fashion before and/or after being stored. In the case of automated data, this occurs through the use of computer programs.

- Storage: The repository of data and information termed a "data base".

- Retrieval: Using the knowledge about the storage technique being used, data are retrieved to answer questions or to be modified.

- Communications: A communication line is needed to connect the user terminal with the place where the dictionary resides.

---

[1]Information Resource Management is whatever policy, action, or procedure concerning information (both automated and non-automated) which management establishes that serves the overall current and future needs of the system. Such policies, etc. would include considerations of availability, timeliness, accuracy, integrity, privacy, security, auditability, ownership, use, and cost effectiveness [Ref. 6].

The environment in which the above processes take place is composed of :

- Data and information. Represents the core of the entire information processing spectrum.

- The users in the system. It is the personnel involved with the system. These are users of data and other information components.

- Physical facilities. Computer hardware and other physical devices used in data processing.

- Processing facilities. These are all the activities which take place in the use of physical facilities.

- Support facilities. All the services which are required by users of data as well as personnel whose responsibilities are primarily in the information systems area.

Each of the above components is refered as an Information Resource and the computer system must provide for an integrated and coordinated manner to manage the entire information resource of the SPLICE system and the data dictionary has to play a major role in conjunction with the database management module.

C. SUPPORT OF SYSTEM LIFE CYCLE

In this section, we present some highlights of how the data dictionary supports the main steps of system development.

The waterfall model of the software life cycle [Ref. 14] consists of the following stages: system feasibility, requirements specification, product design, detail design, coding, integration, implementation, operations and maintenance. Of course there are also other models of a software life cycle but basically the functions of a DDS are the same in whatever model we consider.

During the system's feasibility stage the DDS can be used for new data element collection and to avoid redundancies and inconsistencies. Also the DDS can contain a description of processes that are already available and to help in assessing the true magnitude of the proposed task. During the requirements specification stage, the data dictionary can provide the means to detect existing inaccuracies in definitions and to correct them before the system operation. This is because the DDS contains the overall scope of the requirements to be specified.

During the product design and detail design stages, the DDS can help because it contains the design details of both data and processes, which can be shared by all members of the design team. Particularily in database design the DDS can record multiple user views, pass output from the logical design phase to physical design phase, generate multiple designs for benchmark testing, and verify the existing conversions of data in the system. For the rest of the stages the DDS can help in data collection, coding, and testing, by providing any desired degree of coordination and control over tasks, generating data structures, storing instructions for the staff, describing the various jobs and activities, and finally, providing a means for effective and consistent modification of the system.

Additional benefits that can be derived from the DDS [Ref. 6] are naming standards, aid to auditing, interfaces to application program development tools, and software configuration management. A DDS allows a system to be extended trough the addition of new entity types, relationship types, attribute types, and also can be used to add configuration entity types such as requirements specifications, change notices, etc. The major advantage from the use of the DDS is in the case of an active system where the system not only records the entities, but also controls how they are revised.

I. DATA DICTIONARY-SYSTEM ORGANIZATION

The organizational structure for a DDS that is to be adopted must be commensurate with the size of the activity at any one time. Such a structure is displayed in Figure

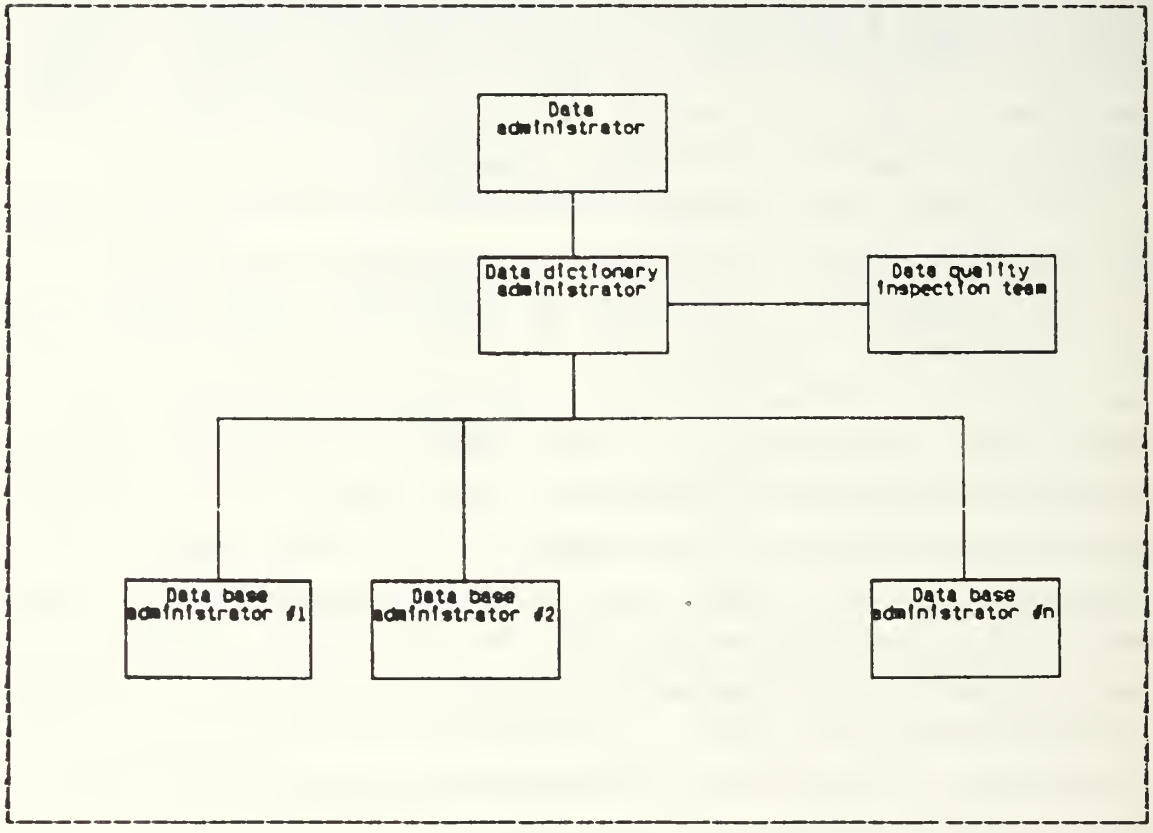

Figure 2.1 SPLICE Data Admin. Function Organization.

2.1 .

The Data Administrator is the person responsible for articulating the data policy after the major guidelines have been laid down by the designing team. That policy includes planning for data collection, its structuring, its storage, and its quality control. For the SPLICE system the Data Administrator can be a person or a team located in any place, whose main function will be the setting of the above policy.

The Dictionary Administrator who the person or team responsible for the dictionary system within the Data Administrator function (eg. recording of all meta- information and meta-data and its maintenance through the use of the dictionary system, along with making its facilities available to the users of this system). Because in the SPLICE system the data dictionary is unique through all the system and no different views of the data dictionary are permitted in the various locations, that team or person must be unique through the system. Only that team (or person) must have the priviledge to maintain the DD. The Database Administrator who the person (or team) responsible for the technical aspects of obtaining, running and maintaining the DBMS. Since SPLICE is a distributed system with databases distributed across 62 different locations, the Database Administrator does not need to be unique. The required policy and definitions are setup by the data dictionary administrator and this is enough to maintain consistency through the whole system. The Data Quality Inspection team has a role also in the hierarchy, and its function is the quality inspection of the information or data, and the quality audit trail of the whole system. This can be one or more teams. In the case of several teams the entire audit effort can be divided among them.

E. CONCEPTS ON DDS SELECTION AND EVALUATION

It is very difficult to find a commercialy available DDS to meet exactly the requirements of a system under development. A selection and evaluation process composed of various steps must be developed in order to select the best system.

Four steps are proposed by [Ref. 6] for the process of selection and evaluation of a DDS:

-Determine the requirements for the dictionary system. These should be classified as either being mandatory or not. If not mandatory establish a scale and assign numbers indicating the importance.

-Develop a list of features of dictionary systems that will be used in the evaluation of systems.

-Determine a mapping from the needs onto these features.

-For each mapping, using descriptions of available systems, a system can be found either to qualify or not. This process leads to eliminate systems that are not qualify.

We cannot say that the above procedure is perfect and does not have a risk for mistakes, because it is subjective and variously depends on the experience and smartness of the selection/evaluation team. Some more common/general reasons leading to mistakes are: The needs were never properly assessed, and potential users were not asked the right questions, unnecessary but apparently "nice" features were given high values, the evaluation of the system was inconsistent because different people evaluate different systems without a well-defined measurement method, undue emphasis was placed on features that will be needed in the future but unimportant now, etc.

For the SPLICE system we cannot follow the above procedure. SPLICE has decided to use Tandem as their "front end" minicomputer. As a result, selecting a DDS is largely a foregone conclusion in this situation. So we have to use Tandem IBMS and the associated dictionary capabilities.

F. ADDITIONAL ASPECTS OF DDS

In the next few years, several extensions to dictionary systems, not available today, will most likely be commercially available. These additions will allow dictionaries

to be more effective in interfacing with the information resources. The use of extensibility facilities allows an installation to customize the dictionary system in order to make it effective in such applications. Such examples are the use of DDS to control the total information resource, to aid in the analysis, design and development of information systems, and to aid in efficient database design. The last application example is the use of DDS as a repository of information for an entire system. This is exactly the major role the DDS has to play in the SPLICE system.

Referring to the SPLICE application environment the DDS would require users and analysts to define the system data elements, files, etc. which would entail updating old definitions, discarding outdated ones, and introducing new ones. In this way standards of data definition and description for application programs can be established over the entire SPLICE system [Ref. 4]. But on the other hand it is a Herculean task to retrofit a dictionary to existing application systems. Because of the many above mentioned difficulties in implementing the dictionary to old application systems, we recommend as much more preferable to implement a dictionary for new applications only. That means that the dictionary will be developed gradualy and a long period will be needed to be fully implemented for the whole SPLICE system.

Although DDSs have many advantages, their disadvantages should be mentioned as well. Dictionary systems are complex software systems and the execution of many dictionary functions may consume a significant part of the system resources. As the scope of the dictionary is enlarged to include always larger number of information resources, the DDS will begin gradually to look like the major resource consumer, and thus the main user of the host computer system [Ref. 6]. When we consider active interfaces of the DDS,

the previous problem becomes more serious. If the DDS controls a process through one of these active interfaces, it follows that this process cannot proceed until such time as the dictionary system has finished its job. This delay time is added to the whole process time. Given that there can be many processes, the continuous use of the DDS and the accumulated service time may eventually result in a bottleneck.

The proposed solution for the SPLICE system [Ref. 4] can avoid (or at least reduce) this overhead by locating one copy of the DDS in each LAN. With this simple and efficient technique each user located in any of the 62 stock and inventory control points only needs to consult the local DDS. The number of users who needs the DDS services remains the same but the overhead from the long queuing time across the IDN will be reduced by a factor close to 62. By locating the master copy of the DDS in one place we can solve the maintenance problem of the DDS, because additions, deletions and updates of the DDS can be done only via the master copy by the Dictionary Administrator. All the other copies can be updated only remotely by the master copy through the DDN, in such a way as to represent the exact image of the master copy. Because changes in definitions (deletions, updates, additions) are not frequent, we estimate that the whole process of updating the local copies of the DDS will not be expensive, and the resultant overhead will not be significant. Of course this assumes all 62 LAN's are working off the same schema, and the application environment is homogeneous across the network.

