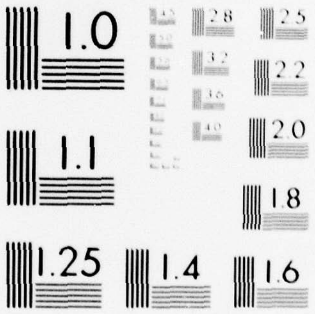
AD-A073 358   IIT RESEARCH INST CHICAGO IL   F/G 9/2
BASELINE SOFTWARE DATA SYSTEM. VOLUME I. SYSTEM DESCRIPTION.(U)
JUL 79 L M DUVALL   F30602-77-C-0052
UNCLASSIFIED   RADC-TR-79-185-VOL-1   NL

1 OF 1
AD
A073358

END
DATE
FILMED
10-79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

RADC-TR-79-185, Vol I (of two)
Final Technical Report
July 1979

BASELINE SOFTWARE DATA SYSTEM
System Description

IIT Research Institute

Lorraine M. Duvall

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-79-185, Vol I (of two) has been reviewed and is approved for publication.

APPROVED: John Palaimo

JOHN PALAIMO
Project Engineer

APPROVED: Wendall C Bauman

WENDALL C. BAUMAN, Col, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
18 1. REPORT NUMBER RADC TR-79-185 ~~Vol I (of two)~~ VOL-I	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER 9
6 4. TITLE (and Subtitle) BASELINE SOFTWARE DATA SYSTEM . Volume I. System Description .		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report . Feb 77 - Aug 78
		6. PERFORMING ORG. REPORT NUMBER N/A
10 7. AUTHOR(s) Lorraine M. /Duvall		15 8. CONTRACT OR GRANT NUMBER(s) F30602-77-C-0052
9. PERFORMING ORGANIZATION NAME AND ADDRESS IIT Research Institute 10 W. 35th Street Chicago IL 60616		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 63728F 16 55500857 17 08
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISIS) Griffiss AFB NY 13441		11 12. REPORT DATE Jul[redacted]979
		13. NUMBER OF PAGES 46
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same 12 46p.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: John Palaimo (ISIS)

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

database management
data requirements
software failure data

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

Volume I of this report provides a feature evaluation of the Management Data Query System (MDQS), a discussion of the contents of the Baseline databases, and a summary of the data required for software reliability modelling.

Volume II is a reference guide for defining and retrieving data from the Baseline databases.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

175 350

PREFACE

This final report, BASELINE SOFTWARE DATA SYSTEM, Volume I, System Description, was prepared by IIT Research Institute, Chicago, IL, as part of Contract Number F30602-77-C-0052. The work was sponsored by the Rome Air Development Center, Griffiss Air Force Base, New York, with Mr. John Palaimo serving as the RADC Technical Monitor for this program. The report covers work conducted during the period from February 1977 through August 1978.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EVALUATION

The objectives of this effort were to implement an experimental data repository and provide information processing tools to assist the in-house software reliability modeling program. This effort was initiated in response to an in-house requirement for a computerized database management capability for software error data. Sizeable collections of software error data had been acquired from several large software development projects for the in-house program.

This effort satisfactorily addressed all major program objectives. The Baseline Software Data System (BSDS) was successfully implemented on the RADC HIS 6180 computer system. Capabilities are available for defining, loading, updating and querying databases. The BSDS also provides capabilities for producing reports, generating data subsets, and interfacing with application programs.

In addition to the software error database, a summary database and a software productivity database were also implemented. The BSDS is currently being maintained by the Data and Analysis Center for Software (DACS) and will be expanded as more data becomes available.

This effort falls within the goals of the RADC Technology Plan, specifically TPO-5, C-3 System Availability (Hardware/Software), in subthrust Software Cost Reduction (Software Data Collection and Analysis).

John Palaimo

JOHN PALAIMO
Project Engineer

PRECEDING PAGE BLANK

Section I

INTRODUCTION

1.1 Study Objectives and Scope

The objectives of this study effort (Contract Number F30602-77-C-0052) were to provide RADC in-house research efforts with easy to use information processing tools to assist in their software reliability modeling efforts and to implement an experimental data repository to serve as a test bed for study and analysis of potential problems and solutions for the establishment and operation of the Data & Analysis Center for Software (DACS). The purpose of the DACS is to upgrade the software development process through the collection, analysis, and dissemination of software development experience information. The results of the study to develop the design for the center are reported in RADC-TR-76-387, Software Data Repository Study (reference 1).

RADC had previously acquired software error data from six large software development projects as reported in references 7 through 12. The data from these datasets were implemented as the Historical Database on the Honeywell 6180 Computer System at RADC using the General Comprehensive Operating Supervisor (GCOS) and the Management Data Query System (MDQS). These datasets were analyzed in terms of data content and compared to the data requirements for software reliability modelling studies. Also, the data from these datasets were summarized along with information from the Final Reports to form the Summary Database.

1.2 Report Contents

This volume, Volume I, provides in Section II a feature evaluation of the MDQS which was the database management software used for the implementation of the Baseline Software System. Section III contains an introductory discussion on each database and a summary of the evaluation of data requirements for software reliability models.

Volume II provides the user of the Baseline Software Data System with instructions for defining and retrieving data from the databases using MDQS.

Section II

MDQS FEATURE EVALUATION

The purpose of this section is to provide a feature evaluation of the MDQS which was used as the database management software for the implementation of the Baseline Software Data System. In this section, references are made to the applicable MDQS Manual and the page number that describes the feature in the form (report reference number, manual page number). Not all of the features of MDQS are discussed but only those that seem most important and had been previously defined as a database management requirement for the Software Data Repository (reference 1).

Included in this section is an overview of MDQS, a discussion on the database management tools provided for each user type and the characterization and structure of the database, a presentation on the MDQS capabilities for loading, maintaining, and retrieving data, a discussion on MDQS data security aspects, and the conclusions and recommendations of this evaluation effort.

2.1 MDQS Overview

MDQS is the Honeywell commercial offering of the World Wide Data Management System (WWDMS) developed for the World Wide Military Control and Command System (WWMCCS) and is a sub-system of the GCOS Operating System using both the time-sharing and batch environments. During this effort two versions of MDQS (designated System/IV (MDQS/IV)) were tested at the RADC Computer Center including:

MDQS Version	GCOS Version	Manual Reference Number
MD 2.0	1G.3	2 and 3
MD 2.2	2H.2	4 and 5

MDQS is a comprehensive database management system which provides the capabilities for database definition, creation, retrieval, maintenance, restructuring, and report generation and operates in both the online and batch environments. The term online is used here to denote the appearance to the user rather than the internal operational mode. The definitions are performed in the batch enviromnent but the job control language can be generated interactively online. There are capabilities to perform retrievals and maintenance in batch, online/batch, or online. The online capability is offered through the use of the

Conversational Management Data Query (CMDQ) which allows a user to interactively generate and execute a procedure from the terminal (5, 7-1).

2.2 Users

MDQS provides database management tools for the database administrator, the applications programmer, the nonprogrammer, and the parametric user. Facilities are provided to the database administrator to define, create, maintain databases and to establish file protection (all of reference 4).

Application programmers are computer professionals who are versed in the current practives of data processing. MDQS provides them tools for writing data subsets and interfacing to application programs, for processing difficult queries, and for generating reports (all of reference 5). A nonprogrammer (or general user) is typically a person who is knowledgeable in the functions of an organization but is not necessarily a computer professional. For this effort it is assumed the "nonprogrammer" is familiar with the software engineering field but does not know the structure of the database. The nonprogrammer can utilize some of the basic procedure and query language features to retrieve data and write simple reports (5, 2-12, and 5, 8-1). CMDQ can also be used by a nonprogrammer to interactively generate and execute simple procedures (5, 7-1). Parametric users are support personnel who do not have programming skills but do have the knowledge required to invoke predefined transactions. Facilities for the parametric user are provided by the capability to generate procedures where parameters are input at exectution time(5, 3-26).

2.3 Database Characterization and Structure

There are three MDQS databases in the Baseline Software Data System: the Historical Database, the Summary Database, and the RADC Productivity Database.

The Historical Database consists of six sequential datasets containing a total of 31,912 eighty-four character records. Below is a summary of the characteristics of each dataset.

Dataset Number	Number of Records	Number of Data Items	Number of Record Types
1	4,970	28	1
2	2,113	46	5
3	2,274	35	2
4	11,730	17	1
5	8,106	18	2
6	2,719	15	1

The Summary Database is an indexed-sequential database containing nine entries (record types) and 135 data items. Each entry contains a key field which is used to uniquely identify each record occurrence. The maximum size of the database is approximately seven million characters. (See Table 2-1 for a break out of the size for each entry for each project in the Historical Database).

The RADC Productivity Database is a sequential database containing 1200 eighty-four-character records consisting of three entires and 31 data items.

A description of the contents of each of these databases is provided in Volume II and in Section III of this volume.

These databases were defined using the three MDQS definition languages (Directory, Data, and Applicaiton). The Directory Definition Language defines the name of the database and the permfile names of the files associated with the database (4, 3-1). The Data Definition Language is a COBOL-like description language which describes attributes (length, data type, etc.) of the data items and the structure of the database. The Data Definition constitutes the schema (4, 4-1).

Sub-schemas are defined using the Application Definition Language which is the user's view of the data. This language defines all of the databases that are to be accessed by an MDQS procedure (4, 5-1).

Values of data items can be decoded using the Table-Lookup option in the Application Definition Language (4, 5-19) or in the Procedure Language (5, C-30). The tables can be generated using the PERFORM subsystem (5, C-25). The ENCODING/DECODING clause within the data definition can be used to specify a user subroutine that is to be executed whenever a data item requiring special conversion is to be processed or updated by a procedure (4, 4-21).

The database directory is available for display for an application definition by use of the Application Definition File Query (ADFQ) subsytem (5, 6-1). This capability allows for the listing of data-item name, type, and number of characters for each entry within an application definition.

