

AD-A082 713

PLANNED SYSTEMS INTERNATIONAL INC CAMBRIDGE MA
DMA MODERN PROGRAMMING ENVIRONMENT STUDY.(U)
JAN 80 L STUCKI, J BROWN, L HAMMOND

F/S 9/2

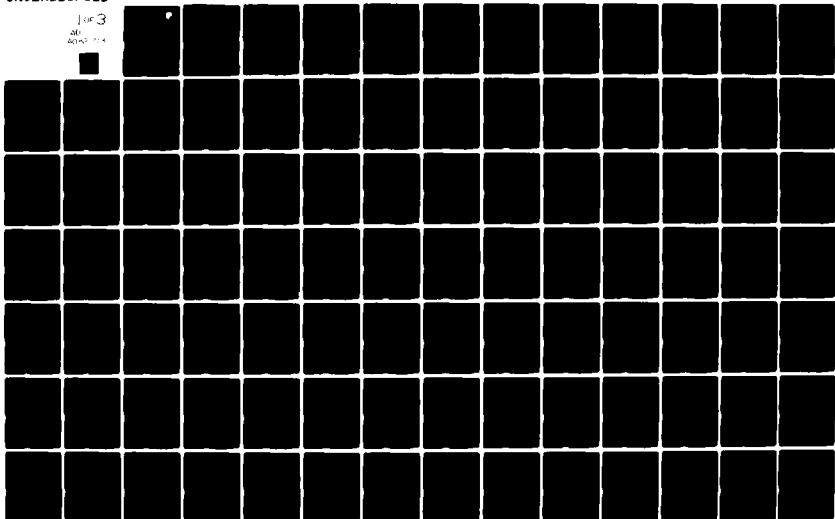
F30602-79-C-0022

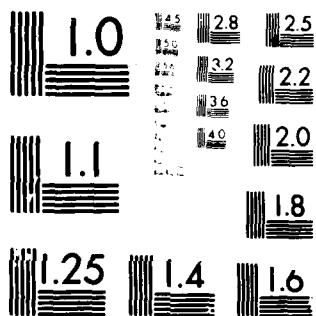
UNCLASSIFIED

RADC -TR-79-343

NL

1 of 3
AD
A082 713





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA 082713

RADC-TR-79-343
Final Technical Report
January 1980

DMA MODERN PROGRAMMING ENVIRONMENT STUDY

Planning Systems International Incorporated

Leon Stucki
John Brown
Linda Hammond

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC
ELECTE
APR 7 1980
S A

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

DDC FILE COPY

80 4 7 042

12
p.s.

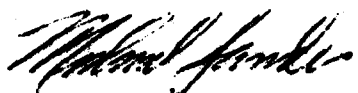
LEVEL



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

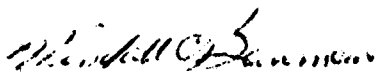
RADC-TR-79-343 has been reviewed and is approved for publication.

APPROVED:



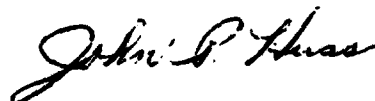
MICHAEL LANDES
Project Engineer

APPROVED:



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

FOR THE COMMANDER:



JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISIE), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 18 RADC-TR-79-343 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER 9
4. TITLE (and Subtitle) 6 DMA MODERN PROGRAMMING ENVIRONMENT STUDY.	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report Nov 78 - May 79	
7. AUTHOR(s) 10 Leon Stucki John Brown Linda Hammond	6. PERFORMING ORG. REPORT NUMBER N/A	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Planning Systems International, Inc. ✓ 238 Main Street Cambridge MA 01242	8. CONTRACT OR GRANT NUMBER(s) 15 F30602-79-C-0022 New	
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISIE) Griffiss AFB NY 13441	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 16 64701B 43050306 17 03	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same	12. REPORT DATE 11 Jan 1980 12/25/81	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.	13. NUMBER OF PAGES 259	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same	15. SECURITY CLASS. (of this report) UNCLASSIFIED	
18. SUPPLEMENTARY NOTES RADC Project Engineer: Michael Landes (ISIE) Subcontractor: Boeing Computer Services Company Seattle WA 98124	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Software engineering programming environment		
ABSTRACT (Continue on reverse side if necessary and identify by block number) This study represents the first formal, rigorous approach to modernizing the DMA programming environment. This study produced a plan for upgrading the current software production practices at all DMA installations using modern software engineering tools and procedures.		

DD FORM 1473
1 JAN 73

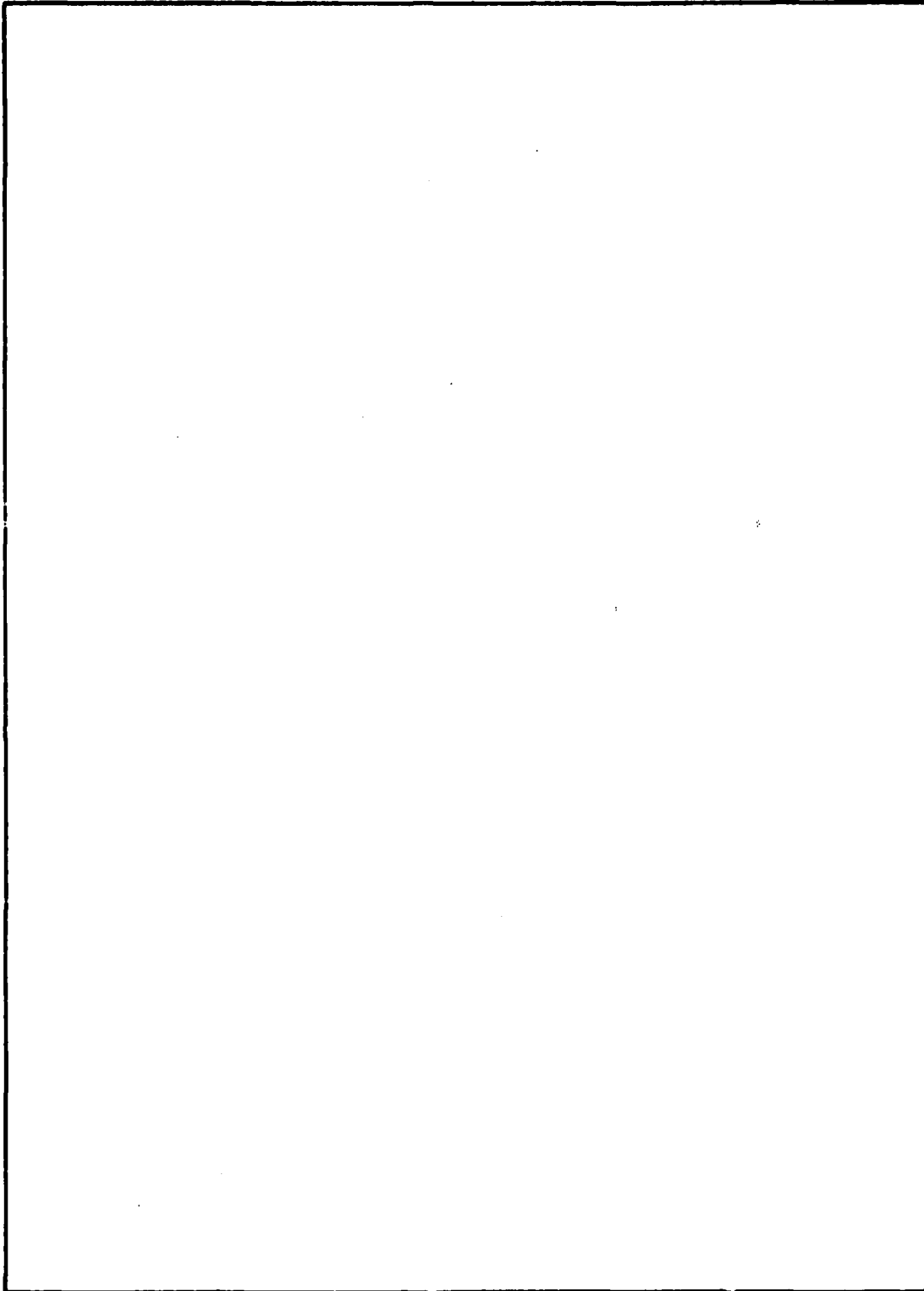
UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