G. HIERARCHY OF DDS

A good hierarchical DDS structure is significant if we want to avoid the "bottleneck" mentioned above. A structure

is proposed in Figure 2.2 and we believe that it is less expensive in consuming the system resources than the structure of having different views of the master dictionary at

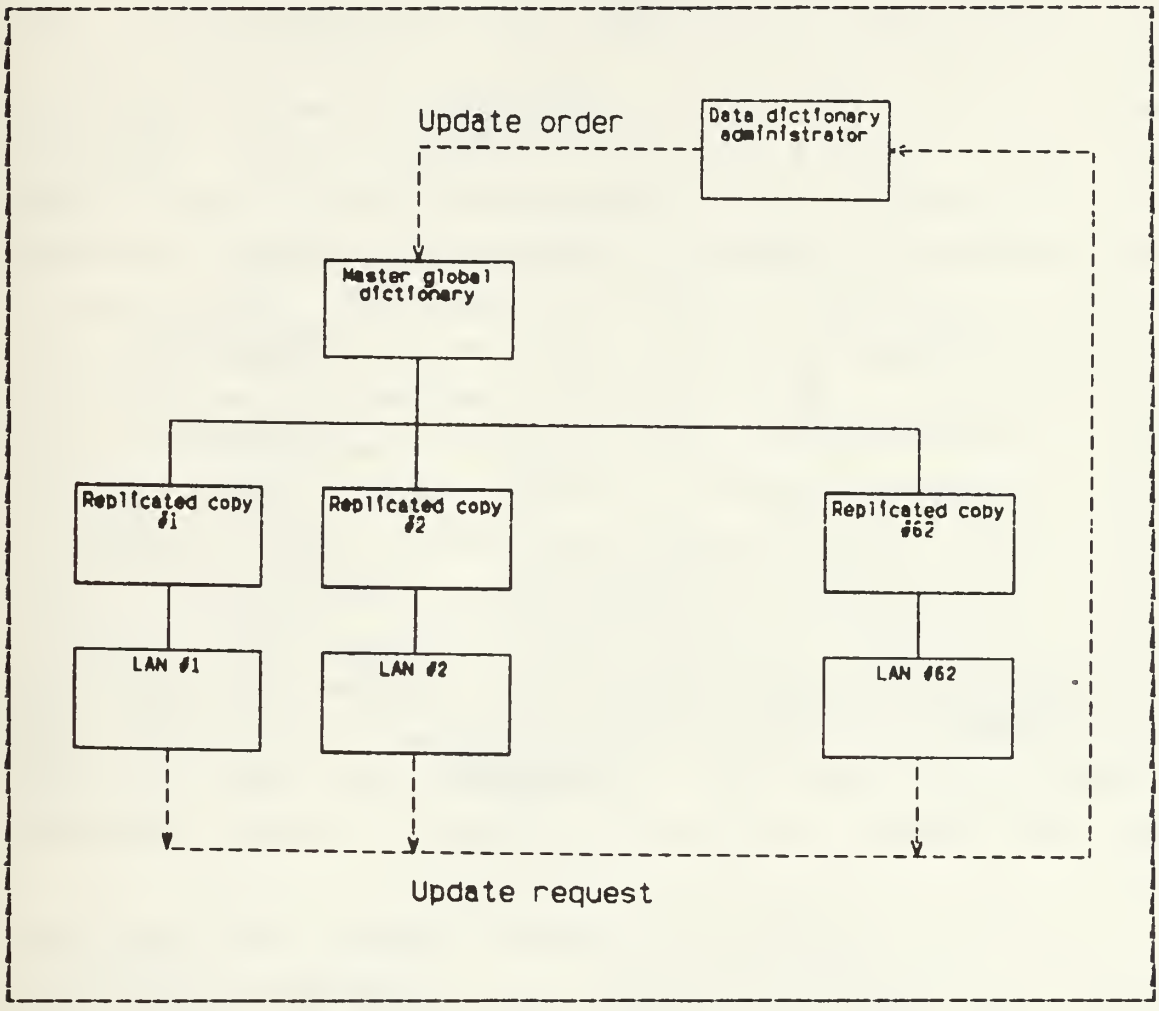

Figure 2.2 A First DDS Hierarchical Structure for SPIICE.

each LAN. In particular suppose the copies of the local dictionaries are not exact images of the master dictionary, but are different views of the master, especially views containing information only for the local database. In such a case it is not useful to separate the definitions from the actual database since the different views of the whole

database are centralized within each LAN. If a spare part for example cannot be found in a local database, then the user has to consult the master dictionary to find the location of the requested spare part because the local copy of the data dictionary does not contain information about other data bases of the system. In this case the user has to access the DDN twice, first to consult the master dictionary and then to consult the local database in which the spare part is located. This procedure can easily lead to long waiting times and finally to "bottleneck" because the master dictionary will have to answer in questions coming from 62 different LAN's. A second hierarchical structure is shown in Figure 2.3 . This structure involves the location of a copy of DDS in selected nodes instead of each node. By this way we reduse the amount of secondary memory needed to store the D/D but we increase the use of DDN. This increase in use of DDN is inversely proportional to the number of D/D replicated copies. The solution of locating exact copies of the master dictionary in each or selected LAN's has the disadvantage of consuming more secondary storage but our estimation is that this is preferable and less expensive than the freguent use of DDN in order to consult the master copy.

We cannot say that distribution instead of replication of DDS is an inefficient method not acceptable for SPIICE. Since there is not enough experience for distributed systems, and especially for data dictionaries, we have to examine carefully every possible architecture, the pros and the cons of each one, in order to make the best decision. But still we believe that the decision will be based more on estimations comming from intuition and less in experience and statistical information. Such an architecture is based on distribution instead of replication of D/D for SPIICE. This is shown in figure 2.4, and will be examined in a next chapter.

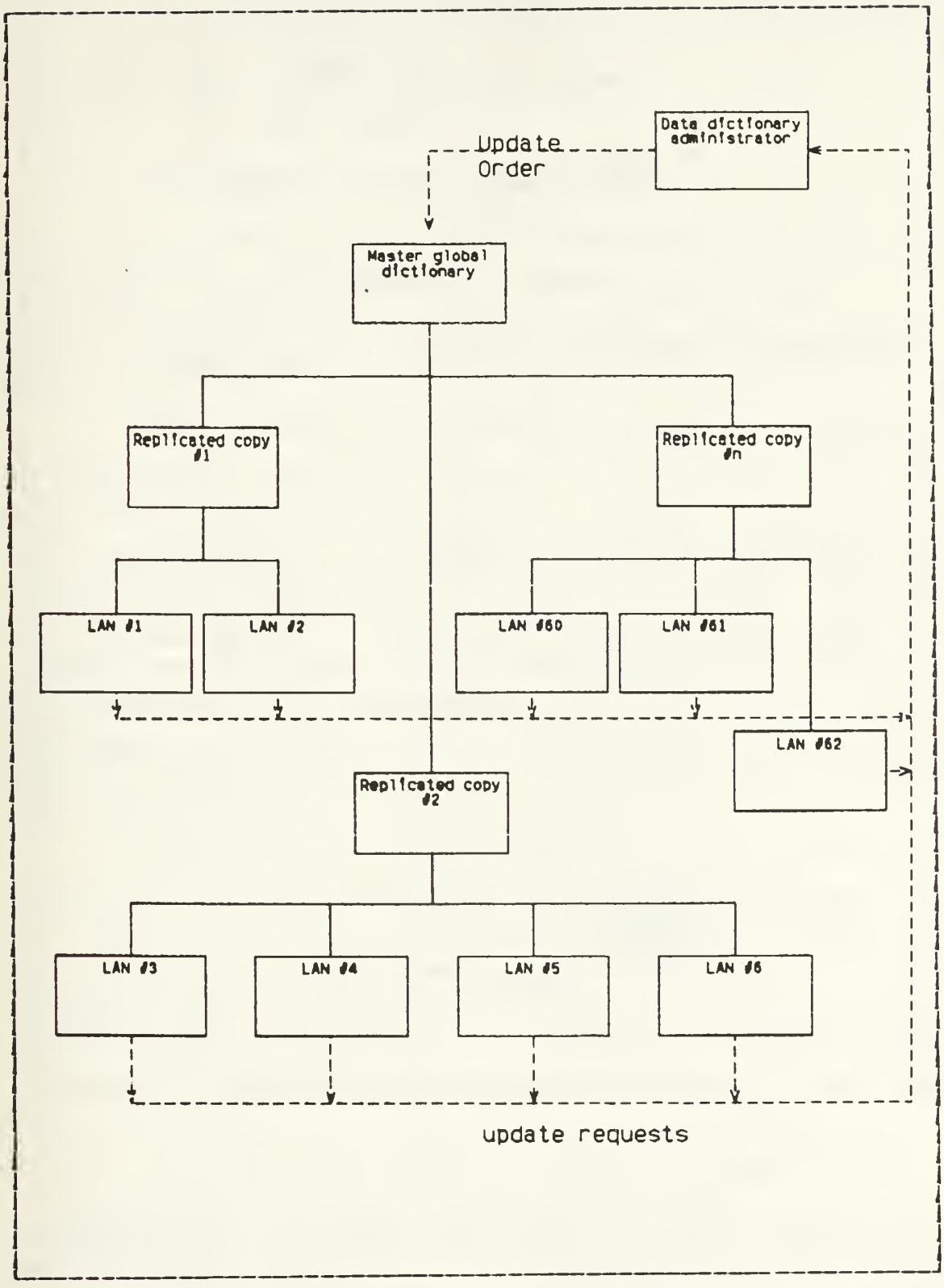

Figure 2.3 A Second DDS Hierarchical Structure for SPIICE.

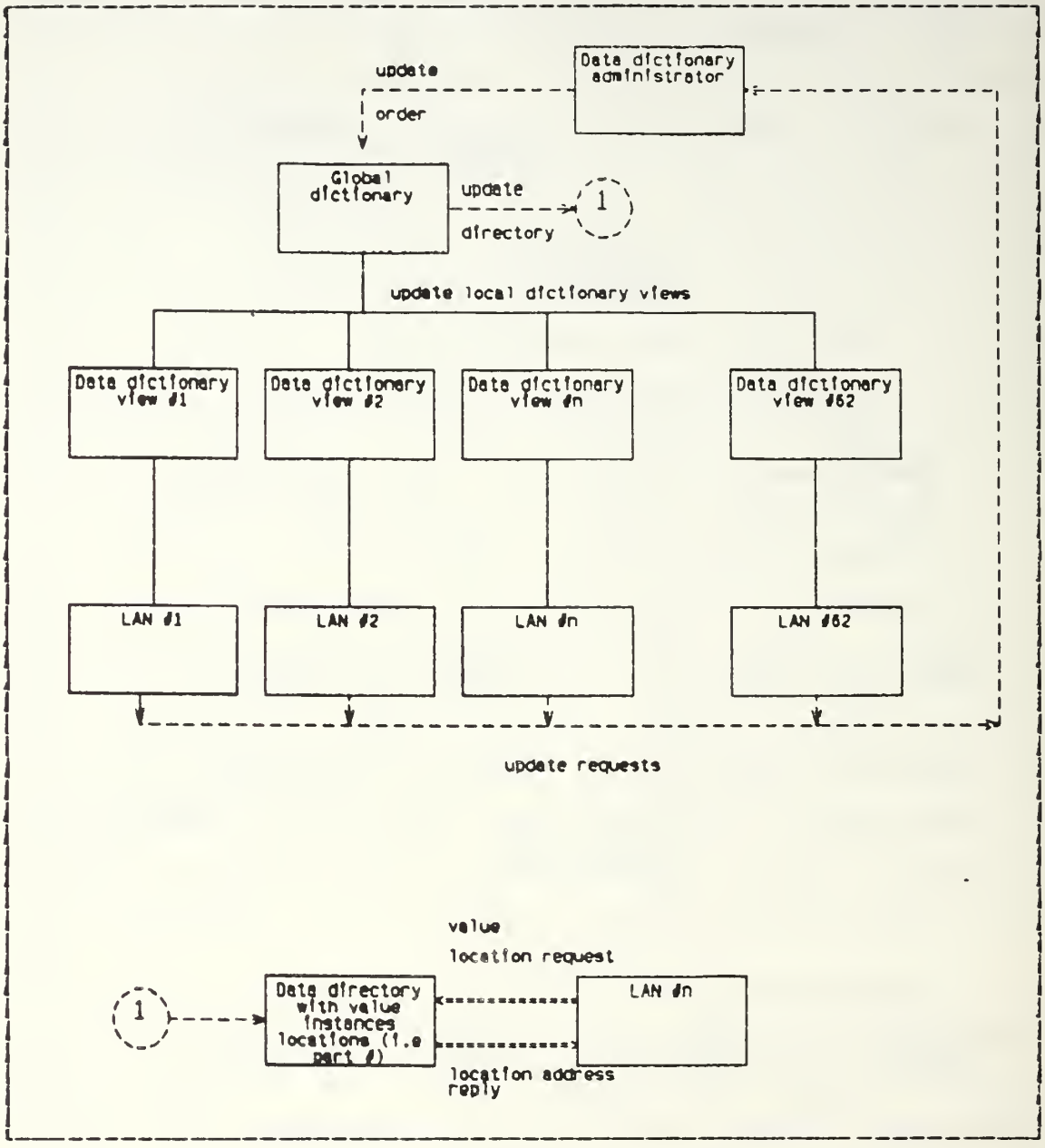

Figure 2.4 A Third DDS Hierarchical Structure for SPIICE.

