

AD-A119 081

IIT RESEARCH INST ROME NY

F/O 9/2

IN THE PAVE PAWS SOFTWARE MAINTENANCE ENVIRONMENT. (U)

JUN 82

F30602-80-C-0223

UNCLASSIFIED

RADC-TR-82-169

HN

10. 2
2018

11

AD A119081



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-82-169	2. GOVT ACCESSION NO. AD-A119 081	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MPP IN THE PAVE PAWS SOFTWARE MAINTENANCE ENVIRONMENT		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 17 June 80 - 17 October 81
7. AUTHOR(s) IITRI Staff		6. PERFORMING ORG. REPORT NUMBER N/A
9. PERFORMING ORGANIZATION NAME AND ADDRESS IIT Research Institute 199 Liberty Plaza Rome NY 13440		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0223
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COEE) Griffiss AFB NY 13441		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 63728F 25280105
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		12. REPORT DATE June 1982
		13. NUMBER OF PAGES 122
		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 Palaïmo (COEE)		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Modern Programming Practices Operation and Maintenance Software Data Collection Software Engineering Tools		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report presents the results of a study to establish a baseline software maintenance experience database and to determine relationships which exist on the use of Modern Programming Practices and Software Engineering Tools on the ease of software maintenance of the PAVE PAWS Phased Array Warning System.		

PREFACE

This document is the final report of Rome Air Development Center contract F30602-80-C-0223 titled "Analysis of PAVE PAWS Ops Data". This contract was performed in support of the U.S. Air Force Electronic Systems Division's (ESD) and Rome Air Development Center's (RADC) mission to provide standards and technical guidance to software acquisition managers.

This report presents the results of a study to establish a baseline software maintenance experience database and to determine relationships which exist between the use of Modern Programming Practices and Software Engineering Tools on the ease of software maintenance of the PAVE PAWS Phased Array Warning System.

To achieve this purpose the IIT Research Institute (IITRI) study team performed a literature search, conducted on-site interviews, collected anomaly and change data, established continuing data collection and analysis procedures, developed a machine readable software maintenance experience database and performed analysis of the anomaly and change data.

Accession For		
NTIS GRA&I	☒	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution/		
Availability Codes		
Distribution and/or		
Distribution		
A		



PRECEDING PAGE B

ACRONYMS

AD	- SPA Division
ADCOM	- Air Defense Command
ADP	- Programming Branch
ANP/ANL	- Programming Branch/Analysis Section
ADP/TAC	- Programming Branch/Tactical Section
ADP/RDR	- Programming Branch/Radar Control Section
ADP/SYS	- Programming Branch/Operating System Section
ADQ	- Quality Assurance Branch
CCB	- Configuration Control Board
CDF	- CPCG Description Form
CMAF	- CPCG Maintenance Activity Form
CPC	- Computer Program Component
CPCG	- Computer Program Configuration Group
CPCI	- Computer Program Configuration Item
CPT	- Chief Programmer Team
CPT & E	- Chief Programmer Test and Evaluation
CPU	- Central Processing Unit
CRB	- Configuration Review Board
DACS	- Data and Analysis Center For Software
DBMS	- Data Base Management System
DR	- Discrepancy Report
DRDB	- Discrepancy Report Data Base
DT & E	- Development Test and Evaluation
ESD	- Electronic Systems Division
HIPO	- Hierarchical Input-Process-Output
HOL	- Higher Order Language
HQ	- Headquarters
I/O	- Input/Output
INTV	- Interview Data
JCL	- Job Control Language
LOC	- Lines of Code
LOL	- Lower Order Language
MAF	- Maintenance Activity Form
MDIS	- Modification Design and Interface Specification

MDQS	- Management Data Query System
MPP	- Modern Programming Practices
MRA	- Memo of Recommended Action
MWS	- Missile Warning Squadron
NORAD	- North American Defense
O & M	- Operations and Maintenance
OT & E	- Operational Test and Evaluation
PC	- Project Control
PCD	- Program Change Document
PCF	- Project Control Form
PDDR	- Program Document Discrepancy Report
PDL	- Program Design Language
PDTR	- PAVE PAWS Data Reduction Miscellaneous Support
PEP	- Programmer Experience Profile
PMR	- Program Modification Request
PPCDF	- PAVE PAWS CPCG Description File
PPCSF	- PAVE PAWS Segment Change History File
PPDRH	- PAVE PAWS Report History File
PMAF	- PAVE PAWS CPCG Maintenance Activity File
PPOS	- PAVE PAWS Operating System
PPPCH	- PAVE PAWS CPC Change History File
PPPEP	- PAVE PAWS Programmer Experience Profile File
PPSCH	- PAVE PAWS Segment Change History File
PPSL	- PAVE PAWS PSL Software
PRCL	- PAVE PAWS Radar Control Software, Signal Processor Software Receiver/Transmitter Test Software System Analysis
PRG	- Summary by Programs
PSL	- Program Support Library
PSIM	- PAVE PAWS Simulation and Target Scenario Generation
PT & E	- Program Test and Evaluation
PTAC	- PAVE PAWS Tactical Software
RADC	- Rome Air Development Center
SAC	- Strategic Air Command
SCRB	- Site Configuration Review Board
SE	- Software Engineering
SEG	- Summary by Segments

SP - Structured Programming
SPA - System Programming Agency
SPO - System Project Office
SVR - Software Version Release
S/W - Software
6MWS - Sixth Missile Warning Squadron
7MWS - Seventh Missile Warning Squadron

TABLE OF CONTENTS

	PAGE
I INTRODUCTION	1-1
1.1 Objectives and Scope	1-1
1.2 Background	1-1
1.3 Management Summary	1-3
1.3.1 Purposes of this Study	1-3
1.3.2 Summary of the Study Approach.	1-4
2 PAVE PAWS MAINTENANCE ENVIRONMENT.	2-1
2.1 Introduction	2-1
2.2 PAVE PAWS Maintenance Organization	2-1
2.3 PAVE PAWS Maintenance Process Overview	2-4
2.4 Configuration Control Summary Description.	2-7
3 DATA COLLECTION.	3-1
3.1 Methodology.	3-1
3.2 Data Identification.	3-2
3.2.1 Questionnaire Interview Procedure.	3-3
3.3 Data Collection (Data Availability).	3-4
3.4 Data Collection Forms.	3-5
3.4.1 Introduction	3-5
3.4.2 Programmer Experience Profile (PEP) Form	3-5
3.4.3 CPCG Description Form (CDF).	3-6
3.4.4 CPCG Maintenance Activity Form (CMAF).	3-6
3.5 Data Processing and Summarization.	3-12
3.6 PAVE PAWS Maintenance Data Base.	3-15
4 STUDY RESULTS.	4-1
4.1 Introduction	4-1
4.2 Summary of PAVE PAWS Maintenance Personnel Interviews. . .	4-1
4.2.1 Top-Down Program Development	4-2
4.2.2 Chief Programmer Team/Librarian.	4-3
4.2.3 Structured Programming	4-5
4.2.4 Structured Walkthrough and Reviews	4-6
4.2.5 Independent Quality Assurance/Testing.	4-7
4.2.6 Program Support Library (PSL).	4-9
4.2.7 HIPO Charts.	4-10

TABLE OF CONTENTS (CONT'D)

	PAGE
4.2.8 Program Design Language (PDL).	4-11
4.2.9 Precompilers	4-12
4.3 Empirical Data Summary	4-12
4.3.1 Introduction	4-13
4.3.2 Maintenance Forms Generation Procedure	4-14
4.3.2.1 Maintenance Packages	4-14
4.3.3 Data Distributions	4-15
4.3.3.1 Productivity Data.	4-15
4.3.3.2 Measurement Data	4-23
4.3.4 Data Collection Refinements.	4-28
4.3.5 Future Research.	4-28
4.4 Conclusions and Recommendations.	4-29
REFERENCES	R-1
APPENDIX A - INSTRUCTIONS FOR COMPLETING DATA COLLECTION FORMS . . .	A-1
APPENDIX B - PAVE PAWS MAINTENANCE DATABASE.	B-1

LIST OF FIGURES

	PAGE
FIGURE 3.1: EXAMPLE OF COMPLETED PROGRAMMER PROFILE.	3-7
FIGURE 3.2: EXAMPLE OF COMPLETED "PEP" CODING FORM	3-8
FIGURE 3.3: EXAMPLE OF COMPLETED CPCG DESCRIPTION FORM	3-9
FIGURE 3.4: EXAMPLE OF COMPLETED CDF FORM.	3-10
FIGURE 3.5: EXAMPLE OF COMPLETED CPCG STATUS CODING FORM	3-11
FIGURE 3.6: EXAMPLE OF A COMPLETED CPCG MAINTENANCE ACTIVITY FORM. . .	3-13
FIGURE 3.7: EXAMPLE OF A COMPLETED CPCG MAINTENANCE ACTIVITY CODING. .	3-14
FORM	
FIGURE 3.8: DATABASE FILES	3-17

TABLE OF TABLES

	PAGE
TABLE 4.1: DR DISTRIBUTION.	4-13
TABLE 4.2: CPCI ASSIGNMENTS	4-14
TABLE 4.3: DISTRIBUTION OF MAINTENANCE PACKETS EXAMINED	4-14
TABLE 4.4: CHANGES BY TYPE AND MANHOURS EXPENDED.	4-16
TABLE 4.5: MANHOURS BY CHANGE	4-17
TABLE 4.6: EFFECTED LINES OF CODE DISTRIBUTION.	4-19
TABLE 4.7: EXPANSION RATIOS	4-20
TABLE 4.8: PRODUCTIVITY MEASUREMENTS.	4-21
TABLE 4.9: MANHOURS PER 1000 LOC.	4-22
TABLE 4.10: LOC GENERATED/MONTH.	4-22
TABLE 4.11: EFFORT VERSUS ERROR CORRECTION/ENHANCEMENT	4-24
TABLE 4.12: ERROR TYPE DISTRIBUTION.	4-25
TABLE 4.13: MEANS OF DETECTION DISTRIBUTION.	4-26
TABLE 4.14: ERROR TYPE VERSUS MEANS OF DETECTION	4-27

ACKNOWLEDGMENT

IIT Research Institute wishes to express sincere thanks to Lt. Col. R. Youngblood, Capt. A. Harriott, Capt. R. Hart, Capt. D. Henry, and Mr. G. Tyrrel of the PAVE PAWS System Programming Agency for permitting us access to their environment, providing time from their busy schedules to support the collection of information, and allowing access to their files that permitted evaluation of Modern Programming Practices, Tools and Techniques in the PAVE PAWS maintenance environment.

SECTION I

INTRODUCTION

1.1 Objectives and Scope

The overall goal of this effort is to provide an evaluation of the impact of Modern Programming Practices (MPP), tools, and techniques on the Operations and Maintenance (O & M) phase of the PAVE PAWS software life cycle. While the use of modern programming techniques and software engineering tools for software development is reported to produce significant improvements in programmer productivity, cost, reliability and quality of software systems, the PAVE PAWS environment is believed to be the first area in which the impact of these approaches may be evaluated in the O & M environment. Structured coding, programming conventions, program design language, top-down design, the program support library, etc. should make software O & M tasks easier, reduce manpower and training requirements, and thus reduce overall costs.

The objectives of this effort are two fold: 1) to determine if the tools used for software development also have an impact on the O & M phase of the software life cycle, and 2) to compile a database of O & M information.

1.2 Background

For a number of years, the U.S. Air Force through its procurement agencies, the Electronic Systems Division (ESD) at Hanscom AFB, Massachusetts and the Rome Air Development Center (RADC) at Griffiss AFB, New York, have pursued the development of methodologies for the reduction of software development costs, maintenance costs and the improvement of software quality. The impetus for this effort has come from increasing demands for error-free, highly reliable software; from the need for software easily i.e., quickly and inexpensively modified to meet newly imposed requirements; and from the recognition that a cost-effective software engineering discipline can be achieved through the identification, application and evaluation of improved production practices (THAY76; MCCA77; STAN77; DONA80; WILL76; BAKE77).

These efforts have culminated in the practical development of methodologies referred to as MPP. These include the structured programming of Dijkstra, as formalized by a number of interpreters; managerial approaches such as those developed by IBM or described by Weinberg and Yourdan; coding conventions; and a variety of software tools, including source code preprocessors, monitors, pre-compilers, verifiers, program support libraries, etc.

The initial experiences in the use of MPP indicated that many of the recommended techniques could lead to major reductions in the costs of producing software for major systems and to a noticeable improvement in the quality of the software product.

To date, the validation of MPP has been primarily limited to the development of software systems. The experience gained thus far has been highly favorable.

The development of the PAVE PAWS Phased Array Warning System at Beale AFB, California and Otis AFB, Massachusetts has presented the unique opportunity to assess the usefulness of MPP during the development of a major software system and additionally to evaluate the effects of MPP and software engineering tools on the O & M phase of the software life cycle.

PAVE PAWS was developed using a complete modern programming environment as described in RADC-TR-79-139 and is believed to be the first such system to use software engineering tools and methods in an integrated and comprehensive manner. Studies of the development cycle of PAVE PAWS (RAYT79; CURT80b) suggest that major benefits were derived from the use of MPP and software tools. These benefits included:

- o the control and management visibility required to guide a major software project to a successful completion and schedule. In particular, management found the mechanism of the Program Support Library (PSL) to be of tremendous value as a management information tool;
- o the disciplined programming environment of modern programming technology used on PAVE PAWS measurably improved the transition of software development from the mysterious and arty to the clear and cohesive world of software engineering;

- o top-down design and implementation was effective in assuring that all system functions were accounted for in the software design and assisted in the tracing of system requirements from the highest level of mission functions to the lowest component of code produced;
- o the commonality and standardization of coding techniques, naming conventions, and the uniform presentation accomplished by indented listings contributed to programmer understanding within and among the groups established to code major system functions. This commonality enhanced design and code reviews by providing a common frame of reference for discussion and continuity. Program concepts and structure could be communicated between programmers and offered the greatest improvement to efficiency and effectiveness.

As stated earlier, it was believed that the tools used for software development would have an even greater payoff during the O & M phase of the software life cycle. Structured coding, programming conventions, program design language, top-down design, the program support library, etc., should make software O & M tasks easier, reduce manpower and training requirements, and thus reduce overall costs.

To form a basis for testing these beliefs, the PAVE PAWS Phased Array Warning Systems at Otis AFB, Massachusetts and Beale AFB, California were selected with the intention of evaluating the impact of MPP and software engineering tools on the system maintenance activities.

1.3 Management Summary

1.3.1 Purposes of this Study

The primary objectives of this study were to:

- (1) Collect, screen and process data from the PAVE PAWS software maintenance effort.
- (2) Compile a computer database of O & M data which can serve as a basis for future software maintenance data collection efforts.
- (3) Evaluate the effects of MPP and software engineering tools on the O & M phase of the PAVE PAWS software life cycle.

1.3.2 Summary of the Study Approach

To accomplish the objectives of this effort, the following tasks were performed:

- (1) The literature was reviewed to identify O & M activities and data collection requirements in order to determine the effects of the use of MPP during development, on the maintenance phase.
- (2) Software maintenance data collection forms and procedures were designed.
- (3) Interviews were conducted with O & M personnel at Beale AFB to identify the benefits and difficulties of utilizing MPP and software engineering tools in the maintenance environment.
- (4) The software maintenance data that was collected by the PAVE PAWS maintenance staff was screened for accuracy, applicability and completeness; and was processed and stored in the PAVE PAWS Software Maintenance Experience Database.
- (5) The data collected in 3) and 4) was summarized and analyzed.

This software maintenance data is continuing to be collected by the PAVE PAWS maintenance staff. The data is transmitted to the Data and Analysis Center for Software (DACS) establishing a potential resource for more extensive and conclusive study and analysis.

SECTION II

PAVE PAWS MAINTENANCE ENVIRONMENT

2.1 Introduction

The delivery of the PAVE PAWS system by the contractor and acceptance by the U.S. Air Force marked the transfer of responsibility for the PAVE PAWS system and the initiation of the O & M phase of the software life cycle. The PAVE PAWS System Programming Agency (SPA) was the organization specifically created by the Air Force and given the responsibility for software maintenance. It is the purpose of this section to provide an organizational description of the SPA and a summary of the configuration control of software problems.

2.2 PAVE PAWS Maintenance Organization

The following is an extraction from the Organization Description of the SPA:

PAVE PAWS SPA Organizational Description:

- o Mission - The SPA mission is to support AN/FPS-115 missile warning system operations and maintenance activities of all computer software for all computer systems at all sites.
- o The SPA is composed of the following groups:
 - (a) The SPA staff (AD) - Provides leadership and management for the PAVE PAWS SPA.
 - (b) The Programming branch (ADP) - Provides analysis, design, development, enhancement and maintenance for all PAVE PAWS software with the exception of the Scenario Generator, Program Support Library and miscellaneous support software. It is composed of four sections organized across subsystem lines. These sections are:
 - (1) Analysis (ADP/ANL) - Provides immediate and long-range analysis of AN/FPS-115 system software effectively. Priority task is assistance in high-priority error analysis. Ongoing task is in-depth analysis of total system to improve software reliability, accuracy, maintainability and availability.
 - (2) Tactical (ADP/TAC) - Provides design, development, enhancement and maintenance for all tactical software which includes the Real Time Monitor, Mission Control, Radar Management,

Calibration and Performance Monitor, Tracking, and Displays and Control.

- (3) Radar Control (ADP/RDR) - Provides design, development, enhancement and maintenance of all Radar Control and diagnostic software. This includes the MODCOMP operating system, the on-line and off-line diagnostic software, the signal processing software, and the tactical radar control function.
- (4) Operating System (ADP/SYS) - Provides design, development, enhancement and maintenance for all CYBER system software to include the CYBER operating system, all external communications (radar controller, displays, communications lines), system reconfiguration, the displays subsystem, and on-line diagnostics.
- (c) The Quality Assurance Branch (ADQ) - Provides central configuration control of all PAVE PAWS system software and documentation. Maintains and controls electronic printed source for all software under configuration control. Acts as administrative control point for all software and documentation change actions throughout all stages of development. (Note: Serves as central point of control for accepting and processing all discrepancy reports). Provides and maintains central library of formal system documentation, vendor publications, and educational and instructional materials. Provides design, development enhancement and maintenance for all system support software to include Real Time Simulation, Target Scenario Generation, Data Reduction, and Program Support Library software. Also responsible for all non-operational software designed to support any activity within the system. Responsible for testing all software prior to release to insure software and system reliability.

