RADC-TR-77-192
Final Technical Report
June 1977

SOFTWARE DATA COLLECTION AND ANALYSIS: A REAL-TIME SYSTEM PROJECT HISTORY

IBM Corporation

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.

This report has been reviewed and is approved for publication.

APPROVED: Alan N. Sukert

ALAN N. SUKERT, Captain, USAF
Project Engineer

APPROVED: Robert D. Krutz

ROBERT D. KRUTZ, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

Do not return this copy. Retain or destroy.

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

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-77-192	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER: AD A041-644	
4. TITLE *(and Subtitle)*: SOFTWARE DATA COLLECTION AND ANALYSIS: A REAL-TIME SYSTEM PROJECT HISTORY	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report 30 Apr 76 - 29 Jan 77
	6. PERFORMING ORG. REPORT NUMBER: N/A
7. AUTHOR(s): W. F. Baker	8. CONTRACT OR GRANT NUMBER(s): F30602-76-C-0161
9. PERFORMING ORGANIZATION NAME AND ADDRESS: IBM/Federal Systems Division, Whippany Road, Whippany NJ 07981	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F, 55811414
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISIS), Griffiss AFB NY 13441	12. REPORT DATE: June 1977
	13. NUMBER OF PAGES: 39
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: Same	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: Captain Alan N. Sukert (ISIS)

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

Data Acquisition collection
Error Data Acquisition
Error Data
Error Data Analysis
Error Analysis
Problem Report Analysis

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

This report discusses the procedures used for, and the results obtained from, an analysis of software error problem reports. The problem reports studied were generated during the development of a large, real-time, highly sophisticated multi-processor data processing system. A brief profile of the development of this system is presented along with discussions of the procedures used in the analysis of the problem reports and the objectives to be met. The results of the analysis are discussed, and statistics reflecting the results

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

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

are presented. Finally, some of the problems encountered during the course of the study are presented, as well as some pertinent observations.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

EVALUATION

The requirement for producing reliable, low cost, quality software, as expressed in such documents as the Findings and Recommendations of the Joint Logistics Commanders Software Reliability Work Group (Nov 1975) and restated in various conferences and symposium sponsored by the Department of Defense and industry, has resulted in the development of new tools and techniques, such as software reliability and error prediction models, and in investigations into the types and causes of software errors, in order to find ways of insuring that all future software produced is reliable. However, much of the research in model development and in software error analysis has been severely hampered by the lack of sufficient software error data from a variety of different software projects, so that statistically valid conclusions can be drawn and model predictions validated.

This effort was initiated in response to the need for software error data, and fits into the goals of RADC TPO No. 5, Software Cost Reduction (formally RADC TPO No. 11, Software Sciences Technology), in particular the area of Software Quality (Software Data). The report focuses on results from the collection, categorizing, and analysis of over 6000 software errors extracted from the test and integration phase of a large DoD real-time, ground-based development project. The importance of obtaining

this data is that it can be used to directly support current software error prediction model development, and can also be analyzed to discover any discernible patterns in the types and categories of errors as functions of different software characteristics. In addition, the results of analysis on this data can be compared with results of similar analysis on software data from both real-time and non-real-time projects, in order to further understand how software errors are introduced and how they can be eliminated or controlled. Finally, this data will be used, along with software error data extracted from other real-time ground-based DoD software development projects, as a means of establishing a baseline for real-time ground-based software projects in terms of the types and number of errors, which eventually will lead to better methods for controlling future real-time ground-based software projects.

Alan N. Sukert

ALAN N. SUKERT, Captain, USAF
Project Engineer

Section 1

INTRODUCTION

1.1 SCOPE

This document is the final technical report under RADC Contract No. F30602-76-C-0161, Software Data Acquisition (SDA). This nine-month study focused on more than 6700 software problem reports for the period from 1 March 1974 through 1 March 1975, which was the Test and Integration (T&I) phase of the software development. Each problem was analyzed, either manually or by computer, as to (1) the type of error reported, (2) the point at which the error was introduced into the development cycle, and (3) the corrective measure taken.

The report is organized into five sections and an appendix. Section 1 discusses the scope and objectives of the study.

Section 2 presents background information about the project that produced the data studied. The discussion centers primarily around a description of the software, its development, the computer systems used in development, and types of data used in the study.

Section 3 describes the data analyzed, results from analysis of the data, the procedures employed in the analysis process, a discussion of the rationale involved in the interpretation of the supplied error categories, and a summary of the new error categories defined by the study.

Section 4 is a limited statistical analysis of the data acquired.

Section 5 presents major conclusions and recommendations: specifically, pertinent observations, the nature of problems encountered during the study, and an evaluation of the data used and acquired.

Appendix A is a detailed list of the SDA error categories.

1.2 OBJECTIVES

The objectives of the SDA study were to:

1. Extract software error data from a large, ground-based, real-time data processing system.

2. Establish a software error data base in support of research in software reliability modeling.

3. Determine from the software error data acquired, and using the error classifications supplied, the types of errors experienced during the development of the software.

4. Determine in which phase of the software development cycle each error was introduced into the system, and identify the type of correction applied to each error.

Section 2

BACKGROUND

2.1 PROJECT INFORMATION

The data utilized for analysis in this study was generated during a ballistic missile project designed primarily to respond to attacks or the threat of attacks of Intercontinental Ballistic Missiles (ICBMs). The development of a large, real-time multiprocessor data processing system brought about some unique situations requiring the development of new and sophisticated algorithms and testing programs, and the extensive use of simulation. The entire software development effort was directed toward meeting the specific needs of a real-time, high-throughput, reliable computing system.

2.2 SOFTWARE DESCRIPTION

Some of the applications of the data processing system consisted of radar surveillance, tracking, target classification, radar management and testing, intersite communication, and command and control display functions. Because the nature of the system demanded high availability, the development of a maintenance system featuring rapid recovery and quick fault isolation and repair was required. The size and complexity of the system compounded the software development problems, imposing the need for a system exerciser to verify as much of the system as practicable.

2.3 SOFTWARE DEVELOPMENT PROCEDURES

A major requirement during development of the software was a test bed that accurately reproduced the software environment, and a system of support functions designed to operate on general-purpose computers.

2.3.1 Requirements Generation

A systems engineering organization defined, established, negotiated, documented, and rigidly controlled system requirements. Changes to the requirements were made as a result of detailed software design by the development organizations, system test program data, system evaluation efforts, and detailed review by the customer.

2.3.2 Design

The design phase consisted of two efforts, process design and program design. Process design was the definition of the system requirements translated into software architecture, global data structures, tasks, task priorities, and task timing requirements. A task was defined as a single unit or program.

The process design activity was complemented by program design, which involved defining internal data bases and developing algorithms and control structures for individual tasks. This combined activity led to a detailed software specification, including specific mathematical equations. The design was dedicated to support early development of a system to which greater capability could be added gradually. Emphasis was placed on modular design to ease system growth.

Size and execution time for individual programs were two major parameters that were controlled and tracked on a monthly basis. Design reviews were held frequently and proved very effective in planning for controlled and systematic changes and refinements to the system.

2.3.3 Coding and Unit Testing

During this phase of software development, the code was written and compiled using an IBM System/360 Model 65 computer. Programs were written in CENTRAN, an extensible intermediate-level language resembling a subset of Programming Language 1 (PL/1). It provided many of the advantages of high-level languages, but could be interspersed with assembly language and system macros when necessary.

To facilitate preparation and testing, a linkage editor, simulator, and disk library system were developed. Unit testing utilized the simulator and drivers and was run on the IBM System/370.

2.3.4 Process and Functional Testing

Tasks were blocked into processes and tested by process integration teams using larger drivers and system exercisers. As testing progressed, processes were, in turn, blocked into functions for more complex system testing.

2.3.5 System Integration

When testing achieved a predefined level of capability, the software was run on the full complement of hardware using the system exerciser.

2.3.6 Evaluation

Evaluation played an important role throughout the entire development cycle. Evaluation was primarily an analytical activity which, because of the complexity of the system, relied heavily on simulation. Also, because there was a practical limit to the level of detail in which the various weapons system functions could be modeled, more detailed simulations of the particularly critical functions were added. By employing simulations in concert, considerable insight was gained into detailed system operation. A feedback mechanism in the form of problem reports re-

sulted in frequent changes and refinements to the software, and a constant updating of the evaluation simulation provided for a more accurate representation of the tactical operation.

2.4 COMPUTER SYSTEMS USED

2.4.1 Central Logic and Control (CLC) System

The Central Logic and Control (CLC) System represented the application of the multiprocessing concept to a large-scale computing system. A modular design was employed in which as many as ten processors and two Input/Output Controllers (IOCs) shared as many as 32 memory racks. Under software control the CLC could be configured to two separate partitions of arbitrary size, each capable of operating as an independent computing system, and complete reconfiguration could be accomplished in less than one second. Application software executed on the larger partition, and the exercise drivers and support activities executed on the smaller.

A single processor can throughput about 1.5 million instructions per second by means of instruction overlap and high-speed arithmetic algorithms. Since processors do not communicate directly with peripherals, processing and input/output on the CLC occurred simultaneously.