Singular, hierarchical, and network are the three allowable MDQS data structures (4, 1-7). The singular data structure consists of only one type of element with no dominant or subordinate relationships while the hierarchical data structure consists of elements that can be related to any number of lower level elements but only one higher level element. The network data structure consists of elements that can be related to any

TABLE 2-1. SUMMARY DATABASE SIZE

Record Type	Project 1			Project 2			Project 3			Project 4		
	#Comp x	#Instances =	Total	#Comp x	#Instances =	Total	#Comp x	#Instances =	Total	#Comp x	#Instances =	Total
Component	283	1	283	77	1	77	112	1	112	65	1	65
Error												
Major	283	20	5,660	77	22	1,694	112	20	2,240	65	21	1,365
Minor	0	184	0	77	185	14,245	0	193	0	0	208	0
Instruction	283	1	283	77	2	154	112	1	112	65	1	65
Corrections	283	5	1,415	77	3	231	112	5	560	65	1	65
Technology	1	2	2	2	5	10	1	4	4	1	1	1
TOTAL			7,643			16,411			3,028			1,561

Record Type (cont'd)	Project 5			Project 6			Total		
	#Comp x	#Instances =	Total	#Comp x	#Instances =	Total	#Records x	Record Size =	# Char
Component	2442	1	2,442	33	1	33	3,012	257	774,084
Error									
Major	2442	21	51,282	33	21	693	62,934	67	4,216,578
Minor	0	205	0	0	0	0	14,245	67	954,415
Instruction	2442	1	2,442	33	1	33	3,089	114	352,146
Correction	2442	1	2,442	33	1	33	4,746	76	360,696
Technology	1	2	2	1	1	1	20	60	1,200
TOTAL			58,610			793	88,046		6,659,119

number of lower level elements and any number of higher level elements. Figure 2-1 contains a pictorial representation of the three data structures. The data structure represents the logical view of the data.

The allowable file organizations (storage structures) for MDQS are sequential, indexed sequential, and integrated. For a sequential file organization the records are stored serially and the only way of physically accessing a record is to read all records that precede it, beginning with the first record in the file.

An indexed sequential file is a collection of records that can be accessed either sequentially in key value order or randomly by a particular key value. It consists of a data file and an index file. An integrated file is a collection of records that may contain complex inter-record relationships where the record association is achieved through chains which provide cross-reference linkages between records. The allowable data structures for each file organization are illustrated in Table 2-2 (4, 1-7).

The integrated file structure is effected in MDQS by the use of Integrated Data Store (I-D-S) (references 13 and 15); and indexed sequential file by the use of the Indexed Sequential Processor (reference 14). These two file structures were studied to determine the feasibility of use for the Summary Database. It was determined that an indexed sequential file organization was the most effective means for implementing the Summary Database. When using an integrated file structure, the data definitions and query procedures become complex because of the need to define chains, retrieval mechanisms, and physical storage requirements (4, 4-34). By defining unique keys in the indexed sequential file for each record occurrence, a relational system was being effected. This then provides more flexibility to expand the definitions and to transfer to another data management system, if requirements dictate.

TABLE 2-2. DATA STRUCTURES/FILE ORGANIZATIONS

	File Organization		
Data Structures	Sequential	Indexed Sequential	Integrated
Singular	X	X	X
Hierarchal	X	X	X
Network			X

The RADC Productivity Database, the transaction files for the Summary Database, and the six datasets for the Historical Database are defined as sequential files with singular data structures. The Summary Database is defined as an indexed sequential file and a hierarchal data structure.

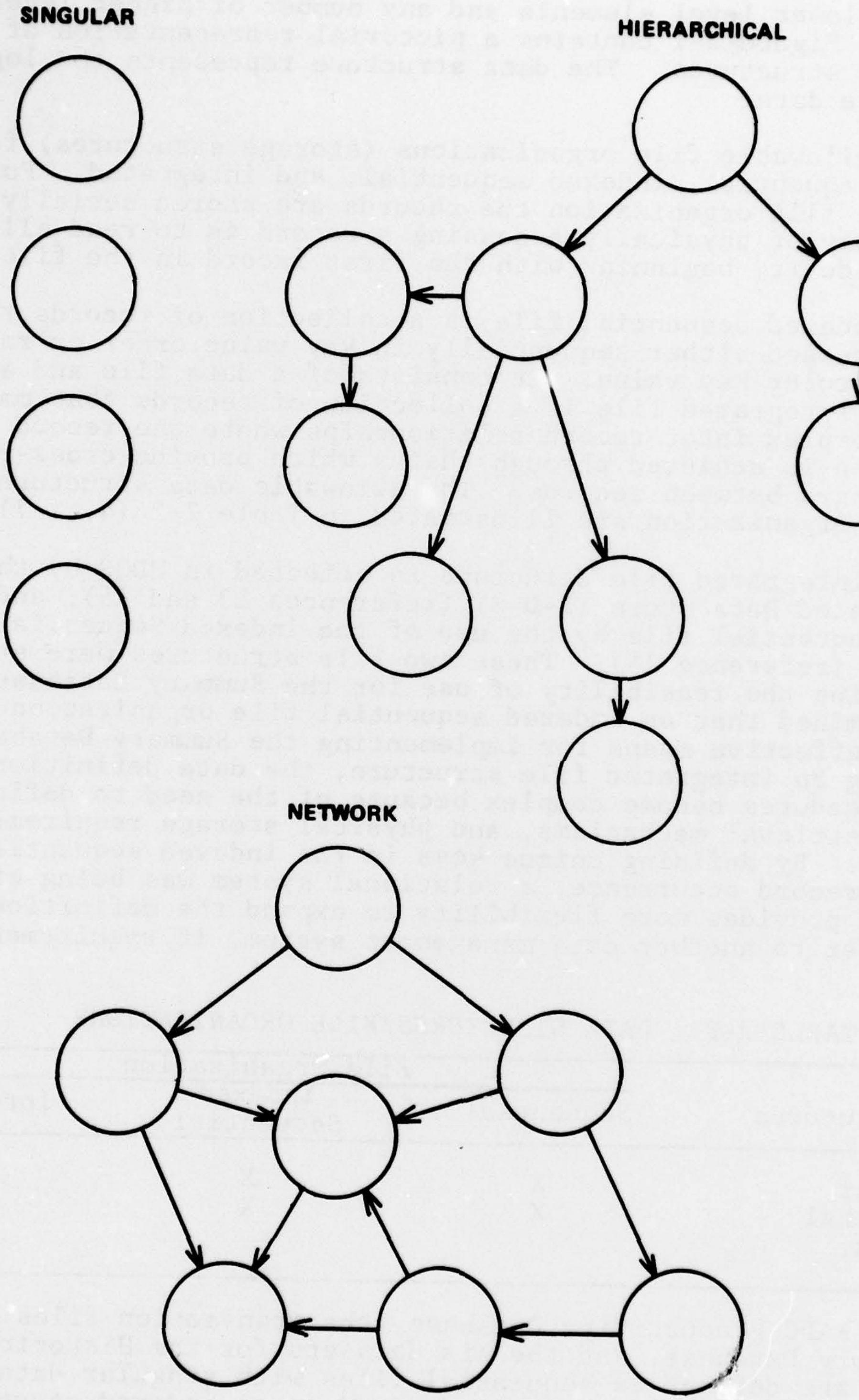


Figure 2-1. Allowable MDQS Data Structures

2.4 Data Loading and Maintenance

There are various options for initially loading a database dependent upon file structure. The data can be loaded external to MDQS through the use of system utilities, HOL programs using the standard I/O Routines, the Indexed Sequential Processor, (reference 15) or Integrated-Data-Store, (reference 13, 14) and must follow the standards for the specific file structure (4, 2-8). The Historical and Productivity Databases were loaded using utilities (see reference 6, Appendix B). The Summary Database was loaded using a combination of the Indexed Sequential Processor, Fortran programs, and the MDQS LOAD function.

Within MDQS the self-contained capability of the data can be loaded using the LOAD function of the Conversational MDQS Language (CMDQ) Subsystem (5, 7-200). The LOAD function is used to generate a new sequential or indexed sequential entry from a terminal using a prompting method for inputting data item values.

The READ statement of the Procedure Language (5, 5-103) causes data to be read from a non-database file into a specified structure and can be read from a permfile on a removable device or a magnetic tape.

Updating is performed (except for sequential) by the use of the UPDATE function within CMDQ (5, 7-11), by the use of the UPDATE statement of the Procedure Language (5, 5-149), and by the use of the UPDATE clause in the RETRIEVE Statement of the Procedure Language (5, 5-131). These are used in conjunction with other statements of the Procedure Language including DELETE (5, 5-48), INSERT (5, 5-63), STORE (5, 5-146), and RESTORE (5, 5-126). There are restrictions on the use of these capabilities and Appendix F of reference 5 provides guidelines for using these functions dependent upon the file structure. The use of this updating feature requires that separate transaction files be initially generated with the updated data and then updating is performed. Figure 2-2 illustrates the overall flow for updating the indexed-sequential Summary Database with a sequential transaction file.

MDQS does not provide for a Host Language capability where an application program can directly access the database through the use of a CALL or language verb. However, if the database is an integrated file I-D-S can be used, if the database is index-sequential the index-sequential processor can be used, and if the database is sequential the GCOS file system can be used which is standard for all the GCOS procedure languages.

The Data Directory feature in MDQS allows for the listing of the attributes of data items and entries within a database, but does not provide cross-reference information in terms of relationships to other data items or the utilization of the items.

