JPL PUBLICATION 82-6

Generic Functional Requirements for a NASA General-Purpose Data Base Management System

Guy M. Lohman

October 1, 1981

NASA
National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 82-6

Generic Functional Requirements for a NASA General-Purpose Data Base Management System

Guy M. Lohman

October 1, 1981

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

ABSTRACT

This report details generic functional requirements for a general-purpose, multi-mission Data Base Management System (DBMS) for application to NASA's remotely sensed scientific data bases. The motivation for utilizing DBMS technology in this environment are explained. The major requirements include: (1) a DBMS for scientific observational data, (2) a multi-mission capability, (3) user-friendly, (4) extensive and integrated information about data, (5) robust languages for defining data structures and formats, (6) scientific data types and structures, (7) flexible physical access mechanisms, (8) ways of representing spatial relationships, (9) a high-level nonprocedural interactive query and data manipulation language, (10) data base maintenance utilities, (11) high-rate input/output and large data volume storage, (12) adaptability to a distributed data base and/or data base machine configuration, and (13) well designed and supported. Detailed functions are specified in a top-down hierarchic fashion. Implementation, performance and support requirements are also given.

ACKNOWLEDGEMENTS

The author wishes to thank Jim Brown and Richard Blackwell of JPL, Pat Gary of GSFC, Richard Hull of UCLA, and the JPL ADAM Team (Kofi Apenyo, Joe Stoltzfus and Jose Ureña) for their valuable comments on a draft of this document. We also wish to acknowledge the valuable inputs and the continuing support and encouragement of H. William Shaffer at NASA Headquarters.

TABLE OF CONTENTS

SECTION 1

INTRODUCTION

"The purpose of a programming system is
to make a computer easy to use."
-- Brooks

1.1 OBJECTIVE

The objective of this document is to define detailed functional requirements for a general-purpose, multi-mission Data Base Management System (DBMS), called the Applications DAtabase Management System (ADAM). The ADAM System is intended to provide DBMS support as one component of the ground data management systems of future missions of the Office of Space and Terrestrial Applications (OSTA) of NASA, and possibly for use by related NASA Offices and/or agencies that collect, archive, process, or distribute remotely sensed scientific data about the earth.

1.2 MOTIVATION

Generic requirements for NASA DBMSs need to be documented for several reasons. First, there has been considerable controversy within NASA as to whether DBMS technology can satisfy the data management requirements of NASA applications. By specifying those requirements most relevant to DBMS technology, both NASA data managers and the DBMS community can more precisely assess the suitability of DBMS for any given application. Second, detailing the requirements in written form will help NASA and commercial vendors to isolate those features that can, and cannot, be provided by commercially available DBMS software. Action can then be taken by NASA or (preferably) by the vendors to augment, alter, or replace current DBMSs with systems more suited to NASA/OSTA applications. Finally, these requirements can serve any NASA application (for example, a future flight mission) as a checklist from which to draw requirements for near-term procurement of the best available DBMS to suit that application.

1.3 SCOPE

It should be emphasized that these requirements are for a general-purpose tool for data base management, which may support -- but is not intended as -- a complete "turn-key" application system such as a geographic information system for Landsat imagery. DBMS requirements for a particular NASA application have been documented before, for example, for the Integrated DBMS of the NASA End-to-End Data System (NEEDS) project [GARY

80]. However, generic DBMS requirements for an indefinite variety of NASA scientific applications have never been specified.

As such, this report contains the "greatest common denominator" requirements. It is intended to be a baseline document that may be altered by applications as needed and that will be updated periodically based upon future NASA/OSTA experience with DBMSs. The requirements contained herein will not suit all applications, especially real-time systems or "pipeline" data processing systems (such as the Landsat Image Processing Facility) where throughput rates are paramount. For any given application, not all requirements will pertain, nor will this list be exhaustive. However, every attempt has been made to reference each functional requirement to one or more known user requirements. Finally, the performance of such a general-purpose tool is difficult if not impossible to specify in absolute quantities, since it is dependent upon the data to be managed and the user response requirements of the individual application. Although some baseline performance figures will be available soon from a few operational DBMS applications [GOUG 81], no standard NASA benchmarks are available for testing DBMS performance.

1.4 ENVIRONMENT

The ADAM System is intended to be a general-purpose tool that can be adapted to many different applications occurring in varying data processing environments. Therefore, there does not exist a specific list of computer hardware and software on which the ADAM must be implemented. Rather, the goal is to achieve maximum independence from the constraints posed by the environment of each application, so that ADAM is transportable between different systems to the maximum degree allowed by the present state of the art.

1.4.1 USER COMMUNITY

The users of ADAM are anticipated to be both discipline scientists -- those well versed in the fields of oceanography, atmospheric or earth sciences, earth resources, etc. -- as well as computer analysts and programmers that will be developing applications programs and systems to be supported by ADAM. More specifically, potential users will include [GARY 80]:

- Principal investigators associated with specific missions and/or research programs
- Investigators analyzing single- or multi-source data
- Scientists developing models of physical phenomena

- o Reseachers in image processing and other data analysis
- o Development programmers responsible for testing and debugging data processing and data analysis software
- o Other applications software development personnel

1.4.2 DATA TYPES

The specific data sets to be managed by ADAM will be different for each application system. However, characteristic NASA/OSTA data types that ADAM will have to accommodate include [GARY 80]:

- o Digital image data
- o Spacecraft *in situ* measurements
- o Data collected by ground sensors
- o Results of application data analysis programs
- o Geophysical parameters extracted from all of the above types of data
- o Application modeling results
- o Information describing data set characteristics

1.4.3 APPLICATION SYSTEMS AND PROGRAMS

An ADAM will need to serve a variety of application systems/programs. These programs will, in general, be developmental and *ad hoc*, rather than operational production programs that are run on a routine basis. Examples of the expected types of application programs are listed in Table 1.1.

1.5 RELATION TO OTHER NASA DBMS EFFORTS

For the reasons outlined in Section 2 below, there has been considerable interest within NASA of late in DBMS technology. Prior to 1978, DBMSs were used within NASA exclusively for administrative types of data bases [BRDK 80]. Several on-going tasks within NASA now have used or plan to use DBMSs for scientific, remotely sensed data. The paragraphs that follow attempt to relate concisely the ADAM to the perceived objectives of these other efforts.

The Packet Management System (PMS) at Goddard Space Flight Center is probably the effort most closely related to ADAM. Its near-term emphasis is upon testing the suitability of commercially available DBMSs (specifically, Oracle and SEED) to a number of applications. One of these applications is managing

Table 1-1. Examples of Potential Application Systems/Programs

Program Type	Example/Description
Customized data access	
o Data set loading and appending of new data	
o Locating data required	
o Display of selected portions of data	
o Data retrieval and filtering of unwanted data	
o Menu-driven (parametric) systems	
Data processing	
o Algebraic and functional operations	
o Algorithm development and application	
o Geophysical parameter extraction	
o Data product generation	
Modeling and Forecasting	
o Finite element models	
o Differential equation models	
o Simulations	
Image display and processing [BLCK 80]	
o Geometric transformations or reprojections	Differing map projections
o Rotation and magnification	Rotation, magnification, and shrinking
o Combination (mosaicing)	Combination of adjoining images into larger ones
o Annotation and display	Add, alter, or remove labels or symbols
o Local operations	Blemish correction or removal, image segmentation
o Point operations	Contrast manipulation
o Algebraic operations	Add, subtract, multiply, divide, average, etc., either between images or with constants/variables
o Correlation and convolution	Comparison and registration of images from different instruments, filters to sharpen features or remove noise
o Frequency domain computation and filtering	Fourier transforms for identification of image components, coherent noise removal, enhancement of specific frequency components
o Image measurement	Extraction of arithmetic and/or statistical characteristics of imagery
o Test image	Creation of test images, grids, noise patterns

Table 1-1. Examples of Potential Application Systems/Programs (cont)

Program Type	Example/Description
Statistical Functions o Interpolation o Smoothing o Averaging o Histogram production o Correlation o Linear regression o Analysis of variance	
Graphics o Map generation o Contour plots o Graphs of geophysical parameters over time o Histograms o Scatterplots	

a catalog of the physical storage granules -- or packets -- that will be received in near-real-time from spacecraft sensors at rates up to 50 megabits per second (but with only a 20% duty cycle). The packets will be archived in the Archival Mass Memory (AMM) that is under development at Marshall Space Flight Center under the NASA End-to-End Data System (NEEDS) Project within the Office of Aeronautics and Space Technology (OAST). Although the PMS plans include a direct interface to the end-user (e.g., a scientist), to application programs that may return more processed data to the archive, and access to the data itself, the initial thrust is upon cataloging the Level 0 and 1 data stored in the AMM [BURN 80].* The functional requirements for the NEEDS DBMS [GARY 80] stress the detailed attributes of a data set that need to be cataloged to help the user locate data of interest, with less emphasis upon functions supporting access to the data itself (such as logical views of the data tailored to users via a subschema definition language). The ADAM is more oriented toward the user interface than the data collection aspect, toward non-real-time rather than real-time environments, and toward DBMS access to smaller data volumes in response to ad hoc types of queries. It is conceivable, for example, that ADAM might interface with the PMS in order to select and retrieve data of interest for more processing or for more intensive study by scientists.

* For a definition of these data levels, see [DJRD 79]

The Pilot Climate Data Base Management System (PCDBMS), sponsored by the Office of Space and Terrestrial Applications (OSTA), is applying the PMS evaluation to the requirements of the climate discipline within OSTA [OAOC 79]. This near-term implementation, like others discussed below, should help to define user and functional requirements for data management tools such as ADAM.

The OSTA Data Inventory and Catalog task is collecting the necessary information about existing data sets to incorporate into a cataloging system to be maintained on-line by the PCDBMS. The ADAM is intended as a tool that is used to maintain and access this kind of information, as well as some of the data that it describes. Although ADAM effort also included an assessment of representative existing data bases and data management systems [APNY 81], the intent was to distill their common characteristics as factors that influence the requirements for ADAM, rather than to perform an exhaustive survey of all data sets as an end in itself.

Also under development by OSTA Data Systems is the Transportable Applications Executive (TAE), an executive that will provide a standard interface to user application software, and facilitate both expert and casual user interaction with the system. A prototype TAE has been implemented that supports a subset of the ultimate TAE services, including user menus, a command language, a parameter selection facility, and an on-line help facility. The TAE -- like ADAM -- is intended to be multi-mission, transportable software. In systems where the TAE is the executive with which ADAM must interface (see Function 4.2, Appendix A), the TAE will help to standardize the interface to the hardware and all other TAE-compatible software, thus simplifying the ADAM interfaces [BCHM 81]. This is particularly true, for example, of the interface with the image which TAE is based.

The Applications Data Service (ADS) Project of OSTA Data Systems is supporting a standards-development activity and three pilot data systems that impact ADAM. The ADAM System subtask to assess DBMS standards and standards-setting activities [URNA 81B] will aid the overall ADS standards work, and in return the standards developed by ADS should simplify the interfaces of the ADAM with users, storage devices, the ADS network, etc. The pilot systems, and particularly the Oceanic Pilot System (OPS) at JPL, will help to refine user and functional requirements for data management and to serve as a testbed for gaining direct experience with the promise and problems of existing DBMSs. The requirements contained herein define a target DBMS that is tailored to the requirements of future systems which are exemplified by the pilots. Therefore, the pilot systems are also prime candidates for prototype implementations of the ADAM.

Finally, the DBMS prototype and the Mark IV Data Records System, under development at JPL for the Flight Projects Support Office (FPSO) in support of OSS (Office of Space Sciences) missions, is another example of a system that is using a commercially available DBMS (Univac's DMS 1100) for a near-term implementation. It too has already provided valuable experience in the advantages and disadvantages of DBMS technology. In particular, the completed prototype demonstrated the feasibility of direct DBMS access to Level 0 and 1 telemetry and engineering data. It also verified user requirements for several capabilities that are uniquely provided by DBMSs, such as data independence, inter-relation of files, and multi-key access to data [APNY 81].

1.6 OVERVIEW OF THIS DOCUMENT

The rest of this document contains the functional requirements for ADAM that were derived from:

- the requirements of users [DJRD 79, FJMT 81, OAOC 79, LHMN 80A];
- the characteristics of existing OSTA and related data bases [APNY 81, BRYN 79A, BRYN 79B, JHNS 80];
- existing standards that were deemed relevant to DBMSs [URNA 81B]; and
- inputs from related efforts such as the NASA/OSTA Data Base Management Systems Panel [BRYN 79A, BRYN 79B, LHMN 80B, URNA 81A], and the NEEDS effort to define functional requirements for a DBMS for a particular environment [GARY 80].

Section 2 first discusses the applicability of DBMS technology to NASA/OSTA problems and inefficiencies. This section answers the question, "When?" or "Under what circumstances?". Section 3 then gives an overview of the major requirements upon ADAM. This section is intended to answer the question "What?" Section 4 contains the implementation characteristics that are required of ADAM, answering the question "How?" Sections 5 and 6 specify the performance and support that is required of an ADAM, respectively, to answer the question "How well?" Appendix A concludes the document with a detailed discussion of the generic functions that ADAM will need to perform. The exposition utilizes a hierarchical breakdown called a "top-down functional analysis".

SECTION 2

DBMS APPLICABILITY TO NASA/OSTA DATA BASES

"The user does not know what he
wants until he sees what he gets."
-- Yourdon

The growing size and complexity of OSTA data bases has created a number of problems and inefficiences in how data is managed. The idea of using a DBMS for NASA's applications data bases originated from similar problems and inefficiencies in private industry that were largely solved by the development and implementation of DBMS technology. What follows is a brief discussion of each of the problems and inefficiencies, the capabilities that DBMS technology offers, and how each capability contributes to the solution of each problem and/or inefficiency.

2.1 PROBLEMS

The problems which follow have impacts that are difficult to quantify but nonetheless interfere to a great extent with the accomplishment of OSTA goals. Many of these problems were independently identified by a major National Academy of Sciences study [BRNS 80].

Problem P1: Lack of Coordination and Communication. Because each mission within each NASA Office has generally been responsible for developing data management systems to support its unique requirements, the concommitant lack of coordination and communication between missions has hampered:

- Standardization. No official standard NASA formats or software could be located by this or related tasks [KUCH 81, URNA 81B], although attempts to specify standard NASA formats are now underway [EDC 79, GDNU 79, WLTN 80, GRNB 81]. Also, some widely used systems (e.g., AOIPS, VICAR [APNY 81]) have formats that have become almost *de facto* standards [URNA 81B].

- Economies of Scale and Cost-Effective Trade-Offs. Virtually every OSTA system examined to date by this task performs in its own way the routine functions of structuring, storing, and retrieving data [APNY 81]. Mission software managers are understandably reluctant to expend their very limited resources to develop general-purpose software that can be used by other missions. The efficiencies of multi-mission software are obvious from the global NASA perspective, but not from the missions' local perspective.

- Adaptability to Change. Most systems that were examined by this task were implemented in FORTRAN or machine language [APNY 81]. Such languages almost always scatter throughout each program the statements which define the structure of the files that they create or access. When errors in these statements are found, or the requirements for its use change, software developers are faced with the dilemma of either suffering serious manpower costs to find and alter all the relevant format statements, or enduring frozen designs which simply do not permit support of the changing requirements.

- Centralized Integrity and Security Constraints. In the systems that we examined, provisions for integrity and security of data sets are quite decentralized. Generally, the principal investigator(s) that collected the data set enforce security only by deciding to whom copies of that data may be sent. They have little or no control of the data beyond that. Except in formal archive centers, individual investigators maintain their own backup copies of their copy of a data set. This can easily lead to redundant or inconsistent backups of the same data set. Furthermore, these investigators must assume the responsibility for locating and accessing any related files that may be compared for validating the correctness of data in a data set.

Problem P2: Difficulty in Finding Useful Data. One of the few user requirements that was identified by every OSTA discipline at the OSTA Data Systems Planning Workshop [DJRD 79], and which is commonly mentioned by actual interviews with users [OAOC 79, FJMT 81, LHMN 80A], is the need to enhance their capabilities to locate data which is both available and of "good quality".

Problem P3: Difficulty in Accessing Relevant Data. Another problem that is widely mentioned is the difficulty of efficiently accessing the portions of data which are relevant to a particular application, based upon multiple keys (e.g., latitude, longitude, time, and sensor [OAOC 79, LHMN 80A, FJMT 81]). Except for small data bases covering regions of intense interest, OSTA archives are stored chronologically on magnetic tapes. Hence retrieval of all the data for a new request pertaining to a particular region involves reading through many tapes and maybe even the entire data base in order to find a small percentage of relevant data. This is a slow process even if catalogs or indexes help the user translate his request into orbit numbers and hence tape reel numbers [APNY 81].

The process is sufficiently clumsy and expensive that it often limits scientific inquiry to those questions for which data is readily available, and excludes other worthwhile investigations that might be possible from the same data base.

Problem P4: Increasing Quantities of Data. Improved sensor technology has resulted in sensors which are returning data at ever-increasing rates, with no change in that trend projected for the foreseeable future [BRNS 80]. For example, the Thematic Mapper instrument to be aboard the Landsat D/D' Spacecraft alone will collect data continuously at a rate of 85 megabits per second. This has resulted in an explosion in data base size -- a projected 10^{14} bits for Landsat D', assuming a lifetime of 5 years -- and bottlenecks in processing that data in a timely way [BLLN 81]. The data rates increase by a factor of 2 to 4 for Multispectral Linear Arrays (MLAs) and Synthetic Aperture Radars (SARs) to be flown on future earth observation satellites [BRCK 80].

Problem P5: Protecting Data. In order to ensure that no data is inadvertantly lost or destroyed during processing and "cleaning" of the data, many redundant copies of entire files are kept, even when changes are made to only 1 to 5% of the records in these files [APNY 81]. This unnecessary redundancy has merely exacerbated the growth in data base size (Problem P4).

Problem P6: Conflicting Access Rights. When data is still "new", a small set of principal investigators need to be granted exclusive access to that data for a limited time. Subsequently, NASA/OSTA would like to encourage the widest possible distribution of that same data. This seeming paradox between very strict and very free access rights to the data has in the past led to systems developed by and for mission principal investigators only, with data bases becoming "orphans" in archival data centers after the termination of that mission [DJRD 79, APNY 81].

In addition to the above problems, the current approach to data management utilizes manpower inefficiently. The inefficiencies which follow may be hidden in the form of decreased programmer productivity, new programs that could not be written, scientific analysis that had to be foregone, etc. Each inefficiency results in the increased use of manpower as a way to minimize expenditures for use of computer hardware. Overall costs increase, therefore, because of the trend throughout the information systems industry toward rapidly rising costs for people and the software they generate, coupled with falling per-unit costs for hardware [ZLKW 78].

Inefficiency I1: Redundant Software Development. As mentioned in Problem P1, OSTA missions have developed all their own

software, usually embedding routine data management aspects within application-specific programs. Many of the data management functions described later in this document are common to all OSTA data systems, but are currently performed by custom software [APNY 81]. Again, no one mission wants to develop multi-mission data management software, yet the inefficiency of this redundant development should be apparent.

Inefficiency I2: Software Maintenance. It has been estimated that over two-thirds of all software development costs in industry are spent maintaining existing software [ZLKW 78]. Software maintenance costs have generally been highest where a machine language rather than high-level programming language has been used [ZLKW 78], and where functions such as data management have not been centralized into one identifiable module. NASA is probably no exception to this industry-wide trend. And in this time of tightening budgets, funds spent for maintenance are funds taken from the development of new software or from scientific analysis.

Inefficiency I3: Adapting Software to Changing Requirements. Inevitably some requirements on any data system change during its lifetime [ZLKW 78]. However, the dispersion of file formats among application programs within OSTA data systems may make any changes too expensive to implement or too time-consuming to meet critical mission milestones. As a result, requirements and file formats must be frozen early in the design of the system, or else considerable effort must be expended determining what portions of the system will be affected by necessary changes. Once data collection has actually begun, system inflexibility becomes even greater. This was the case, for example, with the Seasat mission [BRWN 80].

2.2 DBMS CAPABILITIES

A Data Base Management System (DBMS) is responsible for all access to, and updates of, the files contained in its data base (See Figure 2-1). DBMS technology evolved due to problems and inefficiencies similar to those listed above that were experienced by commercial data base users. The reasons for the development of data base concepts are not invalidated by the types of data found in NASA data bases. The evolution of DBMS-based systems from file-oriented systems is shown in Figure 2-2.

The capabilities of DBMS technology are briefly described below. Further details may be found in [CRDN 79, DATE 77, MRTN 76, etc.]. The OSTA problem and/or inefficiency which each capability addresses is marked by an "X" in Table 2-1, with an "M" denoting its main intent.

Capability C1: Data Independence. This term refers to the insulation of the user from the physical details of how and

where each data record is stored (See Figure 2-3). The DBMS acts as a "middle-man" between the user and the data storage media, much as a clerk in an auto parts store helps customers locate the part(s) that they need. The customers should not have to be aware of how the store organizes its shelves (e.g., by part number), and should be insulated from any changes to this organization. Rather, he or she should need only to describe the logical parameters (car make, year, part type, etc.) of the needed part. It should be the clerk's responsibility to transform those parameters into a part number, and from that to a particular shelf and box number where the part is located.