2.4.2 IBM System 370/165

The System 370/165 is an information processing system designed for very high-speed, large-scale scientific and business applications. The basic Central Processing Unit (CPU) cycle time is .08 microseconds, with a storage cycle time of 2 microseconds. Approximately 1.4 million instructions, on the average, can be processed in one second. Contributing significantly to the speed and power are the main storage capacities, which range from 512K to 3072K, and a high-speed buffer storage that sharply reduces the time required to fetch the currently used sections of main storage. Speed is further increased through the use of multiple storage elements. Reliability and availability are enhanced through the use of instruction retry and main storage error checking and correction.

2.5 TYPE AND EXTENT OF DATA AVAILABLE

The data utilized for this study was extracted from a data base of more than 17,000 problem reports. In accordance with paragraph 4.1.1 of the Statement of Work, only those reports written between 1 March 1974 and 1 March 1975 (a total approximating 6700 reports) were used in the error data analysis. These problem reports, which included hardware problems, came from various areas of the overall project.

2.5.1 Tactical Software Errors

Problem reports in this category were written against the three tactical processes plus the system exercisers and the global data sets. There were approximately 4320 problem reports in this area.

2.5.2 Support Software Errors

Problem reports in this area included all except those written against (1) tactical software items, (2) hardware items, (3) reports written to identify suggested and implemented improvements, and (4) those reports classified, after analysis, not to be errors. There were approximately 1000 problem reports in this category.

2.5.3 Hardware Errors

Problem reports were written against all facets of the hardware, from burned-out lightbulbs to sophisticated electronic design errors. There were 246 hardware reports generated during the Test and Integration phase.

2.5.4 Improvements

Approximately 190 problem reports were written to identify areas of improvement. Some of these improvements were implemented, but the majority were deferred to later periods when time and funding would be available.

2.5.5 Non-Errors

This group of problem reports accounted for a significant number (960) of the reports analyzed and can be divided into three categories. The largest number (709 reports) consisted of duplicates of other reports. The remaining 251 problem reports were considered legitimate non-errors in the sense that the situations described in the reports were not in error with the requirements; or the problem was one that existed only in the simulation environment and a correction was provided simply as an "accommodation" type of correction and subsequently removed when testing took place on the full complement of hardware. (These 251 problem reports involved only those identified during the manual analysis effort; many more had already been eliminated during the automatic analysis period.)

Section 3

DATA ACQUISITION

3.1 DATA DESCRIPTION

3.1.1 Source Data

The data base records of the problem reports consisted of 242 fields, of which only 20 were used in the identification and analysis of the problem reports. Certain fields were used to identify those problem reports that were to be used for the study; other fields were used in the automatic and manual analyses of the problem reports to determine data such as date of correction, type of correction, phase, type of error, etc.

Figure 1 is an example of the printed data base record listing those fields that were pertinent to the SDA study. Explanatory notes on the page following Figure 1 describe each applicable column heading shown in the figure.

The Product Identifier (PIDENT), or program name, incorporates a number of unique features. Figure 2 is a representative example of a PIDENT breakdown; Table 1 lists and describes the alphabetic characters used. The PIDENT type of program naming convention facilitates the identification of the area and function to which the program belongs.

3.1.2 SDA Data

The data acquired for this study was of two types: data related to software errors and data related to the software development process.

Error-Related Data: The data gathered for this portion of the study dealt with software errors and related statistical information. Software errors were controlled and tracked by using an identifying number called a Master Problem Report (MPR) number, and associated with a module by way of a PIDENT name. The date the error was discovered and the date it was corrected were maintained as part of the error-related data, along with descriptions of the error and its solution. The error type, the means used to correct the error, and the point in the software development cycle at which the error was made were items determined through an analysis of the source problem report and stored in the SDA master problem record.

Development Process Data: Data related to the software development process was of the following types:

1. Computer Usage - This data represents the amount of CLC CPU time utilized each month during the T&I period.

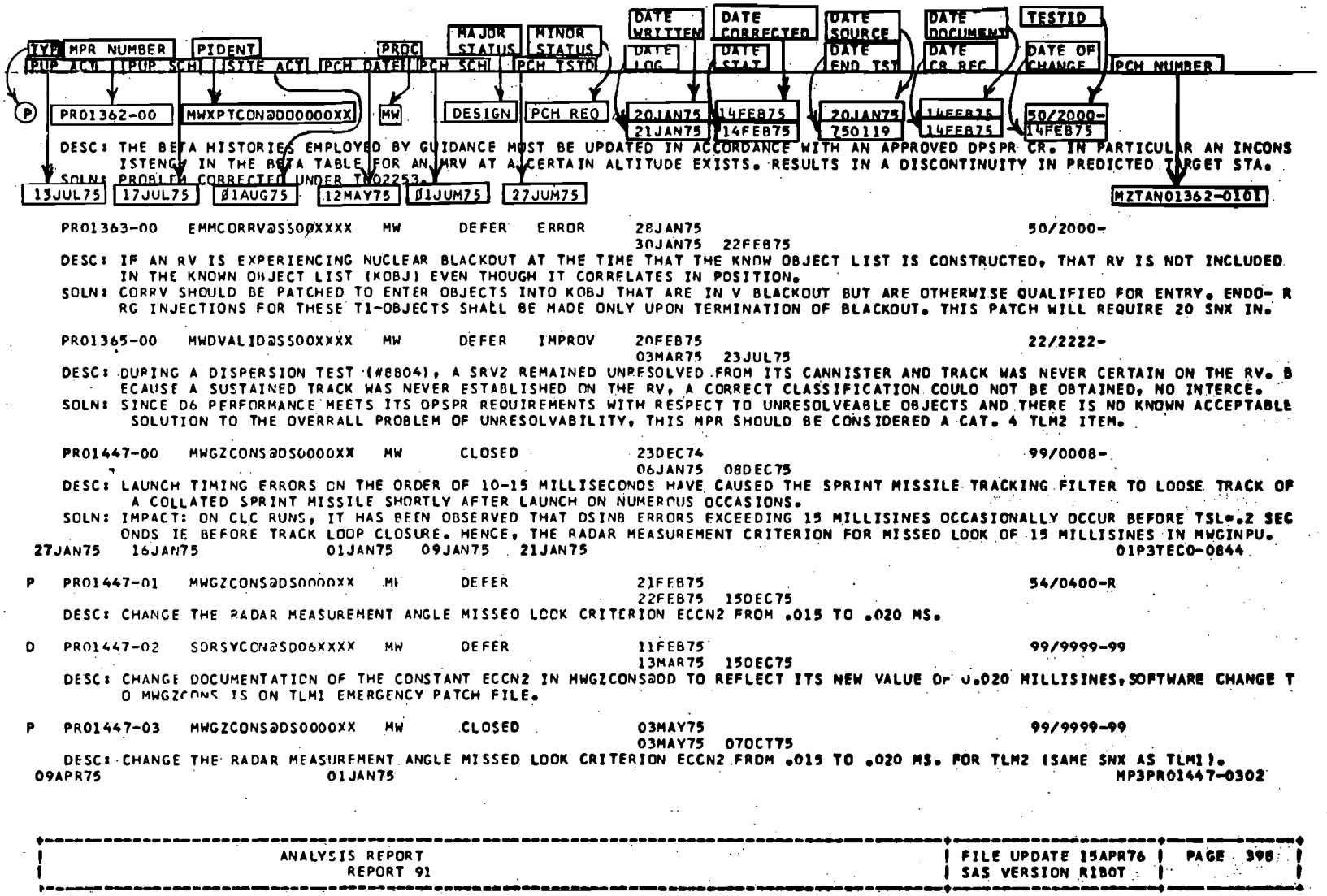
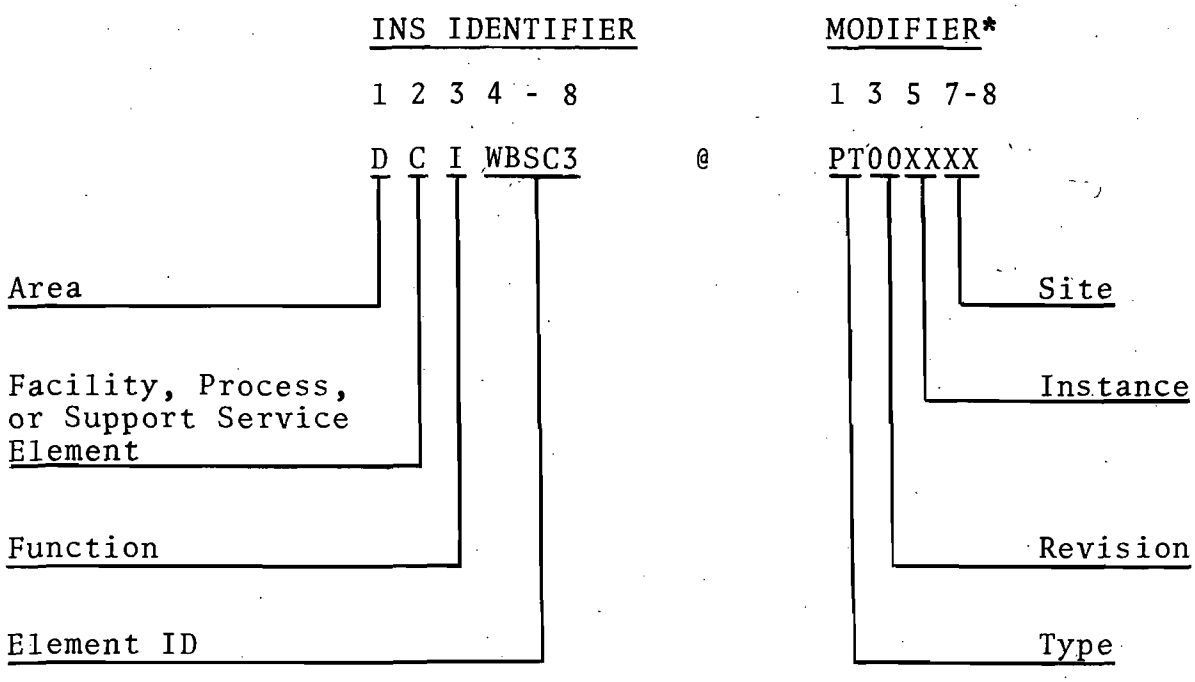