B. FEATURE ANALYSIS OF DDS

In this section the features of DDS and a more detailed analysis of them will be presented. This presentation is a theoretical approach and does not concern any particular

system. A cost/benefit analysis can tell us which features need to be included in a DDS under development. It is more preferable approach than to develop a DDS as described below using the Tandem DBMS capability.

1. Architecture and Implementation

The relationship between DDS and DBMS will be addressed here. The purpose of a DBMS is to manage data and the purpose of DDS is to manage meta-data.[2] The question is whether the DDS must be a free-standing[3] or DBMS-dependent[4] system [Ref. 6].

The free-standing approach is good for commercial systems because each enterprise can evaluate the pros and cons and reach the optimal decisions whether to buy or not. This approach raises compatibility problems between the DDS and the DBMS, especially when the vendors are different companies. There are many factors we have to take into account when deciding whether a DDS must be free-standing or DBMS-dependent. These factors include the method of implementation, the scope of usage, whether the DDS and DBMS are going to be developed together or not, and whether they are going to be supplied by the same vendor or not.

One other feature of DDS architectural structure is whether the DDS should be passive or active. Suppose there is a compiler, application program, or other process that requires meta-data for its execution. There should be DDS available which produces automatically the required meta-data. This functionality is referred to as dictionary interface and can operate in two modes: Passive where there

---

[2]Meta-data is the data that describes data

[3]A dictionary system which does not use a DBMS in its implementation

[4]A dictionary system which does use a DBMS in its implementation

exists an option of whether the process will retrieve the required meta-data (through the dictionary interface or from elsewhere) or, in the case where the process already contains the meta-data, there exists an option for the system to check whether this meta-data is the most current version in the dictionary. Here the dictionary is not in the critical path of a process. Active where the above options do not exist and the process always uses the most current meta-data in the dictionary. The dictionary here is in the critical path of the process and the process must go through the dictionary for the meta-data in order to execute properly.

A DDS can contain both kinds of interfaces. We have to keep in mind that the interfaces of the DDS system do not only concern the DDS itself, but also other modules with which the dictionary has to cooperate in order to maintain the whole system.

2. Logical Schema, Entity Types, Relationships

Dictionary schema is the term denoting the logical structure of a dictionary. Structural characteristics and contents of the dictionary schema determine the kinds of meta-data and the relationships to be established among them. Using the entity-relationship-attribute model [Ref. 6] for the dictionary, we define entities as real world objects or things about which information exists in the dictionary, attributes as properties (quantities or qualities) of the entities, and relationships as connections between entities.

In the DDS, resources such as data, hardware, software, transactions, personnel and documents may be represented, and entities, attributes, and relationships associated with these resources must also be represented. Tables I through V at the end of this chapter taken from

[Ref. 4] indicate possible data element attributes, file entity attributes, hardware entities and attributes, software entities and attributes, and document/report attributes for the SPLICE system.

Similar entities in a DDS establish entity types. Attributes can also have a degree of similarity and in this case we speak about attribute types. Finally similar considerations apply to relationships and so we have relationship types, that are relationships between entity types.

Schema descriptor: In a dictionary schema containing all existing entity-types, relationship-types, and attribute-types, any one of them can be referred to as a schema descriptor. Information existing in the schema can indicate which entity-types are members of a given relationship-type, and which attribute-types are associated with an entity-type or relationship-type.

Entity-types of a DDS can be classified as data entity-types, process entity-types and usage entity-types. On the other hand attribute types can be descriptions, classification and audit attributes created by the dictionary to indicate identification of the person who created the entity, date of entity creation, identification of the person who last modified the entity, date of latest modification, and total number of modifications of the entity [Ref. 6]. These capabilities are very useful for a system, especially one as complex as SPLICE. Using the above capabilities reports and summaries can be presented on request, and also we can have a trace of various interactions on the system using application programs for this reason.

3. Interfaces and Commands

Interfaces must be included in a DDS in order to allow the user to communicate with the DDS via a terminal. The terminal-DDS communication in the SPLICE system is

carried out through the Session Services module. This is a separate topic which will be examined separately. In general an interface can be as shown in Table VI.

On the other hand commands can be classified, on the basis of their functionality, into various categories as shown in Table VII.

A dictionary system can be regarded as a software product that helps in storing information about data that already exists in databases. Both DDS and DBMS deal with descriptions and characteristics of data elements and with the logical structures obtained from these elements and their relationships. A closely integrated dictionary system and automated database design process have much to offer. The interfaces between a dictionary and a database design process can be divided into two broad categories:

-Initial data entry and editing

-Logical model structuring

Initial data entry and editing: For data entry the data requirements information needed by automated database design procedures is almost a complete (proper) subset of the information normally stored in current commercial dictionary systems. For the SPLICE the files already exist but the dictionary does not. Therefore the whole design of IDS must provide for initial detection and avoidance of duplicate entries. As soon as the design takes care of that during the initial steps, then the entry of information about raw data elements has to be made only to the dictionary system. Next an interface must exist in order to allow the design procedures to access information in named aggregations (local views). For editing, the initial data entry is rarely clean in the sense that names, usage, and characteristics of the data elements may not yet be standardized across local views. Synonyms, homonyms and inconsistent characteristics of the same data usually result when

data requirements are gathered from different sources. The editing phases of the automated design procedures, and the reports produced therein, can serve as an input filtering function for the dictionary. When the interactive editing phases are completed, obsolete information (eg. non-standard names) can be removed from the dictionary, such that the information remaining permanently is clean and consistent. Again, as we mentioned in a previous section, this can be done only for new applications because the task of retrofiting a dictionary to existing application systems is very difficult.

Logical model structuring: The structuring procedure for initial design should be able to extract filtered, unstructured data element information in named aggregates (local views) from the dictionary such that the composite model and the derived logical designs can be generated in the normal manner.

For adding new requirements to existing designs and when processing new functions or adding new data to an existing database, the design process should be able to extract from the dictionary a description of the existing design along with the filtered unstructured data element information for that which is new. Various levels of constraints on the freedom of structuring processes can be set here in order to facilitate the whole design effort.

Once the automated design process is completed and a suitable logical design has been obtained, the results must be stored in the dictionary. Assuming the unstructured data elements are already described in the dictionary, the relationships defining segments, databases, logical relations and secondary indexes would now be stored.

TABLE I

Data Element Attributes

Type
Range
Length
Unit of measure
Usage
Language names
Repetitions
88 Levels
Key
Default value
Display format

TABLE II

File Entity Attributes

File name
Locations
Size (in bytes)
Format (seq, random, bin)
Access control
Access security protection

TABLE III

Selected Hardware Entities and Attributes

Entities

- Processing system
- Secondary storage
- Communications system
- Concentrators
- Terminals
- LAN I/O peripherals

Attributes

- Type
- Model
- Model number
- Serial number
- Mfger's number
- Source
- Features
- Description
- Docu. references
- Usage by site
- Cost
- Maintenance activity

TABLE IV

Selected Software Entities and Attributes

Entities

- Operating system
- Operational support system
- Environmental system
- Application software

Attributes

- Program-id
- Revision number
- Revision date
- Date compiled
- Type of compiler
- Patch level
- Change level
- License
- Date released
- Product number
- Source
- Features
- Documentation
- Usage
- Cost
- Maintenance activity

TABLE V

Document/Report Attributes

Name
Number
Product number
Release date
Revision number
Source
Feature
Description
Quantity
Cost

TABLE VI

Kinds of DDS Interfaces

Command language
Screen oriented interface
Fixed format batch data entry facility
Programmatic interface that allows user written applications programs to access the dictionary

TABLE VII

Command Categories for DDS

Dictionary maintenance
Report and query
Data structure interface
Extensibility
Status related
Security
Dictionary processing control
Dictionary administrator

III. INTEGRATION

A. THE PROBLEM

An active[5] data dictionary is desirable for the SPLICE system. It is also known [Ref. 8] that most dictionaries fail to meet this objective. A prerequisite to an active dictionary is a high degree of interaction between the dictionary and various other software elements such as the DBMS itself, but also including query languages, report generators, application development aids, and the like. An architecture for a centered and highly integrated DDS taken from [Ref. 8] is shown in Figure 3.1 .

The existing dictionaries today are noticeably unintegrated, and hence less than active. Such a situation is shown in Figure 3.2 (taken from [Ref. 8]) concerning the IBM DB/DC data dictionary and related software. Notice, in particular, that whereas some batch feeding of data is provided to and/or from the dictionary, there are no fewer than six places where database definition data is stored (in addition to data definitions included in actual programs) [Ref. 8]. These are :

The DB/DC dictionary itself
The DBD/PSB libraries
The CCBCI copy library
The database design aid (DBDA)
The GIS data definition tables
The application development facility (ADF), segment rules in an IMS/DC environment, or in

---

[5]Active to some degree because if it is too active we can lose efficiency

development management system (DMS) files in a CICS environment.

There is no guarantee that each of these descriptions will agree at any point in time. Other data dictionaries may have a higher degree of integration but no one is close

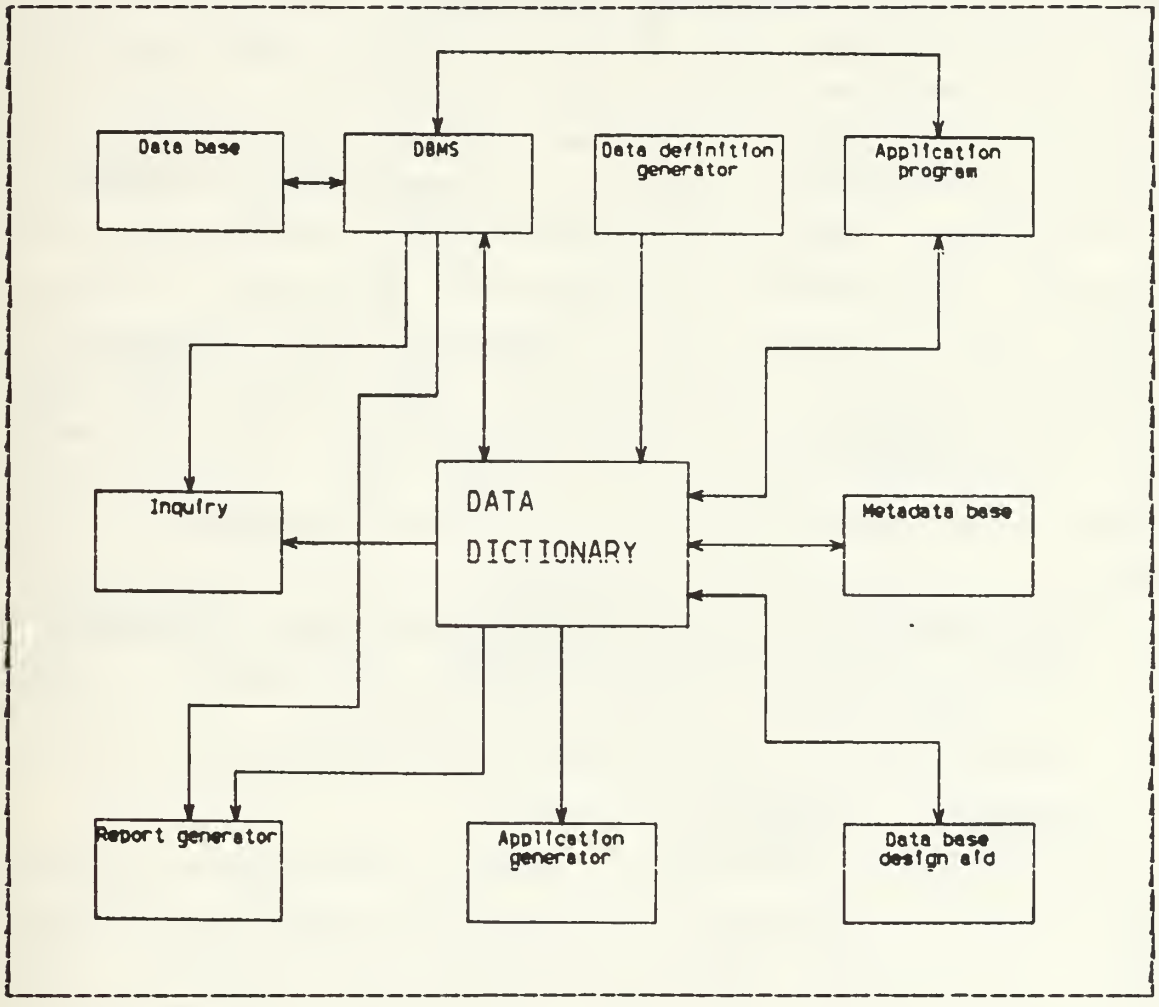

Figure 3.1 Highly Integrated D/D Centered Architecture.