411 6-7 509

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

CONTENTS

		<u>Page</u>
1.0	OVERVIEW	1
1.1	Problem	1
1.2	Purpose	2
1.3	Scope	2
1.4	Objectives and Outcomes	2
1.5	Statement of Work	2
1.6	Study Approach	3
1.6.1	Generic Model	4
1.6.2	Data Collection	4
1.6.3	Application of the Model	4
1.6.4	Detailed Analysis of the Environments	4
1.6.5	Recommend Solutions	5
1.6.6	Recommend Additional Environments	5
1.6.7	Develop Phased Implementation Plan	5
1.7	Study Approach/Statement of Work	7
1.8	Report Organization	9
2.0	ANALYTICAL FRAMEWORK FOR DMA PROGRAMMING ENVIRONMENT	10
2.1	General Approach	10
2.2	Major Components of the Model	12
2.2.1	Programming Environment	13
2.2.2	Applications/Project Environment	14
2.2.3	Center Administration and Management	16
2.2.4	SST and the Center Techniques Offices	17
2.2.5	DMA Customer Environment	17
2.2.6	DMA Headquarters STT	18
2.2.7	Applying the Model to the DMA Centers	19
3.0	THE TOPIC OF PRODUCTIVITY	23
3.1	General	23
3.2	Productivity Metrics	23
3.3	Productivity and Techniques and Tools	23
3.3.1	The BCS Study	24

Contents (Continued)

		<u>Page</u>
3.3.2	IBM Studies	25
3.3.3	Army Ballistic Missile Defense Program	25
3.3.4	The TRW Study	26
3.4	Productivity Summary	27
4.0	RECOMMENDATIONS	28
4.1	Summary Observations and Recommendations	28
4.1.1	Communication and Visibility	29
4.1.2	Product Quality Assurance	36
4.1.3	Facilities	42
4.1.4	Product Management & Control	48
4.1.5	Software Engineering Guidelines & Standard Practices	58
4.1.6	Tools and Software Development Aids	71
4.1.7	Technology Transfer	81
4.1.8	Training	88
5.0	PHASED IMPLEMENTATION PLAN	96
5.1	Immediate, Short-Term and Long-Term Implementation	97
5.1.1	Recommendations for Immediate Implementation	97
5.1.2	Recommendations for Short-Term Application	101
5.1.3	Other Recommendations	103
5.1.4	Recommendations for Long-Term Implementation	104
5.1.5	Tools Recommendations	105
5.2	Implementation Plan-Textual Description	108
5.2.1	Software Engineering Guidelines & Standards Subgroup	108
5.2.2	Technology Transfer and Training Subgroup	109
5.2.3	Tools Subgroup	110
5.2.4	Facilities Subgroup	111
5.3	Implementation Plan Work Breakdown Structure (WBS)	112
5.4	Activity Network	118

Contents (Continued)

		<u>Page</u>
6.0	SELECTED APPLICATION OF MPE DISCIPLINES	120
6.1	ADP Project Complexity	120
6.2	Complexity/MPE Relationship	120
References		124
Appendix A	Characteristics of the DMAAC	A-1
Appendix B	Characteristics of the DMAHTC	B-1
Appendix C	Error off Study Recommendations at HTC	C-1
Appendix D	DMA Survey-Parts 1 and 2	D-1
Appendix E	Recommendation Summary	E-1

Accession No.	
RECEIVED	☒
DATE	☐
TIME	☐
BY	☐

1

A

FIGURES

	<u>Page</u>
1.6-1 Study Approach	6
1.7-1 Relationship of Study Approach to SOW Tasks	8
2.1-1 Conceptual Model of DMA Programming Environments	11
2.2.1-1 Programming Environment	14
2.2.2-1 Applications/Project Environment	16
2.2.5-1 DMA Customer Environment	18
4.1.1-1 Sample Requirements Review Agenda	33
4.1.1-2 Sample Problem Report	34
4.1.2-1 Sample Requirements and Acceptance Criteria Specification	40
4.1.2-2 Sample Requirements and Acceptance Criteria Specification	41
4.1.4-1 Flow of Software and Documentation Through Configuration Control Levels	50
4.1.4-2 Configuration Change Request (CCR)	51
4.1.4-3 Sample Impact Analysis Form	53
4.1.4-4 Sample Software Problem Report	55
4.1.5-1 Sample Software Notebook Cover Sheet	65
4.1.5-2 Sample Programming Standards Policy Statement	68
4.1.5-3 DMA Policy Over the Software Lifecycle	70
4.1.6-1 Relationship of DMA Tools to Software Activities	73
4.1.8-1 Boeing Computer Services Company National Training Schedule	91
4.1.8-2 Courses Available Through TSI	92
5.4-1 Implementation Activity Network	119
6.1-1 Level of Project Complexity	122
A-1 Computer Program Specification Worksheet	A-12
A-2 Sample Site of Submission Report	A-45
B-1 Sample ADP Work Request Form	B-10
B-2 Sample ADP Work Request Estimate Form	B-11
B-3 Top Four Organizational Users of Univac 1108	B-37

TABLES

	<u>Page</u>
2.2.7-1 HTC Department Model Role Cross-reference	20
2.2.7-2 AC Department Model Role Cross-reference	21
4.1.3-1 Summary of Errored Off Runs	45
6.2-1 Application of Disciplines	123
A-1 Sample Monthly Tabulation of 1108 Utilization by Department	A-36
A-2 Run Frequency by SUP Time	A-37
A-3 Run Frequency by Time of Day	A-39
A-4 Idle Time, Down Time, Preventive Maintenance Time & Systems Test Time	A-40

ACRONYMS

AC	Aerospace Center
ADP	Automated Data Processing
ADS	Automated Data Systems
ANSI	American National Standards Institute
BCS	Boeing Computer Services Company
COR	Contracting Officers Representative
CPU	Central Processing Units
CSD	Computer Services Department
DMIS/P	DMA Management Information System for Production
DAD	Data Automation Division
DoD	Department of Defense
EOD	Equipment Operations Division
FAVS	FORTAN Automated Verification System
HQ	Headquarters (DMA)
HTC	Hydrographic/Topographic Center
MC & G	Mapping, Charting and Geodesy
MIS	Management Information System
MPE	Modern Programming Environment (Study)
PAM	Program Assignment Memorandum
PIP	Process Improvement Proposal
PMO	Production Management Office
POM	Program Objective Memorandum
PP	Directorate of Programs, Production and Operations
PRT	Directorate of Plans and Requirements, Techniques Office
PSL	Program Support Library
QA	Quality Assurance
R & D	Research and Development
RDT & E	Research, Development, Test and Evaluation
SAA	Special Activities Area
SDC	Scientific Data Department, Computer Division
STT	Directorate of Systems and Techniques, Advanced Technology Division
SUP	Standard Unit of Processing

SW (or S/W)	Software
TIP	Techniques Improvement Project
TIPs	Technical Information for Programmers
TO	Techniques Office
TSD	Technical Support Division
UPLI	Univac Use Program Library Interchange

Technical Report Summary

Throughout the software industry, recent attempts to upgrade or modernize software production practices through the use of tools and techniques have been largely unsuccessful. From examining these efforts, some of the common factors contributing to the lack of success can be identified:

1. In many cases there is no overall plan to identify, evaluate, acquire or introduce new tools and techniques.
2. Existing tools and techniques interface poorly.
3. Most individuals have a natural resistance to change.