Figure 1. Source Data Base Listing

NOTES TO FIGURE 1

Column Heading	Description
TYP	Type of solution
MPR NUMBER	Problem report number
PIDENT	Program name (Product Identifier)
MAJOR STATUS	Major status code associated with the problem
MINOR STATUS	Minor status code associated with the problem
DATE WRITTEN	Date problem report was created
DATE CORRECTED	Date problem solution was submitted
DATE SOURCE	Date source code delivered from development
DATE DOCUMENT	Date document correction was delivered
TESTID	Test identification
PUP ACT	Date patch actually put on PUP tape
PUP SCH	Date patch scheduled to be put on PUP tape
SITE ACT	Date patch actually sent to site
PCH DATE	Date patch status last changed
PCH SCH	Date patch scheduled for testing
PCH TSTD	Date patch finished testing
DATE LOG	Date problem report logged into SAS system
DATE STAT	Date of last status change
DATE END TST	Date end of source testing
DATE CR REC	Date correction received by CSCM
DATE OF CHANGE	Date this SAS record was last changed
PCH NUMBER	Patch identification number
DESC	Problem description
SOLN	Problem solution description



* Modifier Designators

PT = Type: Policies, Procedures, and Standards (PPS)
00 = Revision: Original Issue
XX = Instance: Not applicable for subroutines
XX = Site: Not used in PIDENTs

Figure 2. Typical PIDENT Breakdown

TABLE 1. PIDENT FUNCTIONAL BREAKDOWN

B = BMDC
- S = Logic Simulation Facility
  - E = System Exerciser

C = CLC Installation and Support Software
- O = Operating System
  - A = M&D Buffer Programs and CLC Monitor Support
  - B = Library
  - C = Configuration Control
  - D = Debug
  - E = Error Control, Interrupt Handler
  - H = Hardware Test Scheduler, Normal Path Diagnostics
  - I = I/O Manager
  - K = Debugging Aids for Real Time
  - L = Loader
  - M = CCDSS Management, Man/Machine
  - O = OS Control
  - P = Communicators
  - R = RSS Management, Overlay Manager
  - S = Scheduler, Main Control
  - U = Utilities
  - X = (functional level designation not appropriate)
- T = Installation and Test Software Support (ITSS) Facility

D = DPS Management Control
- R = Reporting

E = System Exerciser
- G = MSR and PAR Exerciser Process Common Function
  - D = Drivers

G = Global
- X = Routines, subroutines, sources or data sets used in more than one facility or process
  - X = (functional level designation not appropriate)

L = System Test Tapes

M = Missile Site Radar (MSR) Software
- W = MSR Weapons Process

TABLE 1. PIDENT FUNCTIONAL BREAKDOWN (Continued)

- C = Process Coordinator
- D = Data Gathering
- E = Data Reduction
- F = Interceptor Response
- G = SPRINT and SPARTAN Guidance, MDP, and Launch Area Control
- L = Tactical Display Area
- M = MSR Site Manager
- R = Radar Management
- S = Target Selection
- T = Test Coordinator
- V = 360 Driver
- W = Process Design
- X = (functional level designation not appropriate)
- Y = Launch Area Test
- Z = MSR Tests

P = Perimeter Acquisition Radar (PAR) Software

- I = PAR Installation Process
  - T = Receiver Tests - 2nd Interval
- M = Independent Radar Test Monitor (RTM) and PAR Weapons (PW) PIDENTs
  - G = Global Data Sets
  - L = Local Data Sets
  - P = Process Coordinator
  - R = Class B Radar Test
- T = PAR Test Process and RTM Subprocess of PW
  - G = Global Data Sets
  - J = Test and Integration
  - L = Local Data Sets
  - P = Coordinator and Control
  - R = Class A, Class B, or Class C Radar Tests
  - X = (functional level designation not appropriate)

TABLE 1. PIDENT FUNCTIONAL BREAKDOWN (Concluded)

W = PAR Weapons Process
- C = Tactical Communicators/Intrasite
- D = Data Gathering
- G = Global Data Sets
- I = PAR Site Manager/Intersite
- K = Known Object Management
- P = Process Coordinator
- R = Radar Manager
- S = Target Selection
- T = Tracking
- X = (functional level designation not appropriate)

R = PAR Trainer Controller Program
- T = Training Task Initialization

S = Systems Engineering

T = Standard Test Software
- E = 360 Facilities Standard Test Process
- P = Tactical Operating System Cycler

2. Statement Type and Rate - This data identifies the programming language used in writing the software and the rate at which an "average" statement in an "average" statement mix was processed by the CPU.

3. Test Run Data - This data describes the number of different test scenarios used, the number of times all the different tests were run, and the percentage of tests that ran to completion.

4. PIDENT List - This data identifies all modules that were part of the software system during the T&I phase. It lists each program, the size of each in CENTRAN statements, the language in which each was written, and the mode of construction used in development.

3.2 SDA DATA ACQUISITION PROCEDURES

The computer run logs for the period from 1 March 1974 through 1 March 1975 were reviewed manually to extract the CLC CPU time data. Separate run logs had been maintained for Missile Site Radar (MSR) and Perimeter Acquisition Radar (PAR) tests by month and the data was recorded in minutes of CPU time. The data was tabulated by month for MSR and PAR and totaled for each month. Monthly totals were, thereafter, converted from minutes to hours per month, coded, keypunched, and stored on disk in file 2 of the SDA Data Tape data set.

The statement type was the same for all modules since all programs had been written in CENTRAN. The Bell System Technical Journal - Special Supplement* (1975), page S57, was used as a reference for obtaining the statement rate based on a logical statement mix. Using this information in conjunction with the graph found on the same page (S57) led to the determination of a statement rate of 25 microseconds. The information was then written up, coded, keypunched, and added to file 2 of the SDA Data Tape data set.

The test run data was acquired by reviewing a large number of progress reports from several areas covering the period of time under study. The data was tabulated by test scenario, with a column for total number of tests run and a column for number of tests run to completion. After all data was collected, the resulting statistics were calculated, written up, coded, keypunched, and added to the file 2 data set.

Several program libraries containing the desired MW, PW, and support programs were listed indicating PIDENT name, number of instructions, and language used. To these listings was added the mode of construction for each module. The data was formatted,

*The Bell System Technical Journal - Special Supplement, page S57, 1975, American Telephone and Telegraph Company.

keypunched, and loaded on file 3 of the SDA Data Tape data set. Figure 3 is a sample portion of a printed PIDENT data set listing.

During the early stages of the SDA study it became apparent that some of the data, because of its uniqueness, would lend itself to an automatic analysis procedure. For this reason the decision was made to undertake two types of analyses, one automatic and the other a manual process.

The design of the program used to identify the source problem reports written during the T&I phase also incorporated the initial building of the SDA data base and a provision for executing the automatic analysis.

The matching of automatic analysis criteria with appropriate error categories presented some problems, however. Because this activity took place at the beginning of the study, experience in matching error categories with problems had not yet been developed. Moreover, the explanations of many of the major error categories, as set forth in Annex 1 of the Statement of Work, were causing some confusion, and it was not clear that major error categories and/or subcategories could be added if the need arose. As a result there existed some questions concerning the validity of the study team's interpretation of certain error categories.

The SDA Data Base Build program incorporated within its design the task of identifying the source problem reports written during the T&I phase and extracting from them the following data used in establishing the initial SDA data base record.

Source Record		SDA Record
MPR Number	→	Master Problem Number
Date Written	→	Date of Discovery
PIDENT	→	Module in which Error Occurred
DESC	→	Problem Description
SOLN	→	Correction Description

The rest of the design involved the automatic analysis function, wherein the remainder of the SDA error data (date of correction, phase in which error occurred, type of correction, error classification) would be acquired. The criteria devised for the automatic analysis function are outlined in Tables 2 and 3.