to the degree of integration suggested in Figure 3.1 . A high level of integration is very much needed in order to support the advanced functions of an active dictionary. To see that better, consider a user who wants to know what data

is in the database, or a DML routine which wants to edit a field prior to updating the database, or the database access system which needs to know if a user password is valid for updating a certain record. All the above functions require direct access to the data dictionary.

The extent to which a DDS qualifies as being "integrated" is a relative notion determined by the scope of its metadata and the way that it interfaces with other software. The most common use of the term "integrated" is with reference to a D/D that is the sole source of metadata in the system. The integrated D/D is accessed for all references to meta data. Most of the commercially available DDS have reached a high degree of integration with their environments, and this results in multiple sources of descriptors within the systems. The DDS permits these systems to access the D/D indirectly and convert the metadata of each system to the format required by the D/D [Ref. 5]. So for example a DDS might communicate with a compiler in either of two ways:

- By generating file and record definitions that the compiler accepts via copy statements.
- By reading source programs and creating transactions to load the DDS with descriptions of files, records, and elements.

One additional area which demands investigation for the development of a succesful DDS concerns integrating schemas which describe the logical structures of all data types existing in a distributed (like the SPLICE) database. This feature permits the determination of a data file's logical structure as well as its identity and location, and could possibly be essential to the development of query and data model translation shemes. The existence of a master schema also permits the logical relation of data across file boundaries; then all files in the network can be considered as areas within a single large database [Ref. 9].

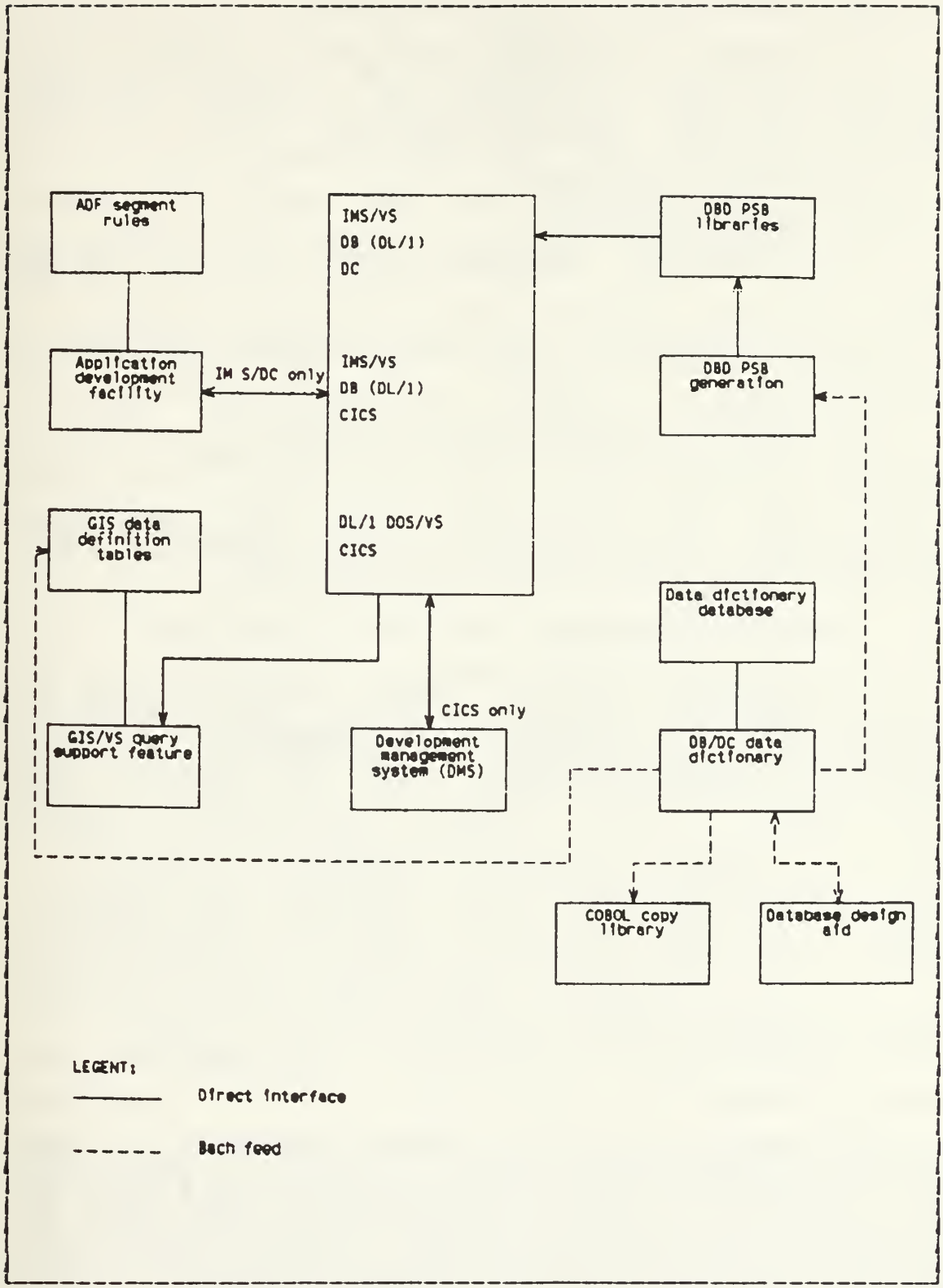

Figure 3.2 IEM Data Management Architecture.

E. INTEGRATION OF DDS

Three aspects of integrated DDS in the centralized and distributed database environment for SPLICE[6] are of great interest and must be emphasized [Ref. 5].

-The software interfaces

-The convert functions

-The environmental dependency between the DDS and the IRMS

A DDS is integrated with other software packages by facilities that:

-Allow direct and indirect access to the D/D

-Automatically capture the metadata used by other systems

In the next three subsections we will examine the three most interesting aspects of an integrated DDS.

1. Software Interfaces

A software interface permits another system to access the D/D either statically or dynamically. First we consider the static interface, which links the D/D with another system indirectly via the extraction of a file of formatted metadata. For the static interface of a DDS and a IRMS, for example, the data dictionary administrator, following the specifications of the data administrator, enters into the DDS all pertinent transactions to define the database and the database administrator using the above definitions describes the database. After reviewing the

---

[6]Our approach for the SPLICE database and data dictionary distribution is hybrid. SPLICE is a distributed system, but the databases are centralized within each LAN. Also the dictionary copies at each of the selected LAN's are exact copies of the master dictionary and different dictionary views are not permitted. So the whole SPLICE system can be viewed as a distributed system, but concerning each particular LAN, the database and data dictionary can be said to follow the centralized database environment concept. So both ideas of centralized and distributed environments can be applied to the SPLICE with slight modifications.

accuracy of this database description, a command is generated for DDS that uses this description to produce a file containing the DDL. The DBMS's DDL processor then translates this generated DDL into a schema file that the run time unit of the DBMS can access. No run-time connection between the DDS and the DBMS exists here; the DBMS's processor is not executing during the DDS's DDL-generation process.

Static interfaces differ somewhat, depending upon whether they interface the DDS with user-written programs or with vendor-supplied software packages. Static interfaces for programs written in languages such as COBOL and PL/I produce file, record, and database descriptions for the user programs from the data dictionary [Ref. 5]. These interfaces sometimes feature edit capabilities, format options, and various other functions to make the interface more flexible. Edit capabilities may include being able to add prefixes and suffixes and even to replace entire names. Format options may control indentation, level-number increments, sequence numbers, and line identifiers. Inclusion of various clauses such as comments, condition names, and initial values also may be allowed.

Static interfaces for software packages, such as DDL processors, communication monitors, and query processors, produce formatted statements for those packages or create specially encoded control files for their use.

Static interfaces are prevalent because of their utility, capability, and efficiency. With powerful static interfaces, the data administrator can quickly change formatted metadata or create new formatted definitions from existing D/D entities. The static D/D can be made compatible with many versions of other software packages and can be developed independently of the source code of particular software packages. A disadvantage to the user of a static

interface is the extra effort that may be required to generate and catalog metadata for the D/D.

More significantly, the static interface itself has no capabilities for updating the metadata of the systems with which it interfaces. Without adequate synchronization and controls, the metadata in the DDS and the metadata in other systems may become inconsistent [Ref. 5].

Dynamic interfaces provide direct access by the DDS to other software modules. This direct access is commonly achieved via high-level interface commands that shield the software package from the physical details of the D/D. The commands activate standard DDS functions, so as to select all entity occurrences that satisfy a particular condition. A DDS can provide a facility that makes commands available through call statements; any program can then access the D/D without knowledge of its physical structure. Dynamic interfaces provide consistency control and capabilities for both update and retrieval. Changes to the D/D are automatically reflected in the next execution of any software packages to which the D/D is interfaced; no intervening procedures are required as with static interfaces. A software package can directly retrieve and update metadata stored in the D/D if the user has the authority to do so, and the software package has a such capability. Otherwise the software package and the user would only have read authority to the D/D.

Here is where special attention must be given when designing a DDS for the SPLICE. We said previously, when we described the first and the second hierarchical structure for SPLICE, that the local copies of the SPLICE DDS will be exact images of the master copy. With this approach one can imagine what will happen if one program in any of the 62 LAN's attempts to update the metadata stored in the DDS. The whole consistency of the system is gone. The local

copies will no longer be exact images of the master copy and many problems can arise. The only solution for the proposed architecture for SPLICE DDS is that requests for update, deletion, or addition of data definitions must be routed via the LAN to the node where the master copy of the DDS resides. Then the data dictionary administrator, who is the only person responsible for DDS maintenance, can approve and make the requested changes in the master copy. These changes must then be transmitted to the various locations where copies of D/D reside and executed. This we believe is the only procedure under the proposed DDS architecture which can maintain consistency over the whole SPLICE system. We cannot say that this kind of operation is purely dynamic, but neither is it static. We might call it is a hybrid interface function wherein the security and validity checks of the DIS are always applied.

The use of dynamic interfaces incurs significant overhead due to the size and complex structure of DIS. Application development support aids, such as preprocessors, source program managers, and design aids generally can afford this overhead because response time is not critical. On the other hand, efficiency is critical for transaction-processing systems that reference the D/D.