Several years ago the Air Force Rome Air Development Center (RADC) recognized this problem. RADC initiated and sponsored studies of the implementation and application of modern programming practices by major software developers. These studies led to the general conclusion that both software quality and development productivity can be substantially improved through disciplined introduction and use of a wide range of modern engineering methods and tools. Results of some of these studies (and other studies performed by commercial software houses) have been documented in Section 3.0. The studies have concluded that potential increases in productivity from the application of modern programming practices range from 50 to 87%.

The DMA Advanced Technology organizations, understanding the significance of these findings, sought and received RADC assistance in formulating and implementing a plan to introduce modern programming practices to the DMA production centers. During the last few years the DMA, with the help from RADC, has studied, acquired and applied selected software development techniques and test tools.

Through application and careful evaluation of these tools and techniques, the DMA began to recognize important additional areas in which significant advances in software development and management practices could and should be made. The DMA also recognized that some important advances in these areas have been made in industry in recent years. Therefore, through the combined efforts of RADC and leaders in advancement of software engineering practice, the DMA sought to systematically address the modernization of DMA programming environments.

The study reported here marked the beginning of this more formal, rigorous approach to modernizing the DMA programming environments. The purpose of the study was to develop a plan for upgrading the current software production practices at the DMA. The ultimate objectives of the modernization approach are increased programming productivity, and improved quality and reliability of DMA software.

To address the systematic modernization of the DMA programming environments, the following approach was chosen:

1. Develop a conceptual model of an ideal Modern Programming Environment (MPE).

2. Collect data from the DMA centers. Information was collected via review of documentation and reports, through interviews and by conduct of a survey/questionnaire.
3. Characterize the DMA in terms of the conceptual (ideal) MPE model. This resulted in definition of characteristics in seven categories, as illustrated by Figure 2.1-1.
4. Conduct a detailed analysis of the DMA model by comparison of the DMA model to the MPE model.
5. Recommend solutions to the problems and potential problems which were identified. Recommend modification of existing DMA characteristics to achieve MPE characteristics.
6. Recommend new practices relating to perceived needs.
7. Develop a phased plan to implement the recommendations.

Each of these steps has been detailed in Section 1.6.

The recommendations of the study, in order for the DMA to achieve a full MPE, fell into eight main areas: Communication and Visibility, Product Quality Assurance, Facilities, Product Management and Control, Software Engineering Guidelines and Standard Practices, Tools and Software Development Aids, Technology Transfer, and Training. Section 4 of the Technical Report contains the detailed recommendations in these categories. Many recommendations are made and some of these interrelate. Although the recommendations do interrelate in some areas, each topic and subtopic should be considered independent and thus may be implemented separately. Some increase in programmer productivity and software quality can be attained by implementation of any one of the recommendations.

As stated earlier, to convert the DMA programming environment into an MPE, a number of modern programming practices and much more formal development/maintenance discipline need to be implemented. The recommendations made in Section 4 describe a fairly comprehensive set of characteristics which would need to be employed to develop a DMA MPE. Obviously, implementation of the full set of these recommendations will be a costly and time consuming effort. However, if the DMA is to achieve a true DMA MPE, certain of these recommendations are critical.

Because we recognize that the DMA (as would be the case with most other programming environments) will not be able to immediately achieve an MPE, we have prioritized the comprehensive set of recommendations detailed in Section 4. In addition, we have identified three of the recommendations which we feel will have the greatest immediate impact on DMA productivity and quality, given the characteristics described in Appendixes A and B.

Very little quantitative data on the potential increase in productivity and quality which could be attained by implementing these recommendations has been collected. The problems associated with quoting the potential benefit associated with any one recommendation have been discussed in Section 3. For these reasons, it is impossible to quote with any great accuracy the potential benefits of a

particular recommendation. It is possible, however, to estimate benefits which may be accrued, based upon our knowledge and experience and extrapolating from the little data which is available. It is also possible to state the qualitative benefits which should result from systematic, disciplined employment of these recommendations.

In order to provide the DMA with a means to justify the costs of implementing the three most critical recommendations made as a result of this study, we have provided estimates of quantitative benefits which we believe may result upon successful implementation. In addition, where quantitative benefits data has been found in the software development community, we have furnished it for the three most critical recommendations. Qualitative benefits generally associated with these three most critical recommendations have also been specified.

The most critical need which faces DMA software developers today is for the establishment and enforcement of programming and development standards. At the present time the programming environments at the DMA are primarily maintenance oriented. Approximately 70% of the programming activities are in support of changes, updates, and corrections to existing production programs. This fact in itself calls attention to the need for the implementation of a rigorous standards program.

In a study conducted by the University of Maryland, research indicates that the employment of a disciplined methodology by a programming team reduces the average costs, both machine and human, of software development, relative to individual programmers and programming teams not employing the methodology by a magnitude on the order of 2 to 1 (i.e., 50%), or better. (See reference 16 for details of the research.)

We have not seen quantitative data which specifically relates the establishment of programming standards to programmer productivity, however, it is a well-known fact that standards must be established as the foundation of cost-effective maintenance and quality program development. A significant reduction in maintenance programmer/analyst time to implement changes can be expected from such a standards program. While we have no data from a well-designed and documented experiment which solely employed programming standards, we conservatively estimate that maintenance programmer time can be reduced by 25-30% if programs are constructed and documented in accordance to standards. Therefore, the following recommendations are made:

1. The programming and software development standards which have been documented at each center (SDCINST 5660.1, SDCINST 8420.39, DMAHTCINST 5660.3) should be incorporated in a preliminary version of a Standards Handbook. The Handbook should also contain references to DoD Standard 7935.1-S.
2. Existing programming standards should be further detailed and then incorporated into the Standards Handbook. New programming standards (as identified in Section 4.1.5) should be defined, formalized and documented in the Standards Handbook. In addition, descriptions of recommended practices and techniques which should be employed in conjunction with phased development (i.e., review and reporting procedures, documentation production, use of a software notebook, etc.) should be incorporated into the Standards Handbook.

3. The reports described in Section 4.1.6 concerning the employment of structured programming without the use of a precompiler should be obtained and studied.

Regardless of whether structured programming is accomplished via support of a precompiler, it should be adopted as a DMA Standard. As described in Section 3.3.2, IBM reported that almost 50% more new code was produced with slightly half the effort for a project utilizing structured programming.

4. Peer code reviews should be conducted prior to final approval of the coding activities. This is necessary in order to verify compliance to established standards. Even the most well-defined set of standards will not result in increased programmer productivity if they are not rigorously employed. Peer reviews are the least-costly manual method of monitoring compliance with standards.
5. After the standards have been defined, documented, and are being employed, the DMA should define the requirements for a Code Auditor tool to be used as an automated means to verify compliance with standards. Depending on the depth of the manual effort to verify standards compliance, varying levels of cost can be incurred. TRW, which long ago established and began to formally employ and verify compliance with standards, developed a Code Auditor of this type. Prior to use of the Auditor, code review was accomplished via peer reviews and by an independent QA review. They estimated cost savings of \$150,000 in the first year of the Auditor's employment to monitor standards compliance.