As the final step in the automatic analysis, the program scanned the newly formed SDA data base record for blank fields. If a blank field was found, the Build program looked at the next record in the source data base. If that record had the same basic problem report number, the Build program performed an analysis of it. If that analysis supplied data on all the SDA fields,

COSRECON	001113	CENTRAN	CONVENTIONAL
COSSTART	000072	CENTRAN	CONVENTIONAL
COSSUCSR	000496	CENTRAN	CONVENTIONAL
COSTAREQ	000727	CENTRAN	CONVENTIONAL
COSTASEQ	000666	CENTRAN	CONVENTIONAL
COSTASKR	000942	CENTRAN	CONVENTIONAL
COSTVPGM	001427	CENTRAN	CONVENTIONAL
COSXAPDG	000003	CENTRAN	CONVENTIONAL
COSZAPDG	000005	CENTRAN	CONVENTIONAL
COSZCOMM	001095	CENTRAN	CONVENTIONAL
COSZDSLT	000034	CENTRAN	CONVENTIONAL
COSZDUMP	000191	CENTRAN	CONVENTIONAL
COSZMAIN	000757	CENTRAN	CONVENTIONAL
COSZMC01	000645	CENTRAN	CONVENTIONAL
COSZMC02	000643	CENTRAN	CONVENTIONAL
COSZMMAP	000242	CENTRAN	CONVENTIONAL
COSZPHSE	000106	CENTRAN	CONVENTIONAL
COSZPLDS	000217	CENTRAN	CONVENTIONAL
COSZRLST	000059	CENTRAN	CONVENTIONAL
COSZSYID	000034	CENTRAN	CONVENTIONAL
COSZUSRL	000068	CENTRAN	CONVENTIONAL
COSZXTCB	000114	CENTRAN	CONVENTIONAL
COUCHDCM	000966	CENTRAN	STRUCTURED
COUCHKCM	000821	CENTRAN	STRUCTURED
COUCMPCM	001661	CENTRAN	STRUCTURED
COUCMPXX	000424	CENTRAN	STRUCTURED
COUCTLVP	000880	CENTRAN	STRUCTURED
COUDIOXX	000941	CENTRAN	STRUCTURED
COUDLTCM	001128	CENTRAN	STRUCTURED
COUDMPDM	001727	CENTRAN	CONVENTIONAL
COUDMPXX	000370	CENTRAN	CONVENTIONAL
COUDPSDM	000519	CENTRAN	CONVENTIONAL
COUDSKVP	002009	CENTRAN	STRUCTURED
COUEDTDM	000894	CENTRAN	CONVENTIONAL
COUEPCVP	001415	CENTRAN	STRUCTURED
COUEPLCM	001148	CENTRAN	STRUCTURED
COUFPOUT	000296	CENTRAN	CONVENTIONAL
COUFWRIT	000710	CENTRAN	CONVENTIONAL
COULCPCM	001196	CENTRAN	STRUCTURED
COULDKDM	002068	CENTRAN	CONVENTIONAL
COUMESCM	000138	CENTRAN	STRUCTURED
COUNTER3	000248	CENTRAN	CONVENTIONAL
COUPCPCM	001426	CENTRAN	STRUCTURED

Figure 3. PIDENT Listing

TABLE 2. AUTOMATIC ANALYSIS CRITERIA

Criteria	Type of Correction	Phase	Error Category
10th & 11th characters of PIDENT name DN FD PD PS SD	DOCUMENTATION	DESIGN	WW020
FN PR PY SF	DOCUMENTATION	REQUIREMENTS	WW010
UM	DOCUMENTATION	REQUIREMENTS	QQ020
First 8 characters of PIDENT name and TYP = P or BLANK MWXSDC-- MWXMSIMP EMXSDC-- PWXSDC-- EPXSDC--	PATCH	REQUIREMENTS	KK010
TYP = S or C MWXSDC-- MWXISIMP EMXSDC-- PWXSDC-- EPXSDC--	SOURCE	REQUIREMENTS	KK010
TYP = D MWXSDC-- MWXMSIMP EMXSDC-- PWXSDC-- EPXSDC--	DOCUMENTATION	REQUIREMENTS	WW010
First 3 characters of PIDENT name and TYP = P or BLANK PMG PML PTG PTL PWG	PATCH	REQUIREMENTS	NN020

TYP = solution type for source problem report

TABLE 2. AUTOMATIC ANALYSIS CRITERIA (Continued)

Criteria	Type of Correction	Phase	Error Category
TYP = S or C PMG PML PTG PTL PWG	SOURCE	REQUIREMENTS	NN020
TYP = D PMG PML PTG PTL PWG	DOCUMENTATION	REQUIREMENTS	WW010
First 8 characters of PIDENT name and TYP = P or BLANK MWGZCONS MWGSCONS MWGLACDS MWDDGCON MWFIRCON MWGSITEC	PATCH	REQUIREMENTS	NN020
TYP = S or C MWGZCONS MWGSCONS MWGLACDS MWDDGCON MWFIRCON MWGSITEC	SOURCE	REQUIREMENTS	NN020
TYP = D MWGZCONS MWGSCONS MWGLACDS MWDDGCON MWFIRCON MWGSITEC	DOCUMENTATION	REQUIREMENTS	WW010
MWX-----@D and			
TYP = P or BLANK	PATCH	REQUIREMENTS	NN020
TYP = S or C	SOURCE	REQUIREMENTS	NN020
TYP = D	DOCUMENTATION	REQUIREMENTS	WW010

TYP = solution type for source problem report

TABLE 2. AUTOMATIC ANALYSIS CRITERIA (Concluded)

Criteria	Type of Correction	Phase	Error Category
Character string of PREFACE in problem description	DOCUMENTATION	CODE	QQ070
CRB Category 5 Rejected - Transient	NONE	NA	SS010
CRB Category 5 Rejected - Duplicate	NONE	NA	PP020
TYP = H or Process Code = HDW or 10th & 11th characters of PIDENT name	HARDWARE	NA	VV000
Major Status = DEFERRED or Minor Status = NOT APPROVED	NOT FIXED	(no further analysis)	
TESTID = 99/0003 or Date source not BLANK or Date end test not BLANK	SOURCE	(no further analysis)	
TYP = P or TYP BLANK	PATCH	(no further analysis)	
TYP = S or TYP = C	SOURCE	(no further analysis)	
TYP = D	DOCUMENTATION	(no further analysis)	

TYP = solution type for source problem report

TABLE 3. DATE OF CORRECTION CRITERIA*

Type of Correction	HIERARCHY OF CRITERIA SELECTION				
	1st Choice	2nd Choice	3rd Choice	4th Choice	5th Choice
PATCH	PUP ACT DATE	SITE ACT DATE	PUP SCH DATE	DATE CR REC	DATE STAT
SOURCE	DATE SOURCE	DATE END TST	DATE CR REC	DATE STAT	
DOCUMENTATION	DATE DOCUMENT	DATE CR REC	DATE STAT		
HARDWARE	DATE CR REC	DATE STAT			

* If the date of correction selected was greater than 10/1/75, a default date of 10/1/75 was used instead

that record was substituted for the previous SDA data base record. If the analysis did not supply the SDA fields, the Build program looked at the next source problem report. This procedure was followed until a source report either furnished all of the necessary data or there were no more source problem reports having the same basic problem report number. This procedure resulted in either an SDA data base record possessing all of the necessary data or a record with one or more blank fields. Those records containing blank fields were set aside for later manual analysis. Execution of the SDA Build program led to the initial generation of the SDA data base, with 2060 records directly resulting from the automatic analysis process.

At this point, during discussions with RADC personnel at the completion of the automatic analysis effort, it became clear that the interpretation uncertainties suspected earlier regarding certain error categories were real. A random check of the automatically analyzed data revealed that results were not as good as anticipated. One of the trouble areas at first, and throughout the manual analysis phase, involved the Preset Data Base and Global Variable/Compool Definition error categories. A new approach to resolving the problems in these two categories had to be devised, and several subcategories had to be added to each as well. The category requiring the most corrections to automatically analyzed data was Requirements Compliance. Initially, this category was interpreted as applying to documentation as well as to software; however, after discussions with RADC personnel, it was used only where the software changed because it did not meet requirements.

A new category, Design/Requirements Logic errors, with several subcategories, had to be defined to accommodate the errors that had originally been assigned to Requirements Compliance. Finally, many of the classifications made in the Documentation Errors category had to be changed because they pertained to design and requirements documents as opposed to other documentation.

After the SDA data base had been built and the automatic analyses run, it was applied against a program that looked for one or more blank fields in the data base record. When a record with blank fields was detected it was listed, showing which data was present and which fields had no data. Figure 4 is a sample portion of an analysis listing which, along with a listing of the source problem reports, was used for the manual analysis phase of the study.

Copies of the analysis listing, the source problem report listing, and Annex 1 of the Statement of Work were distributed, with instructions, to members of the technical staff. The purpose of the initial pass through the analysis listing was to pick out those problem reports having blank fields that were obvious and simple to fill in, and to gain experience in assigning error categories. Subsequent passes through the analysis listing involved increasingly complex analyses.