To reduce the potential overhead, common queries may be precompiled and stored in the D/D. Another technique used to reduce overhead is for the software package to retrieve all the metadata required for a transaction at once; thus future accesses for this transaction only involve memory lookup. Table VIII from [Ref. 5] shows some typical types of software packages interfaces for DDS.

2. Convert Functions

In addition to software interfaces the integration of a DIS into its environment is provided by convert

functions. A DDS organization has a lot of programs, report and files to manage. The data/data dictionary administrator must encode thousands of maintenance transactions to capture the metadata of all these applications. The convert functions of a DDS scan source programs, database descriptions, and teleprocessing environment descriptions and automatically produce maintenance transactions, thus sparing the data administrator many hours of manual effort. Figure 3.3 from [Ref. 5] illustrates the flow of data through a typical convert function.

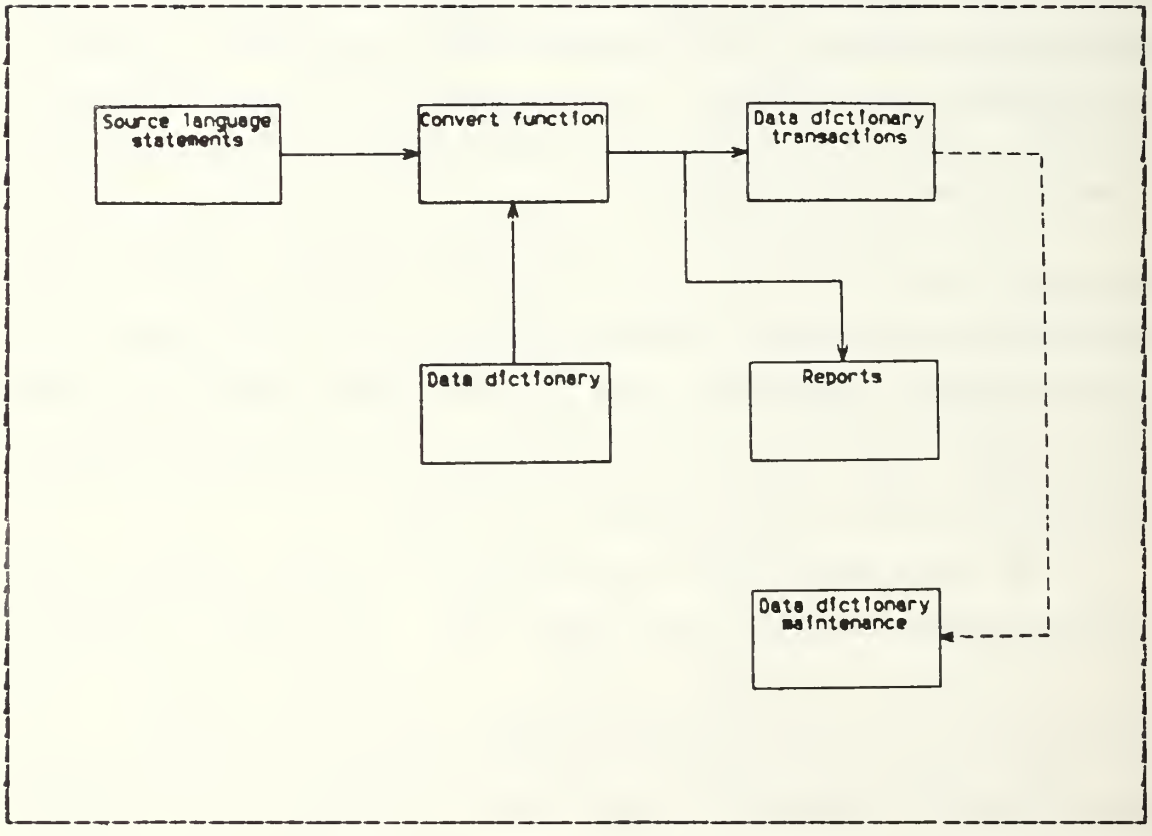

Figure 3.3 System Flow for a Convert Function.

Inputs include the source language statements and the D/D; outputs are a file of transactions to be input to

the D/D maintenance module, (in the case of SPLICE that refers to the maintenance module of the master copy) and a report.

The D/D maintenance transactions include descriptions of databases, files, records, groups, elements and programs. The prime purpose of convert functions is to convert metadata from both user-written programs and from local DBMS and its related components. Table IX illustrates in summary the typical D/D convert function transactions.

Four major characteristics [Ref. 5] for convert functions are:

The content of the generated transactions where the D/D maintenance transactions created by a convert function usually also contains the relationships between data entities.

The input file to a convert function that can be a source program or a library file.

The command options which may include the ability to change names, elect lines to scan, select types of transactions to create, and override generation of some types of metadata, where the ability to analyze the metadata of source programs can make the DDS a valuable tool for auditing adherence to software control techniques.

3. Environmental Dependency

This characteristic of a DDS is determined by its reliance on a specific hardware configuration, an operating system, a DBMS, or a teleprocessing monitor. Under ideal conditions a DDS must have the capability to operate in such an environment without losing efficiency and functionality. But sometimes the practice deviates from theory.

In a completely integrated DDS the DBMS accesses stored databases via the D/D. In a less integrated system, the DBMS may maintain its own directory file for accessing stored databases.

In the independent approach the DDS is completely autonomous, it does not rely on any particular DBMS, and the DBMS maintains its own source of metadata.

In the DBMS application approach the D/D appears to the DBMS as just another database. The DBMS maintains its own metadata for each database and these metadata are separate from the D/D.

For the SPLICE system, it is proposed that the embedded approach be used, where the DDS is actually a component of the DBMS's. This approach provides complete integration of the DDS. The D/D is the only source of

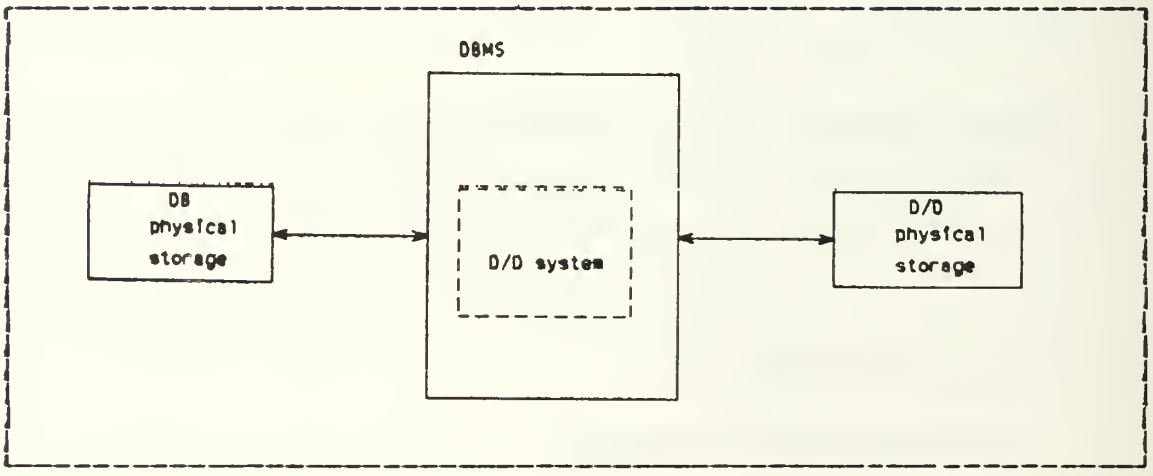

Figure 3.4 SPLICE Embedded Approach to DDS.

metadata. The DBMS utilities provide the D/D management facilities and the DBMS uses the D/D to directly access the stored databases. No other directories internal or external exist for the DBMS, and the DBMS and its facilities rely completely on the D/D for metadata. Such a structure is shown in figure 3.4.

So for example a query processor extracts user views from the DDS and the DBMS applies integrity constraints specified in the DDS by the DDS administrator before storing a data element. A major difficulty here, that the SPLICE designers must overcome, is the fact that the DBMS for SPLICE already exists but the DDS does not. The embedded approach is easier and simpler when both DDS and DBMS are developed in parallel, but this is not the case for the SPLICE. So special attention and effort must be applied during the DDS development phase.

TABLE VIII

Types of Software Packages I D/D System

Module	Description
DDL Processor	Creates a schema file
Database control system	Run-time unit of a DBMS
Preprocessor	Translates DML into CALL statements
Query/update Processor	Provides direct end-user access to stored databases
Batch-code generator	Reduces the time to develop a standard function as compared to a compiler-level language
Source-program manager	Provides security protection, data compression and editing capabilities for source programs
Teleprocessing monitor	Provides the capability of interactive computing to remote terminals.
Test-data generator	Creates test files and databases according to user specifications
Design aid	Analyzes and generates designs of databases or information systems

TABLE IX

Transactions for D/D Convert Function

Module type	Generated transactions
Programming	Element, group, record, file, and sometimes Subschema and process
Database description	Database, file, subschema, relationship, record, group, element
Teleprocessing	Terminal, line, processor, transaction

IV. SESSION SERVICES AND DATA DICTIONARY

A. GENERAL

The term "session" is defined in [Ref. 4] as follows:

"Session: All the activity (message exchange and processing) which takes place between two or more processes for the duration of a single task (e.g. text editing or processing of a transaction file)."

The session services module of the SPLICE has to play the role of coordinating the activity of the other functional modules and providing them with work instructions via the service codes it inserts in messages to the FM's. The sequence of operations may be data dependent or highly interactive, so in some cases, work breakdown cannot be completely determined in advance by the session services. In such cases session services passes control to the first (controlling) FM which is to perform an operation, and subsequent "calls" to other FM's, if any, take place according to processing conditions. In all cases however, session services passes control to the first (controlling) FM. However in some cases, all the FM's which will be involved cannot be determined in advance. Session services retains and maintains state information until either a completion message or error message has been received from the controlling FM. In the case of a message which is destined for an object located in another network, this fact is indicated in the "message type" field. The physical destination address would have been obtained previously from the data dictionary which exemplifies the relationship between session services and data dictionary.