Another critical problem which is occurring at both DMA centers is the availability of the computer. Turnaround time on the mainframe computers was expressed as the number one problem in our interviews with DMA developers and managers. Since the DMA is primarily a maintenance environment, it is reasonable to conclude that a substantial benefit can be incurred by systematically optimizing the most frequently used production programs. This can be accomplished by either manual or automated analyses, followed by program modification. Because of the current shortage of software engineering manpower to conduct manual analyses, we recommend the use of automated optimization aids.

Some data is available (see reference 15) which supports the use of optimization tools:

1. The Federal Aviation Agency (FAA) utilized a set of tools to analyze programs for efficiency and was able to free up \$471,562 worth of computer resources during 1976-1978.
2. NASA used a data flow optimization package to free up 18 hours of computer time per day on two machines (annual savings \$660,000).

While the situations associated with the FAA and NASA cannot be duplicated, it is obvious that a significant amount of cost savings and increased computer

availability is likely to result from employment of optimization tools. Therefore, the following recommendations are made:

1. The production programs which are the biggest consumers of computer resources should be identified. At the Aerospace Center, the Monthly Accounting Program Utilization by Unique Program Number report can be used for this purpose. The HTC should determine whether an equivalent report is produced at the center. If not, the AC Program should be implemented at HTC for identification of frequently utilized programs.
2. Production programs which are big consumers of computer resources should undergo optimization analyses evaluation, and subsequent associated modification.

Another critical need which was identified during the MPE Study centers around training of software development/maintenance personnel. This criticality has occurred for a number of reasons. First, software development has always been more of an "art" than a "science." Only recently have development techniques been employed, and few courses which taught the science of software engineering have been available in the past. Second, most DMA software developers have a background in mathematics, geography and cartography. As such they have not been privy to much software engineering training. Third, the criticality of the production of maps and associated data has imposed a severe constraint in the attendance of training courses.

In the interest of a more effective, efficient and modern programming environment, these situations must be overcome and a formal training program which emphasizes the science of software engineering must be implemented.

No data is available from controlled, well-documented experiments which relate software engineering training to improved productivity and quality. Obviously, any such data (if it were available) would be dependent upon many uncontrollable and virtually unmeasurable factors; individual receptivity to training, preconceived attitudes and biases, effectivity of instructional methods, etc. However, it should also be obvious that achievement of an MPE cannot be realized without familiarity with modern disciplines and concepts. Therefore, the following recommendations are made and should be implemented in the following order:

1. An individual from each HTC department should be identified as a training focal point. Individuals currently performing this function at AC should receive greater visibility in order to better identify the training needs and thereby represent the community of software developers.
2. A catalog of currently available DMA training courses should be obtained and reviewed by each department training representative.
3. While training courses available through DoDCI, CSC and AMETA are identified annually, other training courses which are commercially available should be identified. Courses should address management of software development, software development techniques, testing techniques and strategies, specification and analysis of requirements and planning, scheduling and estimating techniques. Computer and Visual-Aided Instruction

courses and commercially available seminars which address these topics should also be identified.

4. A seminar which will serve to introduce software engineering to DMA management and technical staff should be developed and conducted.
5. A basic course(s) should be identified (or developed) and conducted at each center concerning the use of interactive programming.
6. Training courses in requirements analysis and specification, DMA programming practices, and project management/development methodologies should be conducted.
7. Training courses in test and evaluation methodology and planning, estimating and scheduling techniques should be conducted.

Additional, less critical recommendations are made in Sections 5.1.3 and 5.1.4. Implementation of the full set of recommendations (both critical and less critical) should represent achievement of a DMA MPE.

EVALUATION

The work described in the final technical report represents a significant accomplishment in establishing the framework of a computer software development environment compatible with modern software engineering tools and techniques.

The report describes a basic plan for implementing tools and procedures which will introduce and maintain modern software engineering concepts into the daily operation of DMA computer facilities in Washington DC and St. Louis MO. Use of these tools and procedures will create a disciplined environment that will increase the quality and reliability of software developed at these installations and will lower life-cycle costs.



MICHAEL LANDES

Project Engineer

1.0 OVERVIEW

1.1 PROBLEM

Throughout the software industry, recent attempts to upgrade or modernize software production practices have been largely unsuccessful. From examining these efforts, some of the common factors contributing to the lack of success can be identified. Several of these are outlined below.

In many cases there is no overall plan or approach developed. The problems to be solved are not thoroughly studied or, in some cases, even identified. Ad-hoc attempts are made to initiate new practices and to advocate the utilization of automated tools. This type of bottom-up approach usually results in an unstructured and unsuccessful attempt to acquire and use new technologies. Many efforts to develop a modern programming environment (MPE) are not based upon a comprehensive plan which insures a solution to the real problems via a set of guidelines, practices, techniques and tools which clearly interface with each other.

Management often initiates plans by issuing mandates of various sorts. There is seldom adequate attention to and planning for natural resistance to change. The chances of success of a plan can be diminished by not involving those to be affected in the plan's development, not clearly stating the rationale behind the suggested change, and not including incentives and factors to motivate staff.

Even when planning is adequate, a poor implementation strategy can lead to an unsuccessful outcome. In the past, implementation strategies have been given little attention. These often lack adequate and varied training programs. They are often inflexible, calling for a 100% adoption/implementation, rather than a slow, phased approach, with time for experimentation and tailoring to fit the needs of the environment.

In general, most efforts to either acquire or develop 'Modern Programming Practices' have not approached the problem with the same discipline and rigor that is advocated as the basis for the practices themselves.

Several years ago the Air Force Rome Air Development Center (RADC) recognized this problem. RADC initiated and sponsored studies of the implementation and application of modern programming practices by major software developers. These studies led to the general conclusion that both software quality and development productivity can be substantially improved through disciplined introduction and use of a wide range of modern engineering methods and tools. The DMA Advanced Technology organizations, understanding the significance of these findings, sought and received RADC assistance in formulating and implementing a plan to introduce modern programming practices to the DMA production centers. During the last few years the DMA, with help from RADC, has studied, acquired and applied selected software development and test tools. This DMA initiative to introduce modern tools and techniques involved:

1. Investigations into structured programming concepts and practices.
2. Acquisition of a precompiler (STRUCTRAN-1) to convert a structured extension of FORTRAN to conventional FORTRAN which simulates the basic structured programming control constructs.

3. Acquisition of a translator (STRUCTRAN-2) to structure unstructured FORTRAN.
4. Acquisition of RXVP, or FAVS, a FORTRAN validation and verification tool which incorporates STRUCTRAN-1 and STRUCTRAN-2.
5. Investigations into two other tools: Program Support Library (PSL) and a COBOL validation and verification tool.
6. Acquisition of a meta-assembler.

Through application and careful evaluation of these tools, the DMA began to recognize important additional areas in which significant advances in software development and management practices could and should be made. The DMA also recognized that some important advances in these areas have been made in industry in recent years. Therefore, through the combined efforts of RADC and leaders in advancement of software engineering practice, the DMA sought to systematically address the modernization of DMA programming environments.

1.2 PURPOSE

The study reported here marked the beginning of this more formal, rigorous approach to modernizing the DMA programming environments. The purpose of the study was to develop a plan for upgrading the current software production practices at the DMA. The ultimate objectives of the modernization approach are increased programming productivity, and improved quality and reliability of DMA software.