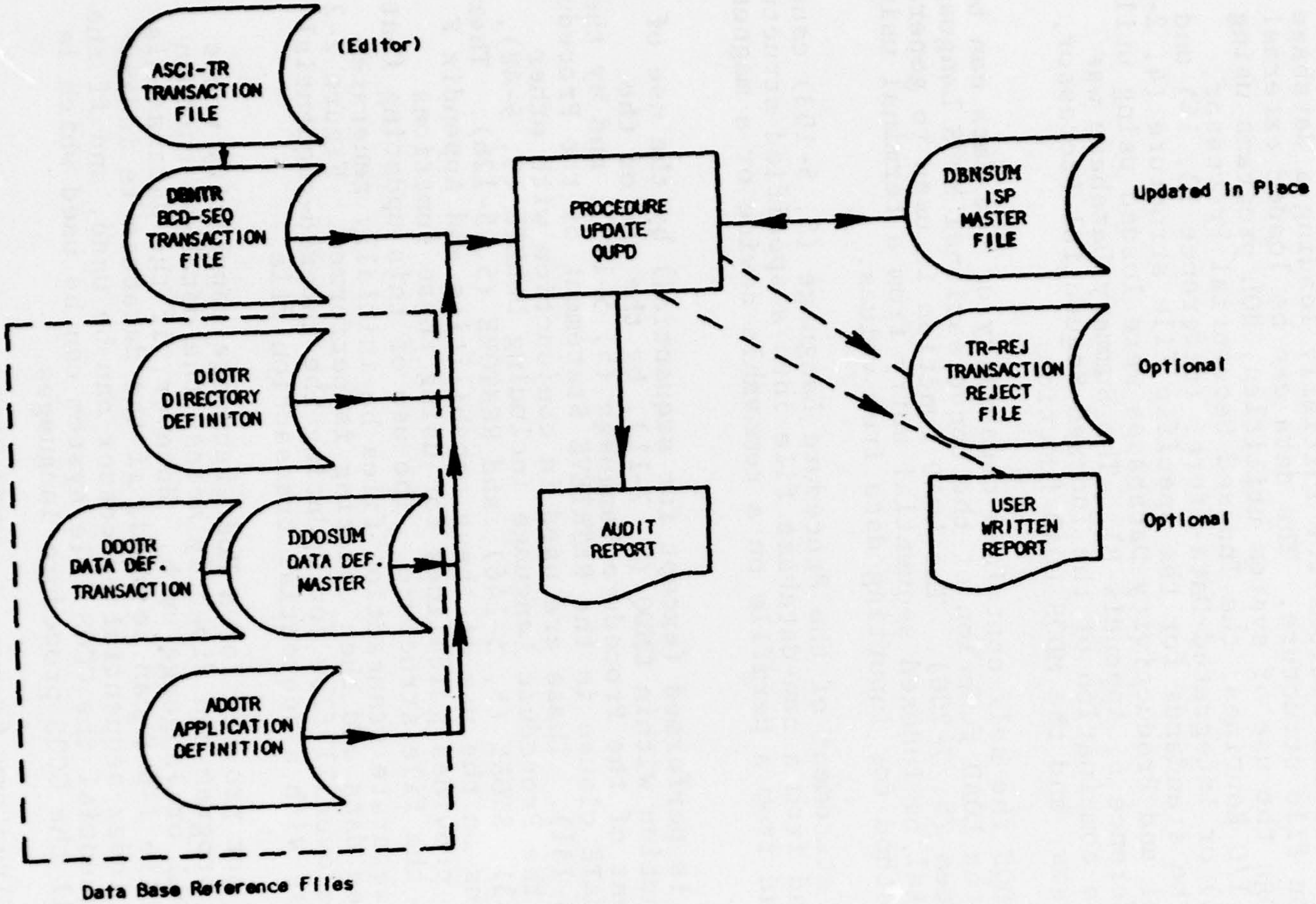

ASCI-TR TRANSACTION FILE
(Editor)
DBNTR BCD-SEQ TRANSACTION FILE
DIOTR DIRECTORY DEFINITON
DDOTR DATA DEF. TRANSACTION
DDOSUM DATA DEF. MASTER
ADOTR APPLICATION DEFINITION
Data Base Reference Files
PROCEDURE UPDATE QUPD
AUDIT REPORT
DBNSUM ISP MASTER FILE
Updated In Place
TR-REJ TRANSACTION REJECT FILE
Optional
USER WRITTEN REPORT
Optional

Figure 2-2. MDQS UPDATE SEQ ISP

Validity checking is performed by the use of the CHECK/IS clause in the data definition (4, 4-19) and is executed whenever a value is changed or added to the data item during a batch execution by procedure. The actual checking is accomplished by a user subroutine and/or by specifying the valid PICTURE clause and a value range.

MDQS provides facilities for reorganizing the PICTURE and USAGE clauses and adding or deleting groups, data items, and records. The picture changes allowed are those as permitted by a COBOL MOVE statement.

New Directory Definitions and Data Definitions must be translated and then the actual restructuring is performed using an MDQS utility function (4, 2-10) 4, 6-1). Figure 2-3 illustrates the basic steps needed for restructuring a sequential or indexed-sequential file. The new and old data definition source code is used as input to an MDQS utility routine and a COBOL program is generated, compiled, and executed performing the restructuring. The process for integrated files uses I-D-S utility programs.

MDQS provides for a checkpoint and restart capability for both the database entry that is being used during a procedure and the coincident memory image of the procedure (5, 5-34 and 5, 5-164). The frequency of checkpoints can be specified and a segment of a procedure is executed through the use of the CHECKPOINT/ROLLBACK statement. The capability is only valid for those databases which have concurrent update protection specified in the Directory Definition and the SHARED or EXCLUSIVE mode in the procedure.

MDQS does not provide for the capability of capturing information about changes made to the database and usage characteristics although various logging facilities and sampling techniques of GCOS can be utilized.

2.5 Retrieval and Report Generation

Through a self-contained procedure language, the MDQS retrieval and report generation capability provides for qualifying a subset of the database, sorting and/or formatting this subset, and printing this subset directly to the requesting computer terminal. The basic retrieval capability is accomplished by the use of the INVOKE and RETRIEVE statements with the incorporation of a conditional expression which qualifies the data subset of interest (5, 5-65 and 5, 5-128). The SORT statement specifies the order of the sort according to a maximum of 50 key fields (5, 5-141).

MDQS procedures may reference user application COBOL, Fortran or GMAP programs that perform data validation, encoding and decoding, table lookups, and data transformation (5, C-1).

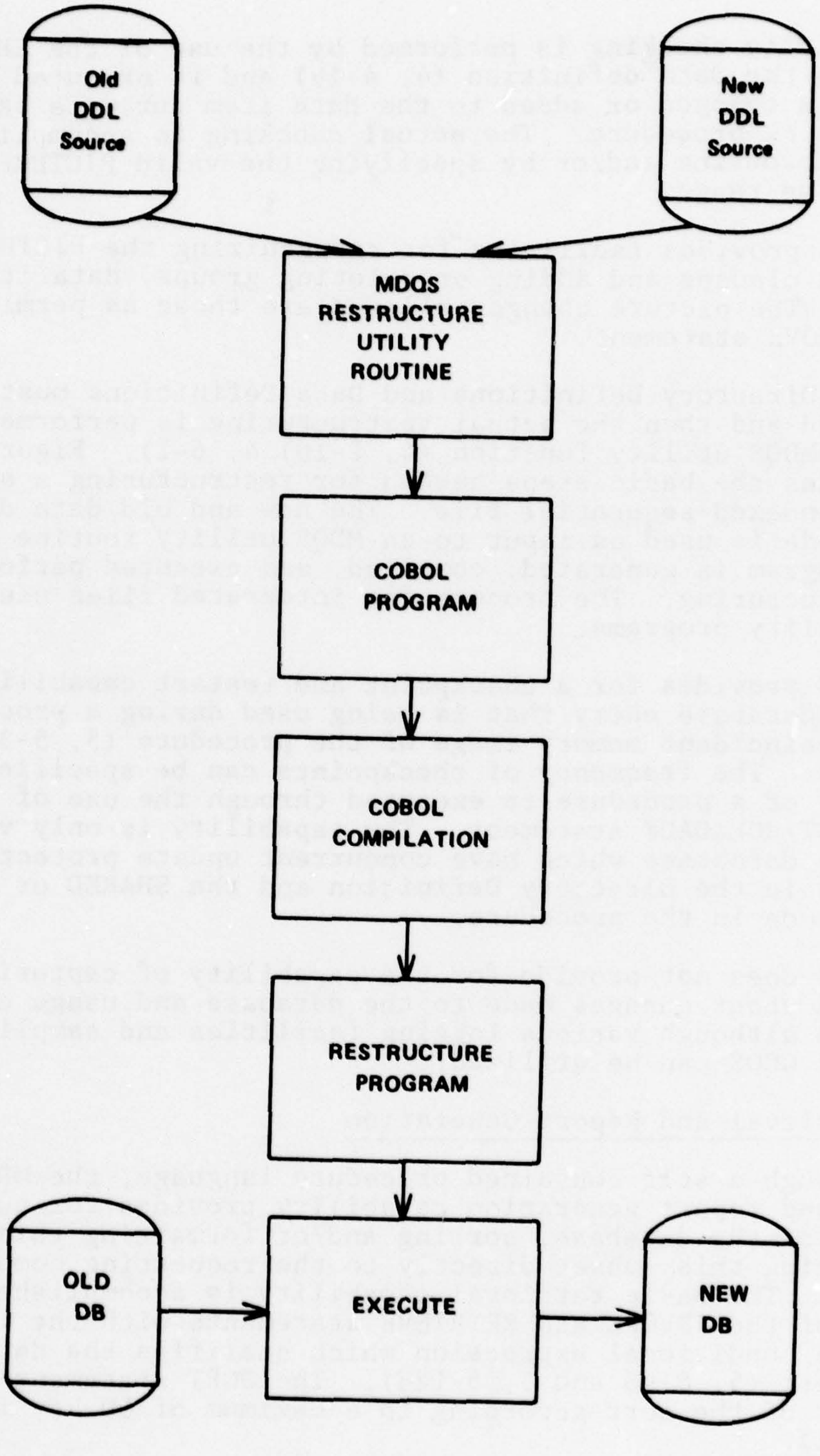


Figure 2-3. The Basic Steps for Restructuring

In addition, the results of a procedure can be written to a system standard permfile and subsequently utilized by an application program.. The results of a retrieval can also be output to the printer, to the online terminal, to a magnetic tape, or to a permfile that can be printed on the terminal (5, 5-81).

A tutorial method for generating MDQS procedures is available through the use of the CMDQ subsystem (5, 7-1) and a more simplified method of retrieving data than the standard procedure language is through the use of the Query Procedure Language (5, 8-1). A capability with the procedure language allows for the definition of parameters to be inserted at execution time (5, 3-26).

An extensive reporting capability is available through the use of the REPORT, LINE, and SPACE STATEMENTS (5, 2-27) and through the use of various editing options (5, 3-30).

Multiple users can access an MDQS database concurrently through a concurrent access environment which protects the integrity of the contents of the files and prevents interference between multiple users (4, 2-10). The databases must initially be established with concurrent update protection by using the GCOS File Management Supervisor (FMS) ACCESS/MONITOR and ABORT/ROLLBACK options. The database access is then defined as PROTECTED in the Directory Definition (4, 3-6).