The organizational structure of the SPA provides a facility to utilize MPP and reflects an understanding of their associated management techniques.

The SPA is designed around the Chief Programmer Team concept. The Programming Branch (ADP) is one chief programmer team, and consists of the four sections previously described. Each section is organized as a programming team, with the section head equivalent to a lead programmer. The branch chief acts as the chief programmer. Each section (or team) is responsible for a major PAVE PAWS maintenance component. These responsibilities are aligned with the Computer Program Configuration Items (CPCI's) of the PAVE PAWS development requirements, new requirements now that the system is operational, and modern programming management techniques. Currently ten CPCIs are defined and assigned as follows:

- o SYS - CPCI 1 -- PAVE PAWS Operating System
- o TAC - CPCI 2 -- Tactical Software
- o RDR - CPCI 6 -- Radar Control Software
 - CPCI 7 -- Signal Processor Software
 - CPCI 8 -- Receiver/Transmitter Test Software
 - CPCI 9 -- Digital Module Test Station Software
- o ANL -- System Analysis

The second chief programmer team (ADQ) is headed by the branch chief. This team maintains the following CPCIs:

- o ADQ - CPCI 3 -- Simulation Software and Target Scenario Generation
 - CPCI 4 -- Program Support Library Software
 - CPCI 5 -- Data Reduction Software
 - CPCI 10 -- Miscellaneous Support Software

In addition to the above CPCIs, the SPA/AD has assigned the ADQ branch the responsibility for configuration management, testing, reliability and maintenance change certification. The ADQ branch is assigned the major responsibility as the "independent test team" for certifying all proposed changes to the PAVE PAWS software. The ADQ branch is also the "quality assurance team", responsible for reviewing all changes and providing a configuration management control and dissemination point.

The SPA Procedure for Testing by ADQ is:

- o After CPT & E is complete, a complete test package is delivered to ADQ. This package includes: the modified software, the programmer test sequence, the CP test sequence, and all updated documentation.
- o The initial testing by ADQ is done on the backup Cyber in an off-line mode. ADQ recompiles the CPCI code to be tested along with the unmodified code and recreate the new CPCI.
- o The first phase of testing is a recreation of the PT & E and CPT & E. If successful, the testing procedure will continue.
- o DT & E (development test and evaluation) by ADQ involves the use of simulation runs against the modified CPCI. Simulation tapes were generated by the contractors as part of the PAVE PAWS deliverables and in

some cases have been extended by the SPA staff. ADQ uses these "Sim tapes" in their DT & E and checks for two things at the end of a run:

- Did the modified software perform properly?
- Was system performance degraded by the modifications?
- o If no problems are encountered, the results of the testing, the test packages, and its documentation are forwarded to HQ SAC for review and approval.
- o Assuming approval by SAC, time is requested on the operational system to perform operational test and evaluation (OT & E) which is performed by the SPA on the operational system. With higher headquarters approval, the PAVE PAWS operational system is reloaded with the new software. During OT & E, the operations personnel continue their normal functions and with the aid of the SPA they attempt to use the functions affected by the modified software.
- o If OT & E is successful, a request for an operational date is forwarded to higher HQ. Depending on circumstances, the software may be left in the system and the communications environment turned on. Alternatively, they may reload the previous version and return to full operational status (communications gear turned on).
- o After approval by higher HQ, an official release date is established. On this date, the PAVE PAWS systems at Beale AFB and OTIS AFB will contain the modified software. Two factors currently require special consideration in the independent test and evaluation process: time on the operational system for testing, and manpower resources to carry out the testing.
- o Requests for time on the operational system must be submitted six weeks in advance. This creates problems for several reasons. All requests for system modifications are "Critical" depending on who is requesting them. As a result, the requests for time are made prior to completion of DT & E. Thus, there is always the possibility that the software won't be ready when the computer is ready. Also the amount of time requested can be a factor since it impacts the operational environment. If too much time is requested, mission schedules are affected. If too little time, the tests may not be completed.

2.3 PAVE PAWS Maintenance Process Overview

The keys to the PAVE PAWS maintenance process are: 1) how the SPA is informed of a system problem (or request for enhancement), and 2) the internal structure that the SPA has created for handling a request once they have been informed.

The mechanism for informing the SPA of a problem is the discrepancy report (DR) and the structure for processing all DRs is the Configuration Review Board (CRB). The following is a brief description of the DR and CRB as applied to the PAVE PAWS maintenance process:

- o A DR is used to report a suspected error or a system problem and to request an enhancement to the software source code or documentation;
- o A DR may be originated by anyone (the users of the Sixth Missile Warning Squadron (6 MWS) or Seventh Missile Warning Squadron (7 MWS), the maintenance staff, higher headquarters, etc.);
- o A DR contains a statement of a problem or a request for enhancement;
- o When a DR is submitted by a site (6 MWS at Otis or 7 MWS at Beale), a Site Configuration Review Board (SCRB) reviews the DR prior to forwarding it to the SPA CRB. All other DRs are forwarded to the SPA directly;
- o The SPA CRB meets on a periodic basis (weekly) to review all DRs. New DRs are reviewed by the CRB, assigned to a SPA branch for analysis and a suspense date assigned for a "Memo of Recommended Action" (MRA) to be returned to the CRB. DRs which have been returned to the CRB by the SPA division with an MRA, are reviewed by the CRB, and if approved by the CRB, forwarded to higher headquarters for further action;
- o After approval is received by the SPA to proceed with modifications to satisfy a DR, the CRB assigns the task to the SPA branch responsible for the changes.

When a SPA branch receives a DR for action, the branch chief assigns the DR to a section chief. The individual assigned is responsible for reviewing all previous documentation accompanying the DR, making the modifications required to satisfy the DR, and extensively testing all changes. In addition, he is responsible for a formal test plan to validate the changes and to update all documentation affected by the changes. This completes the first step in the modification process, the Programmer Test and Evaluation (PT & E).

Upon receipt of the above from a programmer, the branch chief is responsible for the next step, the Chief Programmer Test and Evaluation (CPT & E). Where the programmer was responsible for module testing and validation, the branch chief functions as a chief programmer responsible for the integration testing of all changes. The Chief Programmer Team will load all modified software for a major system module and repeat not only the programmer test plans, but also all

simulation testing that applies to the module. If no problems are encountered, a formalized test plan is created and submitted along with all documentation to the ADQ branch for independent test and evaluation.

The preparation of a system software release is the responsibility of the ADQ branch which has the major responsibility for certifying the modifications before any software release can be considered. Certification in this context means to assure as close to 100% certainty as is possible, that the modified software once introduced into the operational system will not only perform as advertised, but will also not detrimentally impact the operational system. Development Test and Evaluation (DT & E) is the process used by the ADQ branch to certify a proposed system update. Using the backup hardware of the PAVE PAWS system, the ADQ creates a new software system containing all changes which have been incorporated into the proposed release. This candidate system is subjected to extensive testing which includes a rerun of the PT & E and CPT & E sequences. In addition, PAVE PAWS simulation runs are made against the candidate system as well as any operational type tests which ADQ feels are necessary to check out the candidate system. Only when the candidate system has passed all testing successfully, will ADQ consider proposing it for Operational Test and Evaluation (OT & E).

The preparation of OT & E involves not only the SPA but also higher headquarters (HQ SAC, ADCOM, and NORAD). Within the SPA, the ADQ branch prepares complete documentation on the proposed release, and forwards a Version Release Request to HQ SAC/SXMG for approval. (It is important to note that the preparations and procedures for a proposed system release are quite detailed and are only summarized here.) OT & E requires that the PAVE PAWS operational system be in a non-operational status until it can be verified that the newly introduced software is functioning properly.

Upon completion of OT & E, PAVE PAWS is returned to its previous operational state. A final OT & E test report is produced and submitted to HQ SAC/SXMG for approval. Assuming no problems, SAC will designate a deadstart date/time when the new system release will become operational.

2.4 Configuration Control Summary Description

A DR is used to report known or suspected software and software documentation problems to the SPA. A DR may be initiated by any person recognizing a requirement, and all are reviewed by a Configuration Review Board for acceptance or rejection. DR's not rejected will produce an additional series of forms to further control the actions. These include a Program Modification Request (PMR) a Program Change Document (PCD), a Program Documentation Discrepancy Report (PDDR) and a Modification Design and Interface Specification (MDIS). All man hours and computer time used by the division in gathering data and performing initial analysis of the DR is logged on the Project Control (PC) form until the action is completed and the DR is considered closed.

The following is a brief description of the purpose of these forms/documents:

PMR - Program Modification Request

used to request modifications to computer programs which add, modify or delete operational capability so as to require specification changes, and/or provide new programs, and/or provide new systems.

PCD - Program Change Document

used to document program malfunction corrections and data base changes

PDDR - Program Documentation Discrepancy Report

used to identify and correct errors in software documentation and to effect formal changes in software documentation

MDIS - Modification Design & Interface Specification

used to show the proposed design of a software or hardware modification and its interfaces, if any, with other software or hardware

PC - Project Control Form

used to track the milestones reached and computer and manpower resources used

SECTION III

DATA COLLECTION

3.1 Methodology

Data Collection was an iterative process influenced by data needs and the availability of data to satisfy the initial objectives of this study. The steps performed to collect and transform the data are summarized below:

- 1) Reviewed the state-of-the-art in software maintenance to determine the type of data which must be collected to evaluate the effects of MPP and SE tools on the software maintenance process.
- 2) Developed a set of questions which were used to guide IITRI project personnel during interviews with government maintenance personnel. These questions served to obtain information related to personal observations, experiences and opinions of maintenance personnel about the usefulness of MPP and Software Engineering (SE) tools.
- 3) Visited the PAVE PAWS SPO at ESD to become acquainted with the current PAVE PAWS O & M environment, including the level of the current data collection effort.
- 4) Refined the set of interview questions developed in Step 2.
- 5) Interviewed PAVE PAWS maintenance personnel, particularly management, to explain the purpose of the data collection effort, and to determine the extent which they could cooperate and the availability of data. Obtained were copies of the following software documentation and data:
 - o PAVE PAWS source listings (unclassified only)
 - o Discrepancy Report Data Base (DRDB) output. The DRDB is a machine readable file developed and maintained by the PAVE PAWS maintenance staff to track each discrepancy report (DR) from initiation through resolution.
 - o PAVE PAWS software documentation (unclassified), including:
 - Software System Requirements Specifications
 - Detailed Design Specifications
 - User's Manual/Operator's Manual
 - Problem Reports
 - Software Version Release Reports

- 6) Developed a set of data collection procedures and forms, including:
 - o Programmer Experience Profile (PEP) Form
 - o Computer Program Configuration Group (CPCG) Description Form (CDF)
 - o CPCG Maintenance Activity Form (CMAF)
- 7) Coordinated with the maintenance personnel at Beale AFB on the completion of the PEP form. Reviewed the PEP forms and revised the tool usage questionnaires.
- 8) Interviewed maintenance personnel at Beale AFB to determine their personal observations, experiences and opinions on the use of MPP and SE tools.
- 9) Met with the maintenance personnel to explain the purpose of the Maintenance Activity Form and obtain permission to review the documentation on each Discrepancy Report filed to date and to record the desired information on the CMAF.
- 10) Transferred the CPCG/CPC/Segment description and change history data from the PSL management reports to the PAVE PAWS component description and change history forms. Transferred the data recorded on the data collection forms to the data coding forms for input to the PAVE PAWS Maintenance Experience Database.

3.2 Data Identification

The preparation for the data collection trip to the PAVE PAWS site at Beale AFB, California consisted of reviewing all available PAVE PAWS documentation and becoming familiar with the MPP, tools, and techniques that were used in the development cycle of PAVE PAWS and which were included in the delivery and acceptance of the PAVE PAWS system by the Air Force.

Upon completion of the above review, a report was generated by IITRI outlining a data collection and reporting methodology for collecting information which could be used to relate the quality of the software documentation and source code, the use of SE tools, and the experience level of maintenance personnel to the cost of operation and maintaining the PAVE PAWS Phased Array Warning System.

The scope of this report was limited to those data collection activities which were believed to be essential to accurately evaluate the effects of MPP and

SE tools on the O & M phase of the software life cycle. Emphasis was placed on obtaining the opinions of the PAVE PAWS personnel through interviews. The PAVE PAWS maintenance staff was also requested to complete a Programmer Experience Profile questionnaire to determine their experience with MPP and SE tools.

3.2.1 Questionnaire Interview Procedure

The objective of this portion of the data collection effort was to obtain information concerning the personal observations and opinions of the PAVE PAWS maintenance staff with respect to MPP, tools and techniques. Previous efforts have described and reviewed MPP from the development viewpoint and assessed their value during the construction of a major system effort. While the information from these previous efforts was helpful in formulating approaches for the IITRI data collection effort at Beale, it is important to note that this portion of the data collection effort by IITRI was concerned with the use of MPP, tools and techniques by the PAVE PAWS maintenance staff in pursuing their function of maintaining the PAVE PAWS software.

The purpose of the data collection effort was to determine: 1) if the PAVE PAWS maintenance staff continued to use these same MPP, tools and techniques; 2) the value the maintenance staff assigned to these practices, tools and techniques; and 3) the contribution of each to benefits realized in the O & M life cycle.

In pursuing the data collection, a heavy reliance was placed on personnel interviews. The initial portion of each interview was unstructured and involved general discussions on the segment of PAVE PAWS for which the interviewee(s) had been assigned responsibility. In this portion of the interview, the PAVE PAWS segment, its structure as related to MPP, the difficulties and the benefits of MPP as applied to the PAVE PAWS tasking of the segment, and the overall workload of the segment was discussed. The second portion of the interview used a structured approach involving the completion of the questionnaire contained in the "PAVE PAWS Data Collection Scenario". The data collected in these interviews is summarized in Section 4.2.

3.3 Data Collection (Data Availability)

Based on the analysis performed on the previous section the following data was collected. The identified data essentially falls into two categories:

- 1) Data directly applicable to this analysis
- 2) Data collected for future analysis.

Data collected which falls in the first category is comprised of

- o Hardcopy listings of all DR's filed since the maintenance staff became involved during acceptance testing in October 1979. This data is obtained from the online DR data base and contains the following information:
 - DR's opened to date
 - DR's still open
 - closed DR's
 - Software Version Release Documentation
- o Xerox copies of all available DR's and project control forms.
- o Documents describing the organization structure of the maintenance staff (SPA) and the operating instructions for each of the component divisions.
- o Personal observations and experiences of maintenance personnel with respect to the use of MPP and tools for the various CPCIs being maintained.
- o Completed PEP forms for all maintenance personnel employed on-site since November 1980.
- o The following software maintenance packets which describe the attributes of the modification:
 - o Program Change Documentation (PCD)
 - o Program Modification Requests (PMR)
 - o Program Documentation Discrepancy Reports (PDDR)

Within the second category the following data was collected:

- o Microfiche of the majority of the PAVE PAWS program design language (PDL) and source code for CPCIs 1 through 6 as originally delivered by the development contractor.

- o PSL reports by subsystem (CPCG) and program (CPC) as of November 1980 and August 1981.
 - Code progression and durability reports
 - Summary of CPCGs by library
 - Summary of CPCs by CPCG
 - Summary of segments by CPC.

Procedures have been established with the PAVE PAWS SPA to continue forwarding this information to the DACS periodically for maintenance database update and follow-on analysis.

3.4 Data Collection Forms

3.4.1 Introduction

A set of software maintenance data collection forms were designed and utilized to record the PAVE PAWS data collected during this effort. These forms were designed to identify key data collection areas and be flexible enough to accommodate additional data when it becomes available. The first form is the PEP which is used to describe the experience background of the PAVE PAWS maintenance personnel. The second form is the CPCG Description Form (CDF) which is used to describe the development environment, constraints, composition and size of a CPCG (a functionally oriented subsystem) for each CPCI. The third form is the CPCG Maintenance Activity Form (CMAF) which is used to describe the reason and nature of each software change and the resources required to implement the change.

3.4.2 Programmer Experience Profile (PEP) Form

The PEP provides background information on the personnel performing the maintenance of the PAVE PAWS software. This information includes the education and work experience of the personnel as well as experiences in methods of access, programming languages and previous experience on related projects. Specific information obtained from this form is representative of the experience of the maintenance personnel with each tool or technique, each programming language, each operating system, and each of the specific applications of the software for PAVE PAWS.

This form was filled out once at the beginning of data collection and again at the end by each analyst/programmer. It briefly classifies his/her background.

Examples of a completed PEP and a completed coding form are depicted in Figures 3.1 and 3.2. The instructions for completing this form along with a blank form and coding sheet are given in Appendix A.

3.4.3 CPCG Description Form (CDF)

The CPCG Description Form provides information concerning the characteristics of the PAVE PAWS software at the CPCG level. Information provided by this form includes CPCI name, special environmental factors and development constraints, and size of the CPCG including number of CPCs, number of INCLUDED Segments, number of source lines of code and number of machine words of code.

The data on this form is extracted directly from the following hardcopy PSL management reports: Code Progression/Durability Matrix, Summary by Program and Summary by Segments.

Examples of a completed CPCG Description and the associated coding form are depicted in Figures 3.3 and 3.4. The instructions for completing this form along with a blank form and coding sheet are given in Appendix A.

A supplementary coding form was designed to record the status of each CPCG at different points in time. This is called the CPCG Status Coding Form. This form was utilized to record the CPCG version release history by version identification, size and the date last change was made. An example of a completed form is depicted in Figure 3.5.

3.4.4 CPCG Maintenance Activity Form (CMAF)

This form is used to record the maintenance activity performed for each approved DR. Data is recorded at the CPCG level. When a DR is initiated which requires software changes to more than one CPCG, additional forms must be completed. The data provided by this form includes the type of maintenance

PROGRAMMER EXPERIENCE PROFILE

PERSONNEL ID FLASTWA

NAME FIRST LAST NAME AGE 34 DATE 19 AUG 81

PROJECT PAYE PAMS JOB TITLE Chief, S/W Config. Mgt.