AW62409	750129	750228	MWFZNOBJ@SS00XXXX	NOT FIXED		
AW62411	750203	750320	MWSXOCNT@RS00XXXX	PATCH		
AW62421	750207	750404	MWGSOUTP@SS00XXXX	NOT FIXED		
AW62422	750210	750411	MWGINPUT@SS00XXXX	NOT FIXED		
AW62432	750224	750320	MWGPTCHP@SS00XXXX	NOT FIXED		
AW62439	750227	750627		NOT FIXED		
AX58024	750117	751001		NOT FIXED		
AX58729	740807	751001		NONE	NA	PP020
BE70043	750116	750312		NONE	NA	SS000
BE70192	750324	750627		NONE	NA	SS000
BL70019	750109	750627		NONE	NA	SS000
BL70021	750110	750411		NONE	NA	SS000
BL70028	750113	750502		NONE	NA	PP020
BL70052	750117	750627		NONE	NA	SS000
BL70058	750120	750502		NONE	NA	SS000
BL70060	750121	750723		NONE	NA	PP020
BL70063	750121	750502		NONE	NA	PP020
BL70065	750121	751001	CORDSTER@R000XXXX	PATCH		
BL70068	750122	750204	COAZMADP@D00300XX	SOURCE		

Figure 4. Analysis Listing

As problems were encountered they were discussed and the conclusions were circulated as updates to either the set of instructions or to the set of error categories. One of the first problems examined was the problem report having more than one solution. If the solutions were all of the same type, the first record encountered was used. If the solutions were of varied types, the order of priority was: requirements documentation, design documentation, source code, and patches.

The SDA data base was updated each week using the previous week's manual analysis findings.

Upon completion of the manual analysis, a clean-up and review of the SDA data was initiated. The clean-up effort consisted of scanning a copy of the SDA data base for obvious keypunching errors that were not spotted during the updates, and obvious erroneous assignments such as might occur if the phase and type of correction were transposed, for example. The review involved the listing of selected major error categories that had offered particular difficulty during the manual analysis phase, and scanning the error category, phase, and type of correction assignments for consistency.

A final update to the SDA data base was made following the completion of the clean-up and review activities.

With the exception of the Computational and Logic Error categories, the descriptions of the other categories did not seem to be sufficient for the beginner. After reviewing categories with the customer and gaining actual experience in assigning error categories, however, a better understanding of how to apply categories to the problem reports naturally developed. Within a short time it was discovered that additional major and minor categories were needed, and, despite a reluctance to generate new major error categories, it became necessary to do so in two instances: Hardware errors and Design/Requirements Logic errors.

Although the definition of new minor error categories was not as significant in its impact, caution was exercised to hold the number to those few considered essential. Careful attempts were always made to fit problems into the categories already established. Only when a reasonable fit was lacking were new minor categories defined.

One major category that caused little difficulty was User-Requested Changes, for it fit well into the problem report status structure of Deferred Improvement. For this reason, almost without exception, all problem reports with a status of Deferred Improvement were assigned to one of the User-Requested Changes subcategories and assigned a phase of NA (Not Applicable). The phase assignment of NA was used because these were not errors in the generally applied sense, and thus were not introduced into the system. In the

typical case the solution for the problem report was not implemented, resulting in a type of correction assignment of Not Fixed. Occasionally a software change was made requiring the type of correction assignment to fit the type of change.

The definition of several of the major error categories occupied considerable time and attention and it is worthwhile to describe these cases. The two major categories that proved most difficult were Preset Data Base and Global Variable/Compool Definition errors. The main problem was adjusting to a new concept of the two types of data sets because of the manner in which they were used on the project that produced the source data for the study. For purposes of the study, a Global Data Set was defined to be: one used by more than one routine and/or subroutine, and whose data may be defined at requirements or design time, and defined and/or modified during execution time. A Preset Data Set was defined as: one used by only one routine or subroutine, and whose data may be defined and/or initialized by some external source prior to its utilization by the host routine. The data in the data set could be either fixed or variable. The key to differentiating between these two categories appeared to be how the data set was used; by one routine or by several routines.

An initial misunderstanding regarding the Requirements Compliance category led to its use for all documentation errors to requirements and design documents. As a consequent approach to correcting the situation, requirements and design document errors were separated from other documentation errors and a new error category (Design/Requirements Logic errors) was established.

Table 4 lists the new error categories generated during the SDA study.

TABLE 4. SUMMARY OF NEW ERROR CATEGORIES

Category	Definition
FF040	System/system incompatibility
GG110	Routine fails to maintain integrity of interface data
KK020	VS timeout on fetch or store
KK030	Macro definition error
KK040	Delete unneeded macro definition
MM070	Delete unneeded definitions
MM080	Length of definition incorrect
NN060	Add new variables
QQ130	Comments error (this error category was originally listed as NN040)
RR030	Delivered capability in error
SS010	Transient error
SS020	Error the analyst cannot identify in order to categorize
SS030	Error that was fixed, but the designer did not know why the fix worked
TT060	Erroneous input entry
UU040	Noting the existence of numerous non-critical errors
VV000	Hardware Errors
WW000	Design/Requirements Logic Errors
WW010	Requirements documentation errors
WW020	Design documentation errors
WW030	Typographical/editorial error/cosmetic change to design or requirements documentation

Section 4

STATISTICAL ANALYSIS OF ERROR DATA

The information provided in this section is not intended as an in-depth analysis of the error data collected. Its purpose is merely to show some basic relationships that might be used as points of departure for further study and analysis.

4.1 FREQUENCY OF OCCURRENCE BY ERROR CATEGORY

A brief examination of the bar chart illustrated in Figure 5 readily indicates that the Recurrent Errors category (PP) represents a significant percentage of errors. However, when the category is broken down into its component subcategories it can be seen that 86% of the Recurrent Errors are the result of duplicate problem reports. The cross-hatched section of the bar represents the actual recurring errors.

Another category that is somewhat misleading is Documentation Errors (QQ), wherein 85% of the errors pertain to program prefaces. In almost all cases the prefaces were written, but they were not in the format prescribed by the project's Policies, Procedures, and Standards (PPS) manual. The cross-hatched area on the bar represents the remainder of the documentation errors -- with the exception of errors to Design/Requirements Logic, which were assigned a separate error category (WW).

4.2 FREQUENCY OF OCCURRENCE BY PHASE

The NA (Not Applicable) phase is somewhat all-purpose in that it reflects those errors for which there can be no phase, such as the categories User-Requested Changes (LL), Operator Errors (TT), Questions (UU), and the subcategory Duplicate Problem Report (PP020); and those errors for which the phase cannot be determined due to insufficient information, Unidentified Errors (SS).

The large number of errors appearing in the NA phase (see Figure 6) is significant, prompting the observation that many of the problem descriptions furnished in the problem reports were deficient in the description of the problem incurred or were inadequate in the quality of the description.

4.3 NUMBER OF ERRORS BY MONTH

The hypothesis under which the statistics* shown in Figure 7 were collected proposed that, as the T&I phase progressed from begin-