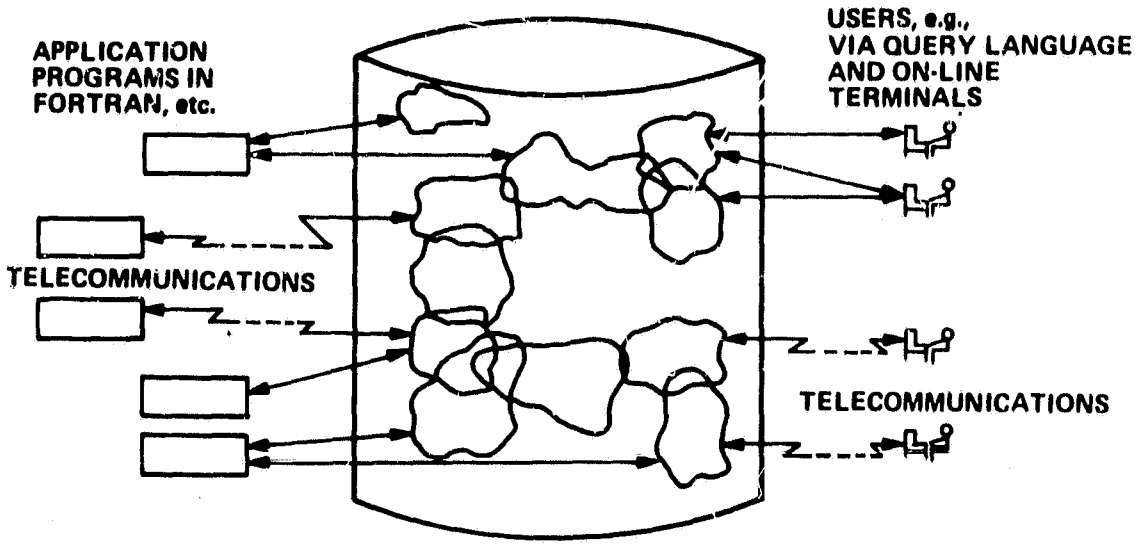


Figure 2-1. Schematic of an Integrated Data Base System [CRDN 79]

Capability C2: Data Shareability and Nonredundancy. Two of the major goals for OSTA's Applications Data Service (ADS) that were determined at the OSTA Data System Planning Workshop are increasing data sharing and reducing data redundancy [DJRD 79]. By centralizing control of all access to and updates of a data base, a DBMS provides a focal point for all users requiring data from the data base. A DBMS acts as a "traffic cop", allowing multiple users to access the same data simultaneously so long as it does not interfere with other users (e.g., those who might be updating the data). By permitting different user views of, and names for, different subsets of the data base, a DBMS provides to a user only the data items needed in the format required, without each user having to retain redundant copies (see Figure 2-4).

Table 2-1. OSTA Data Management Problems (P) and Inefficiencies (I) That Are Addressed by (X), or the Main Purpose of (M), DBMS Capabilities (C)

	OSTA PROBLEM (P) OR INEFFICIENCY (I) / DBMS CAPABILITY	P1: COORDINATION AND COMMUNICATION	P2: FINDING USEFUL DATA	P3: EFFICIENT ACCESS TO DATA	P4: DATA RATES AND VOLUMES	P5: PROTECTION OF DATA	P6: CONFLICTING ACCESS RIGHTS	I1: REDUNDANT SOFTWARE DEVELOPMENT	I2: SOFTWARE MAINTENANCE COSTS	I3: ADAPTATION AND CHANGING REQUIREMENTS
C1	DATA INDEPENDENCE			X					X	M
C2	SHAREABILITY AND NONREDUNDANCY	X	X	X	X		M	M	M	X
C3	RELATABILITY	X	M	X				X		
C4	INTEGRITY	X	X			M			X	
C5	ACCESS FLEXIBILITY		X	M	X			X	X	X
C6	SECURITY	X					M			
C7	PERFORMANCE AND EFFICIENCY	X	X	X	M			M		M
C8	CENTRALIZED STRUCTURING, ADMINISTRATION & CONTROL	M	X	X	X	X	X	X	X	X

KEY: M – MAIN PURPOSE OF DBMS CAPABILITY

X – DBMS CAPABILITY ADDRESSES PROBLEM

Capability C3: Relatability. The major characteristic which distinguishes a true data base from a collection of files is the explicit or implicit relationships which link the files into a structured data base. For example, each spacecraft has one or more sensors, each of which collect observations that are generally stored as separate files. The spacecraft also has engineering and location information (such as ephemerides tables, spacecraft attitude, etc.) that may be relevant to all of the sensors' observations in ways that vary by sensor. Furthermore, one sensor may have multiple data files, containing observations that were collected at different ground stations or have undergone varying stages of processing (e.g., raw bit stream vs. engineering units vs. geophysical parameters). These relationships are shown graphically in Figure 2.5.

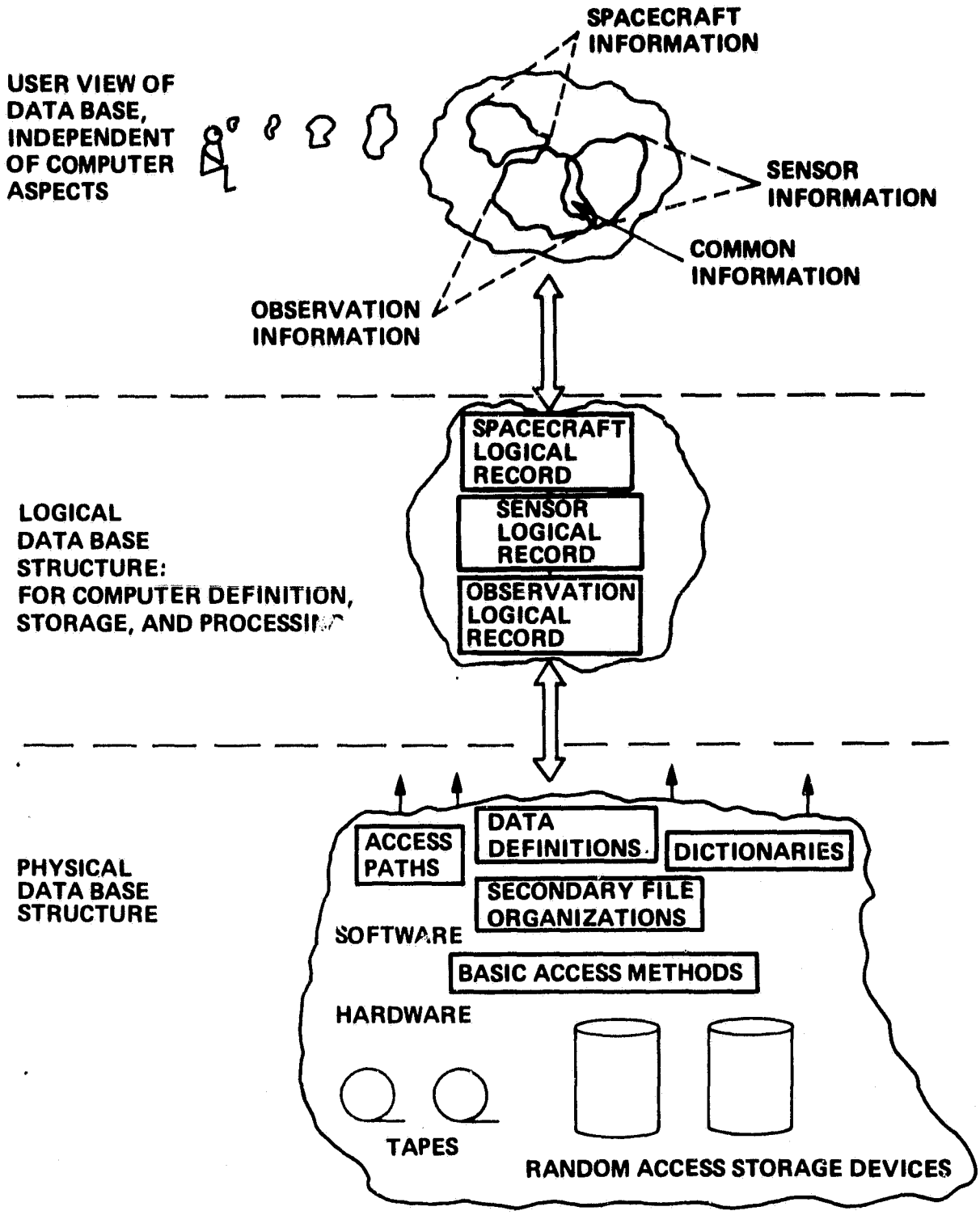


Figure 2-2. The Data Base Organization Realms [CRDN 79]

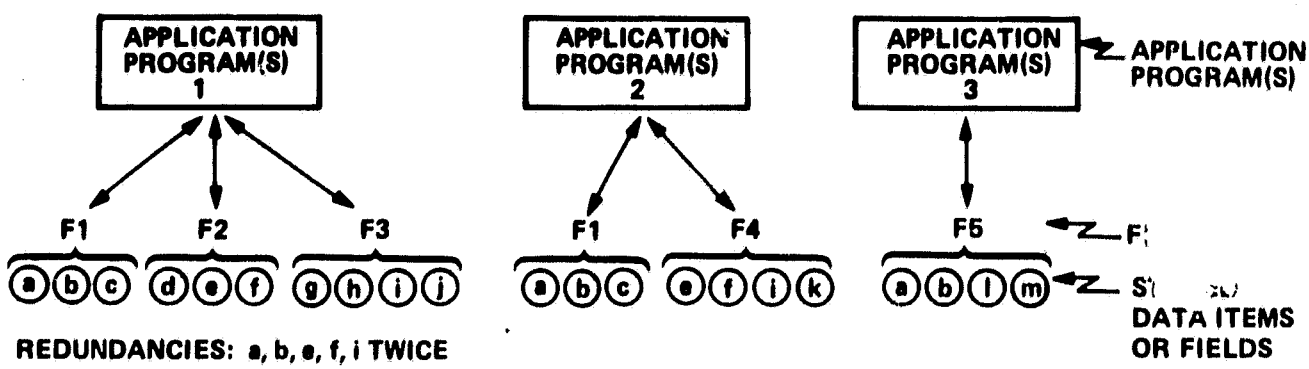


Figure 2-3a. The Traditional Approach to Programs and Data, Leading to Redundancy of Data Stored [CRDN 79]

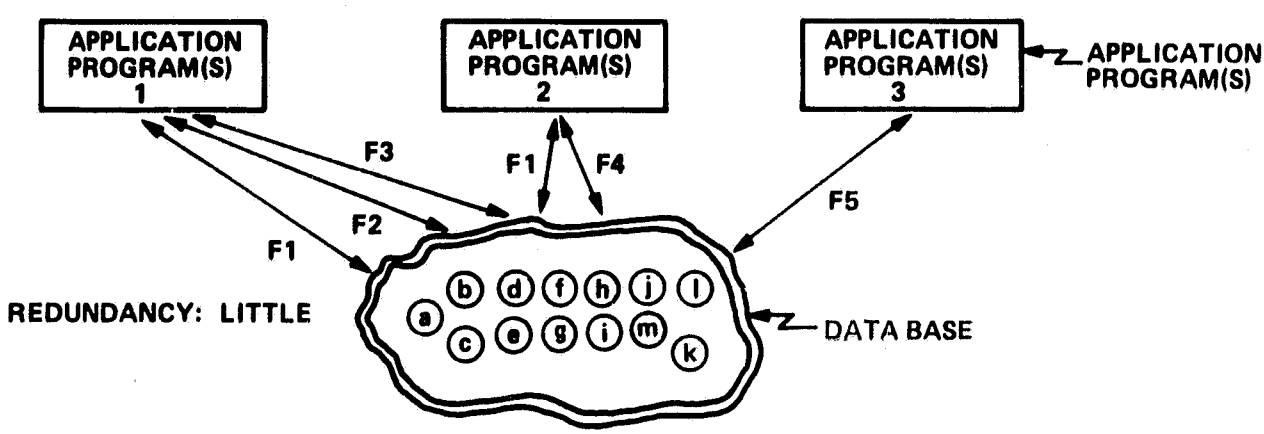


Figure 2-3b. The Data Base Approach to Programs and Data, Avoiding Much Redundancy of Data Stored [CRDN 79]

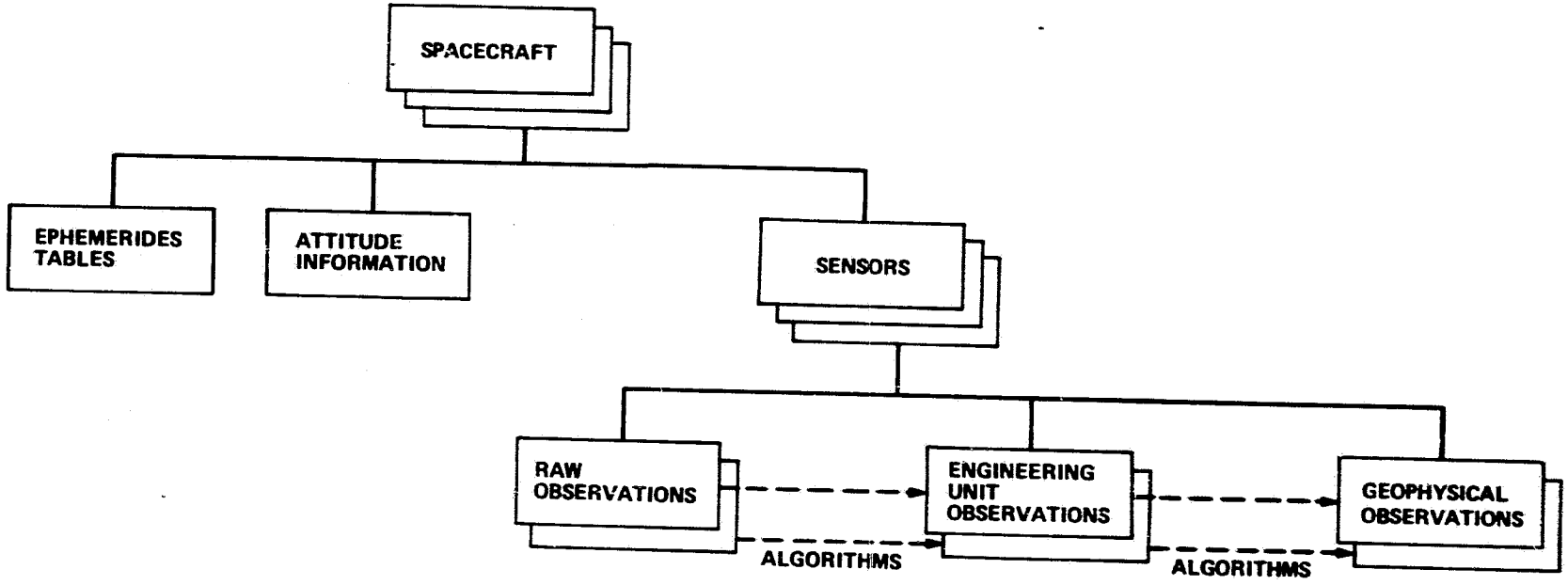


Figure 2-4. Potential Relationships Linking Spacecraft Data Files

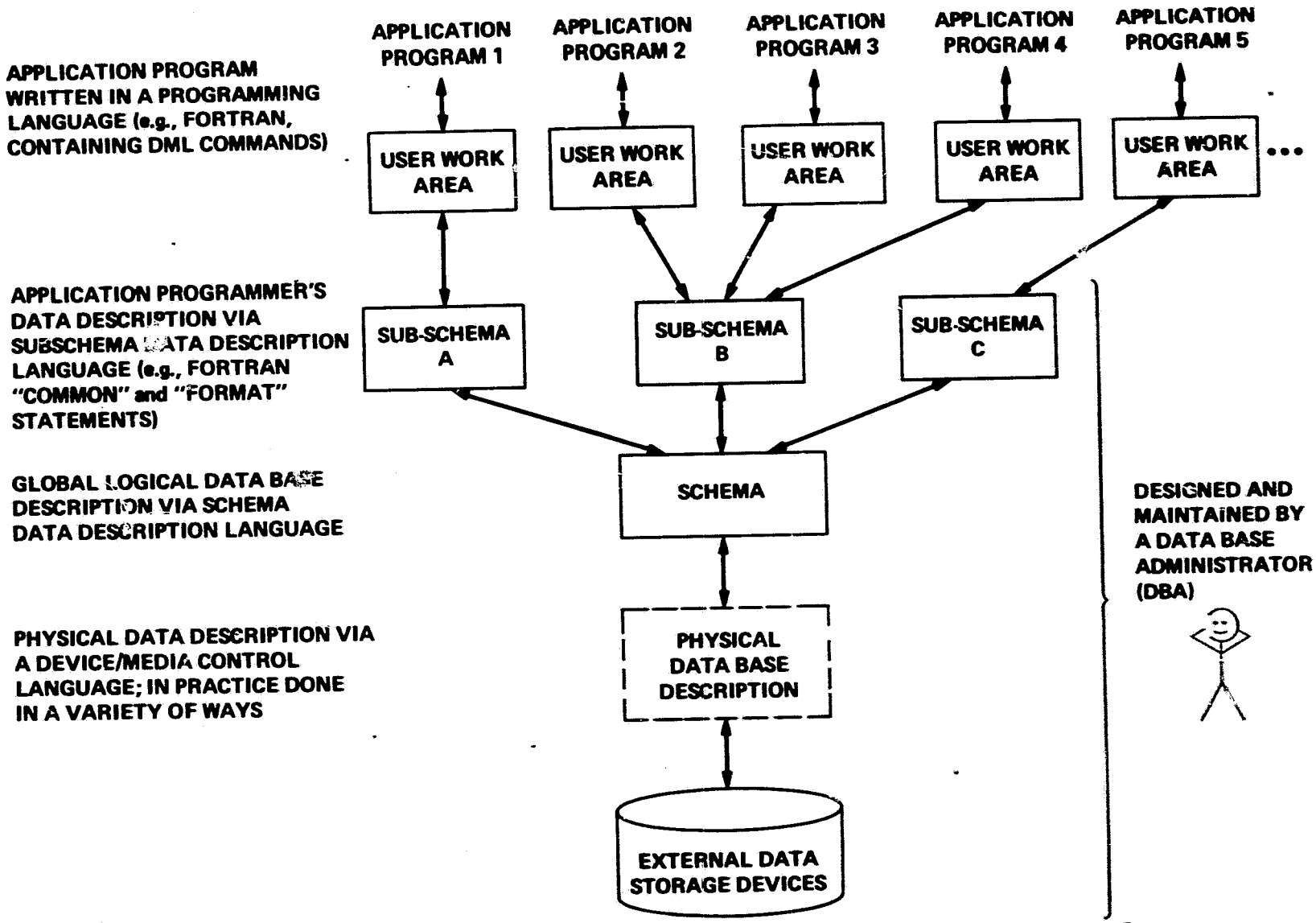


Figure 2-5. Architecture of a Data Base Management System

Capability C4: Integrity. Integrity refers to the correctness of data values and relationships in the data base. The DBMS capability to protect data base integrity is largely drawn from the previous two capabilities: by centralizing control over and relating various data items that were formerly dispersed, a DBMS can monitor for inconsistencies to known relationships, unreasonable values based upon other observations, etc. Furthermore, it can promulgate changes which affect related data items, some of which might not be known to the user who makes the change.

Capability C5: Access Flexibility. Programming languages, using the basic access mechanisms that are offered by the resident operating system, enable the user to efficiently access a file using a single key (e.g., record number). However, most user queries or requests for data from a user program specify multiple keys [OAOC 79, FJMT 81, LHMN 80A], for example: "Find all sea surface temperature data in the region 30N to 35N latitude and 142W to 147W longitude on 7 July 1977". This example has three distinct criteria or keys: geophysical parameter (sea surface temperature), region (latitude and longitude), and time (date). DBMSs permit access of any part of the data base on the basis of many possible access keys, in a way more efficient than generalized file management systems can.

Capability C6: Security. Because the DBMS alone is responsible for all access to the data base, it can assign, control, and remove the privilege of any user or application program to access any portion of the data base. Most DBMSs also have the capability to limit the functions that any individual may perform on a portion of the data base. For example, it can allow a principal investigator, as the creator and "owner" of a data file, to restrict access to that file to only himself and those whom he authorizes, and to restrict updating of that file to only his assistants. As with Capability C4, the DBMS can act as a "policeman", enforcing security constraints that protect the data from malicious or inadvertant intrusion.

Capability C7: Performance and Efficiency. As the data files in a data base grow, classical exhaustive searching for data of interest becomes much less efficient than the multi-key access mechanisms provided by a DBMS. Similarly, copying an entire file in order to change only 1% of its records is a more inefficient use of computer capacity than is the update-in-place capabilities provided by most DBMSs. Consolidation of routine data management operations into one module facilitates the optimization of the algorithms and machine statements which perform those operations. For example, the builders of VICAR found that by developing a centralized READ module that was tailored to the application, they could halve the time required to read a file using the standard FORTRAN READ command [APNY 81].

Capability C8: Data Base Structuring, Administration, and Control. The data base concept treats all data as a pool of information from which many different users draw out and return data. This global perspective includes defining the requirements and design of the entire data base, as well as administering and controlling the data base, for the overall good of the organization rather than according to the interests of a few individuals who create the files. In a data base environment, these functions can be vested in a single individual (or group) called the Data Base Administrator (DBA). The DBA can ensure, for example, that data files generated by different organizations (e.g., missions) are compatible and in a form most usable to the majority of users (e.g., all scientists in a discipline).

2.3 DBMS COSTS

The DBMS capabilities described above are not without their concommitant costs. The design of the internal structures in a centralized data base may favor some application programs at the expense of others. For example, a chronological organization of spacecraft data would favor those programs that process data for a single orbit of the spacecraft, at the expense of programs needing all data for a paticular geographic area. "Binning" the data according to geographical regions would, conversely, favor the latter over the former. However, this is a data base design issue that besets any data management system, not just a DBMS, whenever redundant data bases with different organizations are not permitted.