POSITION D-3 GROUP (DIVISION) ADQ

A. EDUCATION (IN YEARS)

HIGH SCHOOL 4 YEAR GRADUATED 1965

COLLEGE 4

DEGREE	DEGREE YEAR	MAJOR	LOCATION
<u>A.A.</u>	<u>1967</u>	<u>MATH</u>	<u>Ventura College, CA</u>
<u>B.A.</u>	<u>1969</u>	<u>MATH</u>	<u>San Diego State, CA</u>

* COMPUTER SCIENCE COURSES 0 * COMPUTER SCIENCE CREDIT HOURS TAKEN 0

* COMPUTER SCIENCE SEMINARS 0

B. WORK EXPERIENCE

YEARS WITH COMPUTERS 5

% YEARS IN INDIVIDUAL EFFORT 40

% YEARS IN TEAM EFFORT 40

% YEARS IN SUPERVISORY CAPACITY 0 YRS. YRS.

TARGET LANGUAGE(S) NAME JOVIAL 2

TARGET MACHINE(S) NAME CYBER 2

TARGET OPERATING SYSTEMS (NAME) NOS 2

C. SPECIFIC EXPERIENCE (RESPONSE IN YEARS UNLESS OTHERWISE INDICATED)

1. <u>TECHNIQUES</u>		2. <u>PROGRAMMING LANGUAGES</u>	
STRUCTURED PROGRAMMING	<u>2</u>	JOVIAL	<u>2</u>
PDL	<u>2</u>	ASSEMBLER	<u>0</u>
HIPO	<u>0</u>	FORTRAN	<u>0</u>
TOP-DOWN DEVELOPMENT	<u>2</u>	COBOL	<u>0</u>
PSL	<u>2</u>	ALGOL	<u>0</u>
PRE-COMPILERS	<u>2</u>	PL/I	<u>0</u>
CHIEF PROGRAMMER TEAM	<u>2</u>	PASCAL	<u>0</u>
OTHER	<u>0</u>	OTHER	<u>0</u>

3. OPERATING SYSTEMS

MACHINES	OPERATING SYSTEMS	YRS.
CYBER 174	NETWORK OPERATING SYSTEM	<u>2</u>

4. PROGRAMMING APPLICATIONS (YEARS)

BUSINESS	<u>0</u>
SCIENTIFIC/MATHEMATICAL	<u>0</u>
SYSTEMS PROGRAMMING	<u>0</u>
REAL-TIME SYSTEMS	<u>0</u>
DATABASE APPLICATIONS	<u>0</u>
OTHER (SUPPORT, e.g., PSL, DATA REDUCTION, MISC. SUPPORT)	<u>2</u>

FIGURE 3.1 EXAMPLE OF COMPLETED PROGRAMMER PROFILE

PROGRAMMER EXPERIENCE PROFILE CODING FORM

PERSONNEL ID **F L A S T N A** 1-7

NAME **F I R S T L A S T N A M E** 8-27

AGE **3 4** 28-29

DATE **8 1 0 8 1 9** 30-35

PROJECT **P A V E P A W S** 36-50

JOB TITLE **C H I E F S O F T W A R E M G T** 51-70

POSITION **0 - 3** 71-75

GROUP (DIVISION) **A 0 0** 76-78

EDUCATION COLLEGE **4** 79

HIGH SCHOOL **4** 80

YEAR GRADUATED **6 5** 1-2

	DEGREE	DEGREE YEAR	MAJOR
3-11	A A	6 7	M A T H
12-20	R A	6 9	M A T H
21-29			

30-31 **0** # COMPUTER SCIENCE COURSES

32-33 **0** # COMPUTER SCIENCE CREDIT HOURS TAKEN

34-35 **0** # COMPUTER SCIENCE SEMINARS

WORK EXPERIENCE 36-37 **5** # YEARS WITH COMPUTERS

38-39 **4 0** % YEARS WITH INDIVIDUAL EFFORT

40-41 **4 0** % YEARS IN TEAM EFFORT

42-43 **0** % YEARS IN SUPERVISORY CAPACITY

	TECHNIQUES	LANGUAGES
STRUCTURED PROGRAMMING	2 44-45	JOVIAL 2 60-61
PDL	2 46-47	ASSEMBLER 0 62-63
HIPO	0 48-49	FORTRAN 0 64-65
TOP-DOWN DEVELOPMENT	2 50-51	COBOL 0 66-67
PSL	2 52-53	ALGOL 0 68-69
PRE-COMPILERS	2 54-55	PL/I 0 70-71
CHIEF PROGRAMMER TEAM	2 56-57	PASCAL 0 72-73
OTHER	0 58-59	OTHER 0 74-75

PROGRAMMING APPLICATIONS

BUSINESS	0 1-2
SCIENTIFIC/MATHEMATICAL	0 3-4
SYSTEMS PROGRAMMING	0 5-6
REAL-TIME SYSTEMS	0 7-8
DATABASE APPLICATIONS	0 9-10
OTHERS	2 11-12

	MACHINES	OPERATING SYSTEM	YEARS
13-28	C Y B F B	N O S	2
29-44			
45-60			

FIGURE 3.2 EXAMPLE OF COMPLETED "PEP" CODING FORM

CPCG DESCRIPTION FORM

PSL Date: 14 August 1981
 Library Level: ALL
 Data Source: PRG

Software Identification:

CPCI PTAC CPCG COMM

Special Environmental Factors of the Component:

a) Special Display	<u>Y</u>	h) Concurrent Development of ADP Hardware	<u>Y</u>
b) Detailed Operational Requirements Definition	<u>Y</u>	i) Time Sharing (vs Batch)	<u>N</u>
c) Change to Operational Requirements	<u>N</u>	j) Developer Using Separate Facility	<u>N</u>
d) Real Time Operation	<u>Y</u>	k) Development on Operational site	<u>N</u>
e) CPU Memory Constraint	<u>Y</u>	l) Development on other than Target System	<u>N</u>
f) CPU Time Constraint	<u>Y</u>	m) Development at more than one site	<u>N</u>
g) First S/W Developed on CPU	<u>N</u>	n) Programmer Access to Computer	<u>Y</u>

General Program Information:

Number of CPCs 15
 Number of Segments 281
 Number of Source Lines 9042
 Number of Machine Words 7971

FIGURE 3.3 EXAMPLE OF COMPLETED CPCG DESCRIPTION FORM

CPCG DESCRIPTION CODING FORM

SOFTWARE IDENTIFICATION

CPCI	P	T	A	C				
CPCG	C	O	M	M				

1- 4
5-13

SPECIAL ENVIRONMENTAL FACTORS

SPECIAL DISPLAY	Y	14
DETAILED OPERATIONAL REQUIREMENTS DEFINITION	Y	15
CHANGE TO OPERATIONAL REQUIREMENTS	N	16
REAL TIME OPERATION	Y	17
CPU MEMORY CONSTRAINT	Y	18
CPU TIME CONSTRAINT	Y	19
FIRST S/W DEVELOPED ON CPU	N	20
CONCURRENT DEVELOPMENT OF ADP HARDWARE	Y	21
TIME SHARING (vs BATCH)	N	22
DEVELOPER USING SEPARATE FACILITY	N	23
DEVELOPMENT ON OPERATIONAL SITE	N	24
DEVELOPMENT ON OTHER THAN TARGET SYSTEM	N	25
PROGRAMMER ACCESS TO COMPUTER	Y	26

PROGRAM SUPPORT LIBRARY DATA

PSL DATE	LEVEL	DATA SOURCE	NUMBER OF PROGRAMS	NUMBER OF SEGMENTS	SOURCE SIZE	OBJECT SIZE
8 1 0 8 1 4	A L L	P R G	0 1 5	0 2 8 1	0 0 5 4 4 2	0 0 7 9 7 1
27-32	33-35	36-38	39-41	42-45	46-51	52-57

FIGURE 3.4 EXAMPLE OF COMPLETED CDF FORM

CPCG STATUS CODING FORM

CPCG 1-4				PSL DATE 5-10									
C	O	M	M	8	1	0	8	1	4				
CODE PROGRESSION													
PRG	VN	CPT	VN	INT	VN	FIX	VN	TST	VN	FRZ	VN	DEL	VN
1	0	3	9	3	8	1	0	2	8	9	0	9	4
11-17	18-24	25-31	32-38	39-45	46-52	53-59							
CODE DURABILITY													
PRG 60-66	CPT 67-73	INT 74-80	FIX 81-87	TST 88-94	FTZ 95-101	DEL 102-108							
0	0	4	0	0	0	2	0	4	2	0	0	0	0
109-114	115-120	121-126	127-132	133-138	139-144	145-150							

FIGURE 3.5 EXAMPLE OF COMPLETED CPCG STATUS CODING FORM

activity being performed, the precision of the design specifications for that activity, and the complexity of the maintenance activity. Also provided is data concerning how the error was detected and the effort involved in diagnosing it, the reason and nature for a change in the software, and the effort required in each phase of the maintenance to make the change or correction in the software.

This form is a vehicle for summarizing the relevant maintenance activity data contained in the assorted documentation (PCDs, PMRs, MDISs, PDDRs, PCs, DRDB listing) associated with each DR in each of the PAVE PAWS software version releases.

Examples of a completed CPCG Maintenance Activity Form and the associated coding form are depicted in Figures 3.6 and 3.7.

3.5 Data Processing and Summarization

One of the most critical and time consuming functions of the data analysis procedure was that of processing the raw data to assure that the data was valid, accurate, meaningful, and in a form suitable for analysis.

The first step in the validation of the data was to review the documentation associated with each DR to make sure all forms required by the PAVE PAWS configuration management staff were included and that the appropriate fields had been filled out. In addition, the documentation for each Software Version Release which incorporated the modified software was reviewed to determine if all related deficiency reports and supporting data were received.

Procedures were established for processing the collected data depending upon the form in which it was received. Where the data was contained on non-standard forms, the data was first summarized onto the three standard maintenance data collection forms. These forms were revised several times during the data collection effort to reflect the availability and form of the data. After the data was summarized and validated, it was transcribed onto standardized coding sheets which were formatted for each type of form. The data was then entered into the data base.

DR # 80493 COMPONENT MAINTENANCE ACTIVITY FORM DATE 7 November 1980

Location of Activity CPCI PTAC CPCS TRCK CPC/Segment DRIT

Maintenance Type: Error Correction 1 Add Capability 2 Version Release # 85

Delete Capability _____ Optimize/Enhance _____

Brief Description of Change to be made A check was not installed to determine if a missile was on its ascent or descent resulting in impact prediction coordinates being inaccurate.

Specification Precision: Very Precise X Precise _____ Imprecise _____

Size of Change: Source Code Lines _____ Object Code Instructions _____

Urgency of Maintenance Activity (1-3) 0

Complexity of Maintenance Activity: Very Complex _____ Complex _____ Medium _____ Simple X

Very Simple _____

SECTION A Complete if Maintenance Activity was for Error Correction

MEANS OF INITIAL DETECTION - / only for corrections (not New Reqs.)
- More than one category may be /'ed

a. Hand Processing _____ d. Interrupt Error (Code _____) g. Error Message _____

b. Personal Communication _____ e. Incorrect Output or Result _____ h. Code Review _____

c. Infinite Loop _____ f. Missing Output _____ i. Documentation Review _____

k. Maintenance crosscheck (as a result of a change in other software) _____ j. Special Debug Code _____

l. Other, Describe _____

EFFORT IN DIAGNOSING THE ERROR - Do not include effort spent in initial detection

a. No. of Runs to Diagnose _____ Elapsed Computer Time (Minutes) _____ Received _____

b. Working Time to Diagnose: Days _____ Hours _____ Analysis Begins _____

c. No. of Lines of Code: Added _____ Deleted _____ Corrected _____ Project Opens _____

ERROR SOURCE

1 a. Misinterpretation of Spec. _____ e. Specified Interface Not Implemented Correctly _____ j. Deck Setup Error _____ o. Operator Error _____

b. Incorrect Spec. _____ f. Software Interface _____ k. Computational Error _____ p. Due to Prior Modification _____

c. Incomplete Spec. _____ g. Hardware Interface _____ l. Data I/O Error _____ q. Cause Not Found, Workaround Used _____

2 d. Specified Function Not Implemented Correctly _____ h. Operating System _____ m. Logic Error _____ r. Other, Explain _____

i. Support Software _____ n. Data Definition Error _____

SECTION B Complete if Maintenance Activity was to make a change.

NATURE OF CHANGE

a. Documentation (Preface or Comments) _____ d. Structural _____ e. Mission _____ o. Hardware _____

b. Fix Instruction _____ f. Other, Explain _____ b. Engineering Model _____ e. Other, Explain _____

c. Change Constants _____ c. Software Implementation _____

NEW REQUIREMENT - / those which apply

ERROR CORRECTION AIDED BY PROGRAM ANNOTATION? YES _____ NO _____

SECTION C Maintenance Effort Required for Change or Correction

STAGE	DATE		*PERSONNEL HOURS				CPU TIME	PERSONNEL ID
	Received	Forwarded	Management	Analyst	Programmer	Clerical		
Design Effort								
Coding Effort								
Unit Testing								
Integration Testing					35		12	
Testing Review								
Installation								

* Record Hours to nearest tenth of an hour PCD80075

FIGURE 3.6 EXAMPLE OF A COMPLETED CPGC MAINTENANCE ACTIVITY FORM

1-5 DR #

8	0	4	9	3
---	---	---	---	---

 DATE

8	0	1	1	0	7
---	---	---	---	---	---

 6-11

1-5 DR # 80493 DATE 801107 6-11

CPCI

P	T	A	C
---	---	---	---

 12-15

CPCG	T	R	C	K	16-19
------	---	---	---	---	-------

CPC/SEGMENT

Q	B	J	T
---	---	---	---

 20-24 (LEAVE BLANK IF MORE THAN ONE CPC)

25 MAINTENANCE TYPE E 1 26 VERSION RELEASE B 5 29-30

27

A	2
---	---

 28

SPECIFICATION PRECISION

v	p
---	---

 31-32

URGENCY OF MAINTENANCE 33 INTEGERS - RIGHT JUSTIFIED

COMPLEXITY OF MAINTENANCE

S	I
---	---

 34-35 ALPHABETIC - LEFT JUSTIFIED

MEANS OF INITIAL DETECTION	I	O	36-37
			38-39

PROGRAMMER ASSIGNED

--	--	--	--	--	--

 40-46

EFFORT TO CHANGE

0	0	3	5
---	---	---	---

 47-50

0	1	2	0
---	---	---	---

 51-54

MANHOURS

NO. OF FILES AFFECTED 001 55-57

ERROR SOURCE

NATURE
OF CHANGE

NEW RE-
QUIREMENT

M	S	58-59	A	L	64-65
---	---	-------	---	---	-------

N	S	60-61	b	b	66-67
---	---	-------	---	---	-------

62-63 68-69

STRENGTH
70-71

	70-71
	72-73

	74-75
--	-------

ADDED
DELETED
CHANGED 6

DR # 1-5

FILE NAME AFFECTED 7-80

[illegible]

3-14

Some problems were encountered during the processing of the data which restricted the amount of analysis which could be performed. The number of available DR forms completed by the maintenance staff did not agree with the number of DR entries within the DR data base. Also, the effort expenditure shown on the Project Control (PC) was not broken down into distinct maintenance phases i.e. time to detection, corrective design and analysis etc. The PSL data could not be used due to the lack of traceability of specific maintenance actions (resolved DR's).

There were approximately 1180 DR's submitted by the O & M staff (covering the period from November 1979 through September 1981) which required action on the part of the SPA. Of the 1180 DRs, 402 (over 34%) were rejected after analysis for the following reasons:

- o No error
- o Inadequate data collected at time of failure to determine the cause
- o Duplicate DR
- o Request for enhancement not currently needed
- o Inadequate manpower resources available
- o Inadequate computer resources available
- o User misuse of the system

As of September 1981, a total of 201 DRs had been resolved; 577 were still open and were in the process of being resolved by the maintenance staff. Of the 201 resolved DRs, there were only 56 packets which contained complete or near complete PCD and/or PMR and PC data. The results of the analysis of these 56 maintenance packets are included in Section 4.3 of this report.

3.6 PAVE PAWS Maintenance Data Base

The data base was developed so that it could be used for the two major functions of:

- o Supporting the analysis requirements for this proposed effort.
- o Serving as the basis for the development of a large, multiproject collection of software maintenance data which will be managed and further expanded by the DACS.

The design and compilation of the database considered both requirements. Since the database will be used to support future study and analysis of the software O & M process, it was essential that all available data concerning O & M be included. It was also essential that the database design anticipate the requirements of future O & M data analysis efforts.

This data base is comprised of seven sequential files. Each file is in a form which is compatible with Honeywell's commercial Data Base Management System (DBMS), MDQS. It can be readily adapted to MDQS which is currently available on the RADC H6180 computer system. All that must be done is to develop the high level interface software (data definitions and data dictionaries) for each of the sequential files. File/record descriptions are contained in Appendix B of this report.

<u>FILE NAME</u>	<u>DESCRIPTION</u>	<u>DATA SOURCE</u>
PPPEP	Programmer Experience Profile File	PEP
PPDRH	Discrepancy Report History File	DRDB
PPMAF	Maintenance Activity File	CMAF/SVR/DRDB
PPCDF	CPCG Description File	CDF/PSL
PPCSF	CPCG Status File	CDF/PSL
PPPCH	CPC Change History File	CDF/PSL
PPSCH	Segment Change History File	CDF/PSL

FIGURE 3.8

DATABASE FILES

SECTION IV

STUDY RESULTS

4.1 Introduction

This section of the report contains the results of the analysis of the information collected. The purpose of this analysis is to determine the effects of the use of MPPs and SE tools on the O & M phase of the software life cycle. The amount of information analysis which could be performed was very dependent upon the quantity and quality of the input data. The first subsection presents the results of interviews and surveys with the PAVE PAWS maintenance staff as outlined in Section 3.2. The results of the analysis of the empirical data is then discussed.

4.2 Summary of PAVE PAWS Maintenance Personnel Interviews

MPP is a general term encompassing a variety of procedures, standards, programming and design techniques which are considered to improve software development. Some of these techniques are Top-Down Structured Programming, Chief Programming Team and others. These are considered to improve software by making it readable to other programmers, easier to understand and debug, and helping to standardize the development process.