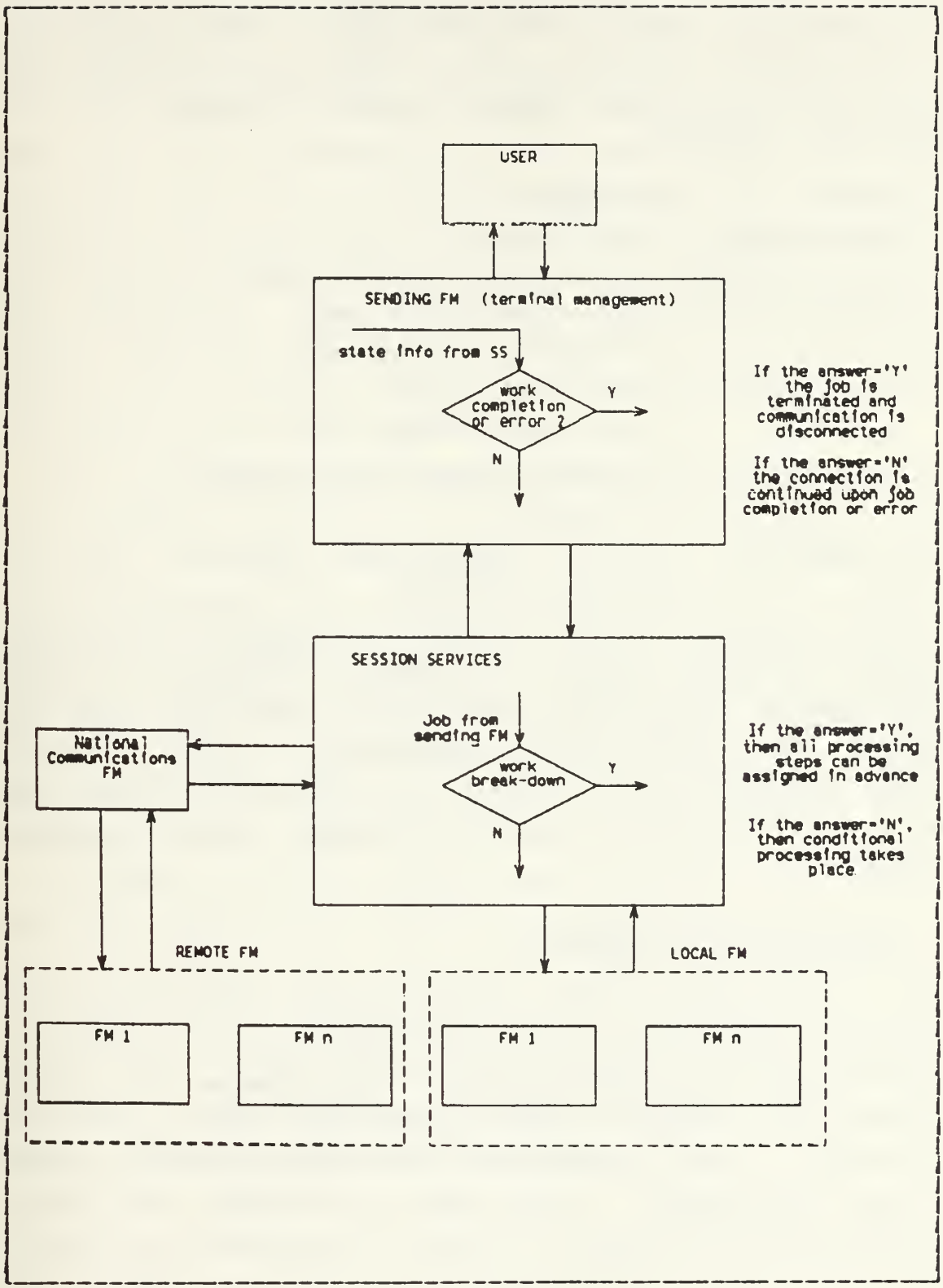

Figure 4.1 Cooperation Between SS and Functional Modules.

Session services is used in a distributed environment and involves the seven layer architecture model of the ISO for distributed networks. The ISO seven layer architecture is a standard one and involves the following layers with the associated functions:

Layer	Function
Application	User process
Presentation	Format data the user wants it
Session	Sets up session between communicating processes
Transport	End to end control
Network	Switching, routing
Data link	Reliable transmission between two nodes
Physical	Physical transmission of bits between two nodes

The complexity of the SPLICE processing environment requires that user terminal processes be given considerable assistance in carrying out their tasks [Ref. 4]. Session services can provide this assistance. User terminal processes specify task environments, largely by task name and the assistance of the data dictionary, where necessary (Figure 1.1).

E. ARCHITECTURE INTERFACES

In the SPLICE layered architecture, the interfaces between the layers are critically important. In particularly we are very interested in the software interfaces between the modules which communicate with the data dictionary. These modules are the session services module and the DBMS module. Some forms of software interfaces between DBMS and D/D can be found in the current literature [Ref. 5]. On the other hand no one has yet defined the

required software interfaces between the D/D and session services modules. We believe that the above mentioned software interfaces must be of the same type and closely related to the interfaces between the end user and the session services. In a centralized system where session services does not exist, the end user has to interface directly with the D/D, but in a distributed system the session services module acts as the mediator between the end user and the data dictionary. As a minimum then, the interfaces between session services and the data dictionary in a distributed system must include the interfaces between end user and data dictionary in the centralized model.

The interfaces between the above modules must be designed to accommodate new mechanisms and, as far as possible, new functions when they may arise. As new mechanisms and network functions come into use in the system, it is highly desirable that previously written programs continue to work. This is achieved by designing the interfaces appropriately and preserving them. In the seven layer architecture, layers 4,5,6 and 7 provide end-to- end communication between sessions in user machines. Layers 1,2 and 3 provide communication with the nodes of the shared network.

Because the SPLICE system uses a modified ISO layered approach, the interfaces between machines need to be defined in terms of the layers. So we will have layer headers and control messages that are passed between the layers. The application programmer does not need to know anything about these. For example any command language, using commands simmilar to GET, PUT, OPEN, CLOSE and DELETE, can refer to data or facilities in a distant machine.

C. THE SESSION SERVICES MODULE

There are differences in the session services provided depending upon type of network. In the distributed environment different types of user software need different types of session services. These differences involve not only the software but also the architecture. So one set of session services may be provided for one manufacturer's architecture and a different set for another. This is very important for the SPICE because the hardware used throughout the system varies. It may be possible that services provided across the system are of different types. However it is desirable to have common session services, because this will facilitate the maintenance task. Also for interfacing purposes want session services to present a common image to the system. This can be accomplished by hidding necessary interface units from the session services. In [Ref. 10 pp 491] there is a description of possible functions of the session services subsystem in a distributed network. These functions are generally divided into three large groups:

- -Functions required when setting up or disconnecting a session.
- -Functions used during the normal running of a session.
- -Functions employed when something goes wrong, such as a node failure or a protocol violation.

More precisely these functions are divided in the following categories:

- --Assistance in establishing a session
- --Basic networking functions
- --Application macroinstructions
- --Program control facilities
- --File access functions
- --Recovery and error control
- --Editing and translation
- --Dialogue software

--Virtual operations and transparency
--Compaction
--Payment functions
--Security and audit functions

D. INTERFACES

Functional interfaces between session services and data dictionary must permit other software modules to access the D/D and convert metadata into the format required by the DDS.

A DDS provides many functions and features such as:

Maintenance
Extensibility
Report processor
Query processor
Convert
Software interface
Exit facility

The software interface function must provide a formatted pathway enabling the DDS to provide metadata to other software systems such as compilers and DDL processors [Ref. 5], to retrieve information from the DDS, to update information where it is permited, and to obtain the restriction protocols for data consistency and integrity. The software interface can generate file descriptions for storage in a program library, or accept the user identification and generate a copy of that user's database view. It is not possible for this study to describe precisely the software interfaces needed for the SPLICE system. Because this system is under development, many aspects of the system are still unknown and the software modules are not yet described in full detail. So, we will only outline some of the software interfaces without claiming that these are sufficient

for the SPIICE system. Interfaces can be added to the system during the later stages of the system life cycle and existing interfaces can also be changed or improved as needed.

Because COBOL is used throughout the system, the COBOL "GENERATE" command can create from the D/D fully formatted file and record definitions that can be stored in a library file. Included can be most COBOL clauses such as 88 levels, SYNCHRONIZED, REDEFINES, and OCCURS. The OPTION clause of this command can permit changes in names, the designation of sequence numbers, level numbers and identifiers, and the inclusion of program comments. An example of the use of this command can be found in [Ref. 5 pp 261]. The generation date, last revision date, and revision number can be automatically recorded in both the listing and the D/D.

The output file can also contain job control statements to be included on the output file. Then the output file can be executed as a job that creates and catalogs the COBOL metadata as a member of a library under control of any of the various source program managers.

A DML processor can be used also to interface between the session services and data dictionary. A source program triggers the DML processor by sending a service code, through the session services, and the DML processor interacts with the data dictionary/directory. The output of the DML processor is an expanded source program that is sent to a compiler for compilation.

Other kinds of interfaces include query processors, source program managers, various user interface facilities, and other software packages.

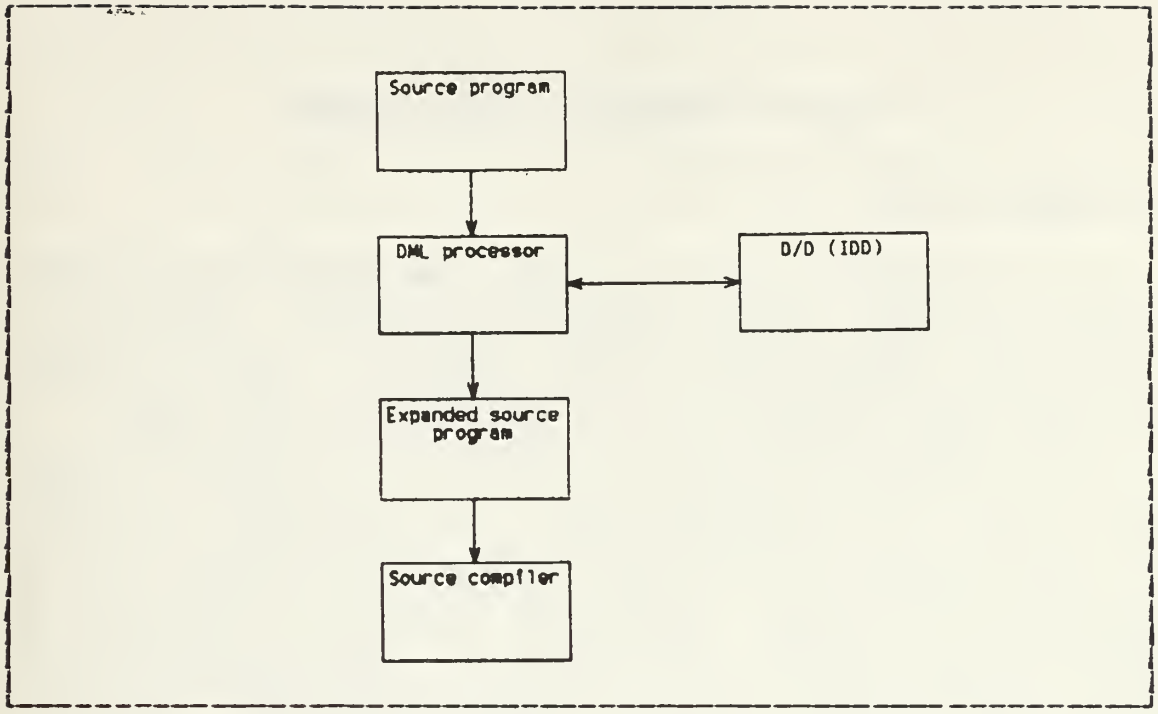

Figure 4.2 Software Interface Using a DMI Processor.

V. D/D IN DISTRIBUTED ENVIRONMENT

A. INTRODUCTION

In this chapter we will consider the design and function of DDS in the distributed database environment. Some extensions to the centralized D/D are needed in order to enable it to function effectively in a distributed environment.

The distributed system is a subset of a general information system. It is not necessary for the user to know how or where the data is stored or in what way the data will be accessed by a program or how and where the processing is accomplished. Unless the dictionary plays a highly active role in the running of the distributed system, there is little need to try to share one dictionary over the entire network. This is because there is not likely to be a large amount of update activity in a dictionary. The dictionary can normally be reproduced at each node and this is the proposed solutions for SPLICE. By using such an architecture, problems of updating the dictionary across the network can be solved without much overhead.

Of course the problem of distributed control in a network is more complex than that of the hierarchical architecture of dictionary systems which has been discussed in chapter two. This is one reason, in addition to the lack of experience with distributed data dictionary systems, why we proposed replication instead of distribution of the data dictionary for SPLICE. The more the dictionary system acts as either the control mechanism or a repository of control information, the more complex the DBMS, network operating systems, and dictionary system interactions become. For example, in the case where we want to determine the best

location for running a query against a distributed and partially replicated database [Ref. 6] the dictionary system is required to retain information on the location of all data. Indeed, this may be highly dynamic itself, and therefore the line between a dictionary and "real" database becomes very fuzzy.

Creation of a distributed information resource implies that the number of hardware and software components are to be designed and integrated into a controlled environment. These components in the SPLICE include several databases and database management systems, user language interfaces, data dictionary/directory catalogue, transaction controllers and data input/output control modules. We will describe the various system components and we will also attempt to demonstrate the integration of them with the international organization for standards (ISO) communications architecture, and a data storage and retrieval architecture (DSRA).

In general, a distributed system must provide to the end user transparency, data sharing, data transfer, process transfer, or a facility for combination of strategic, managerial and operational reporting. In order to do that there are several environmental constraints that must be satisfied [Ref. 12]. These are:

Data communications
Data storage and retrieval
Metadata
User language support
Process and report management
Information representation
System management
Integrity
Security

For the SPLICE system, communication must be integrated with cooperative processing of the various different

existing software and hardware. In order to do that we need to address the considerations of the database interface with distributed system tasks.