---

* Statistics compiled do not reflect hardware errors

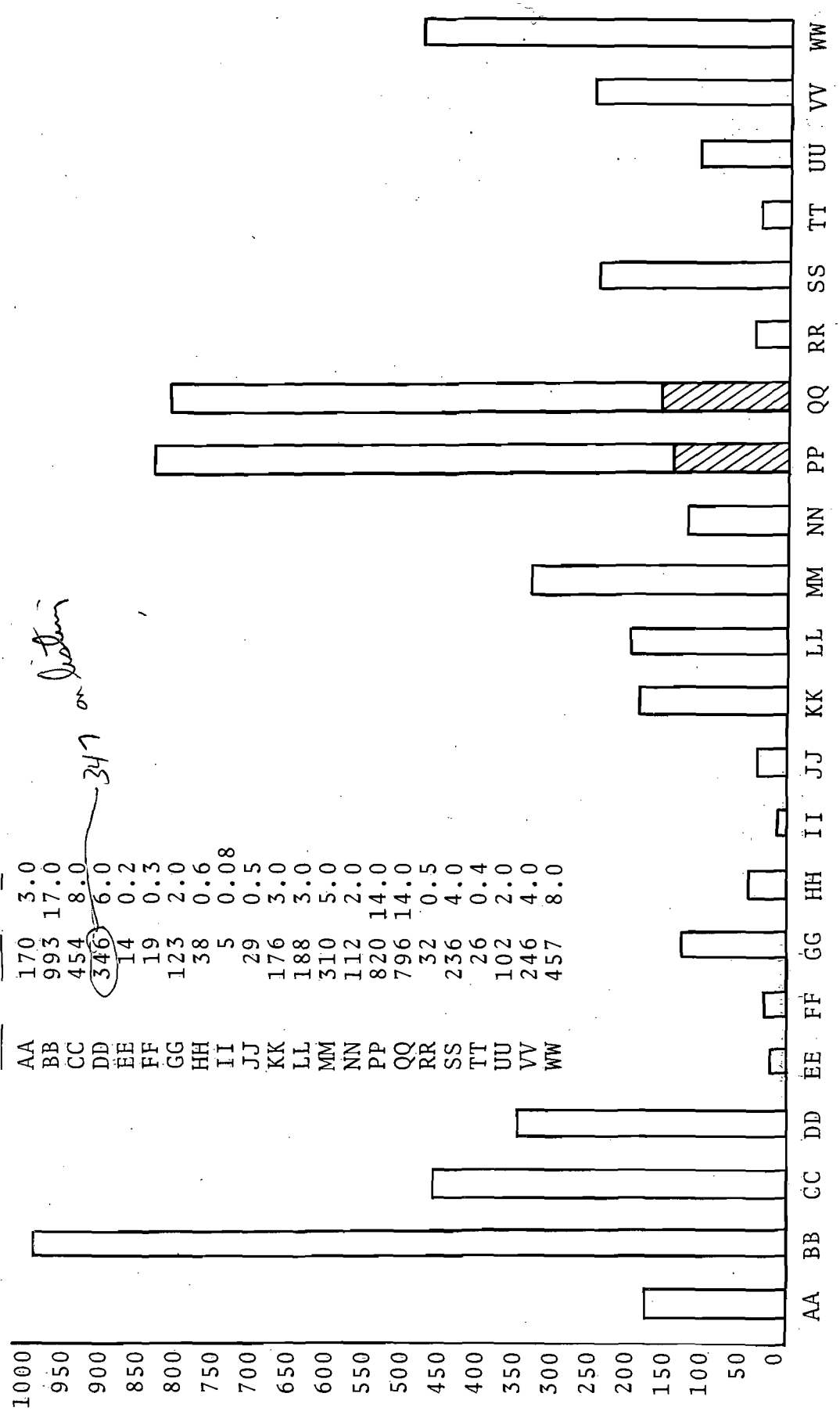


Figure 5. Frequency of Occurrence by Error Category

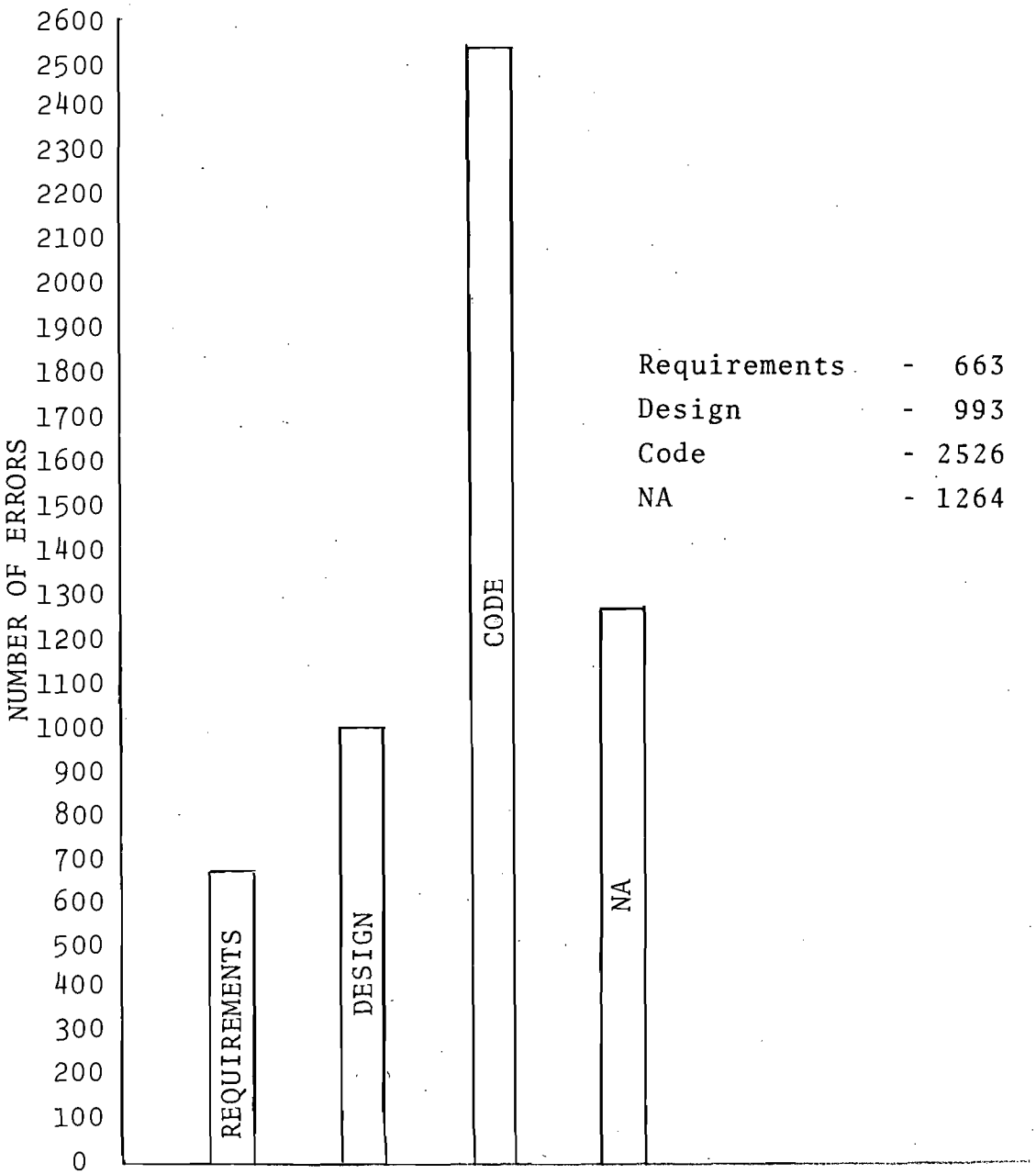


Figure 6. Frequency of Occurrence by Phase

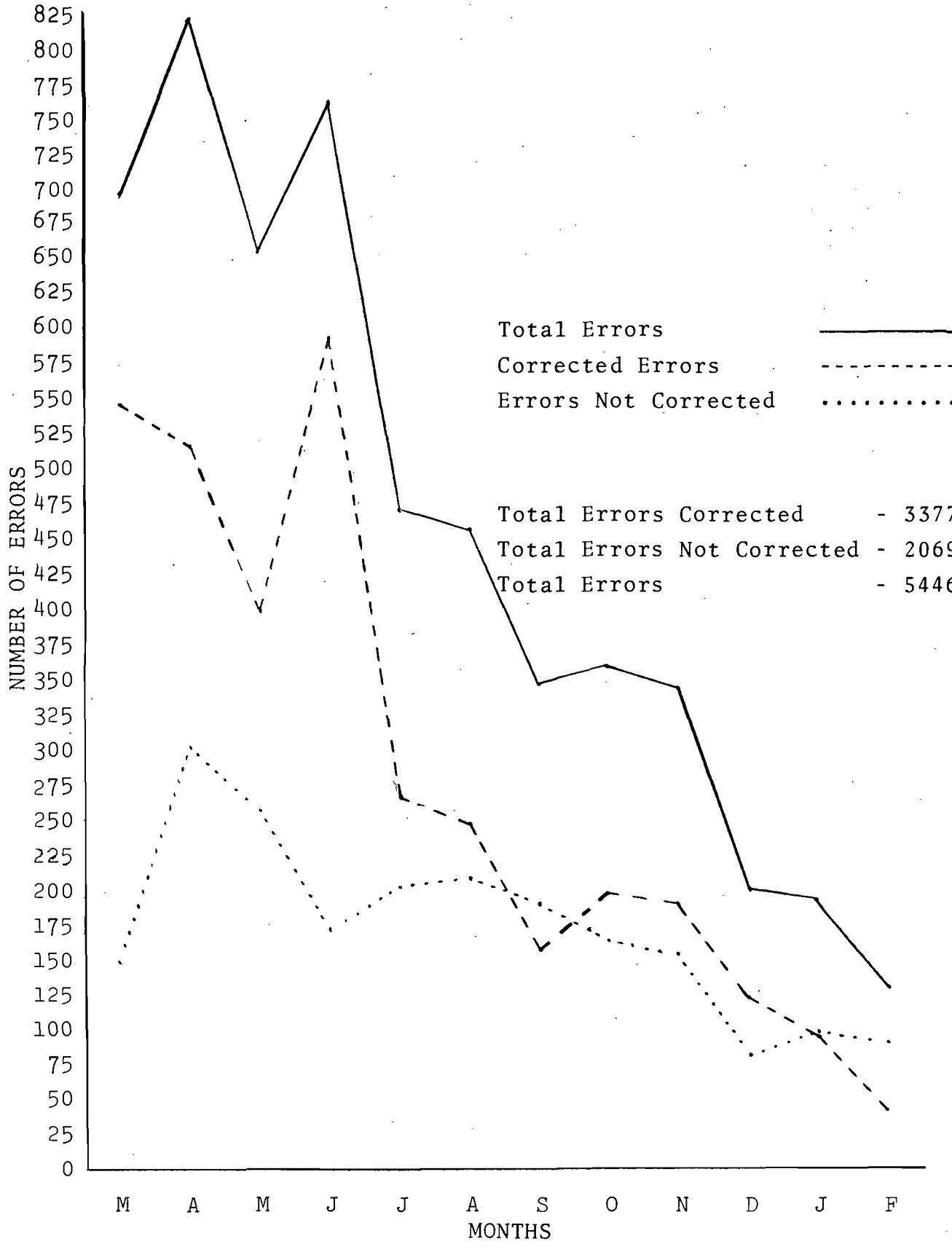


Figure 7. Number of Errors by Month

ning to end, errors would more likely be corrected in the earlier months and, unless essential, would tend to be rejected or deferred (not corrected) toward the finish of the T&I phase. As it turned out, this was not the case. However, it is apparent that even though fewer problem reports (errors) were being written toward the end of the T&I phase, and fewer errors were being corrected, the number of errors not corrected dropped off only slightly. A greater percentage of errors was not being corrected in the latter half of the period, tending to support the hypothesis, but the expected crossing of the two curves (corrected and not corrected errors) did not occur.