SE Tools are those tools used in the practical and methodical application of science and technology in the design, development, evaluation, and maintenance of computer software over its life cycle. These tools include specific charts and diagrams which aid in the design and development of software, and PSLs which aid in the development, evaluation, and maintenance of software. SE Tools can also assist the programmer in using a greater variety of Modern Programming Techniques with greater ease.

In the following sections, the data obtained in the personnel interviews of PAVE PAWS staff has been organized and presented by subject corresponding to the following MPP, tool or technique:

- o Top-Down Program Development
- o Chief Programmer Team/Librarian
- o Structured Programming (SP)
- o Structured Walkthrough and Reviews
- o Independent Quality Assurance/Testing
- o Program Support Library (PSL)
- o HIPO Charts
- o Program Design Language (PDL)
- o Precompilers

4.2.1 Top-Down Program Development

Top-Down Design and Program Development is a technique employed in the design stages of a project which implies an ordering to the sequence of decisions which have to be made in the decomposition of a software system. Essentially, the effort is examined in the most general form first, followed by stepwise refinements which allow for a better understanding of the general requirements of the project before tasks and problems need to be examined.

In the PAVE PAWS maintenance environment, the Top-Down approach considered two areas of activity: 1) maintenance of software designed using the Top-Down approach, and 2) the generation of new software.

All personnel were familiar with the Top-Down concepts, having been exposed to it either through previous experience, education (college or technical school), or Air Force training programs. Without exception, the PAVE PAWS staff felt that the use of the Top-Down approach during the generation of the original software contributed to the maintenance process by 1) making it easier to detect and find errors, 2) reducing the time required to find the error, and 3) contributing to the ease of program modification to correct the software. Where maintenance required the generation of software to either correct an error condition or enhance the system, the staff felt that the Top-Down approach greatly aided the modification process.

4.2.2 Chief Programmer Team/Librarian

This technique involves the concept of division of labor among the project members responsible for code development. Three key individuals are required. They are:

- a. The Chief Programmer, who is responsible for overseeing and coordinating the code development and producing the critical nucleus of the project;
- b. The Backup Programmer who is familiar with all aspects of the system and contributes significant portions of the code;
- c. The Librarian who is responsible for maintaining the status of the program and the test data, updating the source and test data, performing the compilations and test runs, and coordinating the documentation.

The Chief Programmer Team (CPT) concept is the basis for all software maintenance activity in the PAVE PAWS environment. The interviews with the PAVE PAWS staff indicated a firm commitment to the concept and the employment of the concept in designing the organizational structure of the SPA as well as the working formats of each SPA Division. In the past, the CPT concept has been interpreted by some as destroying individual initiative because the concept ignores the human factor in favor of a finely detailed, regimented, controllable structure where the product was produced at the expense of individual creativity. This is not the case with the PAVE PAWS organization. The positive interpretation of the CPT concept is the pooling of the team talent to accomplish tasks. In this approach, all team members are contributing their technical and management abilities. Design work, problem solving and code production tasks are assigned by the SPA/AD and/or the CRB to specific SPA Divisions based on the Division's area of responsibility. Within the Division, the Division Chief functions as a Chief Programmer in assigning the work to a section head, who performs the function of a lead programmer, or Backup Programmer, for the specific section. In the accomplishment of specific tasks, all team members contribute as needed. The team serves as a talent pool. This does not mean that the individual assigned a task is not responsible for its accomplishment. On the contrary, the individual has the total responsibility for the task's accomplishment.

Within the PAVE PAWS structure, the observed difference is that the team members work together. Depending on the problem to be solved, the solution may require the resources of one or two team members or the entire team. In all cases, the work produced is reviewed by another team member. Even though this is mandated by SPA Operating Instructions, it was found that the team members prefer this approach because it contributes to a better product. Team members also felt that they benefited from the pooling of talent and the reviews. Their exposure to the ideas and critiques of other team members increased their skills and allowed them to produce better products.

Within the Divisions, the duties of each individual are well defined. Each division, as well as the entire SPA, knows the steps required for system maintenance. From the time a problem or a request for enhancement is received by the SPA, to the time that the problem is resolved, each member is aware of his individual responsibilities in the maintenance and his contribution to the overall maintenance process.

As with any organization faced with a maintenance project of the magnitude of PAVE PAWS, the allocation and use of resources is critical to the project's success. The PAVE PAWS maintenance personnel feel that the CPT concept is a very positive factor. The benefits derived from this concept as seen by the PAVE PAWS personnel are:

- o Excellent working environment.
- o In an area where resources are limited, the CPT concept makes the best use of the available resources.
- o Compared to the past experience of PAVE PAWS maintenance personnel, the amount of effort required for a maintenance modification is reduced.
- o Because of the CPT approach, the maintenance staff feels that there is less recoding, retesting and errors found in the products produced by the PAVE PAWS maintenance organization.

In the above, it may be noted that the concept of a project Librarian was not discussed. At the time of the data collection effort, manpower limitations prevented the assignment of an individual as a Librarian. Currently, the Librarian responsibilities are supported by team members and the ADQ.

The SPA staff feels that the addition of a Librarian to the PAVE PAWS project is a priority item.

4.2.3 Structured Programming

SP is basically a set of standards for organizing the control structure of a set of computer programs where:

- a. Each program segment has only one entry and one exit point.
- b. Only three basic control structures are needed: Do-While, If-Then-Else, and Sequence.
- c. These basic constructs are augmented with the following practices: Hierarchical and modular block structures, module size limits, indented code, and meaningful variable names.

SP is viewed by the PAVE PAWS staff as one of the more important techniques in the maintenance process. While other practices or tools benefit some specific O & M activities more than others, SP benefits most O & M activities in that SP contributes to the enhancement of the software quality factors of traceability, consistency, simplicity, modularily, self-decriptiveness and expandability. In the maintenance process, the PAVE PAWS staff feels that SP makes it easier to find and correct errors. In addition, the staff felt that SP reduces the effort needed to implement the modification.

The PAVE PAWS staff was asked to rate the characteristics of SP and their contribution to the maintenance process. The following characteristics were rated highly:

- o Indented Code
- o Absence of GOTO's
- o Modular Programming
- o Simplicitiy of Programming Style
- o Enhanced Readability of Code

They also felt that SP aided changeability and maintainability, simplified the testing process, and improved accountability. The only complaint was the size of a program module. They felt that a rule that limited a module to one computer listing page (50-55 lines of source code) could be too restrictive in some cases and could create some confusion for someone reviewing the code if a function were spread over two software modules. PAVE PAWS guidelines are 50-55 lines per module with an in-house procedure available should a larger module size be required.

In the maintenance environment, SP is used for all languages including Jovial, Fortran, and Assembly. Where pre-compilers exist, all code is structured following SP practices--with no deviations permitted. Even if the computer does not have a pre-compiler, the concepts are employed in the generation of new code and the modification of existing code.

4.2.4 Structured Walkthrough and Reviews

Design Walkthrough and Review

This technique is employed primarily after a design is fairly complete and involves a meeting between the user and various members of the project, where a review of a proposal modification is examined for technical rather than managerial purposes. The review is formal and multidisciplinary in nature and often involves hypothetical inputs.

Code Walkthrough and Reviews

This technique is employed during and after the actual code production and involves a meeting between the various project personnel. Like the design walkthrough, this review is primarily technical rather than managerial in nature and is concerned with error detection and not necessarily correction, which takes place after the walkthrough. Hypothetical input is often used, and errors discovered by this technique can be corrected before computer time is used, realizing a cost reduction.

In the PAVE PAWS maintenance environment the overriding concern is to maximize the uptime of the operational system. Even with resources like a backup computer, any potential modification must undergo extensive review prior to changing the operational system. If a modification is needed, it must be correct and complete when it is initially introduced into the operational system. The first step in the validation process of any potential modification is the "Walkthrough and Review".

After receipt of a DR by the Configuration Review Board (CRB), each step of the maintenance process is subject to review.

The PAVE PAWS staff feels that Walkthroughs and Reviews are an absolute necessity. Regardless of whether they are formal reviews involving the CRB on Division Walkthroughs and Reviews of a proposed design change or an actual code modification, each step is subject to review by the SPA, not only by a formal procedure but in fact by staff preference. The staff is divided on whether or not there is a conflict between timely production and proper maintainability. However, the entire staff feels that the correctness of any change is the primary concern. One staff member put it this way: "It is better to be right the first time and any process that gets me there is worth the time". From the staff comments, they feel that walkthroughs and reviews provide more than the obvious benefit of correctness. The pooling of talent, in a formal or informal review, produces a better product and increases the capabilities of the participants.

4.2.5 Independent Quality Assurance/Testing

The ADQ has the responsibility for the independent test and verification of all proposed modifications to the PAVE PAWS software. ADQ has the prime responsibility for insuring that proposed modifications to the PAVE PAWS System perform as expected. In this role, ADQ acts as both "independent test and analysis group" and the "quality assurance group".

The Independent Testing Group is a project group within ADQ and is not involved with the design or implementation phase of the change. They are responsible for testing the proposed software release's accuracy, and designing tests to do this. Since the group is not involved in the design or the

development of the change, the tests produced are objective and complete in scope, programming time is not consumed, and more thorough testing can be accomplished in a shorter period of time.

A separate ADQ testing activity causes sections to prepare better and more complete products because they are aware that ADQ reviews everything in detail prior to committing time for DT & E. The following comment was made by AD and other Branch personnel, "We know that ADQ is going to check for completeness and accuracy, so we might as well do it right the first time". ADQ's basic philosophy compliments the attitude of the rest of the SPA in approaching independent testing, "Assume nothing works until ADQ proves it to itself".

The Independent Quality Assurance Group is responsible for planning a systematic pattern of all the action to be taken to insure that the product conforms to the user's requirements. The efforts of the Independent Testing Group fall under the supervision of this group, in addition to the testing of the design and completed system. The independence of this group from the actual design and development phases assures the same objectiveness and completeness as the Independent Testing Group.

The ADQ provides central configuration control of all PAVE PAWS system software and documentation. It maintains and controls electronic and printed source for all software under configuration control. ADQ acts as administrative control point for all software and documentation change actions throughout all stages of maintenance. ADQ provides and maintains a central library of formal system documentation, vendor publications, and educational and instructional materials. In addition, they develop and maintain all system support software including Real Time Simulation, Target Scenario Generation, Data Reduction, and PSL software.

Within the quality assurance cycle, the ADQ assures the accuracy and completeness of all proposed and finalized modifications to the PAVE PAWS system. From the SPA management standpoint, this configuration control and quality assurance are felt to be critical. The assignment of the ADQ as the focal point for quality assurance reflects management's view of the importance of quality assurance. Complimenting this view, the SPA staff feels that the

benefits received more than compensate for any additional effort required and they willingly support the quality assurance process.

4.2.6 Program Support Library

The PSL is a programming tool designed to provide extensive data collection and reporting capabilities for use by management in making timely assessments of status, specifically on error correction and changes being made to the PAVE PAWS software. In addition, the PSL was designed to support and enforce Top-Down Structured Design techniques and support an orderly progression of software from a development environment through integration and test to a delivered product.

Of all the tools used by the PAVE PAWS staff, the PSL is considered by management to be the most valuable. In an environment where configuration control is a necessity, the PSL provides some of the most essential ingredients required for management and control. The PSL is used for compilation, testing and as a code control mechanism. For Jovial source code, the PSL is the frontend mechanism for utilizing the software. Because of this, management is able to track this software at all times.

Some of the programmers did not like the requirement to use the PSL, and felt that it slowed down the maintenance process and increased the effort. However, they also expressed certain positive aspects, such as availability of copies of the previous source code, current software status, software development level, compilation and load streams, etc.

Currently, the source code for the MODCOMP computers is also maintained in the PSL on the Cyber. This requires the modification of the source language using the PSL on the Cyber, the creation of magnetic tapes containing the modified software, and the transfer of the tapes to the MODCOMP computer where the actual compilations, loads and testing are done. It seems reasonable that the extra steps required in this process would produce negative comments by the SPA staff involved. The unexpected comment from these same ADQ people is, "We like the PSL and its capabilities. We would prefer that the MODCOMP have its own version of the PSL and eliminate the software transfer currently needed".

Overall, while the PSL is considered a good product by the PAVE PAWS staff, there are several areas of the PSL which they feel could be better. The PSL documentation, particularly the user's manual, is considered poor. The user's manual shows how to transition software up and down the various levels of development (i.e. PRG, CPT, INT, FIX, TST, FRZ, DEL); to modify, compile and load a software module; and request a listing (Report) of the status of all modules in the PSL. However, the staff feel that additional useful reports are available based upon other undocumented input parameters used by the PSL as part of the above processes. In the user manual, there is no indication of how to produce additional reports other than the library program segment summary statistics and the code progression/durability reports. The documentation required for the PSL, which would allow the staff to add capabilities to the reporting process, is non-existent and they feel that they would have to list and laboriously review the PSL source to make any modifications.

4.2.7 HIPO Charts

HIPO Charts consist of Hierarchy Charts, a set of blocks, similar to an organization chart, showing each function and its division into subdivisions and the corresponding input-process-output charts, which show the inputs and outputs and the processes joining them. Since these diagrams are visual, they are easier to understand than narrative documentation, contributing to both design and documentation and aiding code production.

All discussions with the PAVE PAWS staff regarding HIPO charts produced the following response, "What are they". Of all the documentation produced on the PAVE PAWS system, HIPO's have the least, if any, use to the maintenance staff. Their major criticisms of the provided HIPO's fall into two categories; they don't go far enough to define software structure and control flow, and they are unclear.

HIPO's do exist in the "A" and "B" Spec documentation and sometimes in the "C" Specs. For HIPO's to be useful for maintenance purposes they must start at the highest level and proceed to the PDL and code levels. They must show the downward progression clearly and provide an upward referencing capability. A significant observation by the PAVE PAWS staff was made concerning the contents

of HIPO's. They feel that the original HIPO's were generated to show the direction in which the software and system development would proceed. The key to this statement is the words "development" and "would". In the maintenance process, a HIPO only has importance if it relates the level and functional category of the software to a specific CPCI/CPCG/CPC. While intent is helpful, specifics are required when tracking a problem through the system. The contents of HIPO's with better and clearer verbiage, a more concise titling approach which would allow a viewer to determine the HIPO's position in the system structure, and the inclusion of specific references to CPCG's, overlays, programs and CPC's are needed for HIPO's to be of use to the maintenance environment.

4.2.8 Program Design Language

The PDL is a design tool used to communicate the concept of the software design in necessary detail, using a form of formal, structured English, and is implemented as a separate language in the PAVE PAWS PSL.

The use of PDL is an area which has caused much reflection by the PAVE PAWS maintenance staff. As with HIPO's, PDL is a product which exists in two environments; development and maintenance. When creating new software or modifying existing software, a PDL is a very useful tool in specifying the software to be created. In the maintenance environment, emphasis is placed on finding a specific location in the software where maintenance needs to be performed. The PDL can be helpful in finding a problem location; but to be helpful, it must be up-to-date. If there is not a one-to-one relationship between the code and the PDL, the effectiveness of the PDL is considerably diminished. PDL existing as an entity separate from the source code slows the tracking process. The PAVE PAWS staff feels that the PDL would be more useful if it were contained in-line with the source code. They feel that would reduce the time required to find a problem and make it easier to specify a solution.

There was one additional comment by the PAVE PAWS staff which they felt was very important regarding PDL, "PDL is not a substitute for good commenting within a source listing". They felt that PDL is a guide, a good design tool and an effective mechanism in the creation of software. However, when software is being

reviewed for correction or possible extension, good commenting is required to better understand the function of software statements.

4.2.9 Precompilers

The precompilers used in PAVE PAWS were designed to translate Structured Programs into compiler compatible statements. This enabled the use of certain constructs, not available in the JOVIAL and FORTRAN languages, to be implemented in the development of the software and the production of code which is structured but not necessarily compiler-compatible.

The PAVE PAWS staff do not feel that the mere existence of a precompiler insures better code or reduces the time required for maintenance activities. Having the precompiler permits the use of structured programming concepts which they feel do contribute to a better product and reduce maintenance time. The advantages of having the precompiler that were cited by the PAVE PAWS staff are:

- o Ease of Programming
- o Structured Code
- o Indented Listings
- o Less Experienced Programmers Produce Better Code
- o Consistency in the Software being Maintained

Staff members at all programming experience levels prefer the structured approach to coding permitted by a precompiler. Without exception, they felt that the use of the precompiler substantially contributes to maintenance and in particular supports path analysis, ease of code segmentation and software.

4.3 Empirical Data Summary

This subsection contains a short description of the processing of the software DR's; the contents of the DR's recorded by the PAVE PAWS maintenance staff from November 1979 through September 1981; and an analysis of the PAVE PAWS error data.

4.3.1 Introduction

As of September 1981, 577 (49% of the total) DRs were still open and in the process of being resolved by the maintenance staff. Of the 201 (17%) DRs which had been resolved, approximately 55% (110) were for system enhancements (such as adding a new capability or refining a display format) and 45% (90) were for error correction, either to the documentation, source code or system specifications.

Responsibility for software maintenance has been delegated to five different sections as described in Section II, PAVE PAWS Maintenance Organization.

The distribution of DRs by responsible section is depicted in Table 4.1.

TABLE 4.1 DR DISTRIBUTION

RESPONSIBLE SECTION	#DR's GENERATED	#DR's ACCEPTED	#DR's RESOLVED	#DR's OPEN
SYSTEM	124	87	19	68
TACTICAL	699	460	95	365
RADAR	164	115	46	69
ANALYSIS	49	37	10	27
QUALITY ASSURANCE	144	79	31	48
	1180	778	201	577

Table 4.2 contains information on the assigned CPCI's per section of the PAVE PAWS maintenance staff.