1.3 SCOPE

The scope of the study included both of the DMA production centers: the Hydrographic/Topographic Center (in Washington, D.C.) and the Aerospace Center (in St. Louis, Missouri). Within each of the centers, the study addressed software production practices of the entire development and maintenance life cycle. The study looked at both the 'closed shop' activities (activities by that group at each of the centers charged with providing software development and maintenance support) and 'open shop' activities (software development and maintenance performed directly by the staff in the project/application departments). The study also included all software activities within the center, whether working on the UNIVAC mainframes or the various minicomputers in-house. In addition, the results of the study were to be applicable to DMA programming environments in future years as they can be anticipated to change due to future hardware changes and enhancements.

1.4 OBJECTIVES AND OUTCOMES

There were three primary objectives of the study:

1. Study and characterize the DMA programming environments.
2. Recommend a plan for modernizing the DMA software production practices, including identifying and recommending solutions to existing problems.

3. Develop a plan for implementing the suggested changes/enhancements to the DMA environments.

The only outcome of the project is this Final Technical Report. It contains a characterization of the DMA programming environments, a set of observations which identifies problems and areas for improvement, recommendations to effect the solutions to the problems and the suggested improvements, and an implementation plan which prioritizes and suggests a relative schedule for implementing the suggested changes.

1.5 STATEMENT OF WORK

The following list of ten tasks was presented in the original Statement of Work.

1. Conduct an in-depth study of current software production practices within DMA production centers at DMAAC, St. Louis, and DMAHTC, Washington, D.C. Current requirements definition, system analysis, design, production and maintenance practices should be considered.
2. Solicit management's role in, and view of, current software practices.
3. Identify and characterize current software development and maintenance problems.
4. Formulate recommendations for improved production practices.
5. Evaluate existing DMA practices and tools.
6. Identify additional concepts, practices, tools and other elements of software engineering discipline which provide potential solutions to problems identified above.
7. Develop a preliminary concept of a Modern Programming Environment (MPE) primary attention to ease of use, extendibility, satisfaction of most critical requirements, and smooth interfacing of practices and tools.
8. Develop recommendations for the phased implementation of preliminary and full scale MPE capabilities. Document implementation approach and steps, with particular emphasis on solving the technology transfer problem (i.e., using broad training programs and other effective reinforcers to assure optimum use of practices and tools in the production environment).
9. Identify for inclusion in the DMA R&D program any required, but not presently available, capabilities.
10. Conduct and participate in review meetings and provide periodic oral presentations on his approach, status, progress, conclusions and recommendations.

1.6 STUDY APPROACH

BCS has followed a seven-step approach to accomplish the ten tasks outlined in the Statement of Work. This approach is outlined in this section.

The principle goal of this study was to access information about the current status of the DMA programming environments in order to formulate a set of recommendations for improving the programming environments at the DMA centers. The long-range objectives of the DMA are increased productivity and improved product (software) quality. To help achieve these objectives, this study developed a set of recommendations which suggest ways to:

1. Solve or alleviate existing problems, and
2. Introduce methods for improving upon the status quo.

The approach taken to develop these recommendations is presented below in a step-by-step fashion, and has been illustrated in Figure 1.6-1.

1.6.1 Generic Model

The first step in the study approach was the development of a conceptual model of a Modern Programming Environment (MPE). This model provides a description of the forces which impact or shape the MPE (e.g., new technology, administration, etc.) and the characteristics or attributes which define the MPE (e.g., what practices are employed during requirements definition, system analysis, design, production and maintenance).

This model was used as a tool during succeeding steps when analyzing the current DMA programming environment and, as such, was refined as the study progressed.

1.6.2 Data Collection

This step represented the data gathering, information-collecting phase of the project. Several methods were employed to collect data, including: a study overview presentation and group discussion session; the collection of various documentation (standards, guidelines, organization charts, code listings, computer usage reports, etc.); phase inquiries; personal interviews with management personnel; and a two-part survey of technical management and staff (see appendix D).

Collected data served to define the characteristics of the DMA programming environments, management's role in and view of current software practices, and the existing DMA practices and tools.

A by-product of the data collection step was the perspective DMA individuals have of current software development and maintenance problems.

1.6.3 Application of the Model

Using the obtained information, the DMA programming environments were characterized using the framework of the generic model. During this step, the generic model was refined to be more representative of the DMA environments.

An important part of this step was application of the determined characteristics of existing DMA practices and tools to the model. This activity lead to an evaluation of the existing practices in terms of their current utility versus their desired or potential utility. The model provided the basis for this evaluation.

The result of this step was a better understanding of the current DMA environments. The model provided the baseline for comparison and the framework for analysis. Characteristics of an MPE which were identified in the generic model, but found lacking in the DMA environments were noted. When the generic and DMA models shared a characteristic, the detail with which the characteristic was addressed in both models was defined and analyzed. Many problem areas were identified directly, while other potential problems were identified by the observation of various symptoms.

1.6.4 Detailed Analysis of the Environments

The problems and potential problems identified during data collection (step 2) and model application (step 3) were studied in more detail during this step. An investigation was performed to determine the cause-effect relationships between existing DMA programming characteristics and current or potential problems. Some of these relationships may have been suggested during steps 2 and 3. The result of this step was an identification of primary problems or areas where recommended improvements could be made. The analysis in effect separated the primary problems from secondary problems (which were essentially effects of the primary problems).

1.6.5 Recommend Solutions

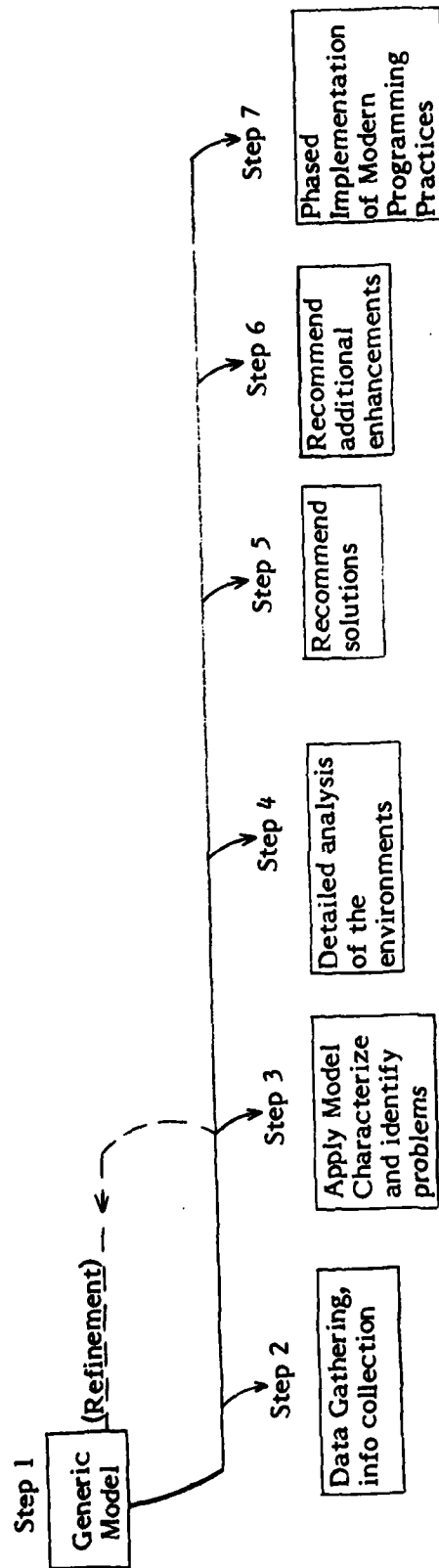
Once high priority areas were identified and the relationship between the problems and the characteristics of the current environment had been determined, recommendations were developed. These recommendations were in terms of the characteristics within the model, suggested changes in the current practices, introduction of new practices, increased use of existing or modified versions of support tools, acquisition of new tools, etc.