Another interesting observation is that in the total figures for corrected errors and errors not corrected, an almost 60/40 relationship existed. Extending the relationship reveals that approximately half of the problem reports written were never corrected.

Section 5

CONCLUSIONS

5.1 PROBLEMS ENCOUNTERED

The major problem encountered during the SDA study was in acquiring a valid and workable understanding of the meaning or interpretation of the error categories. It was important that this information be well understood at the outset to enable the correct coordination of automatic analysis criteria with the appropriate error categories. Not investigating the meanings of the error categories in more depth at the beginning of the study period diminished the effectiveness and efficiency of the automatic analysis process. For similar studies in the future, it is recommended that several days early during the contract period be devoted to study and clarification of the error categories. It is also recommended that any accompanying documentation containing the definitions and error categories be revised and expanded with more detailed definitions and possible examples. The proper assignment of error categories is a key to the study of error reliability modeling.

Another difficulty involved problem reports that often identified problems of a general nature that necessitated corrections to more than one module, and frequently corrections of more than one type. It was not uncommon for problem solutions to result in a correction to both the program and the documentation. When this occurred, all parts of the solution were identified under one problem report number with different suffixes. The task was to pick out a single error and type of correction, along with the other information that represented the problem.

In the automatic analysis process, either the first record of a series or the first record having a complete set of the needed data was chosen. For manual analysis it was felt that errors in documents were of a higher priority than those in programs, and the solution selected to represent the error was, in order: requirements document, design document, source code change, patch change. If the solutions were all of the same correction type, however, the first record of the listing was used.

Similar problems existed where there were multiple solutions to a problem report and it was evident from the different suffix records that there was more than one error involved; or the report identified more than one error and provided more than one solution. Under the problem report number system employed to document errors, only one error per problem report could be identified; thus, in the two instances given above, a choice had to be made as to which

error to document. The priority order was the same as stated previously for the manual analysis procedure: requirements document, design document, source code, patches.

To prevent this type of situation from becoming a problem, the problem reporting system should allow only one error to be identified per problem report, or the data collection and recording methods should be modified so that each error is uniquely identified.

5.2 EVALUATION OF ACQUIRED DATA

Some of the data collected during the SDA study was not as valid as had been hoped. The major portion was collected after the fact, and much of the source information needed had already been disposed of or had never been present. To compensate, certain assumptions and substitutions had to be made. An example of this is the way in which the date of correction was determined (see Table 3). In many cases the type of correction was also deduced through a series of assumptions. For example, if no type of correction -- such as a patch number or a statement in the comments section to indicate a source or document change -- was apparent, the existence of patch dates in the record had to be checked. If patch dates were carried in the record it was assumed the correction was a patch; if no patch dates were present, the next step was to search for source or documentation dates. If none of these types of dates were available, the search would move to the status category of the problem report. When the status was Review, Deferred or Rejected, the assumption was that the error had not been corrected; on the other hand, it was assumed that for any other status of the problem report, the type of correction was a patch.

The assignment of error categories was not as consistent or as accurate as it should have been, for several reasons. The problem and solution descriptions in some of the problem reports were either non-existent or so lacking in detail as to be essentially meaningless. In those cases that defied reasonable assumption, the error category assignment necessarily became Unidentified. This is one important area in which configuration management control could have been exercised to require adequate descriptions.

Other factors that would have contributed to the study became apparent during the course of the contract. For example, it became clear that the time to start collecting data in support of error analysis and error reliability modeling studies is shortly after the commencement of the project from which the data is to be collected. Also, built into the problem reporting and change management control systems of the host project should be those tools necessary to collect and store, on a timely basis, all the data that would be needed to meet the requirements of an error analysis study. Along with this should exist a configuration management control and monitoring system to ensure the accuracy and completeness of the data collected.

5.3 OBSERVATIONS

The SDA study could be the first of a possible series of similar studies possessing unique, built-in characteristics. A number of projects already in progress (PACS, PAVE PAWS), about to begin (SPACETRACK), or projected for the future (Cobra Judy, Pacific Barrier, Space-Based Radar) share an unusual research environment, four aspects of which are of particular interest: common applications skills (Ballistic Missile Defense skills), commonality of code, common systems test architecture, and experienced personnel.

By maintaining a cadre of experienced personnel, the skills and experience acquired on previous similar projects can be applied readily to other projects or to new projects as they come into existence. This aspect of the environment has already been experienced as personnel from the project on which this study is based have moved on to the PACS, PAVE PAWS, and, most recently, SPACETRACK projects.

The PACS and SPACETRACK projects utilize the PAR Weapons software portion of the established data base used for this study. Approximately 37% of the PACS code was changed. Although there is no transferability of code to PAVE PAWS from the data base on which this study was based, the techniques and application of technology are very similar.

In the cases of the PACS and SPACETRACK projects, the presence of a proven test bed and an experienced test team should have a significant positive impact on the time necessary for software installation and acceptance. This facet of the environment should also have considerable influence on the quality and reliability of the delivered product.

By assigning to projects, such as those cited above, experienced personnel who possess the desired applications skills, the training period required for future familiarization would be minimized. Benefits would also accrue in terms of reliability improvement.

The environmental aspects mentioned in these latter paragraphs represent a unique opportunity in reliability and error prediction modeling. The software data base used for the SDA study, plus the PACS and PAVE PAWS projects, can provide valuable data for use in the development of models, whereas the SPACETRACK project can provide an opportunity to test the prediction reliability of those models already developed. The SPACETRACK project could also provide additional data for further model development.

Appendix A

SDA ERROR CATEGORIES

Category ID	Category
AA000	COMPUTATIONAL ERRORS
AA010	Total number of entries computed incorrectly
AA020	Physical or logical entry number computed incorrectly
AA030	Index computation error
AA040	Wrong equation or convention used
AA041	Mathematical modeling problem
AA050	Results of arithmetic calculation inaccurate/not as expected
AA060	Mixed mode arithmetic error
AA070	Time calculation error
AA071	Time conversion error
AA072	Time truncation/rounding error
AA080	Sign convention error
AA090	Units conversion error
AA100	Vector calculation error
AA110	Calculation fails to converge
AA120	Quantization/truncation error
BB000	LOGIC ERRORS
BB010	Limit determination error
BB020	Wrong logic branch taken
BB030	Loop exited on wrong cycle
BB040	Incomplete processing
BB050	Endless loop during routine operation
BB060	Missing logic or condition test
BB061	Index not checked
BB062	Flag or specific data value not tested
BB070	Incorrect logic
BB080	Sequence of activities wrong
BB090	Filtering error
BB100	Status check/propagation error
BB110	Iteration step size incorrectly determined
BB120	Logical code produced wrong results
BB130	Logic on wrong routine
BB140	Physical characteristics of problem to be solved were overlooked or misunderstood
BB150	Logic needlessly complex
BB160	Inefficient logic
BB170	Excessive logic
BB180	Storage reference error (software problem)

Category ID	Category
CC000	INPUT/OUTPUT ERRORS
CC010	Missing output
CC020	Output missing data entries (PH = code)
CC030	Error message not output
CC040	Error message garbled
CC050	Output or error message not compatible with design documentation (including garbled output) (PH = code)
CC060	Misleading or inaccurate error message text (PH = design)
CC070	Output format error (including wrong location)
CC080	Duplicate or excessive output
CC090	Output field size inadequate
CC100	Debug output problem (relative to design documentation)
CC101	Lack of debug output
CC102	Too much debug
CC110	Header output problem
CC120	Output tape format error
CC130	Output card format error
CC140	Error in printer control
CC150	Line count/page eject error
CC160	Needed output not provided in design
CC161	Insufficient output options
DD000	DATA HANDLING ERRORS
DD010	Valid input data improperly set/used
DD020	Data written in or read from wrong disk location
DD030	Data lost/not stored
DD040	Data, index, or flag not set or set/initialized incorrectly
DD041	Number of entries set incorrectly
DD050	Data, index, or flag modified or updated incorrectly
DD051	Number of entries updated incorrectly
DD060	Extraneous entries generated (table, array, etc.)
DD070	Bit manipulation error
DD071	Error using bit modifier
DD080	Floating point/integer conversion error
DD090	Internal variable error (definition or set/use)
DD100	Data packing/unpacking error
DD110	Routine looking for data in non-existent record
DD120	Bounds violation
DD130	Data chaining error
DD140	Data overflow or overflow processing error
DD150	Read error
DD151	All available data not read