TABLE 4.2 CPCI ASSIGNMENTS

RESPONSIBLE SECTION	CPCI	ACRONYM	LONG-NAME
o SYSTEM	1	PPOS	PAVE PAWS Operating System
o TACTICAL	2	PTAC	TACTICAL SOFTWARE
o RADAR	6	PRCL	RADAR Control Software
	7		Signal Processor Software
	8		Receiver/Transmitter Test Soft.
	9		Deptol Module Test Station Soft.
o ANALYSIS			System Analysis
o QUALITY ASSURANCE	3	PSIM	Simulation & Target Scenario Generation
	4	PPSL	Program Support Library
	5	PDTR	Data Reduction
	10		Miscellaneous Support

4.3.2 Maintenance Forms Generation Procedure

4.3.2.1 Maintenance Packets

A maintenance packet is a set of documents and forms used by the PAVE PAWS staff to record and track changes to the software and consists of the DR, PMR, the PCD, and the PC (see Section 2.4). Only those empirical data for which a fairly complete packet was received were examined. This included approximately 56 packets that contained complete or near complete DR, PCD and/or PMR and PC forms, and, another 28 that contained less complete information which included the DR and PC data. The first mentioned packets were required to trace a suspected or known error through its maintenance cycle to the final closing of the DR. The 2nd packet is used for some analysis correlation. This number does not represent the entire number of accepted DR's and subsequent maintenance actions but one can gain valuable insight into an analysis of maintenance activities overall.

Those data that are being utilized for analysis are distributed over CPCI's as shown in Table 4.3.

TABLE 4.3 Distribution of Maintenance Packets Examined

CPCI	# PCD/PMR FORMS	# DR/PC FORMS
PPOS 1	6	5
PTAC 2	22	22
PRCL 6	28	1
TOTAL	56	28

4.3.3 Data Distributions

This subsection summarizes the data from the Maintenance Packets and includes narrative and tabular descriptions of changes by type and manhours expended, manhours by change, distribution of lines of code produced, error distribution with respect to means of detection, and error distribution with respect to error source. Tables presented illustrate the effort required for error correction and enhancement for each Computer Program Configuration Item (CPCI).

4.3.3.1 Productivity Data

Table 4.4 contains data on the number of software changes that were made, where manhours expended for making the change were provided. A change is defined as any maintenance action that was performed on the system and is comprised of one of the following:

- o error correction
- o enhancement
- o documentation correction

Using the data from Table 4.4, the average number of manhours per change and the average number of changes per 100 manhours were calculated and are illustrated in Table 4.5.

Of the total 1710 manhours recorded, almost 50% of the time was spent making changes to PTAC, the Tactical Software. But this CPCI only accounted for approximately 30% of the total number of changes. Also for this CPCI, the average number of manhours for correcting an error was very close to the average number of manhours for implementing an enhancement (67 vs. 50, respectively); whereas, for CPCI 1, the PPOS, the difference in manhours was much greater (20 vs. 93).

The reason that the manhours expended in error correction in PTAC (CPCI 2) is considerably larger than the other two CPCI's was examined. The data shows that

TABLE 4.4 CHANGES BY TYPE AND MANHOURS EXPENDED

CPCI	Changes				Manhours			
	Total Number	Number/Type			Total Number	Number/Type		
		Errors	Enhance	Doc.		Errors	Enhance	Doc.
PPOS 1	7	2	5	-	505	39	466	-
PTAC 2	15	6	8	1	813	402	403	8
PRCL 6	26	17	6	3	393	202	186	5
Totals	48	25	19	4	1710	643	1054	13

TABLE 4.5 MANHOURS BY CHANGE

CPCI	Total	Change Type		
		Error	Enhancement	Documentation
PPOS 1 Average Number of: • Manhours/Change • Changes/100 Manhours	72 1	20 5	93 1	- -
PTAC 2 Average Number of: • Manhours/Change • Changes/100 Manhours	54 2	67 2	50 2	- 8 13
PRCL 6 Average Number of: • Manhours/Change • Changes/100 Manhours	15 7	12 8	31 3	2 60
Totals Average Number of: • Manhours/Change • Changes/100 Manhours	47 3	33 15	58 2	5 37

one error correction performed in CPCI 2 reported a manhours expended which was six times larger than the next largest reported manhours expended and accounted for 80% of all manhours expended in error correction for this CPCI. Eliminating this maintenance action, for CPCI 2, the adjusted total average number of manhours per change equals 35 and the average number of manhours per change in error correction equals 16.

Analysis of these new figures with respect to the complexity of a change leads to an interesting observation. There seems to be a direct correlation between the figures presented in Table 4.5 and the complexity of the change performed. For the most part, changes made to PRCL (CPCI 6) were simpler than changes made to PTAC (CPCI 2) or PPOS (CPCI 1). A ranking of the CPCI's in order of increasing change complexity results in the order PRCL, PTAC, and PPOS. This also happens to be the ranking for increasing manhours per change for both error correction and enhancement.

Productivity Measures

The following productivity measures were calculated and are presented in this subsection:

- o Effected lines of code distribution by CPCI
- o Lines of code effected by manhours expended
- o Effort required for each 1000 lines of code
- o Lines of code generated/month

The distribution of effected lines of code (LOC) by language and CPCI is presented in Table 4.6. Below is a discussion on the calculation of the column titled "Adjusted LOC".

Lines of Code Conversion

For this analysis, the basis of a measure of LOC is Assembly source LOC. Hence, Higher Order Language (HOL) source LOC is converted to an equivalent number of assembly LOC.

TABLE 4.6 EFFECTED LINES OF CODE DISTRIBUTION

CPCI	HOL		ADJUSTED LOC	ASSEMBLY		JCL	DATA	COMMENTS
	LANG	LOC		LANG	LOC			
PPOS 1	PASCAL			COMPASS				
Error Correction Enhancement		-	-		-	-	-	-
		12	90		23	14	0	0
PTAC 2	JOVIAL			COMPASS				
Error Correction Enhancement		37	130		-	-	3	6
		-	-		-	-	-	-
PRCL 6	IFTRAN			ASSEMBLY				
Error Correction Enhancement		51	230		136	0	0	50
		52	234		19	0	0	24

For the conversion of HOL LOC to Lower Order Language (LOL) LOC the following assumption is made. A one-to-one relationship exists between machine executable instructions and LOL instruction. That is, one LOL instruction will perform a single operation (map to one operation code). It is realized that an LOL instruction can generate more than one machine word location, as often occurs in multiple operand instructions or microprocessor instructions, but there is still only one operation performed. An LOL instruction may also generate an operation code which involves a macro. However, assuming this macro is also available to the HOL compiler, this is still considered to be only one operation and hence one LOL instruction. Based on this definition, the LOL language can be used as a basis for LOC generation. That is, we can convert HOL LOC into LOL LOC via a multiplication factor. This will eliminate the bias that results when using this metric for determining productivity, where no distinction is made between HOL and LOL source LOC (JUNE78). It should be realized that this conversion factor varies within compilers of the same language and from HOL to HOL and should optimally be determined at the users site. This was not possible within the scope of this effort, and therefore a literature search was initiated to establish a conversion factor that best fits the PAVE PAWS environment. These conversion factors or expansion ratios are presented in the following table.

TABLE 4.7 EXPANSION RATIOS (JUNK79)

Language	Expansion Ratio	Average Ratio
FORTTRAN(IFTRAN)	4-5:1	4.5:1
JOVIAL	3-4:1	3.5:1
PASCAL*	7-8:1	7.5:1
COMPASS(ASSEMBLY)	1:1	1:1

* An expansion ratio was not available for this language and was assumed to be the same as PL/1

An average of the limits was used in this analysis instead of the upper or lower limit. These figures are substantiated in (RCAP75).

A productivity measure was calculated using the adjusted LOC for three CPCIs and is illustrated in Table 4.8. The total number of LOC effected was calculated from the data in Table 4.6 by adding the adjusted LOC, Assembly LOC, and JCL. Comments were not included. The total number of manhours expended differs from the data contained in Table 4.4. This difference occurs because this productivity measure (i.e. average number of manhours expended per LOC effected) could only be calculated when both the number of LOC effected and number of manhours expended were available.

TABLE 4.8 PRODUCTIVITY MEASUREMENTS¹

CPCI	TOTAL NUMBER LOC ² EFFECTED	TOTAL NUMBER OF MANHOURS EXPENDED	AVERAGE NUMBER MANHOURS EXPENDED/ LOC EFFECTED
PPOS 1			
Error Correction	---	---	---
Enhancement	127	298	2.4
PTAC 2			
Error Correction	133	402	3.0
Enhancement	---	---	---
PRCL 6			
Error Correction	366	161	0.4
Enhancement	253	43	0.2

NOTES: ¹ This table contains data on only those changes for which both number of LOC effected and manhours expended were available.

² Total LOC includes adjusted LOC, Assembly LOC and JCL

From this table it is also possible to compute the effort required in manhours for each thousand lines of code. Figures for enhancement and error correction are presented in the following table:

TABLE 4.9 MANHOURS PER 1000 LOC

CPCI	SIZE	ENH CODE/MANHOURS	CORRECTED CODE/MANHOURS
1	31K	2350	--
2	100K	--	3030
6	125K	170	440

Another means of describing productivity is a function of the LOC generated per manmonth of effort (Table 4.10). For comparison and completeness, the above figures converted using a figure of 173.3 manhours per manmonth.

TABLE 4.10 LOC GENERATED/MONTH

CPCI	LOC/MAN-MONTH FOR ENHANCEMENT	LOC/MAN-MONTH FOR CORRECTION
1	74	--
2	--	57
6	1020	394

The data in these two tables (i.e. Tables 4.9 and 4.10) illustrate that productivity, for enhancements, for the PAVE PAWS Operating System (CPCI 1) is considerably lower than for the radar control software (CPCI 6). This observation is consistent with the findings of Jones (JUNE78) and Zelkowitz (ZELK78) where they concluded that productivity was higher for application software than for operating system software, because of the increased complexity of the operating system.

The wide discrepancy between the error correction productivity figures for CPCI 2 and CPCI 6 is attributable to the effects of the one extreme data point discussed previously. One error problem in CPCI 2 resulted in a manhour expenditure rate which was five times that of the next worst case encountered in error correction. Deleting this data point from the calculation results in an error correction productivity figure of 266 LOC/manmonth, a figure much more consistent with the 394 LOC/manmonth computed for CPCI 6.

In analyzing the data an interesting observation was made. When a considerable number (>20) of lines of code were changed (added, deleted, or modified either for an error correction or enhancement), the manhours required did not increase proportionately. That is, the number of manhours per LOC decreased with an increase in the LOC changed. This may imply that there is an initial cost in the change process which must be paid regardless of the type or extent of the maintenance action. This may be an acclimation cost, directly analogous to a learning curve on a new computer system where productivity is low initially and improves substantially once one becomes proficient with the system. Since for the most part, changes are independent of each other, this learning cost may have to be paid for on each maintenance action performed (regardless of enhancement or error correction).

4.3.3.2 Measurement Data

The maintenance activities of the PAVE PAWS system have been separated into two basically different categories: the activity necessary to find and correct an error and the activity involved in enhancing the software. The relative ease with which these tasks were carried out and the influence of MPPs and software tools on these tasks is examined in this subsection.

The dictionary defines ease as the state of being less difficult or requiring little effort. Subjectively we can relate ease to effort however, in analyzing the empirical data it will be assumed that there is a direct correlation ($\propto$) between amount of time and ease, as follows,

$$\uparrow \text{ time } \propto \downarrow \text{ ease}$$

$$\downarrow \text{ time } \propto \uparrow \text{ ease}$$

The ramifications are that simple but tedious changes are grouped into the first class and are hence interpreted as difficult changes. The reason for establishing this relationship is that effort will be expressed in the amount of time to complete a given action and can hence be related to ease.

This section concerns itself with the following four basic measurements:

- o effort versus change type
 - o enhancement
 - o error correction
- o error distribution with respect to error type
- o error distribution with respect to means of detection

Using Table 4.4, it is possible to compute the percent of effort expended for error corrections and enhancements.

TABLE 4.11 EFFORT VERSUS ERROR CORRECTION/ENHANCEMENT

CPCI	TOTAL CHANGES	# ERRORS	# ENHANCE	% OF EFFORT FOR ERROR CORRECTION	% OF EFFORT FOR ENHANCEMENT
PPOS 1	7	2	5	8%	92%
PPAC 2	14	6	8	50%	50%
PRCL 6	23	17	6	52%	48%
TOTAL	44	25	19	38%	62%

This table shows that the percent of effort spent for making enhancements to the PAVE PAWS software is approximately 61% higher than the percent of effort spent for correcting errors. By eliminating the previously mentioned extreme data point for error correction in CPCI 2, the difference is even greater. The percentages of effort for error correction and enhancement become 23% and 77%, respectively, by deleting this data point. This leads one to the conclusion that, on the average, an enhancement is more difficult to implement than an error correction.

Note that code optimization and capability addition are combined under enhancement (i.e. changes are subdivided into error correction and enhancement with additional capabilities grouped under the latter). Also documentation changes are not included under errors or enhancement, since it is plausible to assume that these do not severely impact software performance (i.e. a DR may have been generated solely to correct a documentation error, which must be distinguished from a software error by our definition).

With the present accounting methods used the effort required in finding errors cannot be readily determined. The total effort expended on a particular change is reported on the PCF and it is uncertain as to whether or not the effort expended includes error detection (finding an error).

Table 4.12 depicts the number of occurrences for each error type, for the most commonly encountered ones. Only the primary error types are presented. For example, suppose in a numerical computation an overflow was encountered and upon investigation it is discovered that the equation used in the computation was incorrect. Since, the primary error is in the specification of the equation the error will be classified as a specification error. The overflow being only a consequence or manifestation of the primary error.

Note that this table includes all of the errors examined from the 56 packets received, whereas Table 4.11 includes only information on those errors for which the effort expended was known (a PC form was available).

TABLE 4.12 ERROR TYPE DISTRIBUTION

ERROR TYPE	NUMBER OF OCCURRENCES CPCI			PERCENT OF TOTAL BY TYPE CPCI		
	1	2	6	1	2	6
SPECIFICATION	2	7	-	100%	78%	--
SUPPORT SOFTWARE	-	-	-	--	--	--
FUNCTION INCORRECTLY IMPLEMENTED	-	2	1	--	22%	4.5%
INTERFACE INCORRECTLY IMPLEMENTED	-	-	-	--	--	--
DATA I/O	-	-	-	--	--	--
SOFTWARE INTERFACE	-	-	4	--	--	18.2%
HARDWARE INTERFACE	-	-	2	--	--	9%
COMPUTATIONAL	-	-	1	--	--	4.5%
LOGIC	-	-	11	--	--	50%
DOCUMENTATION	-	-	3	--	--	13.6%

It is interesting to note that there are no specification errors in CPCI 6, the Radar Control Software, but that 50% of the errors examined were logic errors and 18% were software interface errors.

The following Table depicts a distribution based on the means of error detection and includes only those errors which resulted in a program modification.

TABLE 4.13 MEANS OF DETECTION DISTRIBUTION

MEANS OF DETECTION	1	2	6
INCORRECT OUTPUT OR RESULT	1	8	9
MISSING OUTPUT	-	1	1
PERSONAL COMMUNICATION	1	-	1
ERROR MESSAGE	-	-	3
INFINITE LOOP	-	-	3
MAINTENANCE CROSSCHECK	-	-	1
OTHER	-	-	1
TOTAL	2	9	19

In this table it is interesting to note that 68% of the errors specified in CPCI 6 were related to some form of output anomaly (incorrect, missing, or error message).

Based on the data available it cannot be readily determined which errors are attributable to specific MPP or software tools usage. However, an examination of error types and means of initial detection may provide insight into whether or not the use of MPP or software tools should have detected the presence of errors (e.g. Did any errors slip thru the code and documentation reviews, testing, etc.). Table 4.14 compares the error type to the means of initial detection.

Analysis of this data leads one to the following observations. Almost all of the error types designated as specification errors were detected by an incorrect output or result. This implies that the software testing, which tests for the adherence of the software to the design specification among other things, was not complete with respect to specification testing for all specifications and requirements. Of the errors found, a very small percentage were detected via documentation review or maintenance cross check. This would seem to be intuitively obvious since it is unlikely that the maintenance staff would purposely look for errors. It must be emphasized that these are simply observations. The scarcity of the data precludes the establishment of conclusive results.

TABLE 4.14 ERROR TYPE VERSUS MEANS OF DETECTION

ERROR TYPE	MEANS OF DETECTION							
	Personal Communication	Infinite Loop	Incorrect Output or Result	Missing Output	Error Message	Document Review	Maintenance Crosscheck	Other
Specification Error	1	0	0	0	0	0	0	0
INCORRECT IMPLEMENTATION OF SOFTWARE	0	1	1	0	0	0	1	0
SOFTWARE INTERFACE	0	1	2	0	0	0	0	0
HARDWARE INTERFACE	0	1	0	0	0	0	0	0
COMPUTATIONAL ERROR	0	0	0	0	0	0	0	0
DATA I/O ERROR	0	0	0	0	0	0	0	0
LOGIC ERROR	0	0	0	0	0	0	0	0
OTHER	0	0	0	0	0	0	0	0

4.3.4 Data Collection Refinements

The following recommendations are made to refine the data collection process to provide more usable and consistent data:

- 1) To collect a complete set of maintenance data, it is essential to initiate the data collection process immediately after the acceptance of the software product.
- 2) Recording of error correction information should be refined to provide more exact measurements of manpower and computer resource expenditures. There should be a breakdown of activity recorded (i.e. error detection, correction etc.) and standardized definition of terminology.
- 3) The PCD/PMR packages should be updated to include the specification of lines of code modified. When reviewing a PCD/PMR packet it was often difficult to determine the lines of code that were added, deleted or modified. This can be overcome in one of two ways:
 - o Include before and after source listings within these data packets, or
 - o Include new fields on the forms to report lines of code modified, programming language, and module name.
- 4) Provide distinct accounting of error correction and enhancement for program modification. When a PCD/PMR reports a combined error correction and enhancement, a separate accounting should be kept of the resources expended, lines of code modified etc. This will enable a separation and proper weighting of these activities in future analysis.
- 5) Include better tracking of software modifications from time of modification to testing and, integration within new version release. Once a problem is "corrected" (by a PCD/PMR) it is not known whether or not it will cause a problem in a system upgrade. Also, the amount of time spent in integrating and testing the corrected code is not included in the Program Control form. Nor is it known if a problem is "recorrected" because of a failure in integration and testing. This information is important due to the fact that once a problem is corrected the process is not complete until integration and testing occurs. A correction may cause a problem in another module which does not surface until integration and testing is performed.