The creation and maintenance of the many inter-relationships and multiple access paths permitted by a DBMS impose both space and update time overheads. These overheads may vary from 10 to 20 percent depending upon the number of keys and relationships chosen for that application [CRDN 79].

While a general-purpose, flexible software package such as a DBMS reduces overall programmer time and cost to develop a given application system, the resulting system generally is less efficient with machine resources than is a system tailored to that application only. Furthermore, the ease of adding new applications using a DBMS often results in higher total system cost because more applications will be implemented [CRDN 79].

The cost-effectiveness of a DBMS approach must be judged upon the functions performed per life-cycle dollar expended, including manpower (programming) costs. With the rapidly increasing cost and scarcity of skilled computer personnel and the decreasing per-unit cost of computer hardware, DBMSs have proven cost-effective in the larger commercial applications.

2.4 CONCLUSIONS ON DBMS APPLICABILITY

The data management problems and inefficiencies that were discussed in Section 2.1 above have been found to be common to most NASA/OSTA and related data systems. This includes data at every level of processing. There exist examples of the exploitation of the DBMS capabilities outlined in Section 2.2 above on every level of data known to NASA, including Level 0 and image data [APNY 81].

However, DBMSs are best suited to applications where several of their capabilities are required. In NASA, these tend to be data distribution systems, characterized by occasional ad hoc accesses by a large number of users, based upon multiple access criteria that are inter-related. The discipline-oriented archival systems being developed as pilot systems for the Applications Data Service, as well as most of the systems discussed in [APNY 81], are excellent examples of this type of system.

Applications that are generally not suited to DBMS use include special-purpose, real-time, and/or "pipeline" data processing systems such as the Image Processing Facility at Goddard Space Flight Center. Such applications typically apply a fixed algorithm to a continuous stream of input data, rather than having a variety of application programs that have different and changing requirements for portions of the data base that are selected based upon more than one key. With these production-oriented systems, it is unlikely that the flexibility of access offered by a DBMS would be worth its expense.

SECTION 3

OVERVIEW OF REQUIREMENTS

"No craftsman, if he aspires to the highest work in his profession, will accept [inferior] tools; and no employer, if he appreciates the quality of work, will ask a craftsman to accept them."
-- Weinberg

"If the programmer is working in a language that allows only three dimensions, we are not likely to observe more than three."
-- Weinberg

This section gives an overview of the ADAM functional requirements which are detailed in the rest of this document, as they relate to the perceived user requirements for data management [FJMT 81].

Throughout the remainder of this document, requirements of the form "The ADAM System shall..." may be replaced by "The ADAM System shall, or shall be conveniently modifiable or augmentable to,...". No single DBMS can satisfy all of these requirements at present, but virtually all of the following requirements are satisfied by at least one existing DBMS.

3.1 DBMS FOR SCIENTIFIC OBSERVATIONAL DATA

NASA applications users require simpler, more standardized, more rapid, and more flexible access to observational data that is residing in diverse formats at diverse locations [BRCK 80, DJRD 79, FJMT 81, LHMN 80A, LHMN 80B, LOTS 79A, OAOC 79, OFNS 79]. As discussed in Section 2, most of these objectives are met by the use of a data base management system (DBMS). And most existing DBMSs are designed for applications to the commercial rather than to the scientific sphere.

The Applications DAtabase Management System (ADAM) shall perform as a software tool for access to public archives and private files containing scientific and engineering data. The ADAM is intended to perform routine data management functions that are common to most data management systems of near-earth remote sensing missions of NASA's Office of Space and Terrestrial Applications (OSTA) and related institutions. It will not accomplish any specialized processing that may be under development by, and/or peculiar to, any particular application, although it should provide support and software interfaces to

such routines. The data which it manages may include raw bit streams (Level 0 data) as well as any level of processed data or data products (data Levels 1, 2, 3,...).*

3.2 MULTI-MISSION TOOL

Currently every mission designs and builds its own data management system from scratch [APNY 81, BRNS 80]. The cost of doing this will soon become prohibitive, and is the major impetus for developing multi-mission information systems.

The ADAM is intended to be a tool that may be readily employed in, and adapted to, the data systems of most future OSTA projects; it is not to be a "turn-key" component of any extant or proposed mission. As such, ADAM must be:

Transportable - The ADAM software may be implemented on different brands of computer hardware having different operating systems and assembler languages. Hence the ADAM should itself be written in a standardized, transportable, high-level language and should minimize and concentrate in minimal set of modules its interface with any operating system.

Modular and Expandable - A minimal core of essential data management functions should form the nucleus of the ADAM, and shall be implementable at least on minicomputers or larger machines. Additional capabilities such as telecommunications and image data handling shall be readily appendable as modular enhancements to this system nucleus.

Flexible - Changes to system parameters, commands, data structures, formats, functions, etc., must be simple to accomplish, have minimal impact on other ADAM components, and must not require inordinate amounts of human or computer resources. Both public archives and private, user-defined data bases may be maintained by ADAM.

3.3 USER-FRIENDLY

There is a wide variety of potential users of OSTA data, ranging from programmers who are developing programs to scientists who are unfamiliar with computers and occasionally wish to find some data that is correlative to their own [APNY 81, FJMT 81].

The ADAM must therefore be easy for the programmer and non-programmer alike to use. The system should be easy for the uninitiated user to learn. It should be able to free the user from the often confusing and repetitious tasks of data

* For a definition of these data levels, see [DJRD 79].

buffering, file management, disk and tape mounts/dismounts/positioning, label processing, and applications or analysis program parameter processing.

The command language of ADAM must be a high-level, English-like set of verb-object predicates that are easy for users to understand. Interactive, ad hoc queries as well as routine requests for data from application programs written in any of several standard "host" programming languages (such as FORTRAN, PL/I, Pascal, Basic, etc.) should be expressible with these commands. In formulating an interactive query, the user should have the option of typing in the necessary commands or choosing from a "menu" of commands he may execute. The user and the user's application programs shall not be required to know the precise physical location or physical format(s) of the data accessed. Rather, data items shall be referenceable by specifying only the logical name assigned to that data item, with ADAM responsible for physically finding and accessing the referenced items.

3.4 EXTENSIVE AND INTEGRATED INFORMATION ABOUT DATA

Scientific users generally require access to the "pedigree" of data in a data base: who collected it, when, where, with what instrument, and how it has been processed. Ways to search, scan, and select this information electronically are needed to help the user find data relevant to his application [DJRD 79, LHMN 80A, FJMT 81]. In addition, the data manager and application programmers need to know how system programs, data, and users are inter-related in order to assess the impact of changes to any of these system entities.

Extensive capabilities shall therefore be integrated into the ADAM to store and maintain in machine-processable form information about data in the data base, whether or not that data is itself accessible through ADAM (e.g., non-digital data products). These capabilities shall comprise a Data Dictionary that provides documentation in machine-processable form pertaining to each program or module which accesses the data base, and to each data item or element -- the smallest data unit in the data base that can be accessed. The data dictionary should itself be an integrated data base that is managed by the ADAM (see, for example [JHNS 80]). Entries for data items must contain all information about each item that is relevant to its use: its origin, spatial and temporal extent and resolution, how it was collected and processed, what programs use it, and what it means in physical terms [GARY 80]. Similar attributes shall be stored for each program or module of a data system, and potentially for each user. Finally, aggregations of these items, such as an entire file or data set, should have entries in the Data Dictionary that describe their characteristics and constituents. For details, see Function 1.4 of Appendix A.

3.5 ROBUST LANGUAGES FOR DEFINING DATA STRUCTURES AND FORMATS

Users should not have to concern themselves with the details of data formats, device protocols, etc. Users' application programs should not be impacted by changes to the physical arrangement of the data that they access or by changes to the storage devices on which that data resides.

Data managers and some users need to describe to the system the logical data structure, the user-specific formats, the physical data structure, and the actual layout on a particular storage device. Separate specification of these four aspects, three of which are outlined in the widely accepted ANSI/SPARC Conceptual Framework [URNA 81B], ensures the data independence required by users (see also Section 2).

The ADAM must therefore have commands that enable a data manager or knowledgeable user to define these four distinct aspects of each file in the data base. Each aspect requires a simple, high-level language that is suited to the definition of the appropriate parameters. The languages are:

- Data Definition Language (DDL) to specify the logical format, or "schema", of the data base, which gives the organization of the data base; data item names, types, and sizes, etc. This capability must be far more robust than the standard FORTRAN data definition capability.

- User View (Subschema) Definition Language to describe user-specific subsets of the data base, renaming of data items, and other format differences from the global logical format ("user views") that are easier for users or users' application programs to understand.

- Physical Storage Organization Definition Language to establish the physical arrangement of data in a way that is independent of the device on which it is stored but permits enforcing physical contiguity of related data.

- Physical Device Format and Protocol Definition Language to detail the parameters peculiar to the actual device(s) on which the data may reside in order to optimize storage and retrieval of that data.

Ideally, the user should be able to have as many layers of schemata as are necessary to achieve data independence, and the various definition languages should share common constructs wherever possible for simplicity [JHNS 80]. For detailed specifications on these languages, see Function 1 of Appendix A.

3.6 SCIENTIFIC DATA TYPES AND STRUCTURES

Scientific data is numerically-oriented, and often is generated in binary form by sensors. Users need to store images, ancillary descriptive information, *in situ* observations, system flags, results of calculations, vectors, arrays, and many other varied types of data items [APNY 81, ECSY 79, ERTH 76, LOTS 79B, ODBA 79].

The ADAM shall therefore be required to define, store, retrieve, and update efficiently at least the following atomic data types:

- Fixed point binary numbers, including "double precision".
- Floating point numbers stored efficiently in scientific notation, i.e., signed mantissa and exponent, not as a character string.
- Alphanumeric (or "character") strings, including variable-length strings possibly of great length (e.g., paragraphs of explanatory text).
- Bit strings, which efficiently store binary data in a form that permits access at the bit level.
- Boolean or logical variables that indicate "true" or "false" conditions.

These data types may be either constants or named variables. Ideally, the ADAM should permit user-defined data types with optional listing of valid values (e.g., data type = DAY OF WEEK, values = MONDAY, TUESDAY, ..., SUNDAY).

The user or data manager must be able to define data structures (using the languages described in Section 3.5) that aggregate these atomic data types into multi-dimensional arrays (matrices), hierarchical structures, and/or records. These structures may be of either fixed or variable length. No limit should be placed on the maximum size of any logical constructs, such as fields, records, or files. For example, limiting the maximum length of a field to 255 characters is unacceptable. To illustrate, a single image in raster format is stored as a 2-dimensional array that may well exceed 1 megabyte (1024 x 1024 picture elements), plus header and ancillary information [BLSR 80].

3.7 FLEXIBLE PHYSICAL ACCESS MECHANISMS

NASA/OSTA data bases contain highly inter-related data and unstructured data such as text strings, both of which must be efficiently searched. In addition, the rates at which records

are updated, added, deleted, and queried will differ depending upon the application [APNY 81].

Therefore, the ADAM should support several physical access mechanisms that may be selected by the data base designer through the Physical Storage Organization Definition Language (see Section 3.5). These access mechanism options should include any combination of: (1) indexes using a variety of B-trees or other easily balanced tree, (2) calculated derivation of a key by either generalized hashing or by an application-specific program (e.g., using satellite ephemeris data to convert latitude and longitude to time of overflight), and (3) efficient text pattern searching.

Users should be able to define keys composed of concatenated logical fields. For example, the key field TIME could be defined to be a concatenation of individual fields called YEAR, JULIAN-DAY, HOUR, MINUTE, and SECOND.

3.8 SPATIAL RELATIONSHIPS

A common thread linking all NASA/OSTA data bases is the spatial orientation of the data. In virtually all applications, users must be able to locate observations relative to some subset of space and time. Most commonly, especially for images, this takes the form of the two-dimensional (locally flattened) surface of the earth, often keyed by latitude and longitude. However, global data sets, and oceanographic and atmospheric data, must be represented in 3-space, plus time [APNY 81]. Data management systems that structure data primarily to represent spatial relationships (e.g., distance, contiguity or connectedness, overlap, etc.) and to key upon spatial location are often called Geographic Information Systems. Existing ones tend to be highly specialized towards a particular application, e.g., resource assessment using Landsat imagery [BRYN 76, MRBL 77, NAGY 79, CHCK 81]. However, preserving spatial relationships is also important to data bases containing data other than geographic information, most notably engineering data for computer-aided design and computer-aided manufacturing (CAD/CAM) [DUBE 80].

The ADAM System must be able to support Geographic Information Systems applications by providing data structures that facilitate the representation of spatial relationships, time, and other dimensions of the data. Spatially oriented data to be stored by ADAM may be represented in any combination of raster format (i.e., a matrix of picture elements to be scanned in row-major order) or in vector-graphic form (e.g., points, lines, and polygons represented as variable-length lists of (x, y,) coordinates) [BRYN 76, DUBE 80]. These structures and the query capabilities of ADAM should permit the efficient retrieval of data related by at least the following relationships:

(1) Distance: All data within a certain radial distance of a given location.

(2) Direction: Data in a certain direction from a given location, e.g., "east of".

(3) Connectedness: Whether lines form a single closed region, and if so, the associated attributes of that region such as area enclosed.

(4) Overlap: Whether two intervals or regions overlap, and if so, by how much. High-level constructs in the query language should facilitate queries such as: "Retrieve all data blocks whose range of times intersects the interval (t_1, t_2)"; the user should not have to specify such relationships in complex Boolean combinations using the standard "<" and ">" relational operators commonly available in existing DBMS.

At present, these relationships are not easily representable in any known DBMS.

3.9 LANGUAGE FOR EFFICIENTLY MANIPULATING DATA

Users need to load, store, change, find, access and/or display the data in the data base for their specific application. These data manipulation functions are found in one form or another in virtually all data management systems [APNY 81]. Users need to be able to access data readily, without significant programming effort. Simple queries should be easy to pose and quick to satisfy, but more complicated queries by knowledgeable users must also be possible. Users unfamiliar with the system should be guided through a set of menus, whereas accomplished users should have the option of bypassing the menu through a command language.

Thus, ADAM must permit the user to manipulate data readily, either item-by-item or *en masse*, using a simple, high-level non-procedural language that is easy for scientific users to understand and use (see Section 3.3). The ADAM System shall have an interactive query processor that performs the same basic operations performed by the data manipulation language. The user should be able to store, retrieve, and edit sets of query commands, and ideally should be able to "program" in this language by defining variables, expressions, branching, etc. Queries upon non-keyed as well as keyed fields should be possible, and data base access should be automatically optimized accordingly. The query processor should inform the user of the expected amount of data to be retrieved before significant retrieval is accomplished, and should allow the user to narrow his query by adding additional Boolean constraints. While retrieval of data is in progress, users should be able to

initiate other routines in parallel, determine the status of his query, and/or cancel further retrieval without losing the data retrieved to date. Ideally, the same high-level, non-procedure language will be used for both the interactive query processor and the application program interface, so that applications can be developed first interactively and then migrate to application programs when it is sufficiently well defined.

The required functions are those provided by any existing commercial DBMS. They include those to load the data base, to edit and input new data, to find and access selected data, to rearrange or aggregate data items, to modify the content of data items, to delete unneeded data, and to display data. For a detailed specification of the required functions, the reader is referred to Function 3 of Appendix A.

3.10 DATA BASE MAINTENANCE

Data managers need utilities for protecting and maintaining the data base for the users [APNY 81]. The ADAM System must provide these utilities in a way that can remain transparent to the user. They include the functions of data security, backup and recovery, ensuring data base quality and integrity, the allocation and deallocation of storage space, monitoring and tuning system performance, providing estimates and accounting of system costs, and providing efficient but flexible interfaces to system programs that invoke or are invoked by the ADAM (see Function 2 of Appendix A for details).

3.11 HIGH-RATE INPUT/OUTPUT AND LARGE-VOLUME STORAGE

Earth-orbiting satellites are returning ever larger amounts of data at alarming rates. It has been projected that yearly data volumes from earth resource observation satellites will increase from 10^{13} to 10^{16} bits per year by 1995. The data rates of new instruments will increase from the 15 megabits per second for the current multi-spectral scanner instrument, to 85 megabits per second for the Thematic Mapper, to rates of 200 to 400 megabits per second for Multiple Linear Array and Synthetic Aperture Radar instruments in the late 1980's [BRCK 80]. Present plans for the Thematic Mapper data call for processing and archiving approximately one hundred frames, each composed of 6 spectral images of over 2×10^9 bits each, or over one terabit, per day [BLLN 81].

Therefore, ADAM must, with the appropriate enhancements, be capable of extracting any subset of a large (10^{12} - 10^{15} bytes) data base, by multiple keys. This should require a time proportional to the size of the output data set rather than the size of the data base. Users must be given an estimate of the cost (in terms of disk accesses required) to perform any

significant data retrieval that is requested before the retrieval is performed. The results of a query should itself be a data base that can be further queried. This is so that a user can iteratively narrow the search for relevant data by ANDing additional Boolean conditions with previous conditions without having to re-search the entire data base.

The ADAM must be capable of loading data from the above instruments into the data base, and establishing multiple-key relationships, fast enough to avoid backlog, while still allowing users sufficient time to access the data base. See Section 5 for more discussion of the ADAM performance requirements.

3.12 DISTRIBUTED DATA BASE NETWORKS AND DATA BASE MACHINES

Data which users require is currently distributed among many different data bases and files on differing computer types at various locations across the country [APNY 81, FJMT 81]. Some of these data bases are beginning to be networked together. In particular, the ADAM is intended to be implemented in installations which include the nodes of the proposed Applications Data Service network [OFNS 79, DJRD 79]. A possible configuration of such a network is given in Figure 3-1, showing the heterogeneity of computer architectures and communication speeds as well as the replication of ADAM at certain nodes.

With the appropriate enhancements, ADAM should be capable of operating as one node in a distributed heterogeneous data base network, and/or in a distributed processing network. In the former, request commands for data in the data base of one installation may come to its ADAM from a remote user through his ADAM and an inter-computer network joining the two installations [CHUW 79]. The latter type of network configuration permits the ADAM to reside on a processor different from the processor in which the user's application program is executing. The processors in which ADAMs are executing may thus be specialized "data base machines" that are optimally configured for performing their data management chores [SUSY 80].

3.13 WELL DESIGNED AND SUPPORTED

Inevitably the requirements which ADAM addresses will change and ADAM will have to be maintained and augmented [ZLKW 78, BOHM 76]. The ADAM System software must, therefore, be maintainable and well supported throughout its lifetime by complete documentation, knowledgeable and available user consultants, development programmers, and system programmers capable of resolving ADAM "bugs" or anomalies. It must be a field-proven software package that is well-structured, modular, and uses

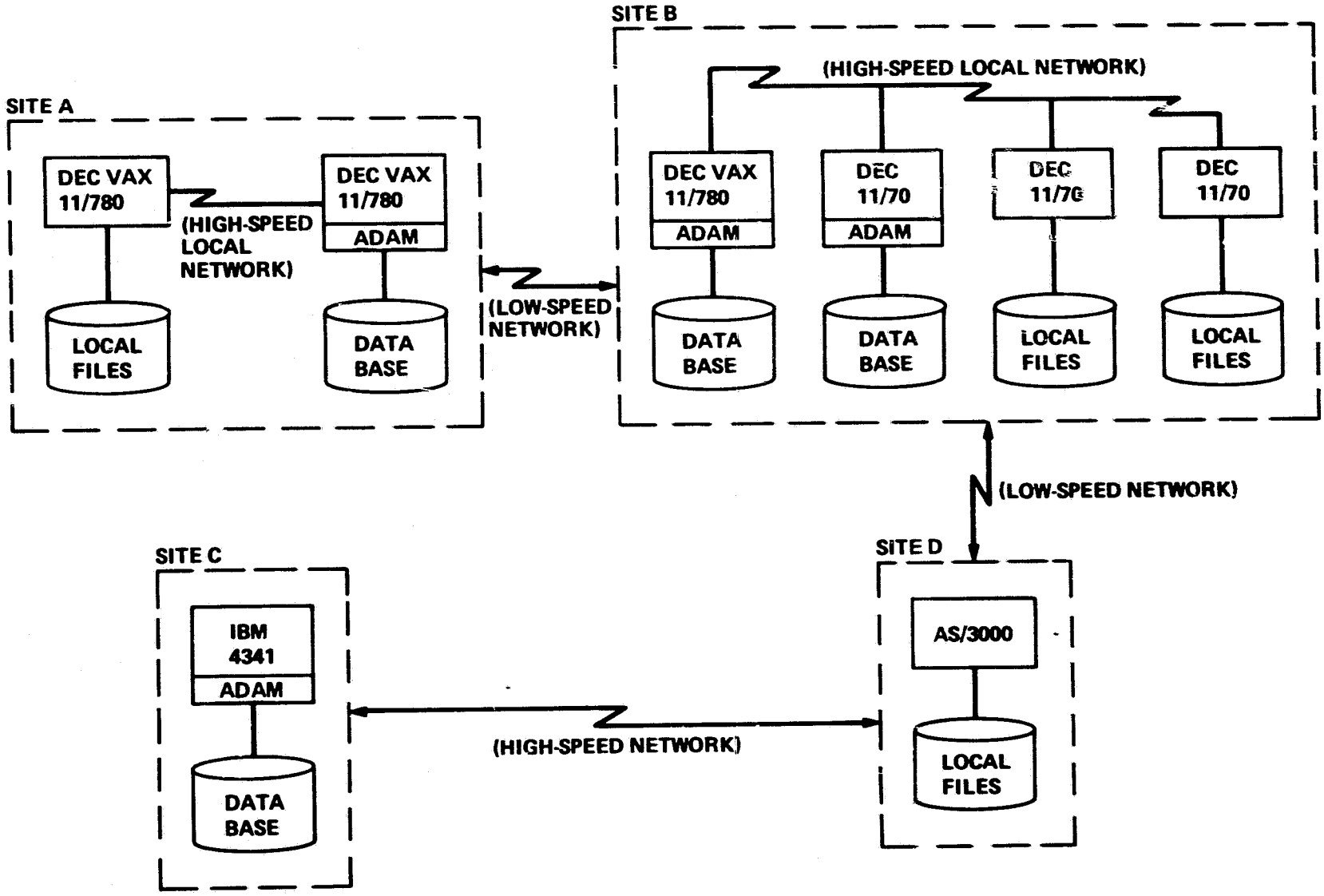


Figure 3-1. Example of Use of ADAM in a Distributed Network Exhibiting Heterogeneous Processors and Communication Link Speeds

accepted software engineering principles. For more discussion of these requirements, see Sections 4 and 6.