A distributed database is particularly useful to applications that involve extensive processing in different locations. SPLICE fits exactly in the above concept as do airlines, banking, retail, and military command and control applications. The distributed database of the SPLICE can be allocated among the nodes of the network according to various existing criteria for fragmentation. To avoid confusion in distributed systems two different terms are used : partitioned database which consists of non overlapping subsets, and replicated database, which has some data redundancy [Ref. 5]. Replication enforces the locality and availability of the database and reduces the frequency of accessing the DDN, but requires the DBMS to provide more sophisticated concurrency and recovery procedures. To avoid expensive overhead in data management, restrictions must be established as to the degree of data replication permitted. SPLICE belongs in the class of replicated database because the same item of the database can be located in several locations and the local databases provides information for items stored in only one location.

Major problems in the development of techniques for a distributed database are due to communication volumes and delays and to the potential for parallel processing. Sometimes it is very difficult to apply working solutions to distributed data processing which are borrowed from the centralized processing concept. These solutions often work well only in one environment and do not transfer efficiently. So excessive delays may occur. Parallel processing also has the potential to increase throughput, but requires complex controls to synchronize concurrent activities at dispersed sites. Because a data dictionary is

a database containing metadata, the same problems existing in distributed databases also exist in a distributed data dictionary.In [Ref. 5] are described five basic problems which must be addressed in distributed data management:

-The coordination of the DBMS with the data transmission network such that reliable delivery of messages can be ensured.

-The decomposition of transactions into atomic parts, selection of nodes to execute those parts, and control of any movement of data between sites necessary to process transactions.

-The synchronization of logically related updates and retrievals that are processed at different nodes.

-The detection and resolution of conditions where a part of the database becomes inaccessible due to node or line failure.

-The management of metadata describing the distributed database and environment. This last problem refers particularly to the data dictionary and deserves special attention.

E. EXTENSIONS TO THE DDS

The role of a D/D in a distributed database environment is very significant because it contains important information about the description of the database distribution, the characteristics of the nodes and other aspects of the data communication network. Some additional entities must be included in the DDS [Ref. 5] :

-The database entity which describes the global view of the database and includes attributes for relation and attribute names, validity constraints, as well as identification of local databases.

-The fragment entity which describes portions of the local database. This entity is not useful for the SELICE because there are not fragments of the local database.

-The topology entity which describes the physical configuration of network components and the links between the nodes.

-The node entity which describes the combination of network subcomponents at a particular site of the network.

-Finally some other entities (terminal, line, multiplexer, processor) describing network design.

We cannot say exactly what new entities should be added to the SPLICE DDS, but at least initially, we believe that a form of topology and node entities must be included. These entities are needed when non-local requests are processed, because the software performing transaction management needs to reference the D/D to determine the location of the needed data, the user's access privileges, the status in addressed nodes, etc. The interfaces needed for this purpose can be dynamic or static exactly as it is in the centralized case.

C. THE DDS AS A DISTRIBUTED DATABASE

Practically, the D/D, when supporting a distributed system, becomes itself a distributed database. The contents of the D/D may reside at various locations. We cannot say that this approach fits exactly in the SPLICE case. The approach we have proposed for the SPLICE is quite different. No partition of the D/D is permitted. That means the D/D cannot be a distributed database as we know it in the original form. For the solution proposed for SPLICE DDS, we can say that it is based on replication instead of distribution of the DDS. On the other hand, there are some other reasonable solutions which follow more closely the distributed concept. Since experience with distributed systems is relatively small, the steps needed to reach a decision must be taken very carefully in oder to avoid mistakes.

The designer of a DDS encounters some similar basic problems as does the designer of a distributed database. When we design a D/D we must determine the extent of environmental dependency between the D/D and the DBMS. As we said before, the distributed D/D is an extension of the centralized one and so the three basic variations to the type of relationships between a DDS and a DBMS are still in force. In the independent distributed approach the DDS has no running connections to any portions of the DBMS and is not actively or directly used in transaction processing by the DBMS. In the DBMS-application approach the D/D is just another distributed database to the DBMS and separate data management functions are not needed to handle the D/D. The DBMS may manage its own run time directory that is separate from the D/D. In the embedded distributed approach the D/D provides the run-time directory for the DBMS. All the components of the DBMS obtain their metadata from the D/D. The size, location, and contents of the D/D would also affect the performance of other DDS functions such as maintenance, reporting, and query [Ref. 5].

D. A MODEL FOR A DISTRIBUTED DDS

In this section we are going to examine a distributed model for SPLICE DDS. Its structure is shown in figure 2.4, and involves the partition of the global DDS into different views containing information for one or more local databases. These different views can be located at each or selected LAN's.

The global (or network) dictionary is the nucleus around which all the management functions of a DDS are centered. It contains [Ref. 11] information to start every management process of the SPLICE distributed database. In particular it contains:

a.-Information for the DDS design

-File access programs
-Total volumes of queries for each file
-Total volumes of updates for each file

This statistical information is very useful especially for evaluating the optimal number of redundant copies.

b.-Information for the distribution function
-Number and types of transmission links, their unit cost, their mean utilization factor
-Routing tables
-CPU workloads
-Disk utilization

This information can help determine the optimal allocation of redundant file copies and of possible operation parallelism.

c.-General information about data and how data is shared among the various nodes of the system. What the number of D/D copies is and where they are located.

d.-Information about existing constraints, status of the system, node failures etc.

e.-Information about data transportability

f.-Information related to data used by applications having a global view. Such applications are for example those where different local databases are involved for execution. We said in a previous section that sometimes data redundancy is preferable over the frequent use of the DDN. That means information about the sites where a component (i.e spare part) is located must be somewhere in a central position. So in the case where the component cannot be found in the local database, the user has to access the global data dictionary to find the places where the particular item is located.

To be able to design and run application or retrieval programs the global D/D must contain information [Ref. 11] about:

Data structures
Data location
Data availability
Data accessibility (related to security, compatibility etc)
Data translation maps, access paths
Data entities
Common procedures
Events and their interrelations

This dictionary must be able to answer queries about DB and DBMS's involved in a transaction and how the transaction can be formulated to obtain the most efficient result.

Local dictionaries include information about local databases and applications, local data entities, local procedures, local interrelations, physical storage structures of local data, access methods, access paths, physical storage devices, and redundancy of data items.

In [Ref. 11] a structure is proposed for a distributed D/D quite different from the SPLICE approach. This structure, as shown in Figure 5.1, involves the existence of:

Network dictionary
Global external dictionary
Global conceptual dictionary
Local external dictionary
Local conceptual dictionary
Internal dictionary

and each one of the above performs a different function.

This architecture which is purely distributed, is probably too complicated to be implemented for the SPLICE. It is a theoretical model and if we try to implement it, we may

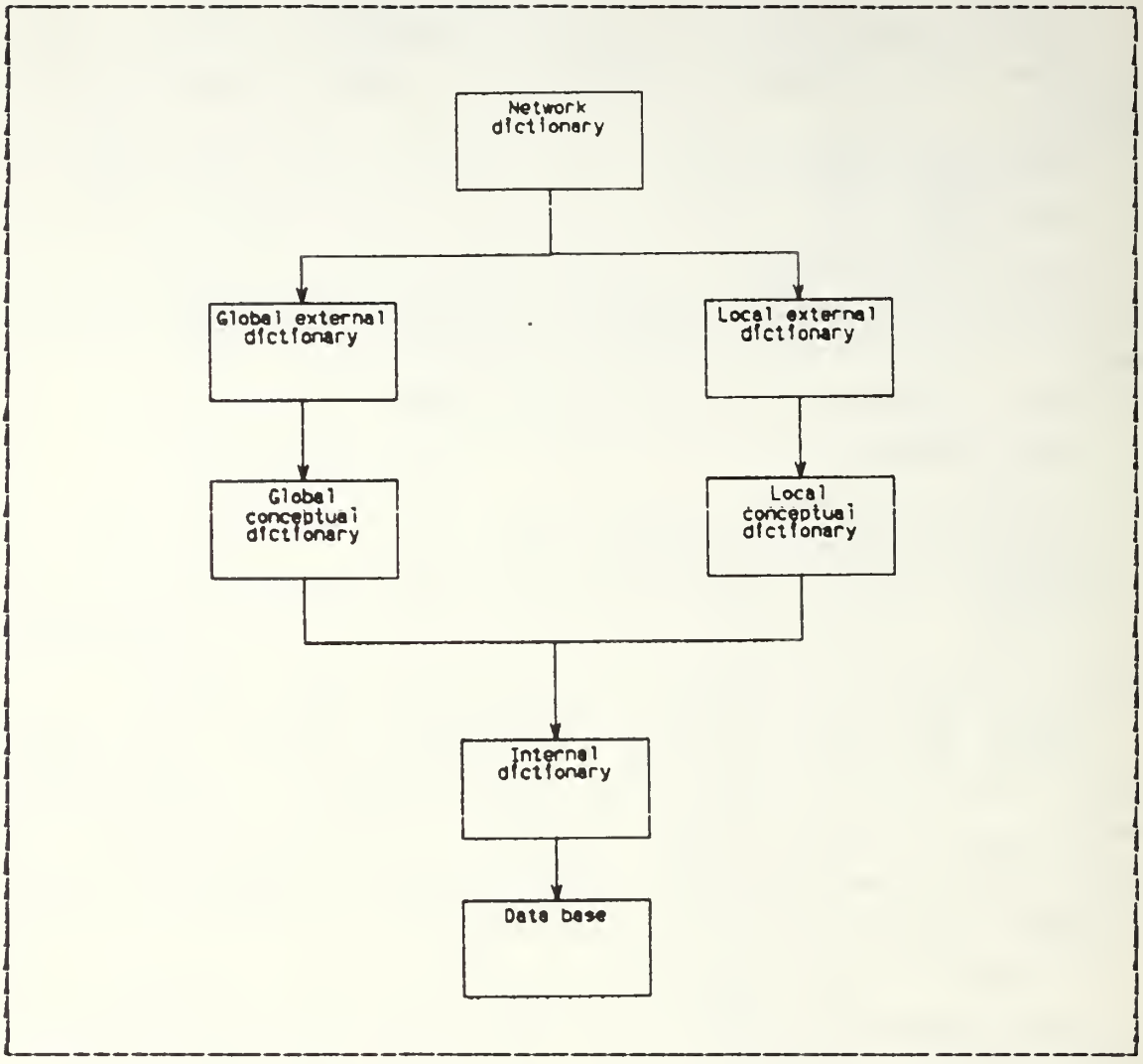

Figure 5.1 A Purely Distributed Approach for a DDS.

face serious interface problems, resulting in the data dictionary becoming the main resource consumer.

The functions we intend to include in the SPLICE DDS will play a major role, if we want to avoid complex structure and saturation. These functions must be the minimum possible needed for the proper operation of the system. We believe, in the case where the distributed instead of replicated approach will be followed, the architecture shown in Figure 2.4 is the more practical.

Following the above architecture a global dictionary located in some node has the role of maintaining consistency throughout the whole SPLICE system. Requests for updates, deletions, and additions are routed through the data dictionary administrator and after an evaluation procedure the global dictionary is updated. Then the changes are transmitted to various locations where the local copies are updated. Also updates are transmitted to the data directory.

Data directories can be located at the inventory control points (ICP). In contrast with the data dictionary, the data directory contains global information only about subject, service code, object name, and address. All the other information is located in the global and the various local dictionaries. The data dictionary administrator is responsible for maintaining the data directory, as well. Different views of the global dictionary are located in various LAN's. Each view can serve one or more LAN's and it is preferable to be located at the LAN where it is most frequently used in order to avoid unnecessary usage of the DDN.