4.3.5 Future Analysis

o Productivity

It is a desirable for future productivity analysis to establish a conversion factor for on site compilers for the purpose of determining the

number of assembly code lines per HOL line of code. Some compilers will give you the option of an assembly listing from a compiled HOL program. By taking some random programs and determining their size in the HOL lines of code and the resultant assembly lines of code a conversion factor for that compiler can be obtained. Once a conversion factor for each HOL compiler used on PAVE PAWS is determined, the accuracy of the factors can be determined. With these factors in hand their use will eliminate the partiality that exists between HOL and lower order languages (LOL) when counting lines of code.

o Determination of Software Quality Factors

Once an adequate amount of information has been collected the reliability and maintainability of specific software modules can be ascertained. By analyzing software modules with high and low degrees of reliability and maintainability it may be possible to determine the factors that impact software quality.

o Comparison to Other Projects

As more data is acquired at the DACS on the maintenance of computer software, it is recommended that studies be performed comparing the results and observations contained in the PAVE PAWS Database with other databases.

4.4 Conclusions and Recommendations

Although the data that was available during this study was scarce and too limited to form conclusive quantitative determinations, the experiences, opinions and comments of the PAVE PAWS maintenance staff provide a sufficient basis to develop some strong conclusions on the use of Modern Programming Practices and Software Engineering Tools.

From the viewpoint of the PAVE PAWS maintenance staff, Modern Programming Practices, Tools and Techniques are a valuable asset, not only from the product quality viewpoint but also with respect to the maintenance structure and process. The use of these practices, tools and techniques during development has produced software which is more consistent and error free than seen in previous systems. Most important to the PAVE PAWS staff is that the code produced is simpler (less complex), more straight forward and structured. In the maintenance environment, these same Modern Programming Practices, Tools and Techniques are an even more valuable asset.

In the PAVE PAWS environment where the prime dictums are "timely response to the operations world" and "accuracy and reliability for any modification to the

system", these practices and procedures are required for the production of reliable software modifications. While the maintenance of the PAVE PAWS software is early in the life cycle of the system, the maintenance staff has already placed heavy reliance on Modern Programming Practices, Tools and Techniques. To date, the staff feels that this reliance has been justified.

The greatest handicap to the PAVE PAWS maintenance staff is caused by the maintenance documentation. The delivered documentation (the "A" Specs, "B" specs, "C" specs, HIPOS, PDL and written commentaries) describe the PAVE PAWS system from the development standpoint as the system was being created. To the developer, this approach provides good documentation and traceability of the system during the development process. For the maintenance process, the documentation must reflect the current status of the system from the top-down as well as from the bottom up. The maintenance staff has pointed out that they have to be able to find the location of an error and be able to determine the conditions that caused the error (top-down). Once found, they need to know what affect a proposed modification will have on the system (bottom-up).

The PAVE PAWS personnel were asked to clarify their remarks and where possible to indicate what additional documentation or tools are required to support the maintenance process. The PAVE PAWS staff feels that for documentation to be useful, it must reflect the current status of the system. It must be organized in a manner that allows a prospective viewer to scan it top-down. In a system the size of PAVE PAWS, functional charts and associated software organization charts are required to present a graphical view of the system. While graphical presentations are helpful, software modifications require detailed knowledge of the system software organization. Graphical representations of the modules (CPCs) which make up an overlay, overlays that form a major system function (CPCG) and the CPCGs that form a CPCI are also necessary to show the software organization.

Because of the limited amount of empirical data available for this study, no conclusive results can be made from analyzing this data to determine the effects of the use of MPPs and SE tools on the maintenance phase of the software life cycle. However, procedures were established for the continual collection and processing of the data to form the basis of a Software Experience Maintenance

Database at the DACS. The existence of this data base and continued receipt of data from the PAVE PAWS maintenance life cycle will greatly facilitate studies on maintenance costs, comparisons with other systems to determine relative effects of comparable or alternative tools and techniques, and productivity based on factors such as programmer experience, program size, software complexity, etc.

REFERENCES

- BAKE77 Baker, W.F. "Software Data Collection and Analysis: A Realtime System Project History." Technical Report RADC-TR-77-192. Griffiss Air Force Base, New York, Rome Air Development Center, 1977. AD#041 644.
- DONA80 Donahoo, J.D.; Swearinger, D. Computer Sciences Corporation, "A Review of Software Maintenance Technology." RADC-TR-80-13. RADC, Griffiss Air Force Base, New York 1980. AD#A082 985.
- JUNK79 Junk, W.S., McCall, J.A., "Reference Manual: Price Software Model", RCA Price Systems, December 1975.
- JONE78 Jones, T.C. "Measuring Programming Quality and Productivity." IBM Systems Journal 1978, 17, (1), 39-63.
- MCCA77 McCall, J.A.; Richards, P.K.; Walters, G.F. General Electric Company, "Factors in Software Quality: Concept and Definitions of Software Quality." RADC-TR-77-369. Volume 1. RADC, Griffiss Air Force Base, Rome New York 1977. AD#A049 014.
- RCAP75 Reference Manual: Price Software Model, RCA Price
- STAN77 Stanfield, J.R.; Skrukrud, A.M. System Development Corporation, "Software Acquisition Management Guidebook: Software Maintenance." ESD-TR-77-327. Contract Number F19628-76-C-0236. Electronic Systems Division, Hanscom Air Force Base, Massachusetts.
- THAY76 Thayer, R.A., et al. "Software Reliability Study." RADC-TR-76-238. Griffiss Air Force Base, New York, Rome Air Development Center, 1976. AD#A030 98
- WILL76 Willmorth, N.E.; Finfer, M.C.; Templeton, M.P. System Development Corporation. "Software Data Collection Study Summary and Conclusions." RADC-TR-76-329. December 1976. Vol I, AD#A036 115.
- ZELK78 Zelkowitz, Marvin V., "Perspectives of Software Engineering, Computing Surveys, Vol 10, No. 2, June 1978.

APPENDIX A
INSTRUCTIONS FOR COMPLETING
DATA COLLECTION FORMS

INSTRUCTIONS FOR COMPLETING
THE PROGRAMMER EXPERIENCE PROFILE

The purpose of this form is to classify the background of the personnel on each project. It should be filled out once at the start of the project by all personnel.

PERSONNEL ID.	Leave blank
NAME	Your full name
AGE	Optional
DATE	Date profile is being completed
PROJECT	Optional
JOB TITLE	Optional
POSITION	GS12, 0-1, E-5
GROUP	Name of company or government branch

A. EDUCATION

Degrees. Fill out educational background.
Courses. Fill in number of university and in-house computer science courses.

B. WORK EXPERIENCE

Give years involved with computers and percent time in each listed activity. Name the languages, machines and operating systems you utilize on this project and the number of years (nearest year) experience with each.

C. SPECIFIC EXPERIENCE

Give years (to nearest year) involved with the following techniques, languages and operating systems.

1. TECHNIQUES - Give number of years (to the nearest year).

Structured Programming - Writing programs using only a limited set of Control structures (e.g., if-then-else, do while).

PDL - A Program Design Language. An algorithmic specification of program as a function of its input and output data.

HIPO - Hierarchical Input Process Output. A graphical technique describing a program as a function of its input and output data.

Top Down Development - A technique where high level modules are developed before the modules that are called by these high level routines.

PSL - The Program Support Library which provides hierarchical levels to ensure qualification of the software.

Pre-Compilers - Software which extends the capabilities of HOLS by allowing structured coding.

Chief Programmer - A technique where an individual programmer writes top level code and major interfaces and delegates responsibility to others to complete it. A librarian manages all source code and documentation.

2. PROGRAMMING LANGUAGES - Give number of years (to the nearest year) you have used these programming languages.
3. MACHINES AND OPERATING SYSTEMS - Give names of machines and operating systems and the number of years experience with each one (to the nearest year). List only the three with which you have the most experience.
4. PROGRAMMING APPLICATIONS - Give the number of years experience you have in programming for each of the listed applications.

PROGRAMMER EXPERIENCE PROFILE

PERSONNEL ID _____

NAME _____ AGE _____ DATE _____

PROJECT _____ JOB TITLE _____

POSITION _____ GROUP (DIVISION) _____

A. EDUCATION (IN YEARS)

HIGH SCHOOL _____ YEAR GRADUATED _____

COLLEGE _____

DEGREE	DEGREE YEAR	MAJOR	LOCATION
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

COMPUTER SCIENCE COURSES _____ # COMPUTER SCIENCE CREDIT HOURS TAKEN _____

COMPUTER SCIENCE SEMINARS _____

B. WORK EXPERIENCE

YEARS WITH COMPUTERS _____

% YEARS IN INDIVIDUAL EFFORT _____

% YEARS IN TEAM EFFORT _____

% YEARS IN SUPERVISORY CAPACITY _____ YRS. _____ YRS.

TARGET LANGUAGE(S) NAME _____

TARGET MACHINE(S) NAME _____

TARGET OPERATING SYSTEMS (NAME) _____

C. SPECIFIC EXPERIENCE (RESPONSE IN YEARS UNLESS OTHERWISE INDICATED)

1. TECHNIQUES

STRUCTURED PROGRAMMING _____

PDL _____

HIPO _____

TOP-DOWN DEVELOPMENT _____

PSL _____

PRE-COMPILERS _____

CHIEF PROGRAMMER TEAM _____

OTHER _____

2. PROGRAMMING LANGUAGES

JOVIAL _____

ASSEMBLER _____

FORTRAN _____

COBOL _____

ALGOL _____

PL/I _____

PASCAL _____

OTHER _____

3. OPERATING SYSTEMS

MACHINES	OPERATING SYSTEMS	YRS.
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

4. PROGRAMMING APPLICATIONS (YEARS)

BUSINESS _____

SCIENTIFIC/MATHEMATICAL _____

SYSTEMS PROGRAMMING _____

REAL-TIME SYSTEMS _____

DATABASE APPLICATIONS _____

OTHER _____

PERSONNEL ID

--	--	--	--	--	--	--

 1-7

Category	Range
PERSONNEL ID	1-7
NAME	8-27
AGE	28-29
DATE	30-35
PROJECT	36-50
JOB TITLE	51-70
POSITION	71-75
GROUP (DIVISION)	76-78
EDUCATION COLLEGE	79
HIGH SCHOOL	80
YEAR GRADUATED	1-2

		DEGREE	DEGREE YEAR	MAJOR																				
	3-11	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>								
	12-20	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>								
	21-29	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>								
	30-31	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							# COMPUTER SCIENCE COURSES															
	32-33	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							# COMPUTER SCIENCE CREDIT HOURS TAKEN															
	34-35	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							# COMPUTER SCIENCE SEMINARS															
WORK EXPERIENCE	36-37	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							# YEARS WITH COMPUTERS															
	38-39	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							% YEARS WITH INDIVIDUAL EFFORT															
	40-41	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							% YEARS IN TEAM EFFORT															
	42-43	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							% YEARS IN SUPERVISORY CAPACITY															
		TECHNIQUES		LANGUAGES																				
STRUCTURED PROGRAMMING	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							44-45	JOVIAL	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							60-61							
PDL	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							46-47	ASSEMBLER	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							62-63							
HIPO	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							48-49	FORTRAN	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							64-65							
TOP-DOWN DEVELOPMENT	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							50-51	COBOL	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							66-67							
PSL	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							52-53	ALGOL	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							68-69							
PRE-COMPILERS	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							54-55	PL/I	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							70-71							
CHIEF PROGRAMMER TEAM	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							56-57	PASCAL	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							72-73							
OTHER	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							58-59	OTHER	<table border="1"><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>							74-75							

PERSONAL EXPERIENCE PROFILE CODING FORM

PROGRAMMING
APPLICATIONS

BUSINESS	☐	☐	1-2
SCIENTIFIC/MATHEMATICAL	☐	☐	3-4
SYSTEMS PROGRAMMING	☐	☐	5-6
REAL-TIME SYSTEMS	☐	☐	7-8
DATABASE APPLICATIONS	☐	☐	9-10
OTHERS	☐	☐	11-12

	MACHINES	OPERATING SYSTEM	YEARS
13-28	☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐	☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐	☐ ☐
29-44	☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐	☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐	☐ ☐
45-60	☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐	☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐	☐ ☐

INSTRUCTIONS FOR COMPLETING
THE CPCG DESCRIPTION FORM

The purpose of this form is to obtain general information concerning the CPCGs in the PAVE PAWS environment. It should be completed for each CPCG initially in the system and everytime thereafter when it is modified.

PSL DATE:

Date of the PSL management report from which this CPCG data is extracted.

LIBRARY LEVEL:

Library level from which the CPCG size data is extracted. Normally set equal to "ALL".

DATA SOURCE:

PSL Management Report from which the size data is extracted. Set to:

"PRG" if from Summary by Programs
or "SEG" if from Summary by Segments
Normally, this parameter is set to "PRG".

SOFTWARE IDENTIFICATION:

Give the CPCI name and CPCG name of the program as listed in the PSL.

SPECIAL ENVIRONMENTAL FACTORS OF THIS COMPONENT:

Answer "Y" or "N" to those environmental factors which apply to this program.

GENERAL PROGRAM INFORMATION:

Number of CPCs: The total number of programs contained in this CPCG.

Number of Segments: The total number of INCLUDED Segments contained in this CPCG.

NOTE: Do not count CPCs or Segments at a higher library level if they are currently at a lower level.

Number of Source Lines: The total number of source statements, including comments.

Number of Machine Words: The total number of words of object code into which the source code compiles.

CPCG DESCRIPTION FORM

PSL Date: _____

Library Level: _____

Data Source: _____

Software Identification:

CPCI _____ CPCG _____

Special Environmental Factors of the Component:

- | | |
|---|--|
| a) Special Display _____ | h) Concurrent Development of ADP Hardware _____ |
| b) Detailed Operational Requirements Definition _____ | i) Time Sharing (vs Batch) _____ |
| c) Change to Operational Requirements _____ | j) Development Using Separate Facility _____ |
| d) Real-Time Operation _____ | k) Development on Operational site _____ |
| e) CPU Memory Constraint _____ | l) Development on other than Target System _____ |
| f) CPU Time Constraint _____ | m) Development at more than one site _____ |
| g) First S/W Developed on CPU _____ | n) Programmer Access to Computer _____ |

General Program Information:

Number of CPCs _____

Number of Segments _____

Number of Source Lines _____

Number of Machine Words _____

A-9

CPCG DESCRIPTION CODING FORM

SOFTWARE IDENTIFICATION

CPCI

--	--	--	--	--

1-4

CPCG

--	--	--	--	--	--	--	--

5-13

SPECIAL ENVIRONMENTAL FACTORS

SPECIAL DISPLAY

--

14

DETAILED OPERATIONAL REQUIREMENTS DEFINITION

--

15

CHANGE TO OPERATIONAL REQUIREMENTS

--

16

REAL TIME OPERATION

--

17

CPU MEMORY CONSTRAINT

--

18

CPU TIME CONSTRAINT

--

19

FIRST S/W DEVELOPED ON CPU

--

20

CONCURRENT DEVELOPMENT OF ADP HARDWARE

--

21

TIME SHARING (vs BATCH)

--

22

DEVELOPER USING SEPARATE FACILITY

--

23

DEVELOPMENT ON OPERATIONAL SITE

--

24

DEVELOPMENT ON OTHER THAN TARGET SYSTEM

--

25

PROGRAMMER ACCESS TO COMPUTER

--

26

PROGRAM SUPPORT LIBRARY DATA

PSL DATE

LEVEL

DATA SOURCE

NUMBER OF
PROGRAMS

NUMBER OF
SEGMENTS

SOURCE SIZE

OBJECT SIZE

--	--	--	--	--

27-32

--	--	--	--

33-35

--	--	--

36-38

--	--	--	--

39-41

--	--	--	--	--

42-45

--	--	--	--	--	--

46-51

--	--	--	--	--	--	--

52-57

CPCG STATUS CODING FORM

CPCG 1-4				PSL DATE 5-10																							
CODE PROGRESSION																											
PRG	VN	CPT	VN	INT	VN	FIX	VN	TST	VN	FRZ	VN	DEL	VN														
11-17		18-24		25-31		32-38		39-45		46-52		53-59															
CODE DURABILITY																											
PRG 60-66				CPT 67-73				INT 74-80				FIX 81-87				TST 88-94				FTZ 95-101				DEL 102-108			
109-114				115-120				121-126				127-132				133-138				139-144				145-150			

INSTRUCTIONS FOR COMPLETING THE
CPCG MAINTENANCE ACTIVITY FORM

This form is used to keep track of the time expenditures by PAVE PAWS personnel from the time a specific maintenance activity is initiated, to its completion. A specific maintenance activity is defined as: 1) the correction of a detected error, 2) the addition of a new function or capability to the PAVE PAWS system, 3) the deletion of an existing function or capability of the PAVE PAWS system, or 4) the optimization or enhancement of an existing PAVE PAWS capability or function.

This form is to be completed each time maintenance is performed on a CPCG. This form is initiated when the Change Review Board (CRB) minutes indicate that a Discrepancy Report has been submitted and a Memorandum of Recommended Action has been approved by the CRB. It is completed when a new Software Version Release containing the associated documentation has been received.

DR NUMBER: Discrepancy Report sequence number.

DATE: Date Discrepancy Report was submitted.

LOCATION OF ACTIVITY: Give the CPCI name and CPCG name of the program being modified. A MAF must be completed for each distinct CPCG being modified, even if all are under the same Discrepancy Report.

MAINTENANCE TYPE: Number the reasons for maintenance (1=primary, 2=secondary) if there is more than one reason for the change. Give a brief description of the change.

SPECIFICATION PRECISION: Check the precision of the design specifications produced for the change.

SIZE: Give the lines of source code and object code size of the code necessary to implement the change if known.

URGENCY: A user assessment of the urgency at some particular phase of the maintenance activity is requested.

The assessment is a rating of E, U, or R where urgency is defined as:

- E. Most Urgent - system critical, respond as rapidly as possible;
- U. Urgent - higher priority than the average maintenance cycle;
- R. Least Urgent - perform in the average maintenance cycle.