SECTION 4

IMPLEMENTATION CHARACTERISTICS

"Neither function alone nor simplicity alone defines a good design"
-- Brooks

The ADAM System should be implemented in a particular way in order to meet certain user requirements. These characteristics are described in this section.

4.1 FLEXIBLE AND ADAPTABLE

As described in Section 1, ADAM is intended to be a general-purpose tool that will be utilized by differing applications, many of which are research or development oriented [FJMT 81]. Hence the ADAM must be adaptable to different applications as well as to requirements varying over time for any given application (e.g., changing data management requirements for a flight project using ADAM).

More specifically, ADAM should:

(1) Permit the addition of functional capabilities or data types (e.g., image data types) to a nucleus of basic functional capabilities and data types without requiring a complete system generation (see also Section 4.2), either as add-on modules of the ADAM or as user-supplied software.

(2) Permit the Data Base Administrator to define new functions and data types without requiring the recompilation of existing commands or schemata.

(3) Have all system parameters specified as variables with default values which may be altered by the Data Base Administrator.

(4) Allow the Data Base Administrator (or users, where authorized by the DBA) to make changes to: data item sizes for storage or display, search key designations, data structures, formats, storage media locations, indexes, data dictionary entries, integrity constraints, security authorizations, and any other specifications. This should be done in a way that is simple to accomplish, has minimal impact on other ADAM components, and requires minimal human and computer resources.

(5) Be able to map user-defined variables in a user view (or "subschema") to the global logical structure (schema) in a

dynamic way; for example, without requiring modification or recompilation of an application program whenever the schema changes the characteristics of data items not used by that program.

Most of these changes should be able to be made dynamically while the system is running.

4.2 EXPANDABLE

Not all users will require the full complement of ADAM capabilities that some users will require -- the requirements and hardware resources of different projects vary considerably [FJMT 81]. Therefore, it should be possible to augment the nucleus of ADAM with additional capabilities required by a particular user. For example, sophisticated graphics and/or report writing capabilities are not required by all users, and can add considerable overhead to ADAM (in the form of memory required to execute) even if they are not used. Addition of these optional capabilities, either as modules of the ADAM or as user-supplied routines, should be easy for the Data Base Administrator to implement and should not require system regeneration.

4.3 TOP-DOWN, MODULAR ARCHITECTURE

The requirement of Section 4.2 implies the need for a top-down, modular software architecture with standardized interfaces between modules, as is illustrated in Appendix A. Modules should be organized along functional lines with the goal of minimizing the impact that any changes to a module might have to related modules.

4.4 TRANSPORTABLE

The ADAM System must be transportable between different computers and different operating systems to the maximum extent possible. The ADAM will be implemented on many different computers, including potentially those of Digital Equipment Corporation, IBM, and Univac. These machines may be either minicomputers or large mainframes, and may vary in their word size, machine language, internal representations, etc. Similar machines may operate under different operating systems. Any one given computer may upgrade or change operating systems over time. New versions of computer hardware and operating systems are inevitable. Therefore, this need for a transportable design of the ADAM software is a stronger requirement than one which specifies the particular machines to which ADAM must be adapted.

To achieve transportability, ADAM should:

(1) be written largely in a high-level language;

(2) concentrate all interfaces with the operating system, and all use of machine language commands, to a small set of well-documented modules. This requirement reinforces the requirement of Section 4.3 for modular software design;

(3) not require any modifications (other than tuning) to the host operating system; and

(4) have no globally reserved keywords.

See [ALLC 80A, ALLC 80B] for an example of an existing file access system that is based upon these principles and is implemented in the Pascal language. Another strategy, using a very high level "macro" language that can be processed into system-specific FORTRAN commands is described in [SCNC 80].

4.5 INTERFACEABLE

The ADAM should have interfaces that are simple, concentrated in a minimal set of modules, and transparent to the user. Interfaces to the following software systems is a minimal requirement:

(1) Operating System. As described in the requirement of Section 4.4, the ADAM interface with the machine's operating system should be concentrated into a small set of modules that can be customized as needed for different computers and/or different operating systems. See Function 4.2 in Appendix A for details.

(2) Language Compilers. Programs written in FORTRAN and machine language shall be callable from the ADAM. Conversely, ADAM functions shall be invokable from application programs written in these languages, either by utilizing CALLS to ADAM and passing functions and their operands as parameters, by interpreting ADAM commands into the host language with a pre-processor, or using any other efficient technique. Interfaces to other programming languages, such as PL/I and Pascal, are highly desirable due to their commands that are high-level, English-like, and consistent with structured programming principles [ANDR 80].

(3) VICAR. As part of the image-handling augmentation to the ADAM, there shall be an interface to the Video Image Communication And Retrieval (VICAR) system that was developed by the Jet Propulsion Laboratory [SDMN 79] and is available in the open domain through COSMIC at the University of Georgia.

(4) TAE. Optionally, ADAM should be able to interface with the Transportable Applications Executive developed by the Goddard Space Flight Center. To see how this is being done with one DBMS, see [BCHM 81].

SECTION 5

PERFORMANCE

As stated in Section 3, absolute performance specifications are dependent upon the data base characteristics and the user requirements for a given application. It is therefore impossible to define quantifiable, absolute requirements upon a general-purpose package such as the ADAM. Furthermore, NASA/OSTA experience with this technology is still quite limited. Experience with prototype applications on the Applications Data Service pilot systems at Goddard Space Flight Center, Johnson Space Center, and Jet Propulsion Laboratory within the next year should help to focus DBMS performance requirements. At a minimum, ADAM shall be required to perform in benchmark tests at least as well as the DBMSs used by these pilot systems, as well as that used by the DBMS-based JPL Mark IV Data Records System, and better than an identical application implemented using a generalized file management system.

Performance superior to the basic requirements of this section is highly desirable and should be weighted heavily in any evaluation of compliance with these requirements.

A brief description of each of the major categories of DBMS performance requirements follows. Some of the following requirements are more important than others. For example, it is anticipated that in most NASA/OSTA applications, large volumes of new data will be loaded into the data base and periodically retrieved, with relatively little updating of the data base. Therefore, Requirement 5.2 is secondary to Requirements 5.1 and 5.3.

5.1 MASS DATA RECEIPT AND LOAD

The development of sensors which generate data at rates ranging from 85 to 400 megabits per second (Mbps) is predicted by NASA/OSTA for the decade in which the ADAM should be operational [BRCK 80]. Applications in which multiple users are simultaneously retrieving images -- each image composed of the order of millions of bytes -- from the data base will pose similar requirements. Already the NASA End-to-End Data System

(NEEDS) Project is developing an archival mass storage device which is capable of accepting data at the rate of 100 Mbps (albeit qualified by a 20% duty cycle) [THMS 80]. Therefore, the ADAM must for some applications be capable of being augmented or enhanced in such a way that it can accept and load data into the data base and maintain the needed multiple keys at these rates, given the machine cycle times and input/output rates expected from hardware technology advances for the same period. Furthermore, it is likely that cost-effective random-access storage for these large volumes of data, most notably digital optical disc media that unfortunately are currently write-once, will not be commercially available for several years [SBSO 80]. Hence, access to some portions of any data base may require the unloading and re-loading of data from a magnetic tape archive to the data base on magnetic disk on a routine basis. This near-term eventuality further underscores the need for efficient loading by ADAM.

5.2 SELECTED DATA STORE/UPDATE

The ADAM overhead which is used to facilitate the location and retrieval of data from the data base shall not place undue burden upon functions which update or insert new data into the data base. For example, the maintenance of inverted lists for many keys needed by users, as is done by Intel's System 2000/80 can significantly increase both the time needed to alter the data base and the space consumed by the inverted lists. However, analysis of existing NASA/OSTA and related data management systems suggests that volatility of these and future data bases is likely to be quite low in ADAM environments other than algorithm development facilities (see Section 1.3).

5.3 SELECTED DATA RETRIEVAL AND DISPLAY

User requirements for data delivery from the processing center to data distribution centers have been projected to increase from the order of days in the 1980s to the order of hours or minutes in the 1990s [BRCK 80]. In the anticipated environments of the ADAM, where the user will be directly interacting with the data base, retrieval and display of selected data will have to be accomplished within seconds to retain an effective interactive user interface that is conducive to scientific inquiry. Interactive users are unwilling to wait minutes or hours for a few lines of data or an image, and usually assume a system failure has occurred if they are not warned every 5 to 10 seconds during long operations. Therefore, the ADAM must be capable of retrieving and displaying data expeditiously, including images comprising tens of millions of bits.

5.4 CONCURRENT USE

The ADAM System shall be capable of concurrent use by at least 8 to 16 users without significant degradation of system response time. A user should be able to execute non-ADAM tasks and lengthy ADAM operations in parallel.

5.5 DATA BASE REORGANIZE/REDEFINE

Respecification by the Data Base Administrator (DBA) of the global logical structure or either component of the physical data structure may result in ADAM having to reorganize the data base, either to retain data base consistency with the new definitions or to re-optimize system performance. Actual reorganization of the data base shall be under the control of the DBA, including an option that will permit the DBA to defer portions of the reorganization, and shall have minimal impact on system operability.

5.6 RELIABILITY

The impact of ADAM and/or system failures upon the integrity and/or operability of the data base shall be minimized through the efficient implementation of Function 2.2 (Backup and Recovery), and by satisfying the support requirements of Section 6 throughout the lifetime of the ADAM. ADAM errors that require re-creation of an entire data base must be more rare than errors that effect a single retrieval.

SECTION 6

SUPPORT

"Documentation is the castor oil of programming...
the managers know it must be good because
programmers hate it so much."
-- Weinberg

The NASA/OSTA requirements for data base management will continue to evolve over time after the initial development of the ADAM, due to changing computer technology, sensor technology, mission objectives, and user requirements [BRCK 80]. Therefore, it is imperative that ADAM be fully supported by a viable, on-going organization which can aid the implementation and augmentation of ADAM for new applications.

6.1 FULLY SUPPORTED SOFTWARE

The ADAM shall consist of software which is guaranteed to be supported by the developing organization for not less than ten years. This support shall include (but not be limited to):

- Thorough documentation, to include at least a system design document, a user's manual, a system reference manual/programmer's guide, a management executive summary, complete internal documentation of all code, a DBA guide/tuning guide, an installation guide for each operating system, and an operator's diagnostic manual. Such documentation shall meet or exceed the JPL internal software standard practices requirements [JPLS 80].

- User consultants who are intimately familiar with every aspect of the system's design, structure, limitations, known problems or workarounds, operation, commands, and debugging.

- System programmers who are capable of diagnosing and correcting system failures within a short period of time (on the order of hours, not days).

- Development programmers who are capable of designing, developing, debugging, testing, and evaluating needed enhancements to the ADAM, as determined jointly by NASA/OSTA and the supporting organization.

6.2 NEW NASA COMPUTER TYPES

Support of ADAM shall include adapting the system to new computers upon which NASA/OSTA wishes to implement the ADAM, including new models or other computers not now in existence, as mutually agreed upon by NASA/OSTA and the supporting

organization. Satisfaction of this requirement should be facilitated by the Implementation Characteristics Requirements 4.1 (Flexible and Adaptable), 4.3 (Top-Down, Modular Architecture), and especially 4.4 (Transportable).

6.3 DIRECT SUPPORT TO MAJOR MISSIONS

In addition to the standard support described in Support Requirement 6.1, certain major future missions may require supplemental support to meet mission-specific schedules, user requirements, operational readiness, or other requirements. This may include the need for on-site and/or round-the-clock availability of support personnel, as determined by NASA/OSTA.

SECTION 7

REFERENCES

[ALLC 80A] Allchin, J. E., et al., "FLASH: A Language-Independent, Portable File Access System", Proceedings of ACM-SIGMOD 1980 International Conference on Management of Data, Chen, Peter P. and Sprowls, R. Clay, editors, ACM Order No. 472800, ACM, New York, 1980.

[ALLC 80B] Allchin, James E., "FLASH: A Language Independent, Portable File Access System, User's Guide", Internal Document, Stanford University, Stanford, CA, 1980.

[ANDR 80] Anderson, Barbara V., Jet Propulsion Laboratory, Pasadena, CA, Personal Communication, 26 August 1980.

[APNY 81] Apenyo, Kofi, Survey of Representative Database Systems, JPL Internal Memorandum (under preparation), Jet Propulsion Laboratory, Pasadena, CA.

[BCHM 81] Bachman, P. W., Helfer, D. P., and Herath, L. W., "Functional Interface Agreement: TAE, ADS, and PCDBMS Projects", Goddard Space Flight Center, Greenbelt, MD, May 1981.

[BLCK 80] Blackwell, Richard J., Jet Propulsion Laboratory, Pasadena, CA, Personal Communication, 27 October 1980.

[BLLN 81] Billingsley, Fred C., Jet Propulsion Laboratory, Pasadena, CA, Personal Communication, 3 September 1981.

[BLSR 80] Blaser, A. (ed.), Data Base Techniques for Pictorial Applications, Florence, Italy, 20-22 June 1979, Lecture Notes in Computer Science, Vol. 81, Goos, G. and Hartmanis, J. (eds.), Springer-Verlag, New York, 1980.

[BOHM 76] Boehm, Barry W., "Software Engineering", IEEE Transactions on Computers C-25, 12 (December 1976), pp. 1226-1241.

[BRCK 80] Bracken, Peter A., "Earth Observation Data Systems in the 1980's, Proc. of the 1980 Annual

Meeting of the American Astronautical Society, Boston, MA, 20-23 Oct. 1980.

[BRDK 80] Bredekamp, J., "NASA Data Base Systems (as of 1978)", Personal Communication, 2 April 1980.

[BRNS 80] Bernstein, Ralph (Chairman), Data Management and Computation: Issues and Recommendations, Draft 7, Committee on Data Management and Computation, Space Science Board, National Academy of Sciences, Washington, DC, September 1980.

[BRWN 80] Brown, James, Jet Propulsion Laboratory, Pasadena, CA, Personal Communication, 22 October 1980.

[BRYN 76] Bryant, Nevin A. and Zobrist, Albert L., "IBIS: A geographic information system based on digital image processing and image raster datatype", Machine Processing of Remotely Sensed Data, IEEE, 1976, pp. 1A-1 to 1A-7; also, IEEE Transactions on Geoscience Electronics, Vol. GE-15, No. 3 (July 1977), pp. 152-159.

[BRYN 79A] Bryant, Nevin, Data Base Management Systems Panel Workshop: Executive Summary, JPL Publication 79-70, Jet Propulsion Laboratory, Pasadena, CA, 1 August 1979.

[BRYN 79B] Bryant, Nevin, Final Report of and Supporting Research to: Data Base Management Systems Panel Workshop, Internal Document, Jet Propulsion Laboratory, Pasadena, CA, December 1979.

[BURN 80] Burns, Thomas E., The Mitre Corporation, McLean, VA, Presentation and Personal Communication, 10 November 1980.

[CDSY 71] CODASYL Systems Committee, Data Base Task Group Report, ACM, New York, April 1971.

[CHCK 81] Chock, Margaret, Cardenas, Alfonso F., and Klinger, Allen, "Manipulating Data Structures in Pictorial Information Systems", working paper (to appear in Nov. 1981 Computer), Computer Science Department, UCLA, Los Angeles, CA, June 1981.

[CHUW 79] Chu, Wesley W. and Chen, Peter P., Tutorial: Centralized and Distributed Data Base Systems, IEEE Catalog No. EH0154-5, IEEE, Long Beach, CA, October 1979.

[CRDN 79] Cardenas, Alfonso F., Data Base Management Systems, Allyn and Bacon, Boston, 1979.

[DATE 77] Date, C. J., An Introduction to Database Systems, Second edition, Addison Wesley, Reading, MA, 1977.

[DJRD 79] DesJardins, Richard (ed.), OSTA Data Systems Planning Workshop Report Draft 2, ADS Study Office, Goddard Space Flight Center, Greenbelt, MD, 20 November 1979.

[DUBE 80] Dube, Peter R., et al., "An Approach for Management of Geometry Data", Internal Document, Boeing Computer Services Co., Seattle, WA, 1980.

[ECSY 79] Compendium of OSTA Space Data Usage Information. Book 3. Appendix A.5: Missions--Sensors--Data Products, Ecosystems International Inc., Gambrills, MD, under contract NAS5-25503 for Goddard Space Flight Center, November 1979.

[EDC 79] EROS Data Center (USGS), LANDSAT-D User CCT Tape Format, NASA/EDC Report FOR-LSD-001A, EROS Data Center, U.S. Geological Survey, Sioux Falls, SD, August, 1979.

[ERTH 76] Earth Observations Sensors, NASA Office of Applications (prepared by Systematics General Corporation, McLean, VA), 25 May 1976.

[FJMT 81] Fujimoto, Brad, User Requirements for NASA Data Base Management Systems. Part 1: Oceanographic Discipline, JPL Publication 81-50, Jet Propulsion Laboratory, Pasadena, CA, 15 June 1981.

[FONG 80] Fong, Elizabeth and Kimbleton, Stephen R., "Database Semantic Integrity for a Network Data Manager", Proceedings of the 1980 National Computer Conference, 19-22 May 1980, Anaheim, CA, AFIPS Press, Arlington, VA, pp. 261-268.

[GARY 80] Gary, J. Patrick, et al., NASA End-to-End Data System (NEEDS) Data Base Management System Functional Requirements, Information Extraction Division, Goddard Space Flight Center, Greenbelt, MD, March 1980.

[GDNU 79] Goodenough, D.G., et al., "Standard Format for the Transfer of Geocoded Information in Spatial Data Polygon Files", Canada Centre for Remote Sensing, Ottawa, Canada, July 1979.

[GOUG 81] Gough, T.L., et al., Data Base Management System Analysis and Performance Testing with Respect to NASA Requirements, GSFC Document No. BTS-FR-81-154, Contract No. NAS525561, Goddard Space Flight Center, Greenbelt, MD, to appear.

[GRNB 81] Greenberg, Ed and Hooke, Adrian, "Standard Message Formatting Protocol for Spaceflight Applications", Draft 3, Internal Document, Jet Propulsion Laboratory, Pasadena, CA, 6 July 1981.

[JHNS 80] Johnson, H.R., Comfort, D. L., and Shull, D. D., "An Engineering Data Management System for IPAD", Internal Document, Boeing Computer Services Co., Seattle, WA, 1980.

[JPLS 80] Jet Propulsion Laboratory, JPL Software Standard Practice: Implementation and Management of JPL Software, Draft, JPL Internal Document 500-152, Jet Propulsion Laboratory, Pasadena, CA, 25 June 1980.

[KUCH 81] Kuch, T. and Sakamoto, R., Survey of Federal, National, and International Standards Applicable to the NASA Applications Data Service, NASA Contractor Report 166675, The Mitre Corp., McLean, VA, March 1981.

[LHMN 80A] Lohman, Guy M. and Renfrow, J. Thomas, A Proposed Concept for a Crustal Dynamics Information Management Network, JPL Publication 79-111, Jet Propulsion Laboratory, Pasadena, CA, 15 January 1980.

[LHMN 80B] Lohman, Guy M. (ed.), Second DBMS Panel Workshop, Monterey, CA, 14-16 January 1980, JPL Publication No. 80-50, Jet Propulsion Laboratory, Pasadena, CA, 1 November 1980.

[LOTS 79A] Loats, Harry L. Jr., et al., Applications Data Service User Requirements Study, Final Report, Ecosystems International, Inc., Gambrills, MD, November 1979.

[LOTS 79B] Loats, Harry L. Jr., et al., Compendium of OSTA Space Data Usage Information, Volume 5: Directory of Sensor, Sensor Data Products and Characteristics, Ecosystems International, Inc., Gambrills, MD, under contract NAS5-25503 for Goddard Space Flight Center, November 1979.

[MRBL 77] Marble, Duane F. and Peuquet, Donna J., "Computer Software for Spatial Data Handling: Current Status and Future Development Needs", working paper, Geographic Information Systems Laboratory, State University of New York at Buffalo, Buffalo, NY, 1977.

[MRTN 76] Martin, James, Principles of Data-Base Management, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[NAGY 79] Nagy, George and Wagle, Sharad G., "Geographic Data Processing", Computing Surveys 11, 2 (June 1979), pp. 139-181.

[OAOC 79] OAO Corporation, User Requirements for NASA Climate Data Base Management System, prepared for Goddard Space Flight Center, Greenbelt, MD, under contract NAS5-23436 Mod. 27, OAO Corp., Beltsville, MD, 1 October 1979.

[ODBA 79] Office of the Data Base Administrator, Scientific and Technical, Spatial, and Bibliographic Data Bases of the U.S. Geological Survey, Geological Survey Circular 817, U.S.G.S., Reston, VA, October 1979.