2.6 Security

MDQS uses the GCOS File Security System (FILSYS) for file security and provides facilities for specifying the privacy protection and for controlling access to the databases by MDQS procedures (4, 1-2). The Data Base administrator is responsible for assigning locks and keys, generating a privacy file, and defining the locks in the Data Definition (4, 7-1).

The Privacy file is created by the use of the Privacy Command within the Privacy subsystem and establishes corresponding locks and keys for User IDs (4, 7-7). The privacy locks at the record level are defined in the record complete entry of the Data Definition where the lock(s) supply to the reading and writing by an MDQS Procedure for all data items within the record (4, 4-8). The locks for each individual data item are defined in the group/item entry of the Data Definition (4, 4-22).

2.7 Conclusions and Recommendations

Overall MDQS provides the basic database management features necessary for the implementation of a Data and Analysis Center for Software (DACS). The three most powerful features of MDQS are its report production capabilities, data structuring alternatives, and the database administrator tools including the schema-subschema facility. It is also very important that during this effort MDQS, with only a few exceptions, performed its functions as described in the documentation. The weakest feature of MDQS is the syntax of the Procedure Language in that it is somewhat cumbersome to use for generating complicated queries. Also MDQS is limited in the tools it provides for database maintenance. These limitations can be compensated for through the use of GCOS utilities and user HOL programs.

It is recommended that MDQS continue to be used as the database management software for the development of the Baseline Software Data System to establish the framework for the evolution into a pilot DACS and then into a fully operational center.

Section III

DATABASE DESCRIPTIONS AND DATA REQUIREMENTS

This section provides an introduction to the Historical, Summary, and Productivity Databases. Also included in this section is a summary of the work performed during this effort on the evaluation of data requirements for software reliability models. The types of data required are listed in Figure 3-1 along with a short description of each data item.

3.1 Baseline Databases

3.1.1 Historical Database. The Historical Database consists of six datasets that contain problem reporting and module descriptive information on six large software development projects. The data items available for each dataset are indicated in Table 3-1 using as a basis the data items listed in the data requirements list (Figure 3-1). There are two columns associated with each project. The first column provides the number of characters that are needed to represent the data item, and the second column indicates the maximum number of occurrences for each problem recorded.

Following is a short description of the six projects that constitute the data for the Historical Database.

Project 1 - This dataset contains Software Problem Reports (SPR) from a large Command and Control System consisting of 115,346 Jovial/J4 source statements and 249 program modules. The Project itself and the dataset is discussed in Reference 7 and is referred to as Project 3.

There is a total of 4,970 Software Problem Report records consisting of the SPR number, the date opened and closed, the module which manifested the error, the module that was changed, the error category and the severity of the error, the test period, the correction type, and the Software Modification Notice (SMN) number. There is a record occurrence for each modification made. Every SPR required at least one SMN, and one SMN could have closed more than one SPR. Therefore, the SPR numbers are not unique and the SMN numbers are not unique.

Project 2 - This dataset contains Software Problem Reports and Module descriptions from an Avionics System consisting of 40,640 Jovial/J3B source statements and 84,065 Assembly Language statements. The description of the collection and analysis of this dataset is contained in Reference 8.

010	PROJ-ID	PROJECT IDENTIFICATION
020	PROJ-VERSION	PROJECT VERSION
030	PROJ-TYPE	PROJECT TYPE
040	SYS-ID	SYSTEM IDENTIFICATION
050	SYS-VERSION	SYSTEM VERSION
060	SYS-TYPE	SYSTEM TYPE
070	SSYS-ID	SUBSYSTEM OR FUNCTIONAL AREA IDENTIFICATION
080	SSYS-VERSION	SUBSYSTEM VERSION
090	SSYS-TYPE	SUBSYSTEM TYPE
100	MOD-ID	MODULE IDENTIFICATION
110	MOD-VERSION	MODULE VERSION
120	MOD-TYPE	MODULE TYPE
130	COMP-ID	COMPUTER IDENTIFICATION
140	COMP-OM	COMPUTER OPERATING MODE
150	COMP-RATE	COMPUTER PROCESSING RATE
160	COMP-OS	COMPUTER OPERATING SYSTEM TYPE
170	TECH-ID	IDENTIFICATION OF THE CONSTRUCTION TECHNOLOGY
180	COMPL-ID	TYPE OF COMPLEXITY MEASURE USED
190	COMPLEXITY	THE COMPLEXITY MEASURE VALUE
200	CONST-TYPE	CONSTITUENT TYPE(EX. JOVIAL,ASSEMBLY LANGUAGE)
210	NUM-OCCUR	NUMBER OF OCCURRENCES OF CONSTITUENT TYPE
220	PHASE	PHASE IN WHICH ACTION OCCURRED
230	NUM-RUNS-TOT	TOTAL NUMBER OF RUNS
235	TEST-PER	THE PERIOD IN WHICH THE TEST WAS PERFORMED
240	NUM-RUNS-OK	TOTAL NUMBER OF CORRECT RUNS
250	AHRS-PER-TEST	AVERAGE NUMBER OF HOURS PER TEST
260	TEST-ID	TEST IDENTIFICATION
270	TEST-TYPE	TYPE OF TEST
280	DATE-RUN	DATE THE TEST WAS RUN
290	STRESS-TYPE	TYPE OF STRESS APPLIED
300	STRESS-MEAS	AMOUNT OF STRESS APPLIED
310	TEST-RESULT	RESULT OF TEST
315	NUM-ERR	NUMBER OF ERRORS DISCOVERED PER TEST
320	SPR-NUM	SOFTWARE PROBLEM REPORT NUMBER
330	DATE-OPEN	DATE THE PROBLEM WAS REPORTED
340	MOD-SOURCE	THE MODULE ID WHERE THE PROBLEM WAS MANIFESTED
350	ERR-CAT-TYPE	ERROR CATEGORY TYPE
360	ERROR-CAT	ERROR CATEGORY CODE
370	SEV-TYPE	SEVERITY TYPE
380	SEVERITY	SEVERITY
390	TYPE-TERM	TYPE OF TERMINATION
400	HRS-TO-DISC	HOURS TO DISCOVERY
405	WORK-CAT	THE TYPE OF DEVELOPMENT TASK PERFORMED
410	SMN-NUM	SOFTWARE MODIFICATION NOTICE NUMBER
420	MOD-CHANGED	THE ID OF THE CHANGED MODULE
430	MOD-CH-VERS	THE VERSION OF THE CHANGED MODULE
440	COR-TYPE	CORRECTION TYPE
450	COR-MECH	CORRECTION MECHANISM
455	ACT-CAT	THE TYPE OF TEST PERFORMED
460	DATE-BEGUN	DATE WHEN PROBLEM SOLUTION WAS INITIATED
470	DATE-CLOSE	DATED WHEN PROBLEM WAS REPORTED TO BE CLOSED
480	DAYS-OPEN	NUMBER OF DAYS BETWEEN DATE OPEN AND DATE CLOSE
490	HHRS-TO-FIX	HUNDRETHS OF HOURS TO FIX
500	NUM-CHANGED	NUMBER OF SOURCE STATEMENTS CHANGED
510	CODE-CONT	A CODE THAT INDICATES AN SPR DOCUMENTS MORE THAN 1 PROBLEM
520	PROB-DESC	A DESCRIPTION OF THE PROBLEM
530	CORR-DESC	A DESCRIPTION OF THE CORRECTION
540	ERROR-DESC	A DESCRIPTION OF THE ERROR

Figure 3-1. Baseline Data Requirements List

TABLE 3-1. ATTRIBUTE MATRIX

AUGUST 1978

ATTRIBUTE	PROJ 1		PROJ 2		PROJ 3		PROJ 4		PROJ 5		PROJ 6	
NAME	NUM	MAX	NUM	MAX	NUM	MAX	NUM	MAX	NUM	MAX	NUM	MAX
	CHR	NUM	CHR	NUM	CHR	NUM	CHR	NUM	CHR	NUM	CHR	NUM
PROJ-ID	2	1			5	1						
PROJ-VERSION	6	1					3	1				
PROJ-TYPE												
SYS-ID			1	1			1	1				
SYS-VERSION							2	1				
SYS-TYPE			1	2								
SSYS-ID			4	1			3	1	1	1	3	1
SSYS-VERSION							3	1	7	1		
SSYS-TYPE							1	1				
MOD-ID			4	1	7	1			8	1	16	1
MOD-VERSION					2	1						
MOD-TYPE					1	1	1	1				
COMP-ID			13	1								
COMP-OM												
COMP-RATE			7	1								
COMP-OS			13	1								
TECH-ID			11	1	1	1			12	1		
COMPL-ID												
COMPLEXITY					1	1						
CONST-TYPE			1	2	1	1			7	1		
NUM-OCCUR			5	2	5	2			6	1		
PHASE			1	1	1	1	1	1	2	1		
NUM-RUNS-TOT			3	1								
TEST-PER	2	1			1	1					1	1
NUM-RUNS-OK			3	1								
AHRS-PER-TEST			3	1								
TEST-ID	8	1										
TEST-TYPE												
DATE-RUN											5	1
STRESS-TYPE												
STRESS-MEAS											6	1
TEST-RESULT											1	1
NUM-ERR											1	1
SPR-NUM	4	1	3	1	4	1	4	1	7	1		
DATE-OPEN	6	1	6	1	6	1			6	1		
MOD-SOURCE	7	1										
ERR-CAT-TYPE												
ERROR-CAT	5	1	5	1	5	1	4	1	5	1	2	1
SEV-TYPE												
SEVERITY	1	1			1	1					1	1
TYPE-TERM	1	1			1	1						
HRS-TO-DISC			5	1								
WORK-CAT											1	1
SMN-NUM	6	1			4	1	6	1				
MOD-CHANGED	7	1	4	13	7	1			8	1		
MOD-CH-VERS					2	1						
COR-TYPE	6	1	1	1	5	1			9	1		
COR-MECH			1	1								
ACT-CAT											1	1
DATE-BEGUN												
DATE-CLOSE	6	1	6	1	6	1	6	1	6	1		
DAYS-OPEN	3	1			3	1						
HHRS-TO-FIX			5	1								
NUM-CHANGED											1	1
CODE-CONT	1	1	1	1								
PROB-DESC									99	3		
CORR-DESC									99	3		
ERROR-DESC							50	1				

There is a total of 2,036 Software Problem Report records containing the SPR number, the date opened and closed, the module(s) that were changed, the error category, the phase in which the error was introduced, the CPU hours to discovery, the correction type, and the hundreths of hours of CPU time to fix. Every SPR number is unique and if more than one module is needed to be changed all the module names are contained in the same record.