(DATA HANDLING ERRORS continued on following page)

Category ID	Category
	DATA HANDLING ERRORS (Continued)
DD160	Long literal processing error
DD170	Sort error
DD180	Overlay error
DD190	Subscripting convention error
DD200	Double buffering error
EE000	OPERATING SYSTEM/SYSTEM SUPPORT SOFTWARE ERRORS
EE010	Language produces erroneous machine code
EE020	OS missing needed capability
FF000	CONFIGURATION ERRORS
FF010	Compilation error
FF011	Segmentation problem
FF020	Illegal instruction
FF030	Unexplainable program halt
FF040	System/system incompatibility
GG000	ROUTINE/ROUTINE INTERFACE ERRORS
GG010	Routine passing incorrect amount of data (insufficient or too much)
GG020	Routine passing wrong parameters or units
GG030	Routine expecting wrong parameters
GG040	Routine fails to use available data
GG050	Routine sensitive to input data order
GG060	Calling sequence or routine/routine initialization error
GG070	Routines communicating through wrong data block
GG080	Routine used outside design limitation
GG090	Routine won't load (routine incompatibility)
GG100	Routine overflows core when loaded
GG110	Routine fails to maintain integrity of interface data
HH000	ROUTINE/SYSTEM SOFTWARE INTERFACE ERRORS
HH010	OS interface error (calling sequence or initialization)
HH020	Routine uses existing system support software incorrectly
HH030	Routine uses sense/jump switch improperly

Category ID	Category	
II000	TAPE PROCESSING INTERFACE ERRORS	
II010	Tape unit equipment check not made	
II020	Routine fails to read continuation tape	
II030	Routine fails to unload tape after completion	
II040	Erroneous input tape format	
JJ000	USER INTERFACE ERRORS	
JJ010	Operations request or data card/routine incompatibility	
JJ020	Multiple physical card/logical card processing error	
JJ030	Input data interpreted incorrectly by routine	
JJ040	Valid input data rejected or not used by routine	
JJ050	Input data rejected but used	
JJ060	Input data read but not used	
JJ070	Illegal input data accepted and processed	
JJ080	Legal input data processed incorrectly	
JJ090	Poor design in operator interface	
JJ100	Inadequate interrupt and restart capability	
KK000	DATA BASE INTERFACE ERRORS	
KK010	Routine/data base incompatibility	
KK011	Uncoordinated use of data elements by more than one user	
KK020	VS timeout on fetch or store	
KK030	Macro definition error	
KK040	Delete unneeded macro definition	
LL000	USER-REQUESTED CHANGES	
LL010	Simplified interface and/or convenience	(PH = NA)
LL020	New and/or enhanced functions	(PH = NA)
LL021	CPU	(PH = NA)
LL022	Disk	(PH = NA)
LL023	Tape	(PH = NA)
LL024	Input/Output	(PH = NA)
LL025	Core	(PH = NA)
LL030	Security	(PH = NA)
LL040	New hardware/OS capability	(PH = NA)
LL050	Instrumentation	(PH = NA)
LL060	Capacity	(PH = NA)
LL070	Data base management and integrity	(PH = NA)
LL080	External program interface	(PH = NA)

Category ID	Category
MM000	PRESET DATA BASE ERRORS (a data set that is generated by an external source)
MM010	Data or operations request card descriptions
MM020	Error message text
MM030	Nominal default, legal, maximum/minimum values
MM040	Physical constants and modeling parameters
MM041	Ephemeris parameters (short-lived parameters or interval parameters)
MM050	Dictionary (bit string) parameters
MM060	Missing data base settings
MM070	Delete unneeded definitions
MM080	Length of definition incorrect
NN000	GLOBAL VARIABLE/COMPOOL DEFINITION ERRORS
NN010	Items in wrong location (wrong data block)
NN011	Definition sequence error
NN020	Data definition error
NN021	Table definition incorrect
NN030	Length of definition incorrect
NN050	Delete unneeded definitions
NN060	Add new variables
PP000	RECURRENT ERRORS
PP010	Problem report reopened or previous fix in error (TC = none, PH = code)
PP020	Problem report a duplicate of previous report (reject duplicates) (TC = none, PH = NA)
QQ000	DOCUMENTATION ERRORS (changes to documents other than requirements and design specifications)
QQ010	Routine limitation
QQ020	Operating procedures
QQ030	Difference between flowchart and code
QQ040	Tape format
QQ050	Data card/operation request card format
QQ060	Error message
QQ070	Routine's functional description (prefaces) (TC = documentation)
QQ080	Output format
QQ090	Documentation not clear/not complete
QQ100	Test case documentation
QQ110	Operating system documentation
QQ120	Typographical/editorial error/cosmetic change
QQ130	Comments error

Category ID	Category
RR000	REQUIREMENTS COMPLIANCE ERRORS (code was changed because it did not meet the requirements)
RR010	Excessive run time
RR020	Required capability overlooked or not delivered at time of report
RR030	Delivered capability in error
SS000	UNIDENTIFIED ERRORS
SS010	Transient error (TC = none, PH = NA)
SS020	Error the analyst cannot identify in order to categorize (PH = NA)
SS030	Error that was fixed, but designer did not know why the fix worked
TT000	OPERATOR ERRORS
TT010	Test execution error
TT020	Routine compiled against wrong Compool/Master Common
TT030	Wrong data base used
TT040	Wrong master configuration used
TT050	Wrong tape(s) used
TT060	Erroneous input entry
UU000	QUESTIONS (MPR used as a vehicle to ask a question or make a statement)
UU010	Data base (PH = NA)
UU020	Master configuration (PH = NA)
UU030	Routine (PH = NA)
UU040	Noting the existence of numerous non-critical errors (PH = NA)
VV000	HARDWARE ERRORS (TC = hardware, PH = NA)
WW000	DESIGN/REQUIREMENTS LOGIC ERRORS
WW010	Requirements documentation errors (MPRs written against requirements documentation)
WW020	Design documentation errors (MPRs written against design documentation)
WW030	Typographical/editorial error/cosmetic change to design or requirements documentation

METRIC SYSTEM

BASE UNITS:

Quantity	Unit	SI Symbol	Formula
length	metre	m	...
mass	kilogram	kg	...
time	second	s	...
electric current	ampere	A	...
thermodynamic temperature	kelvin	K	...
amount of substance	mole	mol	...
luminous intensity	candela	cd	...

SUPPLEMENTARY UNITS:

plane angle	radian	rad	...
solid angle	steradian	sr	...

DERIVED UNITS:

Acceleration	metre per second squared	...	m/s
activity (of a radioactive source)	disintegration per second	...	(disintegration)/s
angular acceleration	radian per second squared	...	rad/s
angular velocity	radian per second	...	rad/s
area	square metre	...	m
density	kilogram per cubic metre	...	kg/m
electric capacitance	farad	F	A·s/V
electrical conductance	siemens	S	A/V
electric field strength	volt per metre	...	V/m
electric inductance	henry	H	V·s/A
electric potential difference	volt	V	W/A
electric resistance	ohm		V/A
electromotive force	volt	V	W/A
energy	joule	J	N·m
entropy	joule per kelvin	...	J/K
force	newton	N	kg·m/s
frequency	hertz	Hz	(cycle)/s
illuminance	lux	lx	lm/m
luminance	candela per square metre	...	cd/m
luminous flux	lumen	lm	cd·sr
magnetic field strength	ampere per metre	...	A/m
magnetic flux	weber	Wb	V·s
magnetic flux density	tesla	T	Wb/m
magnetomotive force	ampere	A	...
power	watt	W	J/s
pressure	pascal	Pa	N/m
quantity of electricity	coulomb	C	A·s
quantity of heat	joule	J	N·m
radiant intensity	watt per steradian	...	W/sr
specific heat	joule per kilogram-kelvin	...	J/kg·K
stress	pascal	Pa	N/m
thermal conductivity	watt per metre-kelvin	...	W/m·K
velocity	metre per second	...	m/s
viscosity, dynamic	pascal-second	...	Pa·s
viscosity, kinematic	square metre per second	...	m/s
voltage	volt	V	W/A
volume	cubic metre	...	m
wavenumber	reciprocal metre	...	(wave)/m
work	joule	J	N·m

SI PREFIXES:

Multiplication Factors	Prefix	SI Symbol
1 000 000 000 000 = 10^{12}	tera	T
1 000 000 000 = 10^{9}	giga	G
1 000 000 = 10^{6}	mega	M
1 000 = 10^{3}	kilo	k
100 = 10^{2}	hecto*	h
10 = 10^{1}	deka*	da
0.1 = 10^{-1}	deci*	d
0.01 = 10^{-2}	centi*	c
0.001 = 10^{-3}	milli	m
0.000 001 = 10^{-6}	micro	μ
0.000 000 001 = 10^{-9}	nano	n
0.000 000 000 001 = 10^{-12}	pico	p
0.000 000 000 000 001 = 10^{-15}	femto	f
0.000 000 000 000 000 001 = 10^{-18}	atto	a

* To be avoided where possible.

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. 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.