[OFNS 79] Ofenstein, W. Thomas, et al., Requirements for an Applications Data Service, Revision 0, The Analytic Sciences Corp., Reading, MA, under contract NAS5-25739 for Goddard Space Flight Center, 19 December 1979.

[SBSO 80] Strategic Business Services, Impact of Optical Memories (Videodiscs) on the Computer and Image Processing Industries, Strategic Business Services, San Jose, CA, May 1980.

[SCNC 80] Science, "At Last There Is a Way to Take It with You", Science, Vol. 207, 15 February 1980.

[SDMN 79] Seidman, J. B. and Smith, A. Y., VICAR Image Processing System (Guide to System Use), JPL Publication 77-37, Revision 1, Jet Propulsion Laboratory, Pasadena, CA, 1 December 1979.

[STNB 81] Stonebraker, Michael, "Operating System Support for Database Management", CACM 24, 7 (July 1981), pp. 412-418.

[SUSY 80] Su, Stanley Y. W., et al., "Database Machines and Some Issues on DBMS Standards", Proc. of 1980

Natl. Comp. Conf. (Vol. 49), AFIPS Press, Arlington, VA, 1980, pp. 191-208.

[THMS 80] Thomas, Doug, Marshall Space Flight Center, AL, Personal Communication, August 1980.

[URNA 81A] Ureña, Jose L. (ed.), Data Base Management Systems Panel, Third Workshop Summary, JPL Publication 81-52, Jet Propulsion Laboratory, Pasadena, CA, 15 July 1981.

[URNA 81B] Ureña, Jose, Survey of Standards Applicable to Data Base Management Systems, JPL Publication 81-88, Jet Propulsion Laboratory, P adena, CA, 1 October 1981.

[WLTN 80] Walton, Barbara, Goddard Space Flight Center, Greenbelt, MD, Personal Communication, September 1980.

[ZLKW 78] Zelkowitz, M. V., "Perspectives of Software Engineering", ACM Computing Surveys 10, 2 (June 1978), pp. 197-216.

APPENDIX A

DETAILED FUNCTIONS REQUIRED

"Conceptual integrity is the most
important consideration in system design."
-- Brooks

This appendix describes in detail the functions that the ADAM must perform. Note that these functions are generic, not application-specific, in keeping with the multi-mission objective of the ADAM. It is intended that missions will use the high-level functions provided by the ADAM as powerful building blocks for constructing individual data management systems that are tailored to specific applications, much as the commands of FORTRAN are now used as the general-purpose raw materials of many systems.

The required functions are structured hierarchically in a top-down or successive refinement manner (see Figure A-1 at the end of this Appendix). This was done to ensure consistency and completeness in the specification of what functions are required. Each function is expressed as a predicate, i.e., a verb-object pair. For any given function in the "tree" of Figure A-1, the functions on the next level of detail answer the question: "What must be done to accomplish this function?" Each of the lower-level functions must be necessary to the completion of the higher-level function, and collectively they must be sufficient to accomplish it. Note, however, that it is <u>not</u> the intent of this "top-down functional analysis" (TDFA) to specify <u>how</u> the ADAM accomplishes each function. Except for the implementation and performance requirements of Sections 4 and 5, how a function is accomplished is a design decision to be made by the developer of ADAM.

The rest of this appendix is devoted to describing in some detail the required functions named in Figure A-1. Where appropriate, examples of their usage are also given. To get just an overview of the functions, the reader can skip lower level functions, which are numbered with more qualifiers. The numbers assigned to functions have <u>no</u> relationship to the sections of the main document.

The overall function of ADAM shall be to provide data from the data base to interactive users or users' application programs and back to the data base. Application programs are application-specific programs which require data from, and/or provide data to, the data base(s) belonging to ADAM. They usually are written in a high-level programming language such as FORTRAN, PL/I, or Pascal, but could be sequences of ADAM commands. Examples of application programs include programs to process raw data into engineering units or geophysical

parameters, image processing and enhancement routines, statistical packages, image co-registration algorithms, or simple user requests to display data (e.g., an image or a subset of observations) or information about the data base (e.g., a catalog) on a terminal or printer. Note that the term "data" here includes both observational data from the data base, as well as information about the data base such as catalogs describing what data is available and information that assists the use of the data base (see also Function 1.4).

1. STRUCTURE AND ESTABLISH DATA BASE

The ADAM must provide mechanisms for specifying storage structures into which both observational data and information about the data base may be loaded, and for initializing the data base by efficiently reading data in arbitrary formats into the data base en masse. In order to achieve data independence (Capability C1 of Section 2.2), separate languages shall be provided to specify the logical data base structure, the user's view (really the user's program's view) of that logical structure, and the physical layout of data on the storage devices. These languages are congruent with those specified by the widely accepted ANSI/SPARC Conceptual Framework for DBMS [URNA 81B]. For example, the logical structure (or "schema" in DBMS parlance) contains the user-oriented names of all data items in the data base, the user view (or "subschema") can specify a user-specific subset of these items and rename them or reorder them to fit his program, and the physical layout specifies the storage device and addresses for each item.

1.1 DEFINE, MODIFY, AND VERIFY GLOBAL LOGICAL STRUCTURE (SCHEMA)

The global logical structure, or schema, describes in outline form the characteristics of data items that are stored in the data base, independent of its physical layout. The function and contents of the schema are similar to those of the DECLARE statement of PL/I. It is "global" in the sense that it encompasses all files in the data base and specifies a standard format for all users of each data base, regardless of how each user might view the data.

The schema shall be specifiable by the Data Base Administrator (DBA) or by a knowledgeable, authorized user who generates a new file, using a language that has at least the functions of the 1978 CODASYL standard Data Definition Language (DDL) [URNA 81B]. Mechanisms for specifying, displaying, modifying, recompiling, and verifying the correctness of the schema shall be provided, either as part of the data dictionary (Function 1.4) or separately.

1.1.1 The schema definition language shall assign a logical name to individual data items or fields (including those calculated from other data items); files; and any combination of other aggregates of data items. These other aggregates include: sets (related records), multi-dimensional arrays (including relations and images represented

as large 2-dimensional arrays) possibly having millions of entries, and hierarchical structures in outline form (like those of PL/I).

1.1.2 The type and size of each data item or data aggregate shall be specified in the schema. Data types shall include integer (in binary form), floating point, (i.e., scientific notation, stored efficiently as signed mantissa and exponent), character (text), byte, and logical (true/false). Special internal representations shall be available to represent "not available" (or "not applicable") and "null", as distinguished from blanks and zeros.

Ideally, the user should be able to define new data types, optionally specifying explicit domains. For example, data type DAY-OF WEEK has the explicit domain (valid values) of MONDAY, TUESDAY, ..., SUNDAY. Also, the ADAM should be able to store values in an encoded form that may be more compact (e.g., MON, TUE, ..., SUN) and decode these back to more readable values for output.

Data items and aggregates may be of fixed or variable length. Fixed-length items may have a size declared for internal storage that differs from the size for display. Variable-length items must be stored such that only the amount of non-blank data to be stored is used, i.e., without padding. Repeating groups should be handled similarly, and permit an unknown number of instances.

1.1.3 The schema should have constructs which enable the DBA to specify integrity constraints upon each data item. For example, a data item called SENSOR.MODE might be limited to having only values which correspond to legitimate sensor modes, say the integers 1 through 100. The constraint types that are required are described in, and enforced by, Function 2.3.

1.1.4 The schema shall also permit the establishment of logical relationships between the component data items defined in Function 1.1.1, such as the ordering of records within a file or of fields within a record, designation of fields to be keyed upon; specification of set membership criteria and "owning" records, stored tables of relations, pointers to related data items, etc.

1.1.5 The schema should have constructs which enable the DBA to specify security constraints upon data base entities such as files, programs, and users. These constraints should include limitations upon and authorizations of the use of any ADAM function, at least at the file level and the data item level, as well as whether an authorized entity may grant that authorization to another entity. For example, ADAM should permit the DBA to authorize a principal investigator to perform most functions on a file that was generated by his sensor and also to grant to selected colleagues, authorization to add to and access but not to modify data already in that file. In other cases, the DBA might wish to restrict all alterations (input, load, and modify) of only one data item in a file -- perhaps a derived geophysical parameter -- to only the chairman of a change

control board, while allowing read-only access to that item and unrestricted access to the rest of the file.

1.2 DEFINE, MODIFY, AND VERIFY USER VIEW (SUBSCHEMA)

The user view, or "subschema" in DBMS parlance, describes in outline form the characteristics of only the data items and aggregates needed by the program(s) of a particular user or group of users. Mechanisms for specifying, displaying, modifying, recompiling, and verifying the correctness of subschemas relative to the schema shall be provided, either as part of the data dictionary (Function 1.4) or separately.

1.2.1 The subschema shall permit the user to select the subset of the data items from the schema that is needed, different names (e.g., shortening a schema's data item name to a 6-character name that is FORTRAN-compatible), as well as any other user-specific format information such as different data type (e.g., converting an integer type to a character type), size (e.g., truncating unneeded decimal digits), or relationships (e.g., a different sort order for records).

1.2.2 The subschema shall specify a mapping from the appropriate schema entry(s) to the corresponding subschema entry(s). The ADAM System shall perform this mapping whenever a using program specifies that subschema, either during compilation of the program or at run time [JHNS 80].

1.2.3 The ADAM shall also be capable of automatically mapping any subschema into the corresponding data definition commands of the using program's host language (e.g., FORTRAN, PL/I, etc.).

1.2.4 Changes to the schema or another subschema should not require modification of any subschema containing unchanged data items, nor the recompilation of programs unaffected by those changes.

1.3 DEFINE, MODIFY, AND VERIFY PHYSICAL DATA STRUCTURE

The physical structure of the data base refers to the actual location of various data items on secondary or tertiary storage devices such as tape, disk, or mass store. This structure, which is usually specified by the DBA, should be independent of the logical structures of the schema and subschemas, with ADAM responsible for mapping from the schema to the physical structure. This explicit mapping by ADAM is analogous to the implicit mapping from the list of variables in a READ statement in FORTRAN to the items of its corresponding FORMAT statement.

Mechanisms for specifying, displaying, modifying, recompiling, and verifying the correctness of the physical data structure shall be provided, either as part of the data dictionary or independently, and preferably separated into the following two functionally distinct and independent specifications:

1.3.1 The physical storage organization specifies physical storage characteristics which affect ADAM performance and resource utilization but which are not impacted by a change of storage media. These characteristics include, for example, any pointers used to relate records, blocks, or sets (Function 1.3.1.1); whether an item is actually stored in the data base or is a "virtual" data item that may be calculated from other items whenever it is needed (thereby saving storage) (Function 1.3.1.2); and any constraints on physical contiguity of certain data items such as the stored sort order of records or the need to have records in a set located together in an area (e.g., a block or page) (Function 1.3.1.3).

1.3.2 The physical device format and protocol specifies physical storage characteristics which are peculiar to a particular storage device. These characteristics include, for example: any device-specific or media-specific parameters or formats such as sectoring and block-size or track-size (Function 1.3.2.1); and addressing mechanisms such as indexes, hashing and collision-resolution algorithms, overflow handling, and standard access mechanisms (such as indexed sequential) provided by the operating system (Function 1.3.2.2).

1.4 DEFINE, MODIFY, AND VERIFY INFORMATON ABOUT DATA BASES

The data dictionary and catalogs are bases of information about the data base that facilitate the use or alteration of the data base. The data dictionary and catalogs are, therefore, themselves, data bases which should be structured and stored in machine-readable form by ADAM in order to enable a user or the DBA to answer questions such as: "What files contain data generated by a synthetic aperture radar (SAR) sensor?", "What files contain data pertaining to sea surface temperature?", "What user application programs access data item Y?", "What users are authorized to access file Z using subschema S?", etc.

Mechanisms shall be provided for specifying, displaying, modifying, and verifying the correctness of information contained in the data dictionary and catalogs, as well as their structure (schemas). The importance of this information dictates that the normal security and integrity constraints (Functions 1.1.3, 1.1.5) must be specifiable at the file, record, and data item levels.

1.4.1 The data dictionary shall be capable of specifying the characteristics of all entities that impact the data base, to include:

- Data items, records, arrays, and other aggregates of data contained in the data base.
- Schemas, subschemas, and physical data structures (both physical storage organization and physical device format/protocol) that structure the data base, and any integrity or security constraints on items contained in

these specificiations.

- Data sources, such as sensors, principal investigators, and documents

- Data products that are output from the data base, such as maps, new files, and reports, overviews, statistical summaries, catalogs, sample data, etc.

- Users of ADAM, including modules, programs, and entire systems which use ADAM, as well as the humans which initiate these routines

- Transactions which may impact the data base, such as particular events or types of events.

1.4.2 The data dictionary shall be capable of specifying any relationships which exist between the data base entities identified in Function 1.4.1, including (but not limited to):

- X accesses Y (e.g., module X accesses data item Y)

- X belongs to Y (e.g., schema X belongs to file Y)

- X uses Y (e.g., program X uses subschema Y)

- X generates Y (e.g., sensor X generates data item Y or program X generates product Y).

1.4.3 Data catalogs serve two major functions: (1) general top-level annotation of the data base, describing how the data in the data base was processed, how good it is, when and how it was collected, from whom it was obtained, etc., and (2) abstraction of the data itself, such as number of data points, ranges or other statistics that characterize the values, subsets of the data that are typical or of high interest, etc.

1.5 LOAD, UNLOAD AND APPEND NEW DATA

Once the data base structure has been correctly defined by Functions 1.1 through 1.4, users must be able to efficiently load, unload, and append data en masse into and out of the data base. This function is distinct from the "Input Data" function (Function 3.1) in that it is used to establish the contents of a new file of the data base or append additional blocks of data to that file, rather than selectively adding data to an established file. Furthermore, the load/unload/append function is intended for manipulating large masses of data, which may already conform to a schema (e.g., an image that was "destaged" or unloaded from secondary to tertiary storage), whereas the input function is intended for smaller quantities of data such as a few records.

The ADAM System must be able to accept and load data in widely varying formats. Therefore, as part of the data loading/unloading/appending process, ADAM should be able to reformat the data automatically to fit the schema and the physical data structure (Function 1.5.1), order data records to conform to that specified in the schema or physical data structure (Function 1.5.2), ensure the quality of the data by enforcing integrity constraints specified in the schema (Function 1.5.3), establish any relationships dictated by the schema or application programs performing the load (Function 1.5.4), and establish whatever control information or delimiters are dictated by ADAM or a file's physical device format/protocol (Function 1.5.5).

2. MAINTAIN AND PROTECT DATA

The ADAM System must be capable of maintaining and protecting the data and information in the data base. Many if not all of these functions should be invisible to most users, but are essential to helping the Data Base Administrator (DBA) to manage the data base effectively and efficiently. Besides protecting the data base from unauthorized or inadvertent destruction or alteration, ADAM should provide mechanisms for monitoring and controlling system performance, costs, and resource utilization.

2.1 PROTECT DATA (SECURITY)

The DBA shall be provided mechanisms to restrict and authorize selectively all access to the system, the data base, or to certain system functions. Security is necessary to assure privacy for some private data files whose owner does not yet wish to share his data, for example a principal investigator's preliminary results which have not yet been published or a file of observations which are undergoing "cleaning" and other processing. The DBA should be able to grant individuals or programs only those privileges which they need and no more, but he should also be able to delegate his authority selectively to responsible users.

2.1.1 Access to ADAM shall be restricted for security as well as accounting purposes, using a user-chosen password or similar mechanism.

2.1.2 Access to files for various purposes (e.g., load, input, modify, delete, or find and access data) shall be limitable depending upon the individual or program requesting such access. Such restrictions should be contained in and enforced by a secured portion of each subschema (Function 1.2).

2.1.3 Use of certain ADAM commands shall be limitable by the DBA or his designee. For example, many users should be able to be restricted to a "read only" set of functions, allowing them to use or copy data without altering it.

2.2 PERFORM BACKUP AND RECOVERY

The ADAM must facilitate the bulk copying of portions of the database from vulnerable but fast storage media such as main memory and disk to less vulnerable but slower storage media such as tape, and vice versa. Such backup and recovery procedures, including the restoration of changes made to the data base since the last backup was made, are necessary to prevent the catastrophic loss of data or prolonged system inoperability. In addition, logging earlier versions of the data base and those changes that are made to it enables the detection and correction of erroneous changes as well as unauthorized system operations.

2.2.1 For event-driven data bases, where updates are made "in place" rather than as complete revisions of large portions of the data base, the ADAM System shall be capable of supporting: (1) an audit trail of all system events or transactions; (2) a copy of the contents of a record before it is changed (a "before-image"); and/or (3) a copy of the contents of a record after it is changed (an "after-image"). Catalogs and files of altimetric observations undergoing "cleaning" are examples of these types of data bases that are updated record-by-record.

2.2.2 The ADAM System shall provide utilities to backup (or "dump") files selectively by copying them to another medium (usually tape) en masse, and to reverse that process in order to restore files which have been damaged. This shall be accomplished without affecting other, unrelated files.

2.2.3 Automatic recovery of the data base to its correct state prior to a failure shall be provided through the use of the restore function (Function 2.2.2) and one or more of the auditing functions (Function 2.2.1).

2.2.4 The ADAM System shall be capable of enforcing system-wide checkpoints (or "quiet periods") at which time the system status and memory contents may be preserved in case of system failure (e.g., power failure or operating system "hangup"). Conversely, ADAM shall be capable of automatic restart of interrupted processes from the last checkpoint, including "backing out" any changes to the data base since that checkpoint. These capabilities are needed in near-real-time applications or applications with a large interactive user population where system failure may otherwise result in unacceptable "down" time or the loss of interim results from long operations.

2.3 ENSURE DATA BASE QUALITY

The ADAM System must provide ways to ensure the correctness of values that individual data items take on, to ensure the consistency between values of various related data items, to control the timing of access to the same data by multiple users, and to enable users to define their own, more complicated quality control procedures that are

automatically invoked by the system. The advent of distributed data bases requires additional constructs to ensure the consistency and integrity of data at various nodes of a network [FONG 80].

2.3.1 Diverse constructs to check the values of data items are essential. These include:

- Ways to validate that data values are of the correct type and format, e.g., are not non-numeric characters to be inserted in a numeric variable or too long to fit within the space defined for that data item. It is highly desirable that the user be able to specify this format using a "template" or mask at the character or bit level, as with the PICTURE attribute of PL/I, as well as by specifying a field or variable "type" (Function 2.3.1.1);

- Ranges or enumeration of legitimate values, which may be specified in a designated table (Function 2.3.1.2);

- Limitations on what operations may be performed on a data item or file (e.g., forbidding any changes to a field containing spacecraft time) (Function 2.3.1.3);

- Constraints on transitions of a data item which relate its new to its old value (e.g., new value should not be different from old value by more than 10%) or its existence to its nonexistence (e.g., to differentiate between new observations and revisions of old observations) (Function 2.3.1.4);

- Ways to ensure that key fields that require unique values are in fact unique, including ways to resolve collisions (Function 2.3.1.5).

2.3.2 A data base is distinguished from a set of files by the relationships that link data items to each other, forming a whole that is greater than its parts. However, the strength of the data base concept lies in utilizing the inter-relationships to help locate related data and to cross-check related items against each other. Therefore, ADAM must be able to validate data items by comparing them to related items. The possible classes of comparison are:

(1) type or domain (e.g., units of measurement must be similar);
(2) an expression relating values of more than one data item, possibly in different records or even different record types (e.g., the distance between the locations of two observations must be less than the product of the platform's ground speed multiplied by the difference of the times of observation); and
(3) an expression relating a data item and a data aggregate

(e.g., an altimetric observation must not differ from the running average of such observations by more than 2.5%).

The last two relationships may be enforced by ADAM by providing automatic "hooks" or "triggers" to execute user-defined checking routines (see Function 2.3.4).

2.3.3 The DBA or authorized user must be able to control the timing of the enforcement of the integrity constraints described above, as well as sharing of data by multiple simultaneous users. These controls shall include the ability: to temporarily turn a constraint on or off (Function 2.3.3.1); to enforce immediate or deferred changes to a data base depending upon the application (Function 2.3.3.2); to specify whether access to a file shall be unrestricted (SHARED), a mixture of many read-only accessors and one updating user (PROTECTED), or limited to only one user (EXCLUSIVE) (Function 2.3.3.3); and whether a file is intended for reading only (RETRIEVAL) or may be updated (RETRIEVAL/UPDATE) (Function 2.3.3.4).

2.3.4 The DBA or authorized user should be able to define quality control routines which may be written in a programming language and which will automatically be invoked whenever an attempt is made to update the affected data item(s). This capability may be used in conjunction with any of the quality control functions described above.

2.3.5 Enforcement of the above logical integrity checks may affect the integrity of the physical access mechanisms (e.g., pointers, indexes) of the data base. Therefore ADAM should automatically check and update these mechanisms if necessary.

2.4 ALLOCATE/DEALLOCATE LOGICAL STORAGE SPACE

The ADAM System should monitor and control the overall use of all storage over which it has control including virtual storage assigned to it by the operating system, in units meaningful to its internal structure. These units may be different from the units which are meaningful to the storage devices themselves, the control of which constitutes a separate function (Function 4.2.3). For example, the ADAM may be required to store all data in standardized logical formats [GRNB 81] that may map into physical pages of memory in complex ways. Additionally, the ADAM shall be capable of controlling the migration of data sets among various storage devices for performance or other reasons. Examples of this migration include the certification of a data set as suitable for exchange and the control of changes to certified data sets (Function 2.4.1), automatically staging frequently used data from tape or mass store devices to disk (Function 2.4.2), and retiring old or seldom-used data to archives or even long-term storage (Function 2.4.3).