There are data on 69 modules which contain the name of the module and a funtional area designation, the programming language(s) used and the number of source statements. There are eight records that contain descriptive information on the type of hardware and software used and descriptions of the testing phases.

Project 3 - This dataset consists of Software Problem Reports and Module descriptions from a real-time control system for a land-based radar system. The software system is made up of 109 modules with a total of 86,780 Jovial/J3 source statements and 49,000 Assemble Language statements. The description of this project is contained in Reference 9.

There is a total of 2,165 Software Problem Report records containing the SPR number, the date opened and closed, the module that was changed, the error category and the severity of the error, the test period, the phase in which the error was introduced, the correction type, and the Software Modification Notice number. There is one record occurrence for each modification made and each SMN number is unique. The SPR numbers and the SMN numbers are the same except that there are some blank SPR numbers.

Project 4 - This dataset contains Software Modification Reports from the flight software of an onboard guidance, navigation and control system for both a command module and a lunar module. There were 16 flight programs (releases) and the total number of computer words for all releases was approximately 610,000 computer words. The sum of the number of words added or changed since the last release was 83,866. The majority of the software was coded using assembly language with interpretive code interspersed throughout. A description of this project and an interpretation of the data is contained in Reference 9.

There is a total of 11,730 Software Problem Report records containing the SPR number, the date closed, the error category, the phase in which the error was introduced, and the SMN number. There is a record occurrence for each modification made and each SMN number is unique. The SPR number references a document that established the basis for the change but is only available for about 13% of the records.

Project 5 - This dataset consists of Software Problem Reports and Module descriptions from a large, ground-based, real-time data processing system. The majority of the Software was coded using CENTRAN (an intermediate - level language resembling a subset of PL/1) interspersed with assembly language and system macros. A description of this project is contained in reference 11.

There is a total of 5,693 Software Problem Report records containing the SPR number, the date opened and closed, the module that was changed, the error category, the phase in which the error was introduced, and the correction type. There is a record occurrence for each problem encountered. If the problem required more than one solution, only one solution was recorded which was established using a priority scheme.

There are data on 2,431 modules which contain the name of the module, the number of instructions, the language used, and the type of construction.

Project 6 - This dataset consists of run and failure analysis data from the development of the Launch Support Data Data Base (LSDB) which includes database management functions and fairly complex scientific calculations.

There is a total of 2,719 run analysis records that report 484 errors. The records contain the module ID, the date and time run, the result of the test, the test period and activity, the severity, error category, and number of errors. There is a record occurrence for each run (test) made.

Below is a summary of the size of the datasets within the Historical Database.

	Software Problem Reports	Module Characteristics
Project 1	4,970	---
Project 2	2,036	69
Project 3	2,165	109
Project 4	11,730	---
Project 5	5,693	2,413
	Run Analysis Reports	
Project 6	2,719	---

3.1.2 Summary Database. The Summary Database was developed so that queries could be formulated across the projects. The failure and correction information from the Historical Database was summarized and incorporated into the Summary Database. The project/module attribute, environment, and productivity data from the Final Reports (references 7-12) were extracted, coded and put into computer readible form.

Figure 3-2 illustrates the three-dimensional aspect of the Summary Database.

Software environment, technology, resource utilization, production, and software characteristics data is stored for various reporting periods for the life-cycle phases. In addition, four levels of descriptive information are used to describe the software: the project, system, functional group, and module levels. A project consists of one or more systems and provides a solution to a problem. A system consists of one or more functional groups and provides a meaningful product to the user. A system is usually capable of operating independently of other systems. A functional group is a collection of modules which together satisfy a set of functional and performance specifications. A module is a discrete identifiable set of instructions handled as a unit by an assembler, compiler, or loader. Queries can be formulated across the projects, modules, systems, and functional groups.

Data summary forms were developed to record information from the technical reports for the six datasets in the Historical Database and to provide summarization requirements to convert the data from the datasets into the format required for the Summary Database. Each form contains eight fields that provide a basis for defining a unique key for each record occurrence within the Summary Database. This key identifies the applicable project, system, functional group, and module that applies to the component information recorded. Also included in this key is information concerning the level of summarization and the record type which indicates the format of the data.

In addition to the key data, the following information is recorded on each form.

Component (see Figure 3-3). Component name, type, and description; developer, contract number, and data source; the number of systems, functional groups and modules; contract type and standards applied; the purpose of the data collection and the procedures used; the priorities and constraints of the product development.

Technology (see Figure 3-4). The phase, reporting level and the applicable dates; the technology utilized, the name of the tool used, and the percentage of usage.

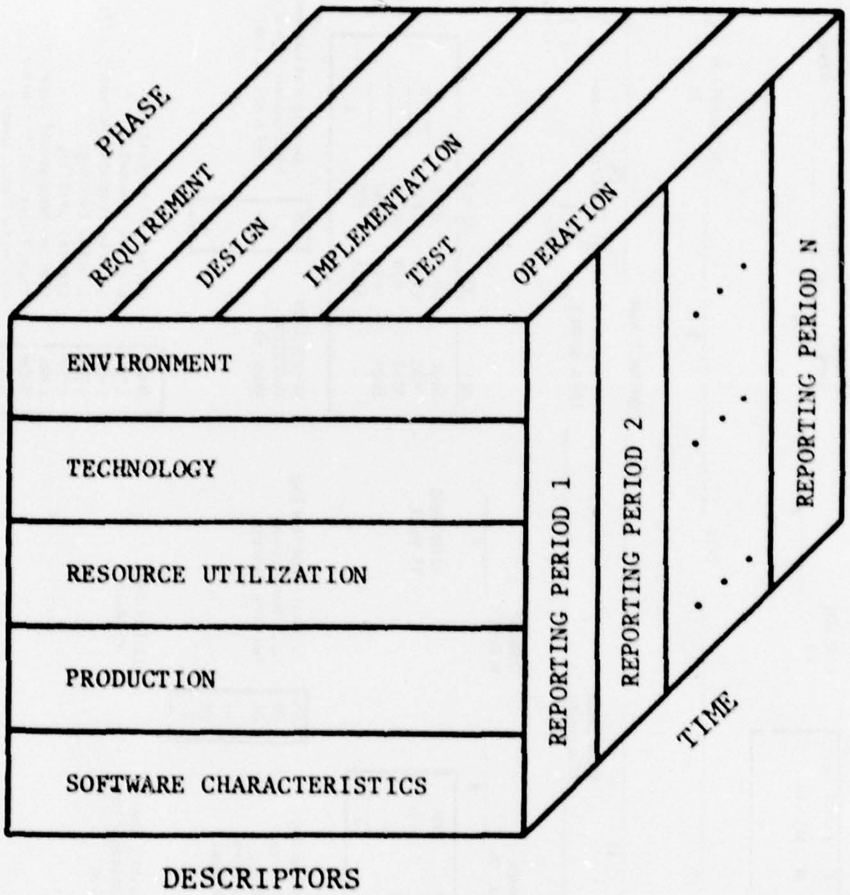


Figure 3-2. Summary Database

DATA SUMMARY FORM - COMPONENT

PROJECT ID ________ 6 / SYSTEM ID ________ 4 / FUNC GP ID ________ 5 / MODULE ID ________ 9

SUMMARIZATION LEVEL

P	Project	F	Funct Gp
S	System	M	Module
1			

VERSION ID ________ 7

RECORD TYPE 010 ________ 3

RECORD NUMBER ________ 3

NAME ________ 12

TYPE ________ 3

DESCRIPTION ________ 28
____

DEVELOPER ________ 11

CONTRACT NUM ________ 24

DATA DATE (Initial) ________ 6

DATA DATE (Current) ________ 6

DATA SOURCE 2 01 RADC Purchased [ ] [ ]
[ ] [ ] [ ]

NUMBER SYSTEMS ________ 4

NUMBER FUNC GR ________ 4

NUMBER MODULES ________ 4

CONTRACT TYPE (5 Max)

PRIM	CPFF	FFP
SUB	CDIF	FPIF
CO	COST	___
INT	TM	___
		4

STANDARDS (5 Max)

AF	DOD	OTHER	
3101	5010	8506	___
483	4120	5233	___
3751	490	4130	___
8003	481	8040	___
	100A	700	
			4

COLLECTION PURPOSE (5 Max)

CO	Contract-During
CA	Contract-After
VI	Visibility
EP	Evaluate Pers.
EA	Evaluate App.
SM	Schedule Monitoring
CM	Cost Monitoring
QC	Quality Control
___	
___	
2	

COLLECTION PROCEDURE (Max 5)

CM	Configuration Management
LI	Automated Library
IN	Internal Manual
___	
2	

PRIORITIES (5 Max)

PR	Processing Speed
CO	Core Utilization
SC	Schedule
CS	Cost
QU	Quality
___	
2	

CONSTRAINTS (5 Max)

HRDW	Hardware Limitations
LISC	Limited Schedule
LICM	Limited Computer Accessibility
LIFN	Limited Funding
LIST	Limited Staffing
LIMS	Limited Management Support
TRCM	Target Computer Different
UNST	Unstable Requirements
NEWA	New Application Area
___	
4	