1.6.6 Recommend Additional Enhancements

The characterization of the current environment served as the basis for developing recommendations to introduce new practices relating to perceived needs (independent of particular problems that had been identified).

1.6.7 Develop Phased Implementation Plan

Finally, these recommendations were prioritized in a manner which would facilitate phased implementation of modern programming practices at the DMA. A short-range plan incorporated recommendations for solution of the most critical problems. A long-range plan addressed an expanded set of problems, providing for the phased implementation of a broad range of modern programming practices.



The generic or conceptional model describes the:

- Views or perspectives and the impacts upon the programming environment
- Roles in and attributes of a Modern Programming Environment

Figure 1.6-1 Study Approach

1.7 STUDY APPROACH/STATEMENT OF WORK

This section provides an elaboration on the study approach. It describes how the approach relates to the Statement of Work. It explains specifically which tasks were addressed during each of the steps in BCS's stepwise approach. Figure 1.7-1 summarizes this cross-comparison.

Step 1 involved the development of a conceptual model of a Modern Programming Environment (MPE). This model formed the basis for the study of the current practices at the DMA (Task 1) and provided a starting point for the development of a DMA MPE (step 7).

Step 2 consisted of data/information gathering (i.e., conducting the two-part survey, the interviews and the multitude of follow-up phone calls, etc.). This information-gathering completed the study Task 1 and also Task 2, which was to solicit the role in and view of the programming environment by DMA management. It also began Task 3, the identification and characterization of the problem areas and areas for potential improvement within the DMA programming environment.

In step 3, the conceptual model was applied to the DMA environment. In reality, the development and application of the model was an iterative process. The conceptual model evolved and was refined through a series of attempts at using it to characterize the DMA programming environment. This characterization, using the model, allowed the evaluation of current DMA programming practices and programming aids/tools (Task 5) and served to further identify and characterize problems (Task 3). This assessment of the current DMA environment was also an important part of Task 7, the development of a DMA MPE.

Step 4 involved a further analysis of the current DMA environment. The analysis continued identification of problems relating directly to current practices at the DMA. This completed Task 3, the identification of problems and areas for improvement, and Task 5, the evaluation of current practices and tools.

Steps 5 and 6 involved the formulation of recommendations to solve or alleviate current problems and to suggest improvements to the current mode of operation, respectively. The formulation of these recommendations corresponded directly with the accomplishment of three tasks: Task 4, to develop recommendations for improved software practices; Task 6, the identification of concepts, practices and tools to solve the problems identified; and Task 9, the identification of potential R&D areas to be addressed during the modernization of the DMA programming environments. These recommendations also completed Task 7, the description/development of a concept of a DMA MPE.

Step 7, the development of a phased implementation plan, corresponded directly with Task 8, which was to recommend phased implementation of the new practices, tools, etc., at the DMA.

Task 10 of the Statement of Work, to conduct and participate in review meetings and periodic presentations, was addressed during all the steps in the approach.

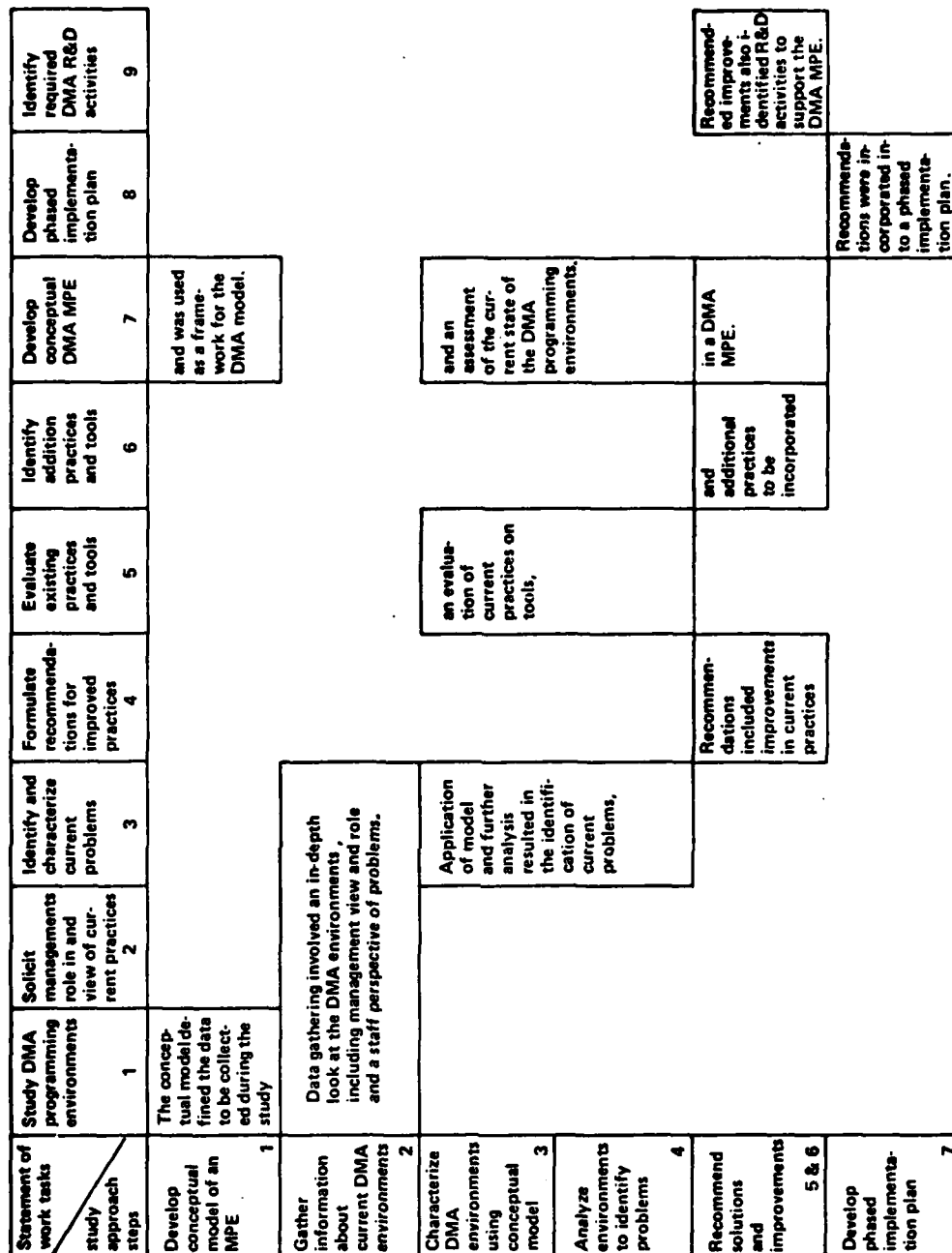


Figure 1.7-1 Relationship of Study Approach to SOW Tasks

1.8 REPORT ORGANIZATION

This section, Section 1, presents an overview to the DMA Modern Programming Environment (MPE) Study. Subsections on the problem, purpose, scope, and objectives document Study background material. The Statement of Work is then related to the seven step approach to the Study.

Section 2 describes the analytical framework and conceptual model which provide a simplified view of the DMA environment and highlight the major roles which impact the programming environment.

Section 3 contains a discussion on the topic of productivity measurement in the software industry. Following this discussion is a description of various studies which have been performed to research the effect of utilization of modern programming practices on programmer productivity.