2.5 MONITOR AND TUNE SYSTEM PERFORMANCE

The ADAM shall be capable of monitoring in real time its own performance and reporting this to the necessary accounting functions (Function 2.6) and periodically to the DBA. It shall also permit the DBA to adjust major system parameters easily so that system-wide trade-offs can be assessed and performance can be optimized. Factors to be measured and controlled by this function include:

- Response time to each user query or command, to operating system commands, to telecommunications commands, and to commands to the input/output (I/O) devices
- Utilization of I/O channels and devices
- Utilization of main memory, secondary storage, and tertiary storage (if any), including percent remaining and percent of storage used for ADAM overhead (e.g., pointers, delimiters, etc.)
- Utilization of the central processing unit(s) (CPU), both in raw seconds and also relative to I/O utilization
- Percent of data physically retrieved that meets the logical retrieval criteria, i.e., the proportion of all data records retrieved that are "hits". This factor is important to adjusting the granularity of physical storage blocks/pages (see Function 4.2.3)

2.6 SYSTEM COST ESTIMATES AND ACCOUNTING

2.6.1 The ADAM System should be able upon request to estimate for a user the approximate cost of performing a function before proceeding to do it. This estimate shall include approximate elapsed time to retrieve (order of magnitude, e.g., seconds, minutes, hours) and volume of output (e.g., number of data points, tapes, pages of printed output). These and the dollar cost can be estimated based upon historical data on similar operations, the parameters specified by the user, and information about the data base that is available from the data dictionary and catalogs (Function 1.4).

2.6.2 Utilizing the information collected by Function 2.5, ADAM shall determine a cost that may be charged the user and shall deduct that amount from the user's account in a file of accounts maintained by ADAM. The user should be able to specify a maximum cost for any requested operation (particularly a large data retrieval request) and to have the cost to date reported to him. Periodic reports of the status of all or selected accounts shall be made by ADAM at the request of the DBA.

3. ACCESS AND MANIPULATE SELECTED DATA

The ADAM System must provide mechanisms that permit users to access and manipulate only the data in the data base that satisfies their immediate needs. All other major functions may be viewed as supporting the goal of this function, which is to get selected data to and from the user, from and to the data base. Selectivity is stressed because the user gains nothing in performance by centralizing all data under the control of the DBMS unless the DBMS concommitantly improves the efficiency of the access mechanisms to that bigger mass of data. Users must be able to access and manipulate multiple data bases, plus other files not created by ADAM, simultaneously.

3.1 INPUT DATA

The ADAM System shall facilitate the addition of new data to the data base in accordance with the format, integrity, and security constraints previously established. This data may come from input files in a variety of formats, from user application programs that have generated new data or processed data from the data base into a new form, through the telecommunications interface, or from users keying in data at their terminal.

Though this function is distinguished from the load and append operation (Function 1.5) in that it is intended for smaller amounts of data that are added to an established data base, the two functions nonetheless share two common supporting functions: ensuring data quality (Functions 3.1.3, 1.5.3) and establishing the necessary relationships between the new data and other data items (Functions 3.1.6, 1.5.4). In addition, the user must be allowed to find the appropriate location to insert the new data (Function 3.1.1, cf. Functions 3.2.2, 1.5.2), display the data before entering it into the data base, and alter the values of the input data items (Function 3.1.2) if need be before inserting the record into the data base (Function 3.1.4). This is accomplished by finding the appropriate information about that item (see Function 3.2.1) and formatting the data suitably (cf. Function 1.5.1). Finally, any access mechanisms such as indexes must be updated to reflect the changes caused by the insertion of the new data into the data base (Function 3.1.5).

3.2 FIND AND ACCESS DATA

A key user requirement is the need to find the desired data in the data base, and retrieve it for further processing [FJMT 81]. The ADAM must therefore permit the user to specify what data is desired, help him or her to determine what data is available and possibly modify the request, find the actual location(s) of relevant portions of the data, retrieve the necessary portions for transfer to the user's work area, and eliminate irrelevant data while reformatting the data into the user's view (subschema).

3.2.1 Mechanisms for finding and accessing information about the data base that is contained in the data dictionary and catalogs are required to help the user determine what data is available, what its format is, who is authorized to use it, and other characteristics of interest to him. This function may be accomplished by (recursively) using Function 3.2 applied to the data dictionary and catalog data bases with at least the functions comprising Functions 3.2.2 and 3.2.3.

3.2.2 The ADAM System shall find the location of the data requested by a user without retrieving the data itself, so that the user can assess the quantity of data satisfying a query and possibly narrow his query. The query may be specified as any combination of relations on either key or non-key data item values (e.g., sensor = SAR, or sea surface height less than 4.5 meters) ranges of values for an item (e.g., latitude between 30N and 40N), a list of values for an item (e.g., sensor mode = 1,2,4,7,15), or any combination of the above (Function 3.2.2.1). These relations may be joined by conjunction (AND) and/or disjunction (OR) (Function 3.2.2.2). The system shall perform the search based upon the appropriate access mechanisms (if any) such as indexes or hash functions that have been established for the keys used and for the values specified (Function 3.2.2.3). Finally, the data base itself shall be searched, if necessary, to find the requested data (Function 3.2.2.4). Search paths will be optimized by ADAM to locate the desired records most efficiently, e.g., key items should be searched upon before non-key items.

3.2.3 Once the data requested has been found, the ADAM shall retrieve the portion needed by the user. This includes possibly staging the data to on-line storage and transferring the selected data to an ADAM-administered buffer (Function 3.2.3.1), selecting the desired subset of records (Function 3.2.3.2) and attributes (fields) within each record (Function 3.2.3.3), and then reformatting the remaining data to fit the user's view (Function 3.2.3.4).

3.3 REARRANGE/AGGREGATE DATA

The ADAM System shall support the rearrangement and the aggregation of selected data records. This function is distinguished from the modification function (Function 3.4) in that it does not alter the basic content of the data base, but merely re-orders, merges, or summarizes it to enable the user to access and/or understand the data more readily.

3.3.1 Some users may wish to re-store data in an order different from the order in which the data are stored according to the schema. Therefore ADAM must be able to sort data that is selected by the user on either key fields (Function 3.3.1.1) or other, non-key fields (Function 3.3.1.2). For example, a file of several images might have different spectral bands interleaved on a row-by-row basis, whereas the user wishes to store or process complete images (all rows) from

each spectral band separately. Or, a file of altimetric observations might be stored in chronological order but the user wishes to interpolate their values in latitude/longitude order. Once this reordering has been accomplished, users may wish to merge two files based upon similar or "near" values of a key or other field which both files have in common (Function 3.3.1.3). For example, users often want to compare synoptic observations of the same geophysical phenomenon by two different sensors for the purpose of validating remote sensing techniques [FJMT 81]. This would involve merging the files or observations from the two sensors whenever the latitude, longitude, and time were all "near" to each other within some user-specified tolerance. As a subset, this merge function shall include the "JOIN" function required for a relational-model data base.

3.3.2 Some users require only an overview of the data, at least initially, i.e., a view of the data at lower resolution than that at which it was collected and/or stored [FJMT 81]. For example, users often wish to display an entire 1024 x 1024 pixel image on a 512 x 512 pixel screen, requiring a 2 x 2 averaging of pixels. Some applications require one observation per 1 degree square for data collected from a variety of sensors, each having a different resolution. Hence ADAM must be able to aggregate records to the appropriate resolution for the application. This function may include user-specified routines for averaging, interpolating, and/or resampling.

3.3.3 In support of the previous function as well as many other functions, the ADAM should permit the user to substitute a mathematical expression involving one or more data items wherever a data item may be specified. In the previous function, for example, the user should be able to average the square of an observation as easily as he can average the observation itself. Therefore, the ADAM must support the ability to recognize, parse, and perform mathematical expressions that use the basic operations of multiplication, division, addition, subtraction, and exponentiation upon data items from the data base.

3.3.4 In addition to the basic mathematical operators, mathematical expressions shall also permit the use of certain built-in functions similar to those of standard programming languages. Availability of these functions in ADAM will allow the user to specify a richer variety of expressions and will save the user calling a user-written routine to calculate the value of these more complicated expressions. Examples of these built-in functions include:

- Minimum (MIN) of an array (any dimension) of values or all selected instances of a data item or expression.
- Maximum (MAX) of an array of values or all selected instances of a data item or expression.

- o COUNT of all selected instances of a data item or expression.
- o TOTAL of an array of values or all selected instances of a data item or expression.
- o Average (AVG) of an array of values or all selected instances of a data item or expression.
- o Standard deviation (STD DEV) of the above average.
- o Logarithms (LOG) base "e" and base 10 of an expression.
- o Absolute value (ABS) of an expression.
- o Exponentiation (EXP) base "e" of an expression.
- o Trigonometric functions (SIN, COS, etc.) of an expression.
- o A substring (SUBSTR) function, similar to that of PL/I, for facilitating character string searching and manipulation.

3.3.5 Users should be able to define functions that can be invoked at least at the program interface level, and preferably at the command language level too.

3.4 MODIFY DATA CONTENT

This function is intended to permit the alteration of the contents of selected data items, including bulk updates of many records using a single command, as well as update-in-place of single records that are stored on storage media which permit it (e.g., disk). Examples of this latter function include the correction of an observation based upon interpolation of surrounding values, and the updating of catalog entries.

3.4.1 The data item(s) to be changed must first be located and accessed. See Function 3.2.

3.4.2 New values may be calculated, possibly using the mathematical operators (Function 3.3.3), the built-in functions (Function 3.3.4), the old value, and/or user-defined routines (Function 3.3.5).

3.4.3 Before the new value may become part of the data base, it must be checked automatically for quality based upon its entry(s) in the schema or subschemas. See Function 2.3.

3.4.4 The new value(s) may now be stored, either by inserting the modified record into a new location found for it, by overwriting the old value(s), or by writing the item to an output file that constitutes the updated version of the file.

3.4.5 If old values are not overwritten, they may need to be deleted.

3.4.6 The modification of key fields may affect the access mechanism(s) used to find that data item (e.g., indexes, pointers, or hashing functions). Therefore, these access mechanisms should automatically be checked and updated by ADAM if necessary.

3.4.7 The modification of any data item may affect the relationship(s) it shares with other data items, as specified in the schema, subschema(s), and/or pointer fields. Therefore, ADAM should automatically check and update appropriately these relationships in order to ensure the integrity of all relationships.

3.5 DELETE DATA

The user should be able to delete, or mark for later deletion, any instance of a data item, record, file, etc., subject to the constraints imposed by the schema or subschema upon the data and/or the user (See Function 2.1.2).

3.6 PREPARE DATA FOR OUTPUT

The ADAM shall have mechanisms for specifying the logical structure of various types of outputs, independent of the physical format details for a particular device. For example, a tabular arrangement of data prepared by this function should be unaffected by the need to display it on a Hewlett-Packard terminal instead of a Tektronix terminal. Adaptations of that sort are the concern of the interface to the operating system (see Function 4.2.2), which ultimately commands the devices in machine language. This function shall include both prior definition and naming of a data product format in a schema and/or subschema (see Functions 1.1 and 1.2) and *ad hoc* specification by user commands (e.g., "DISPLAY...").

3.6.1 As part of the specification of an output product, the user may wish to rearrange or aggregate the data temporarily according to his own view of the data (his subschema) or in an order peculiar to that particular output product. This function is different from Function 3.3 in that it results in no change to the data base. The ADAM shall permit the user to specify the appropriate data item(s) or expression(s) on which to order or aggregate the output, including nested orderings (e.g., "... ORDERED BY SC.TIME BY LONGITUDE" or "AVERAGED BY SC.TIME, BY LOCATION"). See also Function 3.3.

3.6.2 The usual basic report-writing constructs shall be provided, including the ability to specify titles for page headings, footings, rows, columns, subtotals, and totals (Function 3.6.2.1); the ability to space columns (Function 3.6.2.2) and rows (Function 3.6.2.3) automatically on the page in a way that enhances readability; and the actual formatting of each page, including the determination of how

much to fit in a single page, overflow to subsequent pages, and page numbering (Function 3.6.2.4).

3.6.3 As an optional enhancement to the basic system, the ADAM shall permit the user to format graphs formed from the data selected from the data base or provided by the user. This graphing shall include the specification of titles for the graph and labels for the axes, points, keys, etc. (Function 3.6.3.1); the determination of the scale, endpoints, and any breaks in the axes of the graph, either automatically from the data to be graphed (default) or manually specified by the user (Function 3.6.3.2); and the actual plotting of points, vectors, curves, characters, etc., derived from the data provided (Function 3.6.3.3). This capability shall include the plotting of data versus location on various maps, either as symbols, numbers, or contours.

3.6.4 As another optional enhancement to the basic system, ADAM shall enable the user to prepare for display raster images using data extracted from the data base, provided by the user, or generated by another ADAM function (especially graphics outputs, Function 3.6.3).

3.6.5 In order to transfer a block of data to another program within the system, particularly a user's application program, or to another system, the ADAM must be able to specify formats for telecommunications packets (Function 3.6.5.1) or physical storage structures (Function 3.6.5.2) that are compatible with the format of the destination.

3.6.6 For applications where users interact directly with ADAM, the specification and preparation of screen displays for cathode ray tube (CRT) devices is required as an optional enhancement to the basic system. This function shall adapt the outputs of Functions 3.6.2 through 3.6.4 for screen display and response by users, including: any special screen titles (Function 3.6.6.1); reformatting of data or output products for screen display (Function 3.6.6.2); adding any borders or templates needed for forms or other display enhancements that aid user comprehension (Function 3.6.6.3); permitting the user to scroll through to any page of an output product (Function 3.6.6.4); formatting a "menu" of potential user responses to the display (Function 3.6.6.5; cf. Function 4.1.1.1); and accepting user responses (Function 3.6.6.6).

3.6.7 Users must be able to name and store output products for subsequent retrieval and/or transfer to other devices.

4. INTERFACE WITH OTHER SYSTEM PROGRAMS

The ADAM shall provide interfaces to all other system programs with which it must deal, in a centralized and modularized fashion. These programs shall include, but not be limited to, user application programs or query programs, the operating system or executive program, and data communications control programs, if any. For more

details on the sequence of how these programs interact, see [CDSY 71, CRDN 79]. It is assumed in the discussion which follows that the reader is familiar with language compilation/interpretation and with how system programs interact with each other, I/O channels, and the hardware.

4.1 INTERFACE WITH USER APPLICATION/QUERY PROGRAMS

The ADAM shall be able to interface to user's application or query programs, either by being invoked by the user program and/or by invoking the user program from ADAM environment. Application programs are differentiated from query programs (or simply "queries") here by two major factors:

- Queries are generally *ad hoc* in nature, to be used only once, whereas application programs (once debugged) may become operational routines.
- Queries are usually generated interactively with each command translated and verified as it is entered (i.e., interpreted), whereas application programs are frequently submitted as programs which are translated *en masse* before execution (i.e., compiled).

Although this distinction between queries and application programs dictates that they be separated functionally, many of their supporting functions are similar and it is highly desirable that they share a common command syntax so that users need learn only one data manipulation language [JHNS 80].

4.1.1 Due to the interactive nature of queries, the query processor must perform several functions which help the interactive user. These include generating user prompts, preferably including a menu of actions from which the user can choose (Function 4.1.1.1, see also Function 3.6.6); storing and displaying upon request user HELP routines which explain in more detail the use and syntax of a particular ADAM command (Function 4.1.1.5); and maintaining the continuity of the dialogue with the user by optionally providing status information every 5 to 10 seconds during long operations (Function 4.1.1.4). The latter function helps to prevent the user from becoming impatient with the system or from suspecting that the system has forgotten about him. The user must be able to cancel a request at any point during a long operation without loss of data base integrity or loss of data retrieved up to the cancellation. It should also be noted that ADAM should provide only the constructs for building user menus, not the menus themselves: that is an application-specific output product that should be tailored to the user by each organization utilizing the ADAM. However, certain application-independent menus could also be built into ADAM using this same facility.

Functions which the query processor shares with the application program interface include: translating user commands in an ADAM language into a function to be invoked and the parameters needed by that function (Functions 4.1.1.2, 4.1.2.1); transferring ADAM control to the appropriate function(s) (Functions 4.1.1.3, 4.1.2.2); and generating error messages, diagnostics, or return codes to the user to inform him of the status of his query (Functions 4.1.1.6, 4.1.2.5). Several levels of error severity should be supported, including advisories, warnings, recoverable errors, and fatal errors. Error messages should return both an error number and text stating the location of the error (routine name, line and column number, record type and key), its severity level, and the type of error (syntax, invalid reference, integrity or security violation, I/O error, user abort, etc.).

4.1.2 If ADAM interface to application programs is via "CALL" commands within the application program, then the translation function for this interface (Function 4.1.2.1) is simplified: the program's host language will pass to ADAM the function and all qualifying parameters (e.g., data item names or expressions, options, etc.) as parameters of its CALL command. Otherwise, ADAM must parse the "stand-alone" ADAM-language commands just like any other language translator (compiler or interpreter). Since it is desirable that the application program interface use the same high-level language as the query processor, DBMS commands will be interspersed among host language commands. This implies the need for either: (1) a superset of standard programming languages that contain ADAM commands, or (2) a pre-compiler to convert ADAM commands to CALLs in the host language.

Because application programs will be potentially passing large amounts of data between it and ADAM, ADAM must include the functions of accepting data from application programs (Function 4.1.2.3) and returning data to application programs (Function 4.1.2.4), through user work areas and ADAM buffers.

4.1.3 The ADAM System shall enable the user to build, store, and execute "macros" of ADAM commands. Macros are a sequence of ADAM commands that are stored as text files under a referenceable name (Function 4.1.3.1) that permits text substitution (Function 4.1.3.2) and execution of the commands (Function 4.1.3.3) by using the macro name like a single ADAM command. It is desirable that the macro facility include a "call-by-name" substitution of run-time values for macro parameters when entering and exiting the macro, as well as the ability to loop and branch.

4.2 INTERFACE WITH OPERATING SYSTEM

The ADAM must interface with and be under the control of the resident operating system or executive program at each installation upon which it is implemented. The operating system controls the interaction of programs (including ADAM and its application programs), resource

utilization, and all dealings with the hardware (e.g., interrupts, I/O priorities, etc.). Because the operating systems at various installations will necessarily differ, it is imperative that ADAM centralize its interface with the operating system into a minimal set of modules (see also the requirements of Sections 4.3, 4.4), and that it perform as many functions independent of operating-system-specific parameters as possible.

4.2.1 The translation of ADAM commands into invokable ADAM functions is performed by the user interface (Function 4.1). These functions, in turn, must at some point execute some machine language commands which, for performance purposes, must be tailored to the operating system and hardware which execute them. The operating system services which ADAM will use include: single-key access methods that may be provided as part of the operating system (Function 4.2.1.1); READ and WRITE commands provided by the operating system that save having to write channel command sequences (Function 4.2.1.2); and virtual memory versions of operating systems that treat secondary storage as an extension of main memory and automatically move "pages" of data between the two (Function 4.2.1.3). Alternatively, where these operating system services are absent or inefficient for DBMS applications, ADAM may have to provide some or all of these services [STNB 81]. As operating systems become more powerful and sophisticated, interfaces to other such services may need to be added here.

4.2.2 The interface to the operating system should be responsible for mapping the physical storage structures of the data base that were defined in Function 1.3.1 into the device-dependent formats and protocols that were defined in Function 1.3.2. These formats and protocols will therefore be used in generating the specific machine language required for any given transfer to another program in main memory (Function 4.2.2.5), or for any given access to the data base storage devices (Function 4.2.2.4) or any other I/O device. The latter includes device-dependent parameters and formats for data products, such as reports (Function 4.2.2.1), graphics (Function 4.2.2.2), and images (Function 4.2.2.3).

4.2.3 The ADAM interface to the operating system shall be responsible for the monitoring, allocation, and de-allocation of the physical storage space that has been allocated to it by the operating system, as well as the mapping of logical storage structures if any to these physical blocks of data which the operating system manipulates efficiently (e.g., pages or disk tracks). Control of these physical storage spaces include allocation and de-allocation of buffers for I/O devices and user programs ("work areas") (Function 4.2.3.1); allocation and de-allocation of blocks, pages, areas, tracks, or whatever constructs are used to divide up secondary storage (Function 4.2.3.2); and periodic collection of free space vacated by deleted data items, often called "garbage collection" (Function 4.2.3.3).

4.3 INTERFACE WITH DATA COMMUNICATIONS PROGRAM(S)

As an optional enhancement to the basic ADAM system for installations having significant telecommunications traffic and for distributed data base applications, ADAM should be capable of interfacing with specialized programs that optimize data communications resource utilization. This interface will need to include the ability to: send and receive commands, with the proper synchronization, to and from the telecommunications program via the operating system (Function 4.3.1); translate commands to and from either a standardized network language or the language of the destination node (Function 4.3.2); send and receive data to/from the network or a program at one of its nodes (Function 4.3.3); perform necessary protocol and synchronization functions (polling, addressing, scheduling, queueing, prioritization, authorization of resource utilization, security of terminals, and overall control and efficiency of telecommunications) that are either required by a telecommunications program or are not otherwise supplied by it (Function 4.3.4); and reformat or packetize commands and data to conform to the standards of the network and to individual terminals (Function 4.3.5).