When an item is not found in the local database the user routes a value location request through the session services (service code) to the data directory, and the data directory replies with the location address. Using the previous information the user can request and establish a session with the remote database where the requested information resides.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Our objectives, as described in the first chapter, were to investigate the area of data dictionary/directory systems, in a distributed environment, to outline the advantages/disadvantages of these systems, to present the underlying ideas, to examine the benefits for the SPLICE system from using a dictionary/directory system, and finally to delineate the interface requirements between a data dictionary/directory system and other functional modules. In addition to the above objectives we discussed also some ideas concerning the organization of the data administration function, and four hierarchical architectures for DDS, each one with a different degree of distribution.

The first architecture is based on the replication of the D/D. There are no different views of the D/D, only exact copies of one view located in each LAN. Using this architecture we have 62 replicated copies of the D/D (the same as the number of LAN's), each containing the information (metadata) about all SPLICE data base definitions and functions residing in each LAN. This architecture minimizes access to the DDN but has the drawback of requiring a lot of secondary storage. The size of the D/D, statistical and other information concerning the frequency of using the DDN, and the amount of information included in the D/D, all will have an impact on the effectiveness of this architecture.

The second architecture which allocates replicated copies of the D/D to selected nodes (the most active) is more conservative. In the case of a huge dictionary, this saves a significant amount of secondary storage, but

requires heavier use of the DDN. Here the size of the D/D and the appropriate nodes at which to install the replicated copies seriously affect the effectiveness of this architecture.

The third architecture is based on distribution of the D/D. Different views of the D/D reside in each LAN and contain information only concerning the local data base. This architecture involves the use of a data directory (we propose two replicated copies, one located in each ICF). The use of the data directory (which contains limited information) provides a kind of "relation or connection" between the various views. Also a global dictionary is needed in order to provide consistency and global function facilities throughout the system. This architecture is more dynamic than the previous two discussed so far. It has the advantage of saving secondary storage but, on the other hand, increases even more the use of the DDN.

A fourth architecture was discussed just to mention another possibility for a distributed architecture, but our estimation is that it would be too expensive in system resource consumption for the SPLICE.

Three environmental dependency options for the DDS (independent, completely integrated, and DBMS dependent) were also discussed. The main reason for choosing the embedded (DBMS dependent) approach is because the data dictionary is going to be used only for the SPLICE system (so the independent approach does not make any sense), and also the SPLICE data base already exists. Also the embedded approach (DBMS dependent) was chosen because of the homogeneity of the DBMS environments across LAN's. The independent and completely integrated approaches are too costly at this time although the latter could be implemented eventually from an embedded environment.

E. RECOMMENDATIONS

From the investigations performed, we have the following main recommendations for the SPLICE system:

a.- The TANDEM data dictionary that already exists should be the basis for the SPLICE data dictionary.

b.- The D/D should be implemented only for new applications because it is a herculean task to retrofit the D/D to the existing old applications.

c.- The embedded (DBMS dependent) approach should be used for the D/D.

d.- Two candidate architectures should be examined further based on statistical and other information (not available for the present thesis):

- -Replicated architecture (Figure 2.3) with selection of nodes where each copy will reside.
- -Distributed architecture (Figure 2.4) with the use of two replicated copies of the data directory located at each ICP.

e.- A DML processor should be used to interface between data dictionary and session services.

APPENDIX A
TANDEM DATA DICTIONARY

1. Overview

This appendix is included to mention some features (hopefully the most important) of TANDEM data dictionary, since the TANDEM DBMS will be used in the SPLICE system. For a more detailed description of the TANDEM D/D, see [Ref. 13].

A data definition language (DDL) is a language used by the data dictionary administrator to describe record and file structures of a database. After the description, the resulting source file is input to the DDL compiler, and the DDL compiler can create data declaration source language for database records in three languages, COBOL, FORTRAN, and TAL. The DDL compiler can also produce FUP (file utility program) file creation commands for database files. The most significant feature of DDI is its ability to create and maintain a data dictionary. The TANDEM data dictionary is a set of seven files that documents the structure and location of each file in a database.

The DDI provides facilities for updating a dictionary as the database it describes grows and the structure of the database files changes. The DDL compiler and the dictionary it creates serve as a central point of control over a database.

TANDEM defines a database as a collection of files structured to serve one or more applications. When a list of DDI statements --a DDL source schema-- is given to the DDL compiler, the compiler can produce any of the following files:

* A data dictionary.

* A FUP-file creation command source.
* A data declaration source for COBOL, FORTRAN, or TAL.
* A schema report summarizing each record's structure and each file's access keys.

The data dictionary produced by the DDL compiler is a set of files that forms a permanent record of the database schema. Thus the database schema, stored as a set of dictionary files, becomes a system resource. The dictionary gives database managers information about each file in the database and also shows how the files relate to each other. After the dictionary has been created, the DDL compiler can read the dictionary and produce COBOL, FORTRAN, or TAL data declaration source for any record defined by the schema. The dictionary is also used by ENFORM, TANDEM's database query language and report writer.

2. Creating a Dictionary

The data dictionary files can be created on any subvolume in the system. The subvolume that is to contain the data dictionary is specified with the DDL DICT command (for example ?DICT $STOCKNO.QNTY). The DDL compiler first creates the dictionary files on the quantity subvolume of the $ STOCKNO volume, and then opens the files for access.

3. Dictionary Reports

TANDEM provides DDL users with ENFORM source for twelve dictionary reports. The twelve reports document all of the DEFINITION and RECORD entries in the dictionary, describing not only their structures, but how they relate to each other as well.

Once a schema describing a database has been compiled by the DDL compiler and a dictionary has been produced, information about the database can easily be

obtained with a set of TANDEM provided ENFORM queries. The reports produced by these queries provide:

* Database documentation.
* Database analysis information.
* Quick access to dictionary contents.

The dictionary reports are produced from ENFORM source that is available to the user. This means that in addition to the standard reports, you can obtain customized reports, tailored to answer specific questions, by simply editing the TANDEM supplied ENFORM source. The ENFORM dictionary report source file consists of 12 queries that produce 12 different reports. Each query is a separate section. Thus the queries can be run as a complete group, individually, or in any combination. The 12 dictionary reports are shown in Table X.

4. Updating the Dictionary

As the database changes, its dictionary can be updated to reflect the changes by adding, deleting or modifying DEFINITION and RECORD entries. In Table XI is a summary of TANDEM dictionary modification function.

TABLE X

Dictionary Report Summary

Query Name	Report description
R1	DICTIONARY OBJECTS- R1 describes each DEF and RECORD in the dictionary, giving the time and date of creation, the time and date of the last modification, and the version number for each object.
R2	DEFINITION STRUCTURE- R2 lists all of the component groups and fields for each DEF in the dictionary.
R3	RECORD STRUCTURE- R3 lists all of the component groups and fields for each RECORD in the dictionary.
R4	DEFINITIONS USING DEFINITIONS- R4 shows which DEFs are referenced by other DEFs. The referencing DEFs are listed with each of its elements that references another DEF and the referenced DEF's name.
R5	RECORDS USING DEFINITIONS- R5 shows which DEFs are referenced by RECORDS. Each RECORD is listed with each of its elements that references a DEF and the referenced DEF's name.
R6	DEFINITIONS WHERE USED- R6 lists each DEF that is referenced by another object, be it a DEF or a RECORD. The referencing DEF or RECORD is shown in each case.
R7	RECORD ACCESS- R7 lists the file name and access keys (both primary and alternate) for each RECORD in the dictionary.
R8	RECORD DEFINITION METHOD- R8 shows the method used to define each RECORD. The source DEF is listed for those RECORDS defined with the DEF IS <def name> clause.
R9	REPORT HEADINGS- R9 lists all of the ENFORM report headings declared for fieds and groups within each DEF and RECORD in the dictionary.
R10	DISPLAY FORMATS- R10 lists all of the ENFORM display formats declared for fields and groups within each DEF and RECORD in the dictionary.
R11	RECORD COMMENTS- R11 lists the comments that immediately preceded the defining RECORD statement for each RECORD in the dictionary.
R12	DEFINITION COMMENTS- R12 lists the comments that immediately preceded the defining DEF statement for each DEF in the dictionary.

TABLE XI

Dictionary Modification Function

Operat./Ent.type	Procedure
ADD/DEF	Open dictionary with ?DICT and compile new DEF statement.
ADD/RECORD	Open dictionary with ?DICT and compile new RECORD statement
DELETE/DEF	Open dictionary with ?DICT, delete all dictionary entries that reference the DEF, and then delete the DEF itself with DELETE
DELETE/RECORD	Open dictionary with ?DICT and then delete the RECORD entry with the DELETE statement.
MODIFY/DEF	Open dictionary with ?DICT command, then delete all other RECORD and DEF entries that reference the DEF, delete the DEF, recompile the edited DEF, and, finally, recompile the DEF and RECORD statements that reference the DEF.
MODIFY/RECORD (with no DEF changes)	Open dictionary with ?DICT and recompile edited RECORD statement.
MODIFY/RECORD (with DEF changes)	Open dictionary with ?DICT and delete the RECORD with the DELETE statement. Then modify any DEF entries that need to be changed, and finally, recompile the new record statement.

LIST OF REFERENCES

1. Schneidewind Norman F., Functional Design of a Local Area Network for the Stock Point Logistics Integrated Communications Environment, NPS-54-82-003, Naval Postgraduate School, Monterey, Ca, December 1982

2. Barrett, K.M., Integration Considerations for the Stock Point Logistics Integrated Communications Environment (SPLICE) Local Area Network, Master's thesis, Naval Postgraduate School, Monterey, California, December 1982.

3. Carlsen David L. and Krebill Dan P., The national communications module of the stock point logistics integrated communications environment (SPLICE) local area network, Master's thesis, Naval Postgraduate School, Monterey, California, June 1983.

4. Schneidewind Norman F. and Dolk Daniel R., A distributed operating system design and dictionary/directory for SPLICE, NPS-54-83-015, Naval Postgraduate School, Monterey, Ca, November 1983.

5. Allen Fr., Loomis Mary, Manino Mich., "The Integrated Dictionary/Directory System", ACM computing surveys, Volume 14, Number 2, June 1982.

6. Iefkovits, Information resource/data dictionary systems, QED information sciences, Inc, 1983.

7. Sakamoto J. G. and Ball F. W., "Supporting Business Systems Planning Studies with the DB/DC Data Dictionary", IBM systems journal, Volume 21, Number 1, 1982.

8. Cirtice Rob., "Data Dictionaries: An Assessment of Current Practice and Problems", ACM Proceedings, Seventh International Conference on VERY LARGE DATA BASES, September 1981.

9. Deppe Mark E. and Fry James P., "Distributed Data Bases. A Summary of Research", Computer Networks, Volume 1, Number 2, September 1976.

10. Martin James, Design and strategy for distributed data processing, Prentice Hall, 1981.

11. Schreiber F. A. and Martella G., "Creating a Conceptual Model of a Data Dictionary for Distributed Data Bases", Data base, Volume 11, Number 1, Summer 1979.

12. Swager James R., "Architecture for a Distributed Data Base Information Resource", AFIPS, Conference Proceedings, Volume 50, May 4-7, 1981

13. TANDEM Computers Incorporated, Data Definition Language (DDL) Programming Manual, December 1981.

14. Boehm Barry W., Software Engineering Economics, Prentice-Hall, 1981

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Library, code 0142 Naval Postgraduate School Monterey, California 93943	2
2.	Department chairman, code 52Hq Department of computer science Naval Postgraduate School Monterey, California 93943	2
3.	Prof. Norman F. Schneidewind, code 54Ss Naval Postgraduate School Monterey, California 93943	4
4.	Prof. Daniel R. Dolk, code 54Dk Naval Postgraduate School Monterey, California 93943	1
5.	Defence Technical Information Center Cameron Station Alexandria, Virginia 22314	2
6.	Hellenic Navy General Staff Education department Stratopedo Papagou Holargos Athens, GREECE	2
7.	CDR Vassilios Paragiaris Hellenic Navy General Staff SDENE Stratopedo Papagou Holargos Athens, GREECE	4

210525

Thesis
P1454 Panagiaris
c.1 A dictionary/directory system (DDS) for the SPLICE system.