Section 4 contains the recommendations which have resulted from the Study. Recommendations are made in eight major areas: communication and visibility, product quality assurance, facilities, product management and control, software engineering guidelines and standard practices, tools and software development aids, technology transfer and training. Prior to the recommendation statement, a summary of related DMA characteristics is given in order to illustrate the rationale for the recommendation.

Section 5 describes the plan to implement the recommendations which were presented in Section 4. The plan is illustrated via use of a Work Breakdown Structure (WBS), an activity network and text which briefly describes interrelationships between the recommendations.

Section 6 contains a scheme by which to compute a complexity factor (between 1 and 5) for DMA projects. The factor is then used as guidance for the application of particular disciplines and techniques to DMA projects.

Appendixes A and B contain the studied characteristics of the DMAAC and DMAHTC, respectively. The characteristics are documented in an outline-like format for each of the seven components of the model. In some cases the material may be repetitive, when characteristics of one of the components overlap characteristics of another.

Appendix C contains the results and final recommendations associated with the Error-Off Study which was conducted by an in-house group at HTC. These recommendations address characteristics which were observed during the MPE Study and have therefore been reproduced in this report.

Appendix D contains Parts 1 and 2 of the Survey which was administered at each of the centers in February.

Appendix E contains a summary of the contents of the recommendations in an outline form for easy reference.

2.0 ANALYTICAL FRAMEWORK FOR DMA PROGRAMMING ENVIRONMENT

2.1 GENERAL APPROACH

A conceptual model of the DMA programming environment has been developed. This model was developed in order to provide a simplified (less detailed) view of the DMA environment and to highlight the major roles that need to be considered in this study.

Section 2 presents this model and subparts of the model in various forms. All of the forms of the model depicted in figures are consistent and are presented to illustrate the various perspectives from which the DMA and its programming environments can be viewed.

In addition to the model, an information framework (IF) has been developed. This framework outlines the data or information which was necessary in order to describe and characterize the DMA environments. The framework is described fully throughout this section and is used to represent the characteristics of the DMAAC and DMAHTC programming environments in Appendixes A and B.

Figure 2.1-1 presents a tiered, functional view of the DMA environment. This view highlights the programming environment and the interfaces between it and the other components of the DMA which directly and indirectly affect it. This model of the DMA is general and applies to both DMA centers -AC and HTC.

Four distinct levels are portrayed in this model. Level 1 represents the external forces impacting each of the DMA centers. Level 2 corresponds to the internal forces within each center that have a direct bearing on or interface with the programming environment. Levels 3 and 4 comprise the programming environment itself. The programming environment contains the computing resources and software support services (personnel, methodologies and tools) necessary for software development and maintenance.

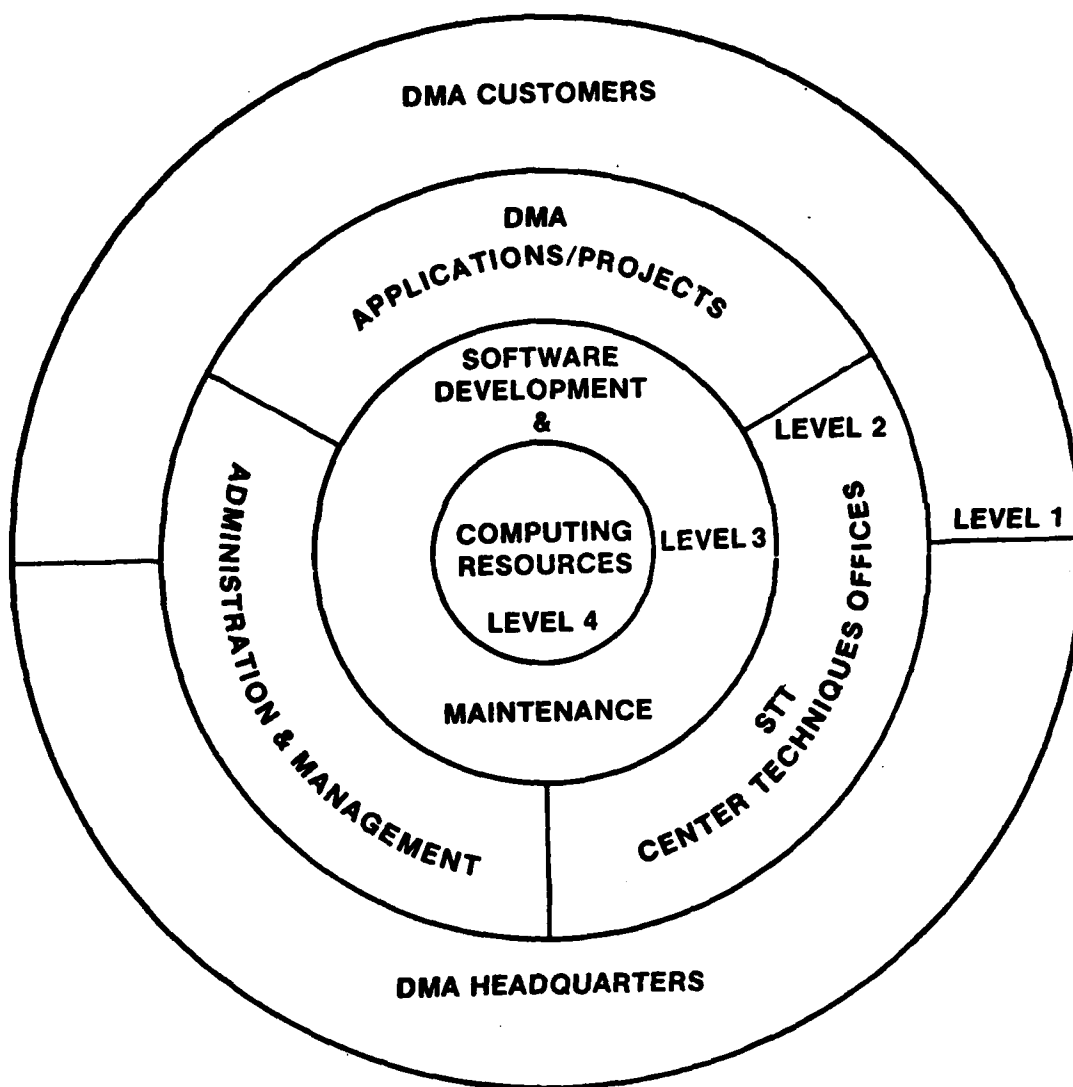


Figure 2.1-1 Conceptual Model of the DMA Programming Environments

Each of these levels includes a number of generic functions important in coming to a good understanding of the DMA programming environment:

1. The outermost circle (Level 1) represents the external world as seen from each of the centers under study. This aspect of the model includes the customers (i.e., SAC, NOAA, DoD, etc.) that come to the centers with support requests for projects. These requests are then passed to the various technical organizations within the DMA represented by the application/project domain (Level 2).
2. Level 2 contains the applications/projects area which is the primary customer serviced by the programming environment. These applications areas are responsible for formulating the requirements for software support. Once formulated, this information is then passed into the programming environment. This is done either by the initiation of a closed shop computing project within the computing support department or directly, via an open shop approach.

Two other functions within the center are also apparent at this level. Administration and management are involved in the sense that they must review and approve the expenditure of resources for computing. They are also vitally interested in the productivity, efficiency and quality of the work being performed. The Techniques Offices and the Directorate of Systems and Techniques (STT) are similarly interested in the efficiency and quality of the work performed within the programming environment.