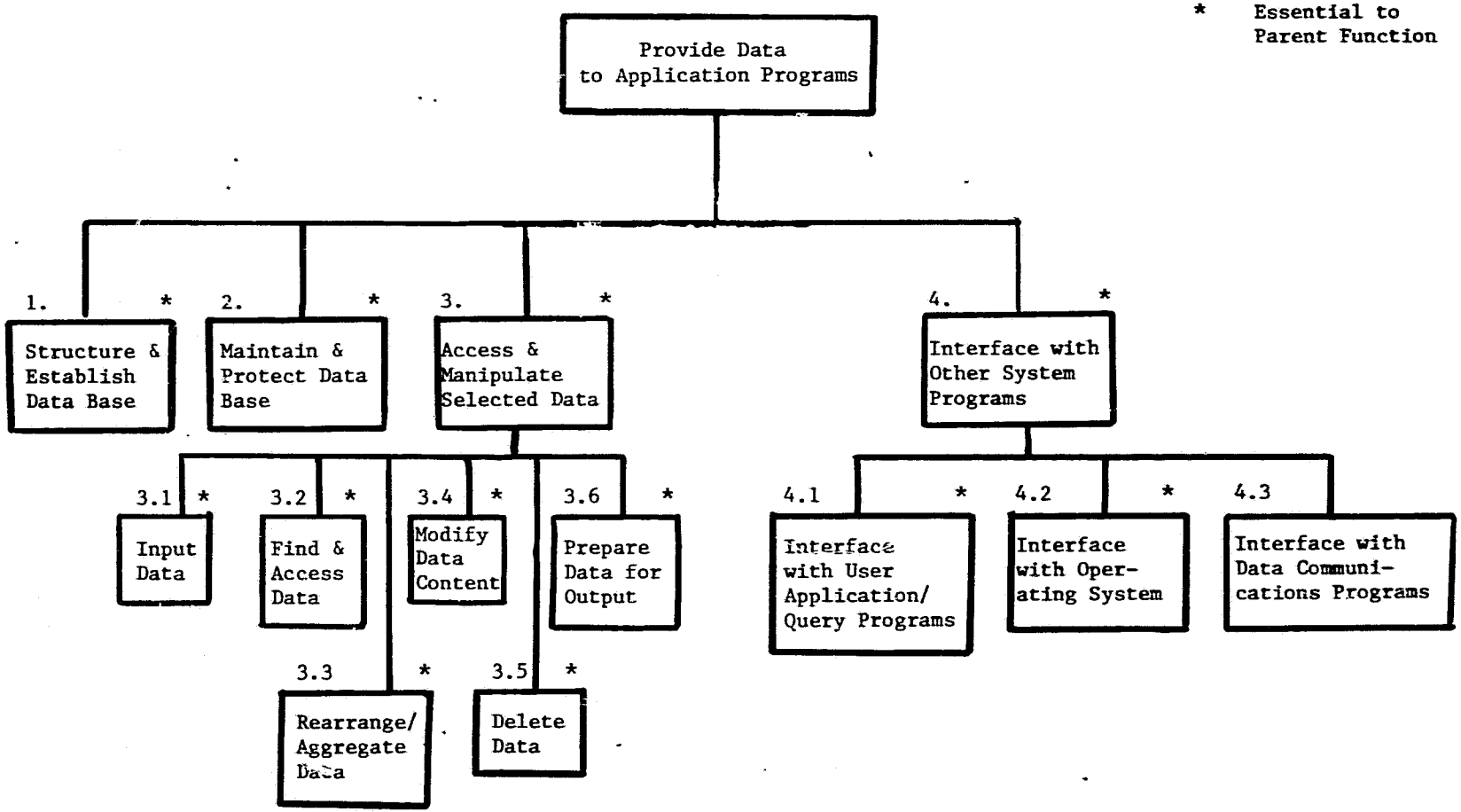


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM

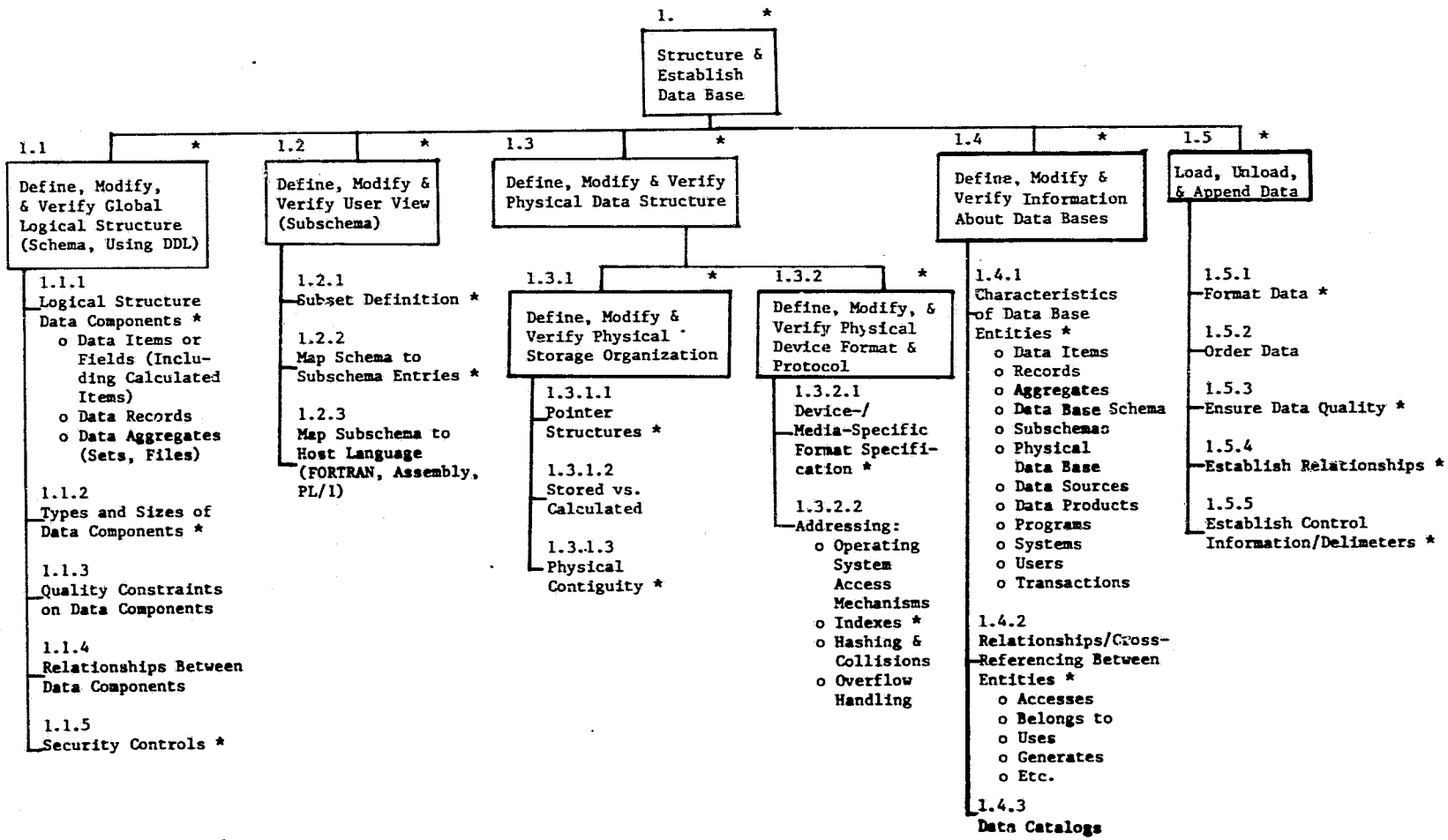


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)

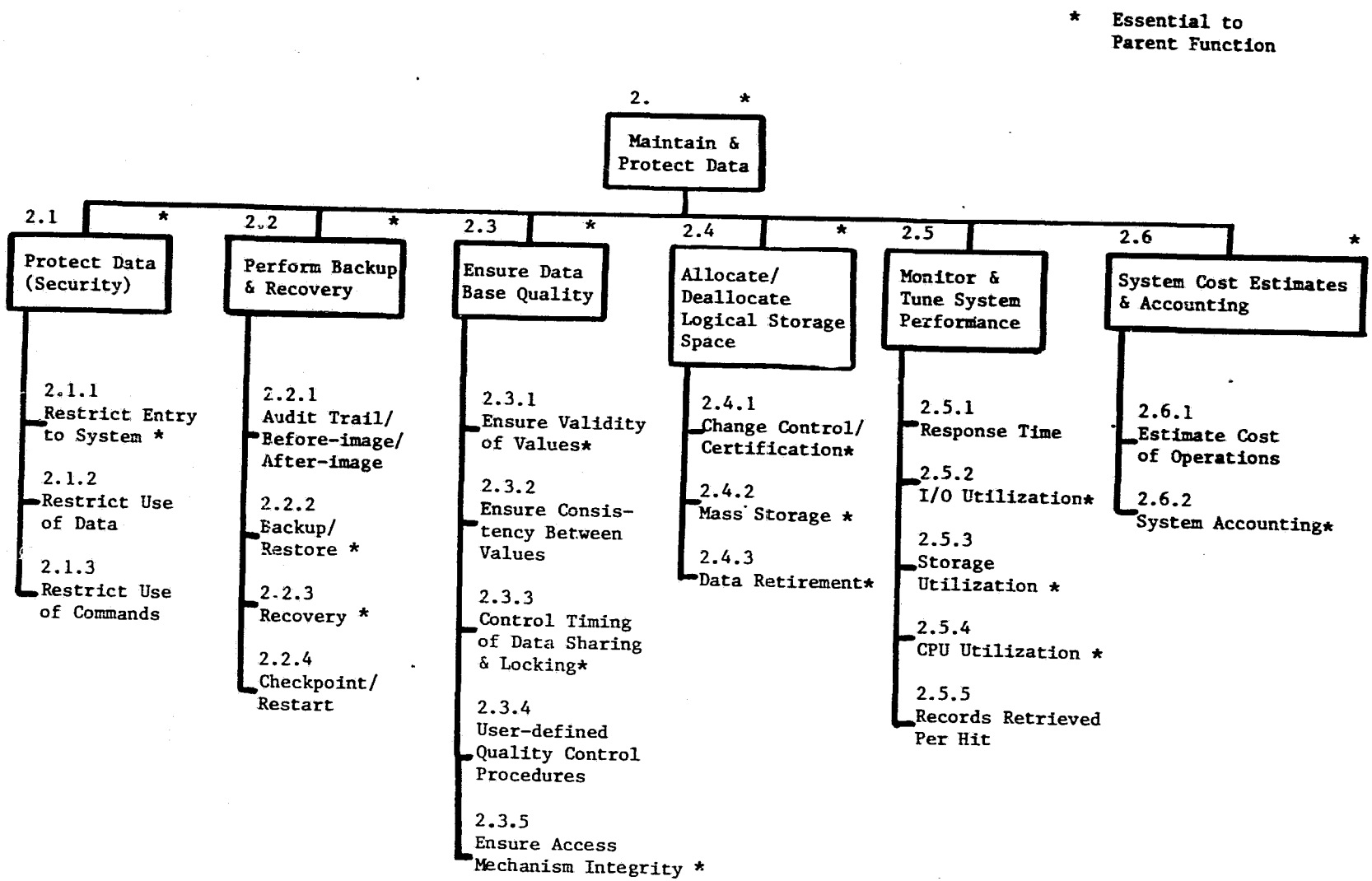


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)

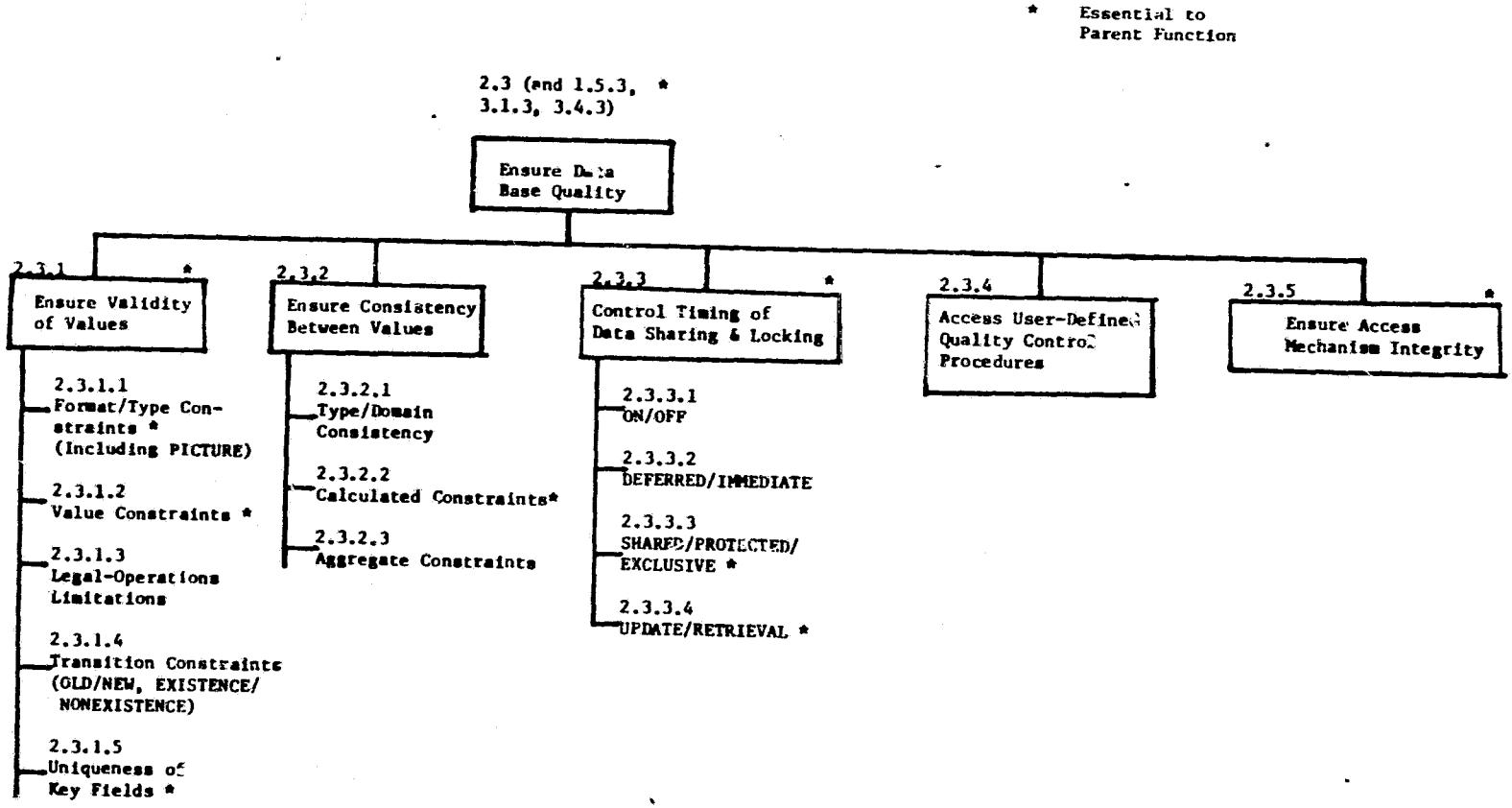
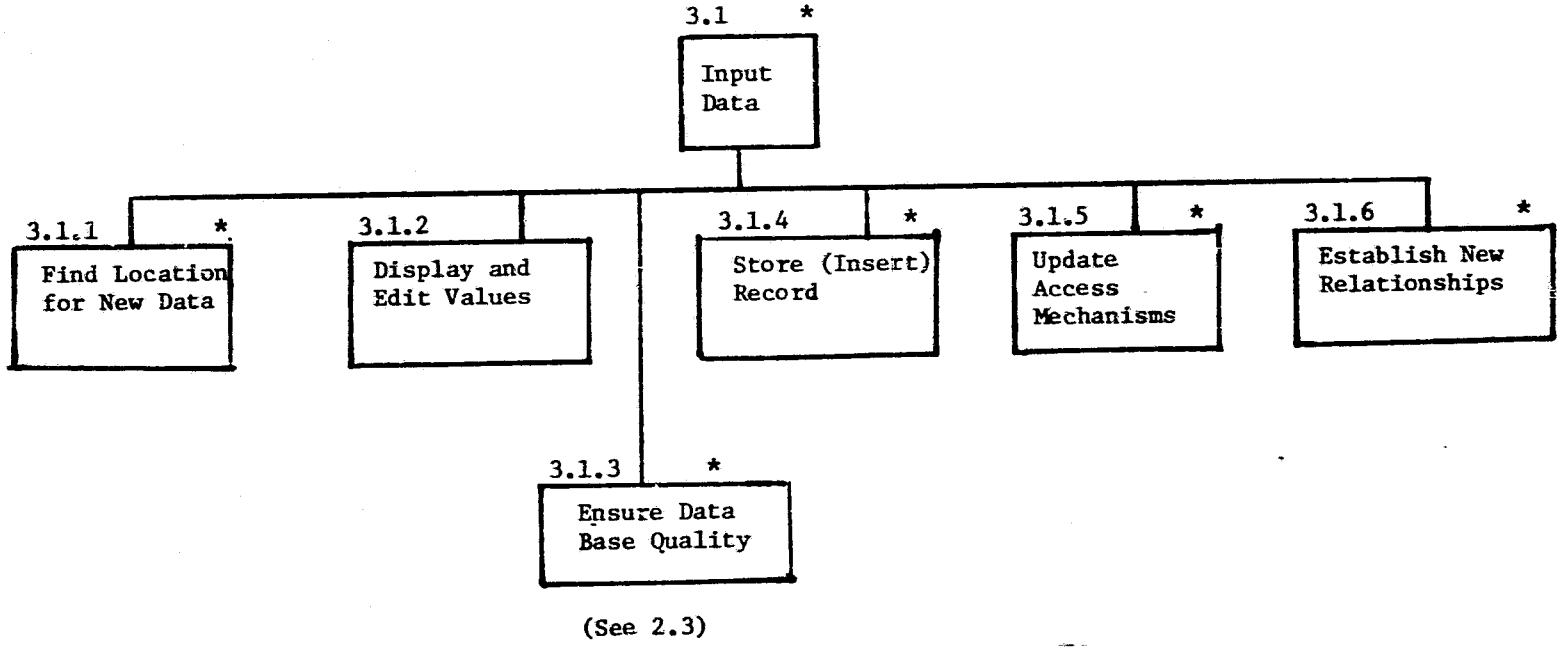


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)

* Essential to Parent Function



(See 2.3)

Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)

* Essential to Parent Function

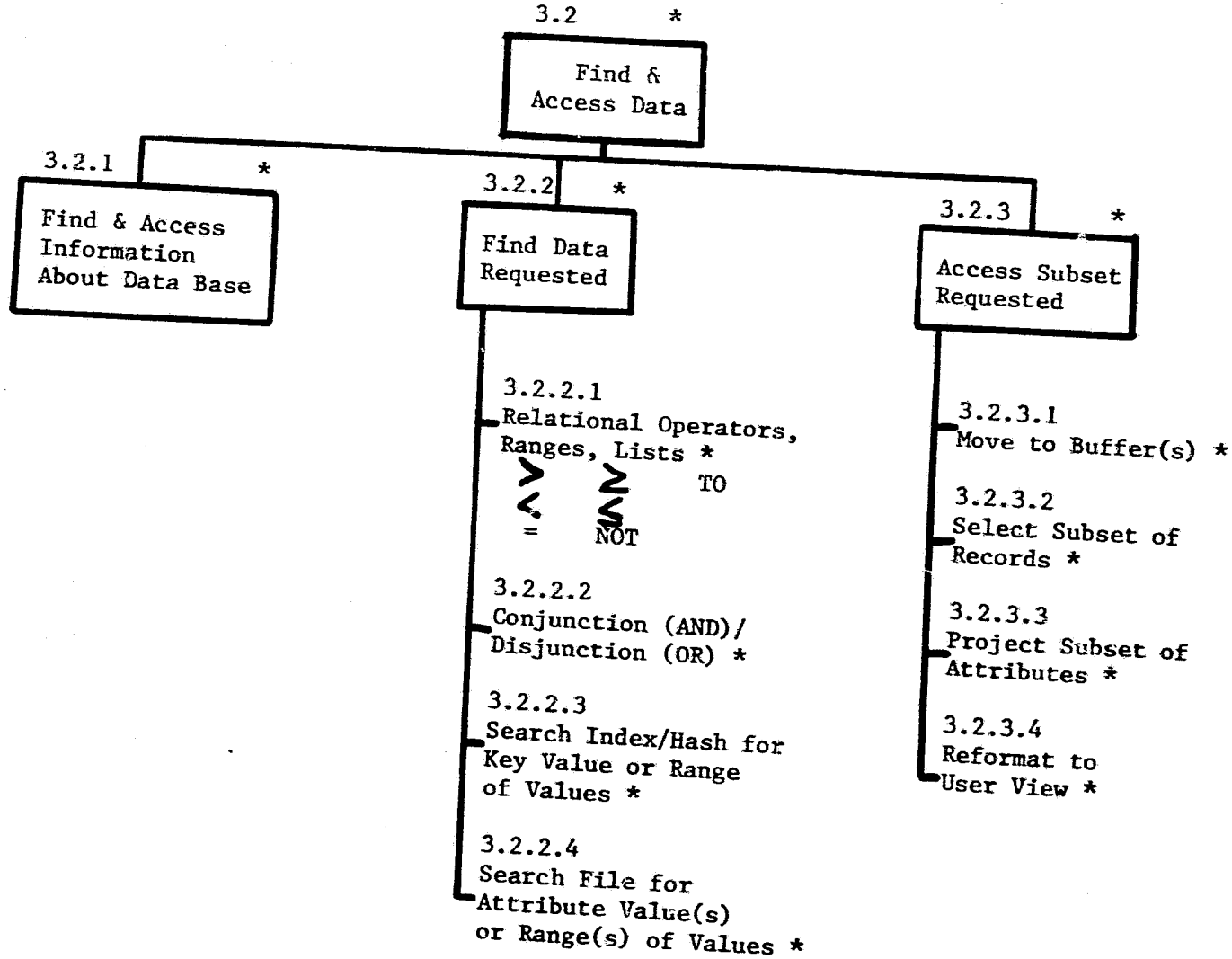


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)