DIFFICULTY OR COMPLEXITY: Requires an assessment by the evaluator of the degree of difficulty in performing some operation in the maintenance activity. The appropriate level of complexity should be checked.

- 1. Very Difficult or Complex - impacts more than one CPCI;
- 2. Difficult or Complex - impacts more than one CPCG;
- 3. Medium Difficulty or Complexity - impacts more than one CPC in a CPCG;
- 4. Simple-minor modifications to several CPCs;
- 5. Very Simple-minor modification to a single CPC.

SECTION A: Complete this section if a change is being made for a reason other than new requirements.

MEANS OF INITIAL DETECTION: Check the appropriate means by which the error was initially detected. Enter 1 for the primary reason. If there was a secondary reason, indicate by entering 2. Up to two may be marked.

EFFORT IN DIAGNOSING THE ERROR: Number of Runs to Diagnose: Give the number of computer runs used in correcting the error.

ELAPSED COMPUTER TIME: Give the number of hours of computer time used in correcting the error.

WORKING TIME TO DIAGNOSE: Give the total time involved in determining change to be made.

PROBABLE ERROR SOURCE: Enter 1, 2 or 3 for those items which most likely explain the source of the error. Enter 1 for the primary source. If there was a secondary source, indicate by entering 2. If there was a third reason, indicate by entering 3. Up to three sources may be marked.

SECTION B: Complete when the maintenance activity involved making a change to the software.

SOFTWARE CHANGE REQUIRED:

New Requirements: If the change is a result of new requirements, check those appropriate new requirement categories. Up to three may be marked. Indicate primary, secondary, and tertiary by entering 1, 2 or 3, respectively.

Nature of Change: Check the appropriate nature of the change involved. Up to three may be marked. Indicate order by entering 1, 2 or 3.

SECTION C: To be completed on the form.

For each stage of the maintenance activity give the following information:

Date: Give the date each stage was started and completed.

Personnel Hours: Give the amount of time in each personnel category that was devoted to the maintenance activity in tenths of hours. For some stages of the maintenance activity some personnel categories may not have devoted time to the maintenance activity in which case time should be recorded as zero.

CPU Time: Give the amount of computer time needed to complete each stage of maintenance.

Personnel ID: Identify the individuals performing the activity in each stage.

SECTION D: Flip the MAF over and list the names of the segments changed and the number of source lines of code added, deleted or modified in each segment.

DR # _____ COMPONENT MAINTENANCE ACTIVITY FORM DATE _____

Location of Activity CPCI _____ CPCG _____ CPC/Segment _____

Maintenance Type: Error Correction _____ Add Capability _____ Version Release # _____

Delete Capability _____ Optimize/Enhance _____

Brief Description of Change to be made _____

Specification Precision: Very Precise _____ Precise _____ Imprecise _____

Size of Change: Source Code Lines _____ Object Code Instructions _____

Urgency of Maintenance Activity (1-3) _____

Complexity of Maintenance Activity: Very Complex _____ Complex _____ Medium _____ Simple _____

Very Simple _____

SECTION A Complete if Maintenance Activity was for Error Correction

MEANS OF INITIAL DETECTION - / only for corrections (not New Reqs.)

- More than one category may be /'ed

- _____ a. Hand Processing _____ d. Interrupt Error (Code _____) g. Error Message _____
- _____ b. Personal Communication _____ e. Incorrect Output or Result h. Code Review _____
- _____ c. Infinite Loop _____ f. Missing Output i. Documentation Review _____
- _____ k. Maintenance crosscheck (as a result of a change in other software) j. Special Debug Code _____
- _____ l. Other, Describe _____

EFFORT IN DIAGNOSING THE ERROR - Do not include effort spent in initial detection

- a. No. of Runs to Diagnose _____ Elapsed Computer Time (Minutes) _____ Received _____
- b. Working Time to Diagnose: Days _____ Hours _____ Analysis Begins _____
- c. No. of Lines of Code: Added _____ Deleted _____ Corrected _____ Project Opens _____

ERROR SOURCE

- _____ a. Misinterpretation of Spec. _____ e. Specified Interface Not Implemented Correctly _____ j. Deck Setup Error _____ o. Operator Error
- _____ b. Incorrect Spec. _____ f. Software Interface _____ k. Computational Error _____ p. Due to Prior Modification
- _____ c. Incomplete Spec. _____ g. Hardware Interface _____ l. Data I/O Error _____ q. Cause Not Found, Workaround Used
- _____ d. Specified Function Not Implemented Correctly _____ h. Operating System _____ m. Logic Error _____ r. Other, Explain
- _____ i. Support Software _____ n. Data Definition Error _____

SECTION B Complete if Maintenance Activity was to make a change.

NATURE OF CHANGE

- a. Documentation (Preface or Comments) _____ d. Structural _____ e. Algorithmic _____
- b. Fix Instruction _____ f. Other, Explain _____
- c. Change Constants _____

NEW REQUIREMENT - / those which apply

- a. Mission _____ d. Hardware _____
- b. Engineering Model _____ e. Other, Explain _____
- c. Software Implementation _____

ERROR CORRECTION / IDE BY PROGRAM ABORTATION? YES _____ NO _____

SECTION C Maintenance Effort Required for Change or Correction

STAGE	DATE		*PERSONNEL HOURS			CPU TIME	PERSONNEL ID
	Received	Forwarded	Management	Analyst	Programmer	Clerical	
Design Effort							
Coding Effort							
Unit Testing							
Integration Testing							
Testing Revis							
Installation							

* Record Hours to nearest tenth of an hour

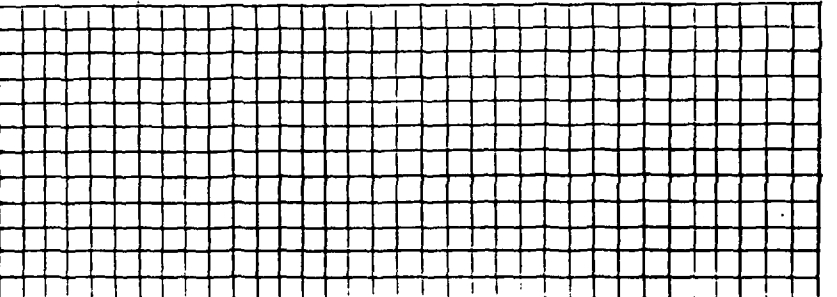
1-5	OR #					DATE						6-11
	CPCI					12-15						
	CPCG					16-19						
	CPC/SEGMENT					20-24	(LEAVE BLANK IF MORE THAN ONE CPC)					

25	MAINTENANCE TYPE			26		VERSION RELEASE			29-30
27				28					
	SPECIFICATION PRECISION			31-32					
	URGENCY OF MAINTENANCE			33		INTEGERS - RIGHT JUSTIFIED			
	COMPLEXITY OF MAINTENANCE			34-35		ALPHABETIC - LEFT JUSTIFIED			
	MEANS OF INITIAL DETECTION			36-37					
				38-39					
	PROGRAMMER ASSIGNED							40-46	
	EFFORT TO CHANGE								
				47-50				51-54	
				MANHOURS				COMPUTER HOURS TENTHS	
	NO. OF FILES AFFECTED			55-57					
	ERROR SOURCE			58-59		NATURE OF CHANGE		NEW RE- QUIREMENT	
				60-61					70-71
				62-63					72-73
									74-75

ADDED
DELETED
CHANGED 6

DR #1-5

FILE NAME AFFECTED 7-80

A large rectangular grid of small squares, likely a placeholder for a drawing or a data table. The grid is composed of approximately 20 columns and 25 rows of squares. It is located in the center of the page, below the header information.

APPENDIX B

PAVE PAWS MAINTENANCE DATABASE

APPENDIX B

PAVE PAWS Maintenance Database

1. INTRODUCTION

The purpose of this Appendix is to describe the contents of the PAVE PAWS Maintenance Database. The database consists of the following sequential files:

PPPEP	Programmer Experience Profile File
PPDRH	Discrepancy Report History File
PPMAF	Maintenance Activity File
PPCDF	CPCG Description File
PPCSF	CPCG Status File
PPPCH	CPC Change History File
PPSCH	Segment Change History File

2. Programmer Experience Profile File (PPPEP)

This file is used to describe the experience of the members of the PAVE PAWS maintenance staff. Each experience profile is recorded in a set of three (3) records as depicted in Figure B-1. The input source is the PEP form. Major categories of experience include:

- Age and Rank
- Formal Education and Major Course of Study
- Computer Science Education
- Data Processing Experience
- Modern Programming Practices Experience
- Programming Languages Experience
- Computer System Experience
- Applications Experience

The following remarks apply to the data items in this file:

- (1) The Personal ID Code is formed from the first initial followed by the first six characters of the person's last name (Note 1).
- (2) Years of experience is recorded to the nearest whole year.
- (3) Experiences not applicable are recorded as blanks in the appropriate columns of the record.
- (4) Some data are not critical and may be recorded on a voluntary basis. Data of this type includes age, project name and job title.

This file should be updated periodically as new members are added to the PAVE PAWS maintenance staff. In most cases, a personal telephone call to the individual should be all that is needed to allow the individual responsible for maintaining the PAVE PAWS Software Maintenance Database to complete a PEP coding form.

The data contained in this file can be utilized to compare the effort and time required to resolve a Discrepancy Report with

the experience of the maintenance personnel with regard to formal education, on the job training, experience with specific MPPs, languages and computer systems and specific types of applications experience.

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
<u>RECORD 1</u>			
1- 7	Personal ID Code	A7	Note 1
8-27	Name of Individual	A20	
28-29	Age		YYMMDD
30-35	Date Profile Completed	I6	
36-50	Project Name	A15	
51-70	Job Title	A20	
71-75	Position (GS12, E-6, O-1)	A4	
76-78	Section Name (ANL, ADQ, ADT)	A3	
79	Years of College Education	I1	
80	Years of High School Education	I1	
<u>RECORD 2</u>			
1- 2	Year Graduated (High School if no College)	I2	Blank if not needed
3- 5	First College Degree (BS, AAS)	A3	
6- 7	Degree Year (Last Two Digits)	I2	
8-11	Major Course of Study	A4	Blank if not needed
12-14	Second College Degree (MS, MEE)	A3	
15-16	Degree Year (Last Two Digits)	I2	

FIGURE B-1 PROGRAMMER EXPERIENCE PROFILE FILE (PPPEP)

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
<u>RECORD 2 CONTINUED</u>			
17-20	Major Course of Study	A4	
21-23	Third College Degree (PHD, MAT)	A3	Blank if not needed
24-25	Degree Year (Last Two Digits)	I2	
26-29	Major Course of Study	A4	
30-31	Number Computer Science Courses	I2	
32-33	Number Computer Science Semester Hours	I2	
34-35	Number Computer Science Seminars	I2	
36-37	Number Years With Computers	I2	
38-39	Percent of Years Individual Effort	I2	
40-41	Percent of Years Team Effort	I2	
42-43	Percent of Years Supervisor	I2	
44-45	Number Years with Struc- tured Program	I2	Round frac- tion of years to nearest year in the following fields
46-47	Number Years with Program Design Language	I2	
48-49	Number Years with HIPO	I2	
50-51	Number Years with Top- down Dev.	I2	
52-53	Number Years with Program Support Library	I2	
54-55	Number Years with Pre- compilers	I2	
56-57	Number Years with Chief Programming Team	I2	

FIGURE B-1 PROGRAMMER EXPERIENCE PROFILE FILE (PPPEP) CONT.

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
<u>RECORD 2 CONTINUED</u>			
58-59	Number Years with Other Techniques	I2	
60-61	Number Years with JOVIAL	I2	
62-63	Number Years with ASSEMBLER	I2	
64-65	Number Years with FORTRAN	I2	
66-67	Number Years with COBOL	I2	
68-69	Number Years with ALGOL	I2	
70-71	Number Years with PL/1	I2	
72-73	Number Years with PASCAL	I2	
74-75	Number Years with Other Languages	I2	
<u>RECORD 3</u>			
1- 2	Number Years with Business Applications	I2	
3- 4	Number Years with Scientific/Math Applications	I2	
5- 6	Number Years with System Programming Applications	I2	
7- 8	Number Years with Real-time Applications	I2	
9-10	Number Years with Database Applications	I2	
11-12	Number Years with Other Applications	I2	
13-19	Name of Computer (Primary)	A7	CDC CYBER
20-26	Name of Operating System (Primary)	A7	NOS
27-28	Number of Years This System (Primary)	I2	
29-35	Name of Computer (Secondary)	A7	IBM 370
36-42	Name of Operating System (Secondary)	A7	OS

FIGURE B-1. PROGRAMMER EXPERIENCE PROFILE FILE (PPPEP) CONT.

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
<u>RECORD 3 CONTINUED</u>			
43-44	Number of Years this System (Secondary)	I2	
45-51	Name of Computer (Tertiary)	A7	H6180
52-58	Name of Operating System (Tertiary)	A7	MULTICS
59-60	Number of Years this System (Tertiary)	I2	

FIGURE B-1 PROGRAMMER EXPERIENCE PROFILE FILE (PPPEP) CONT.

3. Discrepancy Report History File (PPDRH)

This file is a copy of the PAVE PAWS Discrepancy Report Database (DRDB) developed and maintained by the SPA ADQ branch at Beale Air Force Base. Figure B-2 describes the data contained in this file.

The DRDB is updated by a set of support programs developed by ADQ whenever a DR is opened, submitted for analysis, closed, or its status otherwise changed.

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>SAMPLE DATA</u>
<u>RECORD 1</u>			
1- 5	Discrepancy Report Number Y = Last Digit of Year NNNN = Sequence Number	A5	10167
7- 9	Discrepancy Report Origin	A3	7th
11-14	Local DR Number	A4	B270
16-55	DR Description	A10	
57-63	Date DR Received	A7	100ct79
65-67	Responsible Section	A3	TAC
69	DR Priority	A1	R
<u>RECORD 2</u>			
1- 7	Date Analysis Started	A7	06Nov79
9-15	PMR/PCD/PDDR Open Date	A7	12Dec79
17-56	PMRYNNNN, PCDYYNNN, PDDRYNNN, PDDRYNNN, PMRYNNN/PDDRYNNN, PCDYYNNN/PDDRYNNN, or any remarks desired	A40	

FIGURE B-2 DISCREPANCY REPORT HISTORY FILE (PPDRH)

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>SAMPLE DATA</u>
<u>RECORD 3</u>			
1-15	Name of Programmer Assigned to the Project	A15	N. NAHE
17-23	Estimated Completion Date	A7	10Feb80
25-31	Scheduled Version Release	A7	PTAC-D0
37-39	Scheduled Operation Date	A7	13Apr80
41-47	Date Forwarded to NCCB	A7	18Dec79
49-51	CCB. Action (APP, REJ, DEF)	A3	
53-59	Date of NCCB Action	A7	31Dec79
61-67	Date Project Closed	A7	13Apr80

Note: APP Approved
 REJ Rejected
 DEF Deferred

FIGURE B-2 DISCREPANCY REPORT HISTORY FILE (PPDRH) CONT.

4. Maintenance Activity File (PPMAF)

This file summarizes the information provided periodically by the PAVE PAWS System Programming Agency concerning each Software Version Release (SVR). A set of records is defined for each software modification described in the SVR.

The source of the data provided in this file is the CMAF form and includes, for each software modification, the type of maintenance activity being performed, the redesign specifications precision for that activity, and the complexity of the maintenance activity. Also provided is data concerning how the error was detected, the effort involved in diagnosing it, the reason and nature for a change in the software, and the effort required to make the change or correction in the software. This data was summarized from the following Air Force forms:

ADCOM form 103 DR	-	Discrepancy Report
ADCOM form 542 PMR	-	Program Modification Request
ADCOM form 544 MDIS	-	Modification Design and Interface Specifications
ADCOM form 547 PDDR	-	Program Documentation Discrepancy Report
ADCOM form 549 PCD	-	Program Change Document

Additional maintenance data related to each problem report was contained on:

MRA	-	Memo for Recommended Action
PC	-	Project Control Report

Accompanying each batch of discrepancy reports are:

- Version Release Request (ADCOM form 540)
- Version Description Document
- DT & E Version Test Report
- Recommended OT & E Procedures

AD-A119 081

IIY RESEARCH INST ROME NY
MPP IN THE PAVE PAWS SOFTWARE MAINTENANCE ENVIRONMENT. (U)
JUN 82

F/G 9/2

UNCLASSIFIED

RADC-TR-82-169

F30602-80-C-0223

ML

20 2

10 10



The PPMF file consists of a variable number of two record types: The first record type contains, for each Discrepancy Report, all previously mentioned data with the exception of the names of the programs and segments which were affected by the modification. The names of these programs and segments are described in record type 2, one record for each program or segment.

The PPMF file format is depicted in Figure B-3. A set of records is defined for each unique Discrepancy Report/CPCG combination. If a Discrepancy Report references more than one CPCG, another Maintenance Activity form must be completed, resulting in an additional set of records in the PPMF file for the same Discrepancy Report number.

This file should be updated periodically as documentation on new PAWS Software Version Releases is received.