DOCUMENT ACCESSION NUMBERS (Max 5) ________ 5 ________ ________ ________ ________

Figure 3-3. Component Data Summary Form

DATA SUMMARY FORM - TECHNOLOGY

PROJECT ID ______ 6 / SYSTEM ID ______ 4 / FUNC GP ID ______ 5 / MODULE ID ______ 9

SUMMARIZATION LEVEL

P	Project	F	Funct. Gp
S	System	M	Module

I

VERSION ID ______ 7

RECORD TYPE 020 ______ 3

RECORD NUMBER ______ 3

PHASE

C	Conceptual
R	Requirements
D	Design
I	Implementation
T	Test
O	Operation

2

REPORTING LEVEL

MO	Monthly
PH	Phase
T	Total

2

BEGIN DATE ______ 6

END DATE ______ 6

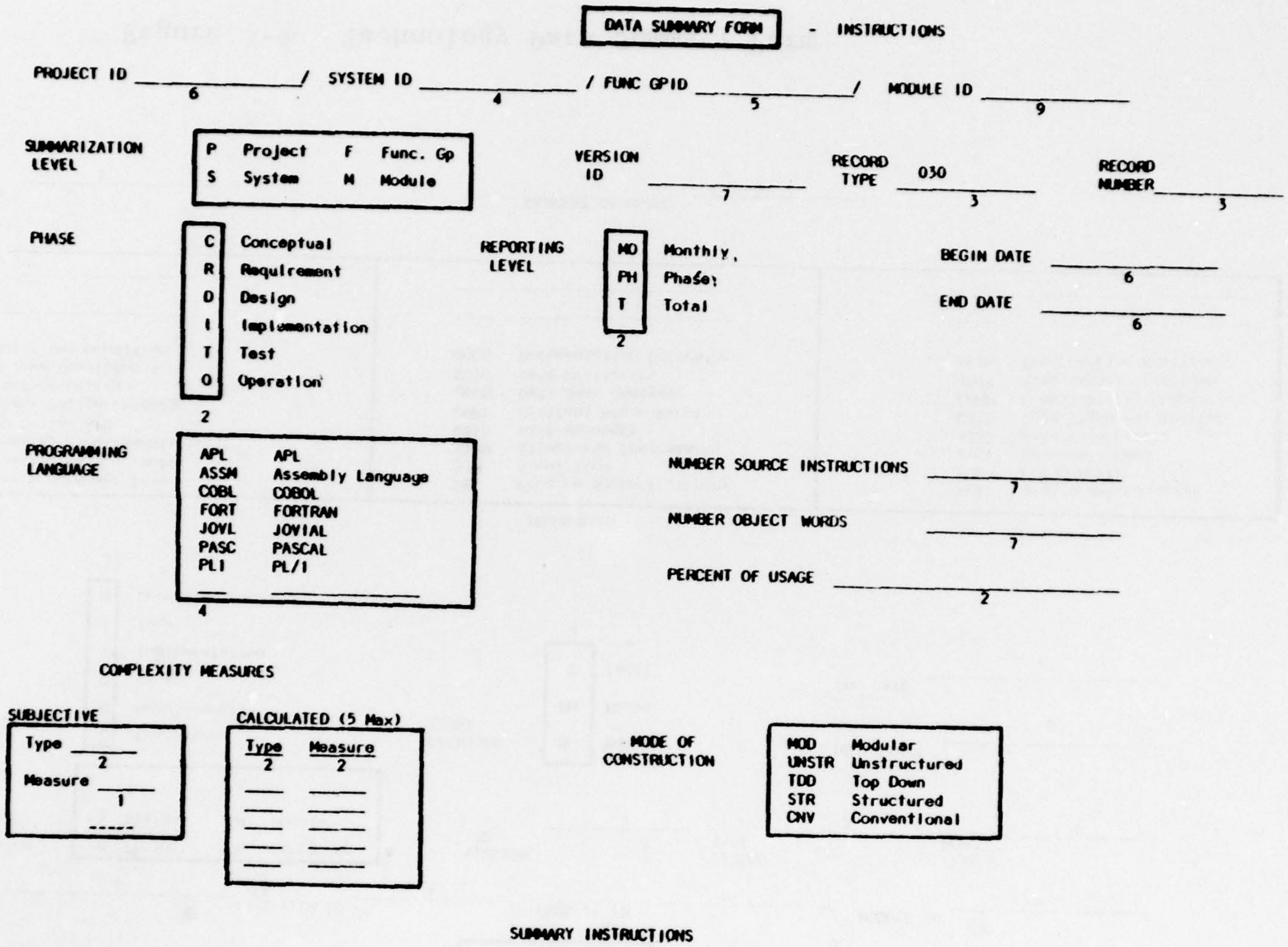
TECHNOLOGY

CPT	Chief Programmer Team	PSL	Program Support Library	HOL	High Order Language
AUDT	Automated Design Tools	SIM	Simulation	PCOM	Pre-compiler
AURT	Automated Requirements Tools	STPR	Structured Programming	RUSE	Re-usable Code
HIPO	HIPO Design Aid	WALK	Walk-throughs	ASSP	Assertion Proofs
PDL	Process Design Language	CRPF	Critical Piece First	CSTA	Code Standards Auditor
STCH	Structure Charts	DBRN	Data Base Analyzer	COMP	Compatability Checker
TDD	Top-down Development	DDIC	Data Dictionary	INTT	Independent Test Team
MODE	Modular Decomposition	DOCG	Documentation Generator	PFAN	Program Flow Analyzer
___	______	___	______	___	______
___	______	___	______	___	______

TOOL NAME ______ 5

PERCENT OF USAGE ______ 2

Figure 3-4. Technology Data Summary Form



DATA SUMMARY FORM - INSTRUCTIONS

PROJECT ID ______ / SYSTEM ID ______ / FUNC GPID ______ / MODULE ID ______
6 4 5 9

SUMMARIZATION LEVEL

P	Project	F	Func. Gp
S	System	M	Module

VERSION ID ______ 7

RECORD TYPE 030 ______ 3

RECORD NUMBER ______ 3

PHASE

C	Conceptual
R	Requirement
D	Design
I	Implementation
T	Test
O	Operation

2

REPORTING LEVEL

MO	Monthly,
PH	Phase;
T	Total

2

BEGIN DATE ______ 6

END DATE ______ 6

PROGRAMMING LANGUAGE

APL	APL
ASSM	Assembly Language
COBL	COBOL
FORT	FORTRAN
JOVL	JOVIAL
PASC	PASCAL
PLI	PL/I
___	______

4

NUMBER SOURCE INSTRUCTIONS ______ 7

NUMBER OBJECT WORDS ______ 7

PERCENT OF USAGE ______ 2

COMPLEXITY MEASURES

SUBJECTIVE

Type ______ 2

Measure ______ 1

CALCULATED (5 Max)

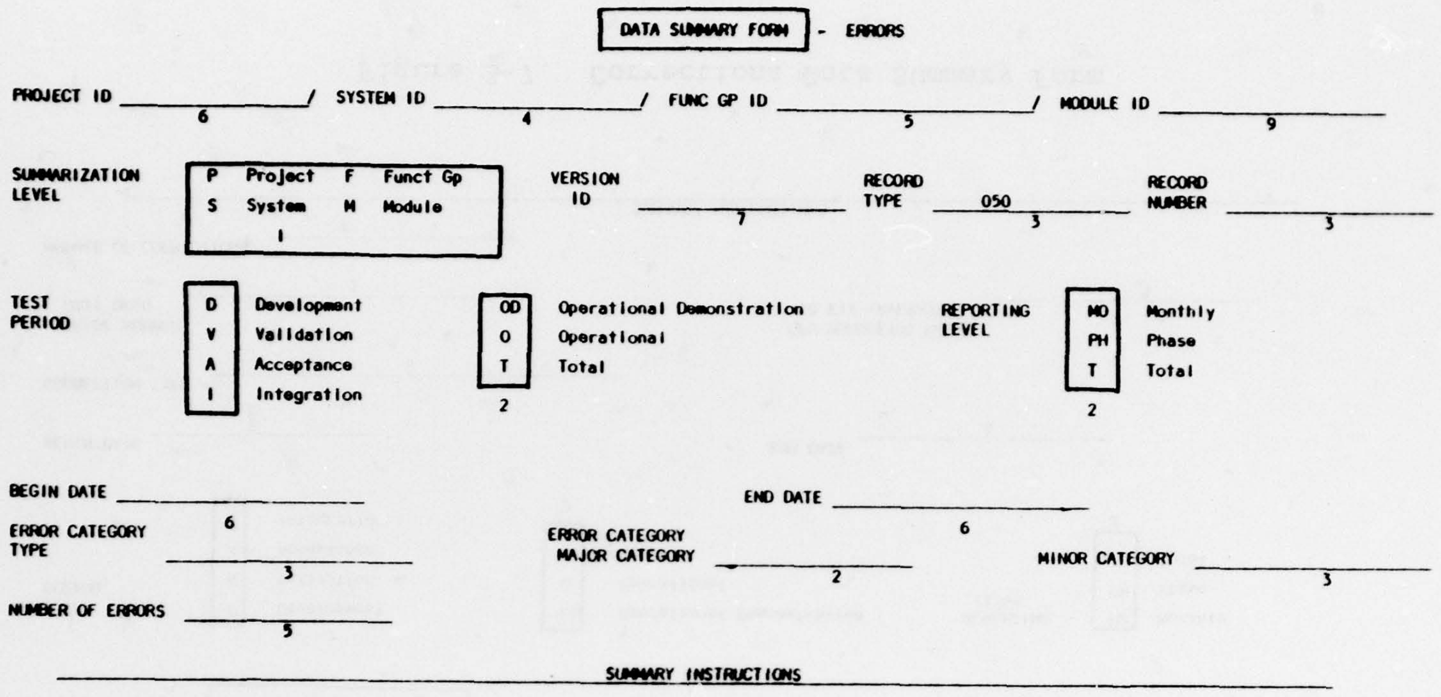
Type 2	Measure 2
___	___
___	___
___	___
___	___
___	___