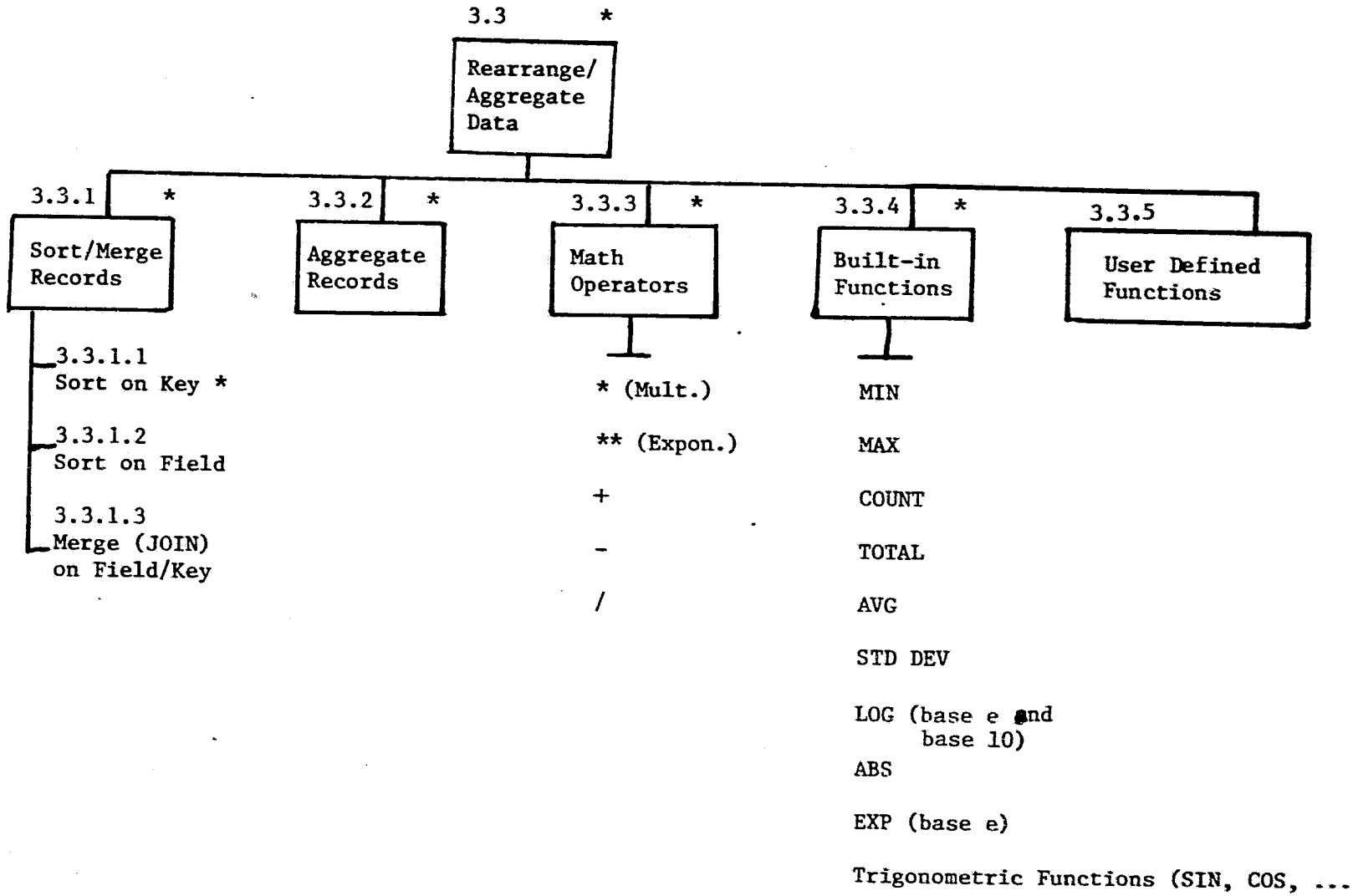


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)

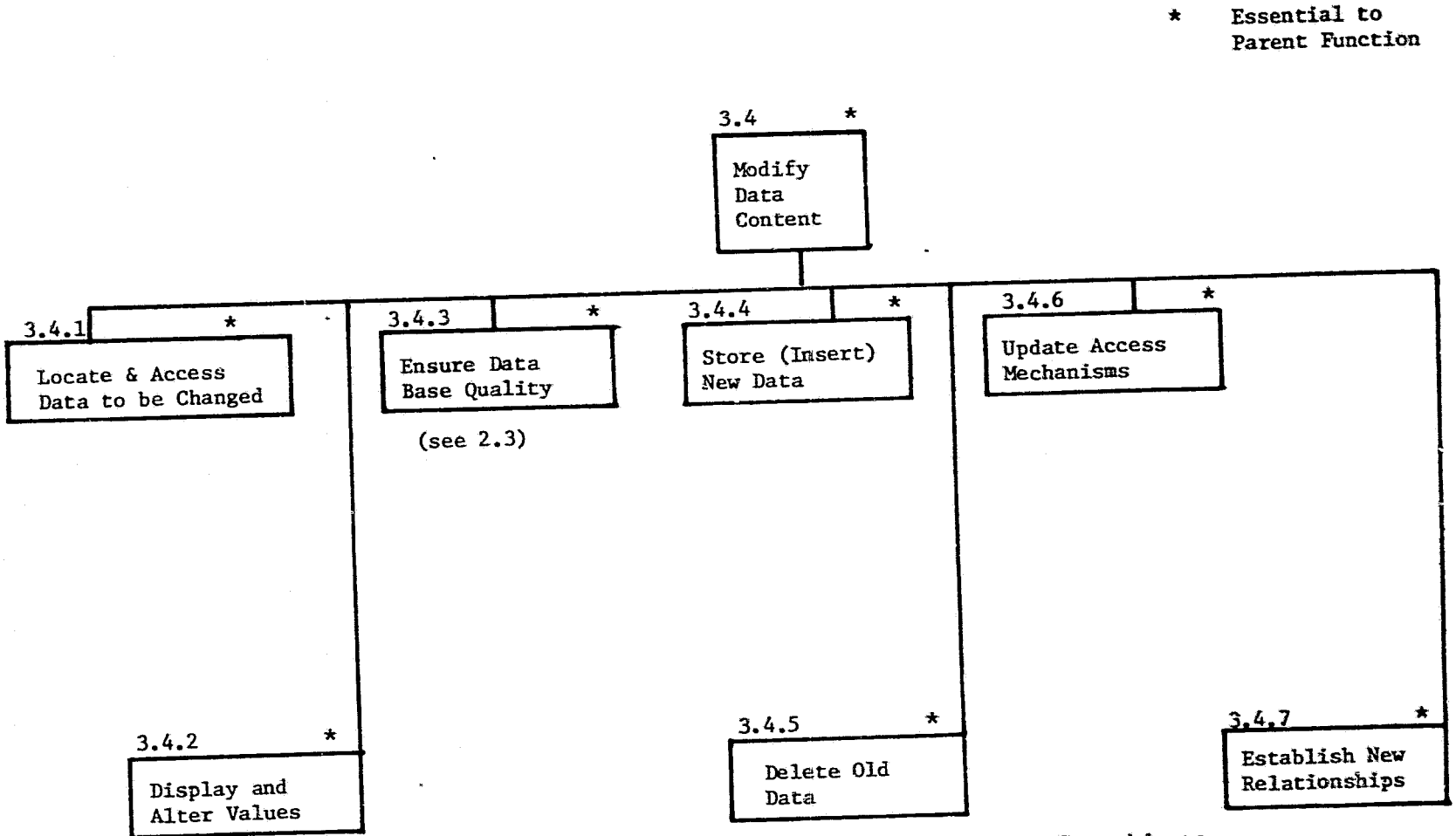


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)

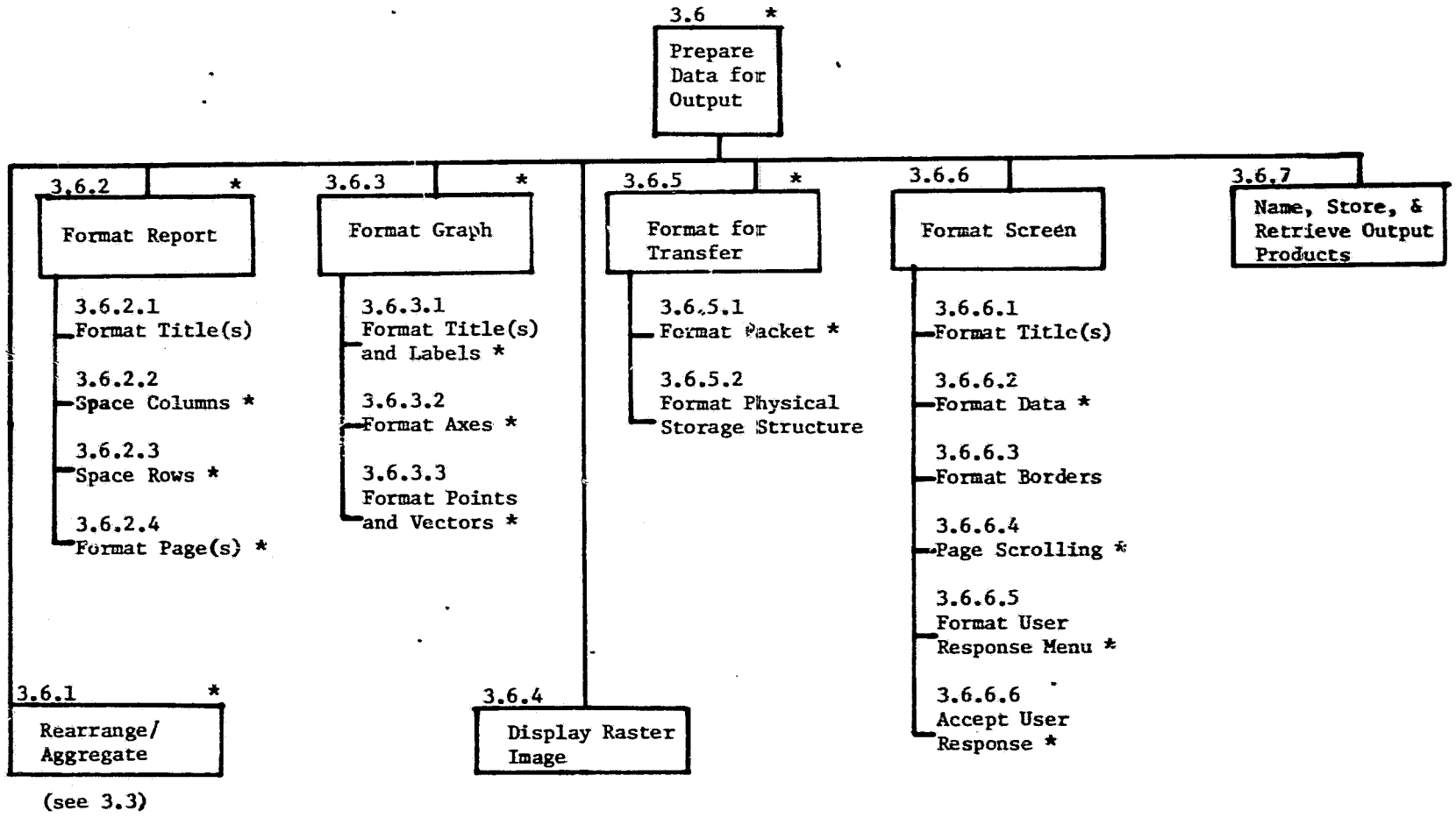


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)

* Essential to Parent Function

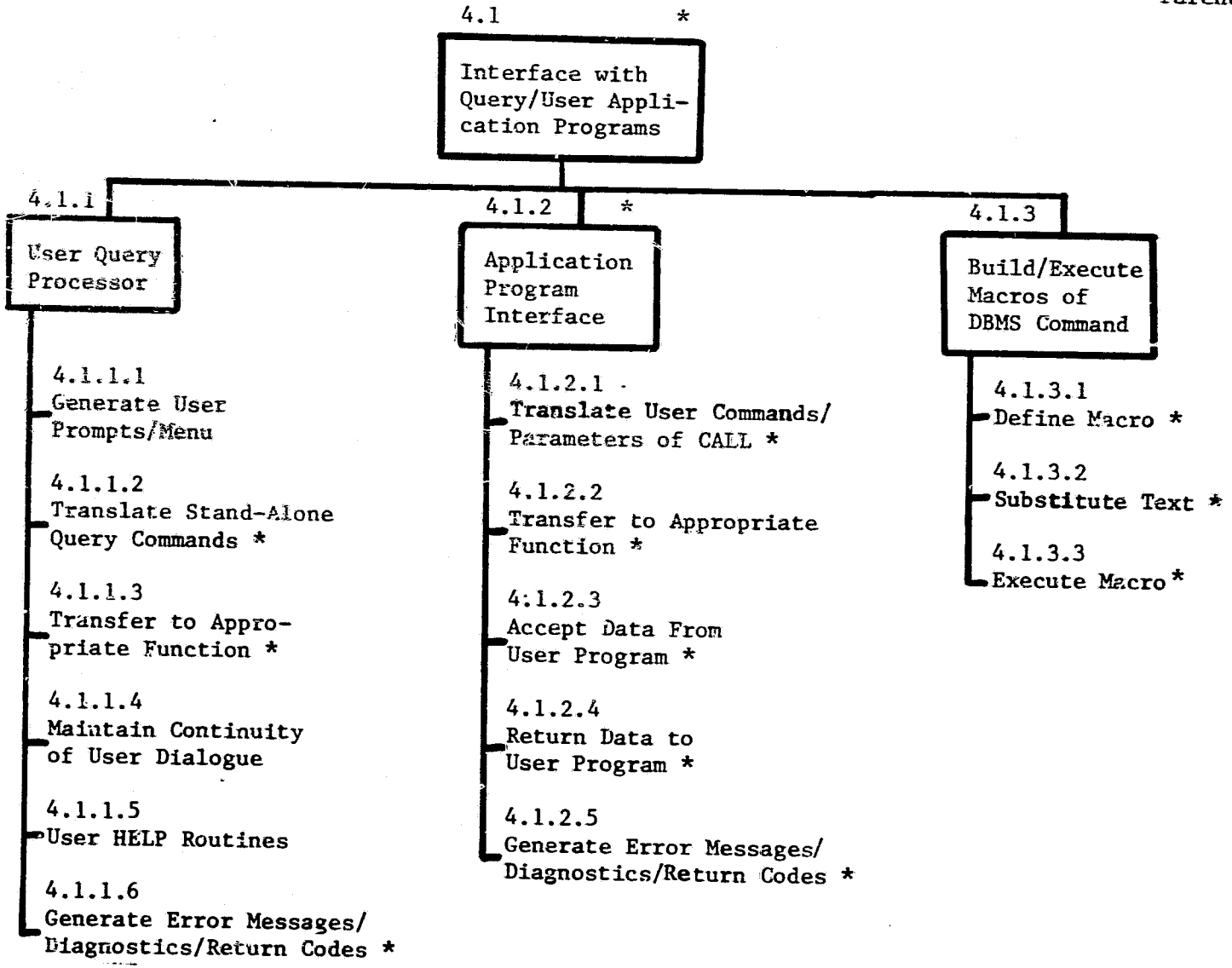


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)

* Essential to Parent Function

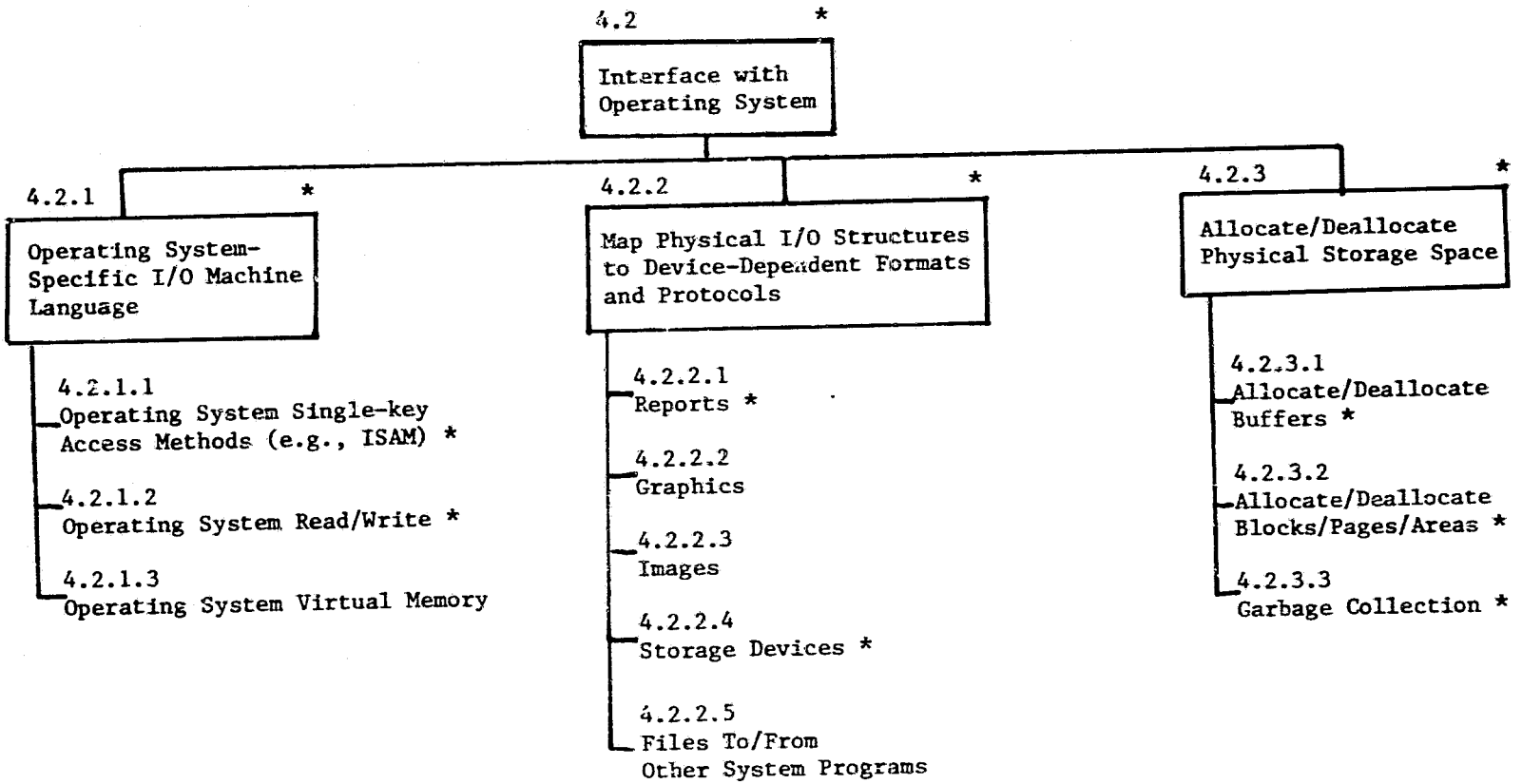


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)

* Essential to Parent Function

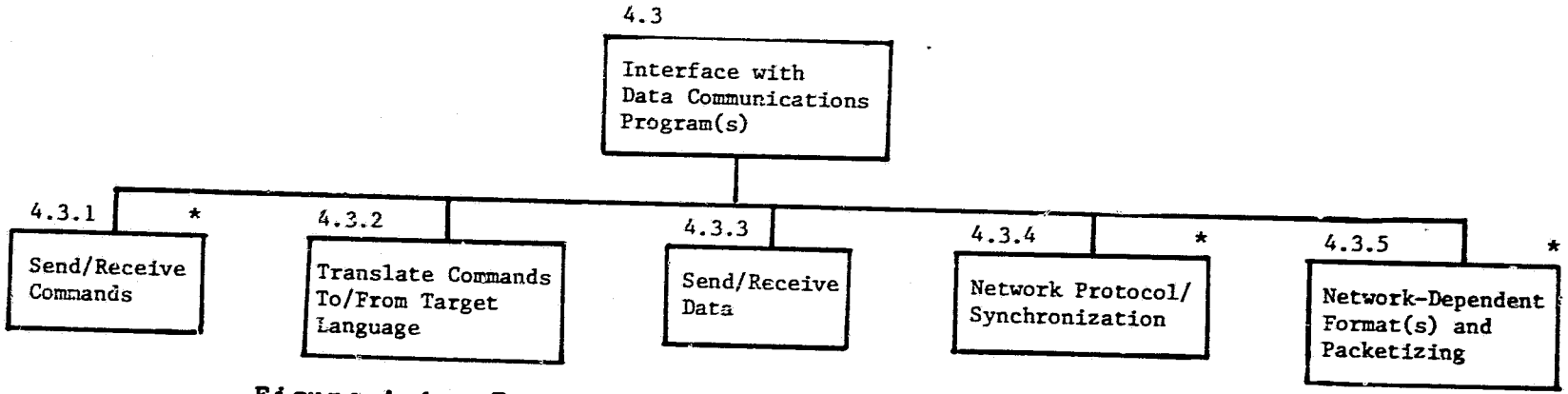


Figure A-1. Top-Down Functional Analysis of the Functions to be Performed by ADAM (cont)