The record format for the PPMF file refers to eleven (11) notes in the comments column of Figure B-3. Explanations of these notes are as follows:

Note 1: Maintenance Type - Enter up to two codes which describe the type of maintenance being performed. The primary reason code is suffixed with a one (1) and the secondary reason code is suffixed with a two (2). The list of possible codes is:

- E - Error Correction
- A - Add Capability
- D - Delete Capability
- O - Optimize/Enhance

Example:

Columns 25 - 26 E1
Columns 27 - 28 A2 or ~~DD~~ if not needed

Note 2: Precision of Change Specification - Enter the code which best describes how detailed the documentation (MRA, PMR, PCD, MDIS, PDDR) describes the change to be performed. The list of possible codes is:

VP - Very Precise
PØ - Precise
IM - Imprecise

Example:

Columns 31 - 32 PØ

Note 3: Urgency of Maintenance Activity - Enter the code which best describes the priority associated with the resolution of this change request. These codes are usually found on the DR or MRA forms. They are always found in the machine readable DR Database (Discrepancy Report History File PPDRH) received periodically from the SPA. The list of possible codes is:

E - Emergency
U - Urgent
R - Routine

Example:

Column 33 - U

Note 4: Complexity of Change Activity - Enter the code which best describes the difficulty of making the software modification. The list of possible codes is:

VC - Very Complex
CØ - Complex
MØ - Medium
SØ - Simple
VS - Very Simple

Example:

Columns 34 - 35 VS

Note 5: Means of Initial Detection - Enter up to two codes which most accurately describe how the problem which initiated this change exhibited itself. In this case, the distinction is made between primary and secondary reason codes by the order which they are defined. The list of possible codes is:

- HP - Hand Processing
- PC - Personal Communication
- IL - Infinite Loop
- MC - Maintenance Crosscheck due to making an unrelated software change
- IE - Interrupt Error
- IO - Incorrect Output or Result
- MO - Missing Output
- EM - Error Message
- CR - Code Review
- DR - Documentation
- SD - Special Debug Code
- OT - Other

Example:

Columns 36 - 37 MO
Columns 38 - 39 MC or ~~MO~~ if not needed

Note 6: Programmer ID Code - Enter the code associated with the programmer who performed the major portion of the change. The code is constructed by concatenation of the first six characters of the surname to the first name initial.

Example:

Given the name Mary Smither
Columns 40 - 46 MSMITHE

Note 7: Number of Segments Affected - Enter the number which indicates how many programs and segments were changed, added and deleted from the CPCG undergoing change. If this is a documentation change only, enter zero (0). This number determines how many type 2 records follow record type 1.

Note 8: Source of Error - If the reason for change was due to the detection of an error, enter the appropriate code which corresponds to the probable error source. Up to three (3) error sources may be entered. The order they are entered determines whether the source of the error is primary, secondary or tertiary. If no error occurred, enter blanks in columns 58-63. The list of possible codes is:

- MS - Misinterpretation of Specifications
- IS - Incorrect Specifications
- NS - Incomplete Specifications
- SF - Specified Function not Implemented Correctly
- SI - Specified Interface not Implemented Correctly
- SO - Software Interface to another Program
- HI - Hardware-Software Interface
- OS - Operating System
- LE - Logic Error
- CE - Computational Error
- DE - Data I/O Error
- DD - Data Definition Error
- CN - Cause Not Found - Workaround Used
- IO - I/O Software
- PM - Due to Prior Modification
- SS - Support Software
- DS - Deck Setup Error
- OE - Operator Error
- OT - Other

Example:

Columns 58 - 59 IO
 Columns 60 - 61 SI
 Columns 62 - 63 (not needed)

Note 9: Nature of Change - If the software documentation and/or source code was modified due to the correction, addition, deletion or enhancement of a function, enter the type of change made to the system in columns 64 - 69. Otherwise, enter blanks. The list of possible codes is:

DO - Documentation
FI - Fix Instruction
CC - Change Constants
ST - Structural Change
AL - Algorithmic
OT - Other

Example:

Columns 64 - 65 AL
Columns 66 - 67 DO
Columns 68 - 69 ~~XX~~ if not needed

Note 10: Type of New Requirement - If the source code or documentation was not in error but was modified to delete or add a capability or optimize performance, enter the new requirements code from the list below in columns 70 - 75. If the change was not due to a new requirement, enter blanks. The list of possible new requirements codes is:

MI - Mission Changed
EM - New Engineering Model
SW - More Efficient Software being Implemented
HW - New Hardware being Added or Old Hardware being Removed
SS - New Support Software being Implemented
OT - Other

Example:

Columns 70 - 71 MI
Columns 72 - 73 ~~XX~~ if not needed
Columns 74 - 75 ~~XX~~ if not needed

Note 11: Name of Segment Affected - This entry has three possible formats, depending upon whether 1) only one CPC was affected; 2) more than one CPC was affected by the change; or 3) the CPC name is longer than 5 characters.

One CPC Affected - When only one CPC is affected by the change, columns 20 - 24 should contain the name of the CPC. The normal longname format of a program or segment may contain up to 40 characters. An example is the following for CPCG = TGDB and CPC = LOAD:

"TGDB.LOAD.SPARE.SPACE.FILE"

Since the first two strings are already identified, enter "SPARE.SPACE.FILE" in columns 7 - 22 of record type 2.

CPC Name Longer than Five Characters - When the CPC name is longer than five characters, enter blanks in columns 20 - 24. The CPC name is included in the string which starts in column 7 of the record type 2 associated with the segment. For example, given CPC = LOADER in the segment "TGDB.LOADER.COMPOOL.DATABASE", leave columns 20 - 24 of record type 1 blank and enter:

LOADER.COMPOOL.DATABASE in columns 7 - 29 of record type 2.

More than One CPC Affected - When more than one CPC is affected by a software modification request, columns 20 - 24 are left blank and the CPC name is included in the string which starts in column 7 of record type 2. An example of this case is when two CPC's are modified due to one Deficiency Report. Suppose the following segments are affected: "TGDB.LOAD.SPARE.SPACE.FILE" and "TGDB.LOADER.COMPOOL.DATABASE". The CPCG name is already defined in columns 16 - 19. The integer 2 will be entered in columns 55 - 57, right justified. Two type 2 records are required:

Example:

- #1 - columns 7 - 27 LOAD.SPARE.SPACE.FILE
- #2 - columns 7 - 29 LOADER.COMPOOL.DATABASE

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
<u>RECORD 1</u>			
1- 5	Discrepancy Report Number	I5	
6-11	Date DR Submitted	I6	YYMMDD
12-15	CPCI Affected	A4	
16-19	CPCG Affected	A4	One Record for each CPCG
20-24	CPC Affected	A5	Leave blank if more than one
25-26	Maintenance Type (Primary)	A2	Note 1
27-28	Maintenance Type (Secondary)	A2	Blank if not needed
29-30	Version Release Affected	A2	
31-32	Precision of Change Specification	A2	Note 2
33	Urgency Code	A1	Note 3
34-35	Complexity of Change	A2	Note 4
36-37	Means of Initial Detection (Primary)	A2	Note 5
38-39	Means of Initial Detection (Secondary)	A2	Blank if not needed
40-46	Programmer ID Code	A7	Note 6
47-50	Manhours to Change	I4	
51-54	Tenths of Computer Hours	I4	
55-57	Number of Segments Affected	I3	Note 7
58-59	Source of Error (Primary)	A2	Note 8
60-61	Source of Error (Secondary)	A2	Blank if not needed
62-63	Source of Error (Tertiary)	A2	
64-65	Nature of Change (Primary)	A2	Note 9
66-67	Nature of Change (Secondary)	A2	Blank if not needed
68-69	Nature of Change (Tertiary)	A2	

FIGURE B-3 MAINTENANCE ACTIVITY FILE (PPMAF)

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
<u>RECORD 1 CONTINUED</u>			
70-71	Type of New Requirement (Primary)	A2	Note 10
72-73	Type of New Requirement (Secondary)	A2	Blank if not needed
74-75	Type of New Requirement (Tertiary)	A2	
<u>RECORDS 2 THRU N+1:</u> where N = # Segments Affected			
1- 5	Discrepancy Report Number	I5	
6	Segment Added (A), Deleted (D) or Changed (C)	I1	
7-46	Name of Segment Affected	A35	Note 11

FIGURE B-3 MAINTENANCE ACTIVITY FILE (PPMAF) CONT.

5. CPCG Description File (PPCDF)

This file is used to describe the physical characteristics of each Computer Program Configuration Group (CPCG).

This file consists of a variable number of 57 character records; one record for each CPCG/PSL date combination. It should be updated periodically so it can be used for CPCG/CPCG growth studies.

The PPCDF record format is depicted in Figure B-4. The name and development environment data in columns 1 - 26 should remain constant between CPCG updates. However, columns 27 - 57 may vary between PSL Management Report Dates. The data in this file defines the physical characteristics of each CPCG at a specific point in time given by the PSL Management Report Date (record fields 27 - 32).

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
1- 4	CPCI Name	A4	
5-13	CPCG Name	A9	Key is first 4 Characters
14	Special Display	A1	Y or N
15	Detailed Requirements Definition	A1	Y or N
16	Change to Operational Requirement	A1	Y or N
17	Real Time Operation	A1	Y or N
18	CPU Memory Constraint	A1	Y or N
19	CPU Time Constraint	A1	Y or N
20	First Software Developed on CPU	A1	Y or N
21	Developed Concurrently with Hardware	A1	Y or N

FIGURE B-4 CPCG DESCRIPTION FILE (PPCDF)

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
22	Time Sharing (Vs Batch)	A1	Y or N
23	Developer Used Separate Facility	A1	Y or N
24	Developed on Operational Site	A1	Y or N
25	Developed on Other than Target System	A1	Y or N
26	Programmer had Direct Access to Computer	A1	Y or N
27-32	PSL Management Report Date	I6	YYMMDD
33-35	PSL Library Level	A3	
36-38	Source of Data	A3	Summary by SEG or PRG
39-41	Number of Programs	I3	
42-45	Number of Segments	I4	
46-51	Number of Source Lines	I6	
52-57	Number of Object Words	I6	

FIGURE B-4 CPG DESCRIPTION FILE (PPCDF) CONT.

6. CPCG Status File (PPCSF)

This file is used to describe the status of each Computer Program Configuration Group (CPCG).

This file consists of a variable number of 150 character records; one record for each CPCG/PSL date combination. It should be updated periodically so it can be used to relate CPCI/CPCG status, date of change and growth rate to the occurrence of maintenance change activity and distribution of efforts.

The PPCSF record format is depicted in Figure B-5. The data contained in each record is obtained from the following Program Support Library Management Reports: Code Programming Durability Report, the Summary by Programs Report and the Summary by Segments Report.

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
1- 4	CPCG Name	A4	
5- 10	PSL Management Report Date	I6	
11- 15	Number of Lines Effective Code at PRG Level	I5	
16- 17	Highest VN at this Level	A2	
18- 22	Number of Lines Effective Code at CPT Level	I5	
23- 23	Highest VN at this Level	A2	
25- 29	Number of Lines Effective Code at INT Level	I5	
30- 31	Highest VN at this Level	A2	
32- 36	Number of Lines Effective Code at FIX Level	I5	
37- 38	Highest VN at this Level	A2	
39- 43	Number of Lines Effective Code at TST Level	I5	

FIGURE B-5 CPCG STATUS FILE (PPCSF)

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
44- 45	Highest VN at this Level	A2	
46- 50	Number of Lines Effective Code at FRZ Level	I5	
51- 52	Highest VN at this Level	A2	
53- 57	Number of Lines Effective Code at DEL Level	I5	
58- 59	Highest VN at this Level	A2	
60- 64	Number of Lines Durable Code at PRG Level	I5	
65- 66	Highest VN at this Level	A2	
67- 71	Number of Lines Durable Code at CPT Level	I5	
72- 73	Highest VN at this Level	A2	
74- 78	Number of Lines Durable Code at INT Level	I5	
79- 80	Highest VN at this Level	A2	
81- 85	Number of Lines Durable Code at FIX Level	I5	
86- 87	Highest VN at this Level	A2	
88- 92	Number of Lines Durable Code at TST Level	I5	
93- 94	Highest VN at this Level	A2	
95- 99	Number of Lines Durable Code at FRZ Level	I5	
100-101	Highest VN at this Level	A2	
102-106	Number of Lines Durable Code at DEL Level	I5	
107-108	Highest VN at this Level	A2	
109-114	Last Change at PRG Level	I6	YYMMDD
115-120	Last Change at CPT Level	I6	YYMMDD

FIGURE B-5 CPCG STATUS FILE (PPCSF) CONT.

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
121-126	Last Change at INT Level	I6	YYMMDD
127-132	Last Change at FIX Level	I6	YYMMDD
133-138	Last Change at TST Level	I6	YYMMDD
139-144	Last Change at FRZ Level	I6	YYMMDD
149-150	Last Change at DEL Level	I6	YYMMDD

Note 1: VN Version Release Number

Note 2: Within the PSL, seven hierarchical library levels are defined. Starting with the highest level in the PSL, these levels include:

- DEL - software which is in the field
- FRZ - software which has been qualified
- TST - software undergoing qualification test
- FIX - software corrections for TST level
- INT - software undergoing integration test
- CPT - software undergoing group test
- PRG - software under development/unit test.

FIGURE B-5 CPCG STATUS FILE (PPCSF) CONT.

7. Program Change History File (PPCH)

This file contains a list of CPCs (programs) which have been created or changed since the last software version release.

Figure B-6 depicts the format of this file.

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
1- 40	Program Longname	A40	
42- 47	Program Shortname	A6	
49- 52	Language	A4	
55- 61	Date Program Last Changed	A8	YY/MM/DD
64- 71	Time Program Last Changed	A8	HH.MM.SS
72- 76	Number of Segments in Program	I5	
77- 82	Total Size (Including all Segments)	I6	
83- 86	Number of Stubs	I4	
88- 89	Program Version (Maximum of All Segment Versions)	A2	
90- 93	Program Edition (Sum of All Segment Editions)	I4	
94- 98	Program Instance (Incremental For Each Compile)	I5	
101-108	Date Compiled	A8	YY/MM/DD
110-117	Time Compiled	A8	HH.MM.SS
118-123	Object Size (Decimal Words)	I6	

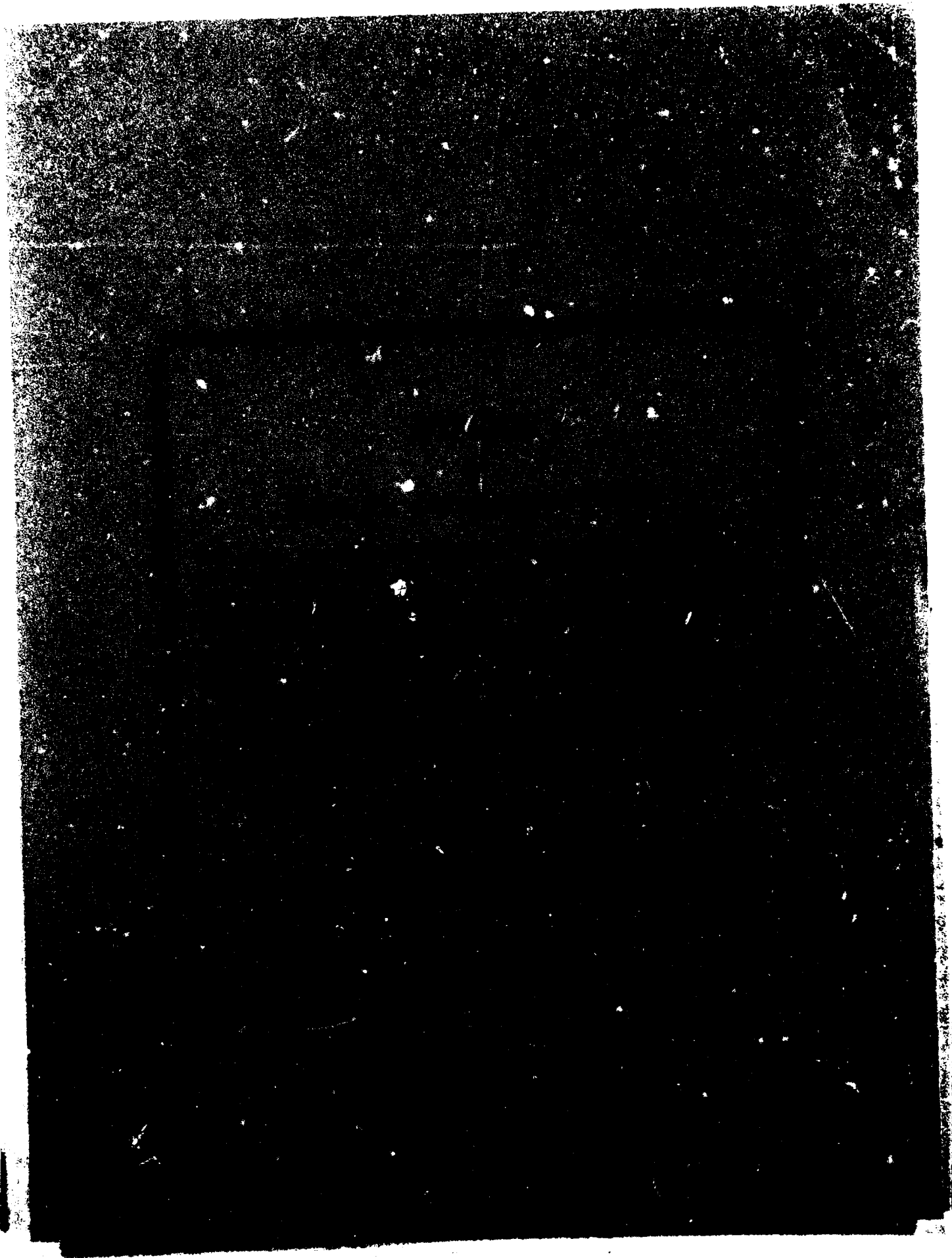
FIGURE B-6 PROGRAM CHANGE HISTORY FILE (PPCH)

8. Segment Change History File (PPSCH)

This file contains a list of INCLUDEed segments which have been created or changed since the last software version release. Figure B-7 depicts the format of this file.

<u>LOCATION</u>	<u>DESCRIPTION</u>	<u>FORMAT</u>	<u>COMMENTS</u>
1- 40	Segment Longname	A40	
42- 47	Segment Shortname	A6	
49- 52	Language	A4	
54- 57	Segment Type	A4	
59- 66	Date Segment was Created	A8	YY/MM/DD
67- 70	Current # Lines in Segment	I4	
71- 74	Gross Size Including Lines Deleted	I4	
76- 83	Date Segment Last Changed	A8	YY/MM/DD
85- 92	Time Segment Last Changed	A8	HH.MM.SS
94- 95	Segment Version	A2	
96- 99	Segment Edition	I4	
100-103	Total Number of Times Segment has Been Changed	I4	
104-107	Number of Changes Made to Current Version	I4	
108-111	Number of Lines (Gross) for Current Version	I4	
113-118	ID of Creator	A6	
120	Special Flag	A1	See p. 25 RADC-TR-79-137
128-130	ID of Person who Last Changed the Segment	A3	

FIGURE B-7 SEGMENT CHANGE HISTORY FILE (PPSCH)



ATE
MED
8