MODE OF CONSTRUCTION

MOD	Modular
UNSTR	Unstructured
TDD	Top Down
STR	Structured
CNV	Conventional

SUMMARY INSTRUCTIONS

Figure 3-5. Instructions Data Summary Form



DATA SUMMARY FORM - ERRORS

PROJECT ID ______ / SYSTEM ID ______ / FUNC GP ID ______ / MODULE ID ______
6 4 5 9

SUMMARIZATION LEVEL
P Project F Funct Gp
S System M Module
1

VERSION ID ______
7

RECORD TYPE 050
3

RECORD NUMBER ______
3

TEST PERIOD
D Development
V Validation
A Acceptance
I Integration

OD Operational Demonstration
O Operational
T Total
2

REPORTING LEVEL
MO Monthly
PH Phase
T Total
2

BEGIN DATE ______
6

END DATE ______
6

ERROR CATEGORY TYPE ______
3

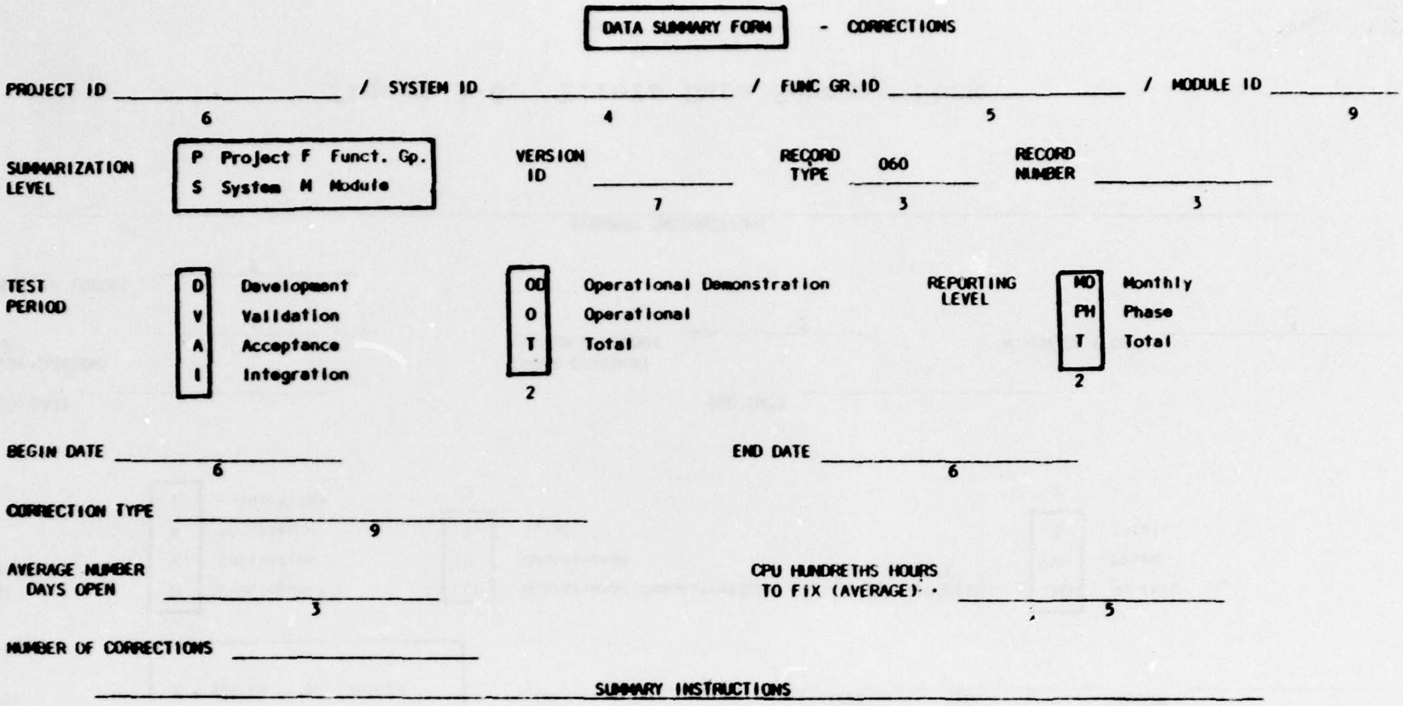
ERROR CATEGORY MAJOR CATEGORY ______
2

MINOR CATEGORY ______
3

NUMBER OF ERRORS ______
5

SUMMARY INSTRUCTIONS

Figure 3-6. Errors Data Summary Form



DATA SUMMARY FORM - CORRECTIONS

PROJECT ID ____________ / SYSTEM ID ____________ / FUNC GR.ID ____________ / MODULE ID ____________
6 4 5 9

SUMMARIZATION LEVEL

P	Project	F	Funct. Gp.
S	System	M	Module

VERSION ID ____________
7

RECORD TYPE 060 ____________
3

RECORD NUMBER ____________
3

TEST PERIOD

D	Development
V	Validation
A	Acceptance
I	Integration

OD	Operational Demonstration
O	Operational
T	Total

2

REPORTING LEVEL

MO	Monthly
PH	Phase
T	Total

2

BEGIN DATE ____________
6

END DATE ____________
6

CORRECTION TYPE ____________
9

AVERAGE NUMBER DAYS OPEN ____________
3

CPU HUNDRETHS HOURS TO FIX (AVERAGE) ____________
5

NUMBER OF CORRECTIONS ____________

SUMMARY INSTRUCTIONS

Figure 3-7. Corrections Data Summary Form

DATA SUMMARY FORM - COMPONENT - MODULE SUMMARY

PROJECT ID 6	SYSTEM ID 4	FUNC GR ID 5	MODULE ID 9	SUMMARIZE LEVEL 1	VERSION ID 7	RECORD TYPE 3	REC # 3	NAME 12

Figure 3-8. Component-Module Data Summary Form

Instructions (see Figure 3-5). The phase, reporting level and the applicable dates; the programming language used; the number of source instructions, object words, and percent of usage; complexity type and measure; and the mode of construction.

Errors (see Figure 3-6). The test period, reporting level and the applicable dates; the error category type, the error category, and the number of errors.

Corrections (see Figure 3-7). The test period, reporting level and the applicable dates; the correction type, the average number of days open, and the number of errors.

Component-Module (see Figure 3-8). This form is used to establish the key in a concise manner for any of the record types.

3.1.3 RADC Productivity Database. The summary database compiled by Richard Nelson of RADC was defined using MDQS and was called the RADC Productivity Database (reference 16). This database contains information on software projects including project and company name, attributes of the programming language and documentation, a productivity measure, an SPR rate, and the type of construction used.

This database contains information on approximately 400 software development projects encompassing 21 million lines of code. Below is a listing of the number of project descriptions that contain specific data.

Number Projects	Data Type
370	productivity
30	error
365	language usage
200	implementation technologies

3.2 Data Requirements

The purpose of this section of the report is to provide a summary description of software reliability models and an explanation of the kinds of data that are required to use these models. This information was compiled to determine what kinds of data were needed for modelling purposes and the type of coverage provided by the data in the BSDS.

The Baseline Data Requirements List (Figure 3-1) contains the majority of the model requirements (except for reference 5 and the personnel availability as in reference 8). The stress type and measure should be expanded to include explicitly whether the stress involves CPU time and/or a measure of the quality of the tests. The constituent and complexity measure types should be defined to provide a set of measurements applicable across projects and modules.

The majority of the Data Acquisition Projects provide the date of error detection and the date of correction. The one exception is the Project 4 data where only the date of correction is reported. The only dataset that included CPU time is the Project 2 dataset and none of the datasets include any information on each test performed. However, the data from Project 6 does include information on each test.

Hecht (2) differentiates between measuring, estimating, and predicting software reliability. Measurement implies that the software operates over a period of time and segments of operation are scored as failure or success. A measurement reliability numeric is normally calculated during acceptance testing before the software is turned over to the user to determine if a reliability requirement has been met. This reliability numeric can also be used to determine if the software is deteriorating over the life of the product and to determine the effect on reliability of different development and testing tools and techniques.

Estimation is taking sample reliability measurements in order to approximate when testing will be completed and to determine if a reliability goal can be met. The estimation reliability numeric must take into account any differences from the operational environment including test data selection and reliability growth.

Prediction is a reliability statment not based on a measurement of the operation of the software but on the actual or anticipated attributes of the software such as the number of lines of code. Prediction is used for project management purposes to estimate test and correction effort needed, to forecast operational downtime, and to guide software design to meet reliability requirements.

The data requirements for measuring and estimating are very similar, but the data needed for prediction varies because of the difference in the nature of the assumptions. For measuring and estimating, it is assumed that the system is operating, and the data reflects the operational characteristics of the system. With prediction, only the static characteristics are considered and data can be acquired or determined before the program is operational.

The Hecht reports (2,3,4) present the essential concepts in the numerical evaluation of software reliability and a simple mathematical relations (models) that have been found useful in the field.

The measurement models assume that the tests or runs (trials) performed are those that are meaningful for the actual operational environment. The most simplistic measurement model provides a reliabilty numeric for a batch software system or a real-time system dealing with discrete operations using the ratio of successful trials to the total number of trials. This numeric can be normalized to program length to account for differences in exposure to failure between programs.

For real-time systems dealing with continuous data streams, a practical reliability numeric is mean-time-between-failures expressed as total running time (t), divided by the number of failures (F) in the interval 0 to t. A normalizing factor for this case is the number of instructions executed per unit-time.

For software reliability estimation, if the software is being tested in the operational environment and the test cases are representative of inputs for the operational environment, then the reliability indices calculated during measurement can be used as unbiased estimators to estimate future reliability taking into account reliability growth as applied to operating time and error removal.

In the case where test data contains more severe requirements then actual usage, he discusses using the techniques of partitioning the input data sets and calculating the probability of failure ascribed to the selection of input data.

Littlewood (5,6) discusses a model for estimating the reliability of a software system based upon the attributes of each component (or subprogram) and the interactions. The inputs needed for this model include a transition probability matrix which gives the probability of each subprogram given that it will switch to another program, and the failure rate for each subprogram. The model is also extended to include cost of failure by inputing the mean and variance failure costs for each subprogram. He states that the failure rates and cost parameters for each subprogram can be estimated from test data, that the transition probability matrix in a large system would be sparse, and that the mean-time spent in each subprogram should be able to be estimated.

Littlewood (6) discusses the need to examine the special requirements of software and that many of the software reliability measures rely too much on hardware analogies. He specifically argues that we should be concerned with operational reliability and not with how many faults are in the program. He defines operational reliability as the reliability of the program as it performs (failure rate, distribution of time to next failure, etc).

The results of a project to develop software reliability prediction models using regression analysis methods are presented by Motley and Brooks (7). The authors concluded that the predictability of programming error measurements varies from very low to very high and the variability is related to the functional differences of the modules, the differences in the programming language used, the length of time formal failure data collection was carried out, the amount of thoroughness of testing, inadequacy of the linear model to provide perfect predictability, and other programmer, project, and management factors affecting the software development process. They recommend the establishment of a baseline set of predictor variables initially starting with their five and ten predictor summaries. These five and ten predictor variables are dependent upon the project or functional grouping. This baseline list should then be expanded to reflect the results of further studies.

Their results indicated that the length of the program and the number of program interfaces per 100 lines of source code were found to be the best single predictors and that program complexity variables contributed significantly to predictability.

Musa (8,9) postulates a software reliability model based on execution or CPU time, and a concomitant model of the testing and debugging process that permits execution time to be related to calendar time. The main input consists of a set of execution time intervals between failures experienced in testing, along with the number of days from the start of testing on which the failures occurred. Auxilliary inputs consist of 23 parameters including dates, computer time, and man-hours required per correction, personnel and computer availability, and mean-time-to-failure (MTTF) objective.

The output consists of measurement numeric of the present MTTF, and estimate of MTTF objective attained, remaining number of faults to be uncoverd and corrected to achieve the MTTF objective, and an estimate of the remaining execution time and calendar time required to meet the objective.

Thayer (10) and Thayer with Lipow (11) discuss what they have termed a phenomenological approach to software reliability prediction. Phenomenological is used in the sense of relating to measureable software characteristics that experience has shown are well correlated with reliability. They have used both standard and nonstandard linear regression analysis techniques applied to numbers of software problems as a linear function of defined software reliability characteristics.

They have hypothesized that the best single predictor for the number of problems is the number of branches and is a slightly superior predictor than the number of statements. They also state that the number of application program interfaces, number of computational statements, and number of data-handling statements are also good predictors but that the number of branches and the number of data-handling statements are highly correleated and should not be used together. These predictors cover the period for the number of software problems during formal demonstration.

Their data also showed that, for each software function, the number of preoperational problems is a fairly good predictor of the number of operational failures and that the number of design problem reports is a good predictor of the number of problems encountered in testing. Additional analysis shows that operational failures for each software function are reasonably well correlated with the number of design problem reports for that function.

An examination of some of the more widely used software reliability models is presented in reference 12. This paper addresses analytic models that predict the number of indigenous errors remaining in the program, the mean-time to the next failure, the time required to discover all remaining errors, and the standard deviation associated with the predictions. Although the authors of this paper state that all error prediction models are deficient in the accuracy of the model predictions, the insights gained from studying the problem have provided guidelines for developing and testing the software.

Sukert (13,14) reports on a study to analyze the results of several software reliability models against failure data obtained during formal testing of several large DoD and NASA software development projects. No consistent patterns emerged in this study. Results varied depending upon data content and application type. He recommended that more detailed analysis is needed with additional datasets and that better ways are needed to statistically determine the accuracy of model predictions.

Goel and Okumoto (15) have developed a stochastic model for software failure phenomena based on the case where errors are not corrected with certainty. The following quantities of interest are derived in this report: distribution of time to a completely debugged system, distribution of time to a specified number of remaining errors, distribution of number of remaining errors, expected number of errors detected by time (t), and the distribution of time between software failure.

The required data for models described in (reference 12) include the date the error was detected. From this information the calendar time between failures, the number of failures per reporting period and cumulative number of errors detected at a certain date can be computed.

The required data for models described in reference 2 and 2 include the date the failure occured and the date corrected. In addition to the above mentioned data, cumulative number of corrections made and the number of errors detected and corrected per time-interval can be determined.

The data required for models described in reference 2, and 5 - 9 include the amount of execution (CPU) time expended before an error was detected. From this information CPU time-to-failure, total running time-to-date, CPU time-between-failures, and CPU time-per-interval can be computed. This CPU time can be reported either for cumulative time before an error occurs or for each trial, whether it be a success or a failure. Reference 2 includes discussions on models that required a recording on the date of each test (trial) and the result.

The prediction models discussed in references 7, 10 and 11 also require module length, number of interfaces, number of branches, number of computational statements, number of data handling statements, and other statement type counts (these attributes have been termed constituent types). Also required are complexity measures and the number of design problems encountered.

Additional information that provide more meaning to the results include dates for testing phases, error descriptions including type and severity, operating mode and processing rate of the computer, stress type and measure (i.e., a measure to indicate how well the test(s) correlate to the operational environment and/or a measure of the amount or percentage of code exercised), module attributes, resources spent in correcting the error, and the phase in which the error was introduced. The model discussed in references 8 and 9 requires additional information including personnel and computer availability and the project mean-time-to-failure objective.

The semi-markov model discussed in reference 5 is actually a system level reliability estimation model in that an estimate is made dependent upon the failure rate for each subprogram, a probability matrix for subprogram "switching", and a measure of how much time would be spent in each subprogram.

3.3 Data Requirements' References

1. Lloyd, D.K., Lipow, M. Reliability Management, Methods, and Mathematics, Second Edition, Redondo Beach, CA: Published by the Authors, 1977.

2. Hecht, H. (The Aerospace Corporation, E. Segundo, CA). Measurement, Estimation and Prediction of Software Reliability. Hampton, VA: NASA Langley Research Center, NASA-CR-145135, January 1977.

3. Hecht, H., Sturm, W.A., Trattner, S. Reliability Measurement During Software Development. El Segundo, CA: The Aerospace Corporation, NASA-CR-145205, 1977.

4. Hecht, H., Sturm, W.A., Trattner, S. "Reliabiltiy Measurement During Software Development", A Collection of Technical Papers. AIAA/NASA/IEEE/ACM Computers in Aerospace Conference, November 1977, pp. 404-412.

5. Littlewood, B. "A Semi-Markov Model for Software Reliability with Failure Costs", Proceedings of the Symposium on Computer Software Engineering. New York: Polytechnic Press, 1976, pp. 281-300.

6. Littlewood, B. "How to Measure Software Reliability, and How Not to....", Third International Conference on Software Engineering. New York: The Institute of Electrical and Electronics Engineers, Incorporated, and the Association for Computing Machinery, May 1978, pp. 37-45.

7. Motley, R. W., Brooks, W.D. (IBM Corporation, Arlington, VA). Statistical Prediction of Programming Errors. Griffiss AFB, NY: Rome Air Development Center, RADC-TR-77-175, May 1977 **A041106.**

8. Hamilton, P.A., Musa, J.D. "Measuring Reliability of Computer Center Software", Third International Conference on Software Engineering. New York: The Institute of Electrical and Electronics Engineers, Incorporated, and the Association for Computing Machinery, May 1978, pp. 29-36.

9. Musa, J.D. "The Use of Software Reliability Measures in Project Management", Second International Computer Software and Applications Conference. New York: The Institute of Electrical and Electronics Engineers, Incorporated, November 1978, pp. 493-498.

10. Thayer, T.A. (TRW Defense and Space Systems Group, Redondo Beach, CA). Software Reliability Study. Griffiss AFB, NY: Rome Air Development Center, RADC-TR-76-238, August 1976 **A030798.**

11. Lipow, M., Thayer, T.A. "Prediction of Software Failures", Proceedings 1977 Annual Reliability and Maintainability Symposium. Washington, D.C.: Reliability and Maintainability Symposium, January 1977, pp. 489-494.

12. Schick, G.J., Wolverton, R.W. "An Analysis of Competing Software Reliability Models", IEEE Transactions on Software Engineering, Volume SE-4, Number 2. New York: The Institute of Electrical and Electronics Engineers, Incorporated March 1978, pp. 104-120.

13. Sukert, A.N. "An Investigation of Software Reliability Models", Proceedings 1977 Annual Reliability and Maintainability Symposium. Washington, D.C.: Reliability and Maintainability Symposium, January 1977, pp. 478-484.

14. Sukert, A.N. "A Four-Project Empirical Study of Software Error Prediction Models", Second International Computer Software and Applications Conference. New York: The Institute of Electrical and Electronics Engineers, Incorporated, November 1978, pp. 577-582.

15. Goel, A.L., Okumoto, K. (Syracuse University, Syracuse, NY). Bayesian Software Prediction Models, An Imperfect Debugging Model for Reliability and Other Quantitative Measures of Software Systems. Griffiss AFB, NY: Rome Air Development Center, RADC-TR-78-155, July 1978.
VOL I, A057870; VOL II, A057871; VOL III, A057872; VOL IV, A057873.

REFERENCES

1. Duvall, L. M., Software Data Repository Study. Griffiss Air Force Base, NY: Rome Air Development Center, RADC-TR-76-387, December 1976 A050636,

2. Honeywell, Management Data Query System/IV Administrator's Guide, DD94, Rev. 0, March 1975.

3. Honeywell, Management Data Query System/IV User's Guide, DD92, REv. 0, March 1975.

4. Honeywell, Management Data Query System/IV Administrator's Guide, DD94, Rev. 1, August 1976.

5. Honeywell, Management Data Query System/IV User's Guide, DD92, Rev. 1, August 1976.

6. Curtis, C., Duvall, L., Baseline S/W Data System User's Guide. Griffiss Air Force Base, NY: Rome Air Development Center, October 1978.

7. Thayer, T., et. al., Software Reliability Study. Griffiss Air Force Base, NY: Rome Air Development Center, RADC-TR-76-238, August 1976 A030798.

8. Fries, M. J., Software Error Data Acquisition. Griffiss Air Force Base, NY: Rome Air Development Center, RADC-TR-77-130, April 1977 A039916.

9. Willman, H. E., et. al., Software Systems Reliability: A Raytheon Project History. Griffiss Air Force Base, NY: Rome Air Development Center, RADC-TR-77-188, June 1977 A040992.

10. Rey, P., et. al., Software Systems Development: A CSDL Project History. Griffiss Air Force Base, NY: Rome Air Development Center, RADC-TR-77-213, June 1977 A042186.

11. Baker, W. F., Software Data Collection and Analysis: A Real-Time System Project History. Griffiss Air Force Base, NY: Rome Air Development Center, RADC-TR-77-192, June 1977 A041644.

12. Hecht, H., et. al., Reliability Measurement During Software Development. NASA-CR-145205, September 1977.

13. Honeywell, I-D-S/1 Programmer's Guide, DC52, Rev. 0 and Addendums A and B, April 1976.

14. Honeywell, Indexed Sequential Processor, DD38, Rev. 0 and Addendum A, January 1975.

15. Honeywell, I-D-S/1 User's Guide, DC53, Rev. 0 and Addendums A and B, April 1976.

16. Nelson, R., Software Data Collection and Analysis. Griffiss Air Force Base, NY: Rome Air Development Center, September 1978.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.