3. The software design, construction, and maintenance functions are contained in Level 3 of the model. This is the real heart of the programming environment. The software production and maintenance functions are performed by both the professional analyst/programmer and the open shop programmer. The functions provided to both classes of programmers are similar, however, the skill with which they are used is a significant issue. The characterization of the programming environments contained in this document is designed to facilitate the identification of the key elements of this programming environment and to study their interrelationships.
4. Level 4 portrays the computing resources available within the DMA centers. Included in this portion of the model are all the hardware elements (maxi and mini), the operations personnel, and the vendor supplied support software (i.e., operating systems, utilities, etc.).

2.2 MAJOR COMPONENTS OF THE MODEL

The principle goal of this study is to formulate a set of recommendations regarding the programming environments at the DMA centers. The recommendations will focus upon changes intended to increase programmer productivity and software product quality. Therefore, the intent of the model developed is to (1) illuminate the programming environments at the DMA and their operation, and (2) understand the forces which impact the programming environments.

The following sections will explain the components of the model in more detail, starting with the programming environment (Levels 3 and 4). Then, working

out from this central point, subsequent sections explain the forces shaping the programming environments (Levels 1 and 2).

2.2.1 Programming Environment

. The Programming Environment (PE) has been divided into two components. The first component (Level 3) embodies the major functions involved in software development and maintenance (the software D&M component). This houses the staff who perform the functions and includes the procedures, standards and guidelines which govern or influence software development and maintenance. The staff is divided according to function: technical management, development staff, and support staff (e.g., clerical). The procedures/standards/guidelines are described in terms of what they pertain to, e.g., project organization and management, software development, or quality assurance.

The second component of the PE (Level 4) is the Computing Resources component (CR). It includes the computer(s) used for development, testing and installation. (Computer is used in a broad sense, referring to not only the hardware, but also the operating system and utility software.) The CR component also includes the hardware, software and staff which enhance the usability and utility of the computing system. Operations support (the computer operators and related functions), and software tools or packaged systems which automate commonly performed functions (DBMS, MIS, Accounting, etc.) are part of this subcomponent. The CR also includes that part of the overall system which provides the users with access to the system, including facilities such as terminals, keypunches, job submission sites, and also the procedures governing the operation of these sites - hours of operation, job scheduling algorithms, usage restrictions, etc.

The computing resources component of each of the DMA centers includes a wide variety of hardware. In each case it not only includes the central site mainframe computers, but also the on-site minicomputers that are used in many of the project/application departments.

Another way of viewing the programming environment is by building a pyramid, where each level of the pyramid is supported by the next lower level. The computing resources component comprises the bottom level. It supports both open and closed shop software development and maintenance activities (see Figure 2.2.1-1).

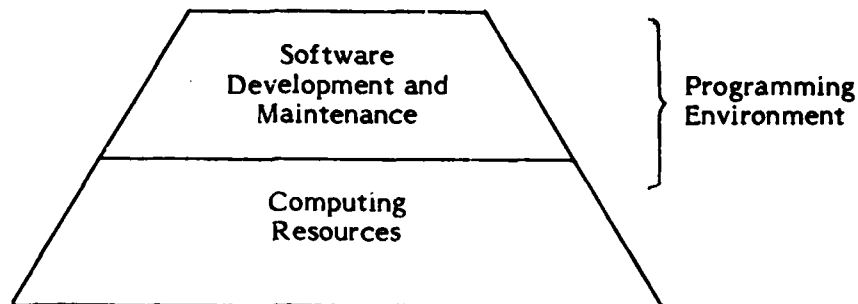


Figure 2.2.1-1 Programming Environment

To fully describe and characterize the programming environments of the DMA centers as depicted by the model, the following types of information are necessary:

- Software Development and Maintenance
 - Project Types
 - Project Organization
 - Project Planning and Initiation
 - Project Management
 - Software Lifecycle - Activities/practices by phase
 - Project/Product Documentation
 - Evaluation
 - Tools and Techniques
 - Project Support
- Computing Resources
 - Hardware configurations
 - Common job attributes
 - User support/consultation
 - System support/system programming
 - Operations/Job handling
 - Utilities (software)
 - Tools
 - Types of services
 - Access
 - Program Library

2.2.2 Applications/Project Environment.

The basic reason for the existence of the programming environment is to provide its user community with software development services. This user

community is broadly defined as the consumer of the products and services of the programming environment. This community is thus integrally involved in determination of the overall objectives of the programming environment and in identification of the most relevant qualities of those products and services.

The user community also acts in another directly related role - that of specifying the requirements for the software products, and then reviewing progress at intermediate steps during product development.

The user community of the programming environment is the application/project departments of the DMA (Level 2). These departments collectively make up the next level of the pyramid presented previously (see Figure 2.2.2-1).

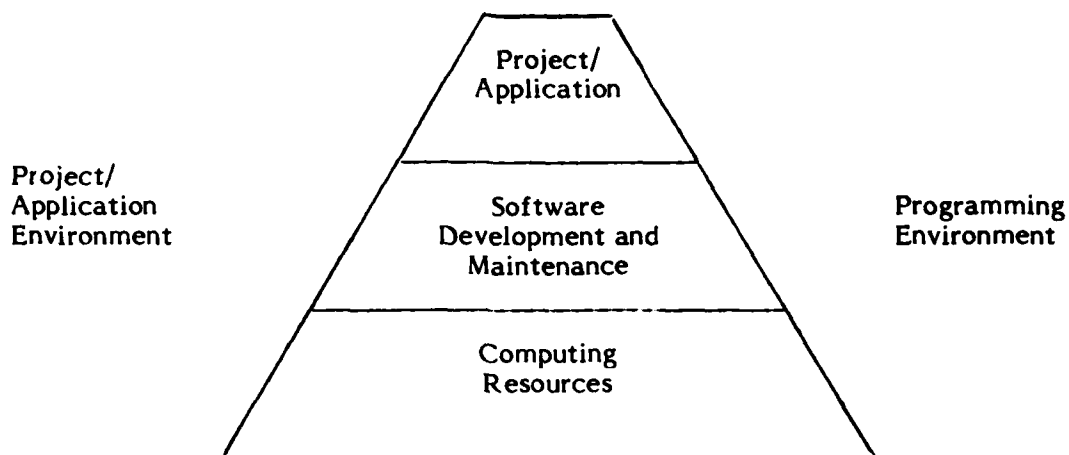


Figure 2.2.2-1 Applications/Project Environment

To characterize the project/application environment(s) at the DMA, the following information must be studied. This information is part of the information framework previously discussed. The emphasis is to understand the interface between the programming environment and those who use its services, i.e., the role of the user in software development.

Characteristics of the DMA Application/Project Environment

- Interface with DMA customers
- Interface with Programming Environment (both open and closed shop)
- Role in Project Planning and Initiation
- Role in Project and Product Management
- Role/Participation during Software Development
- Role/Participation during Software Maintenance
- Role in Quality Assurance

2.2.3 Center Administration and Management

The Center Administration and Management component (also at Level 2 of the conceptual model) is a composite of organizational elements responsible for establishing policies, making management decisions, allocating funds, hiring staff, etc., which greatly influences the operation of the programming environment. It includes the functions of the technical director, comptroller, facilities engineering, program planning, etc.

In order to characterize this group of functions at the DMA, the following items are part of the information framework. The intent is to characterize the impact of Administration and Management upon the programming environment.

Characteristics of Administration and Management

- Services to be provided
- Administration
 - Resource Customer Interface
 - Comptroller
 - Personnel
 - Resource Acquisition
 - Training
 - Standards, procedures, guidelines
 - Tools and Techniques (administrative aids)
- Management
 - Project Support
 - Decision making
 - Management Tools and Techniques (for project monitoring and control)

2.2.4 STT and the Center Techniques Offices

The introduction/implementation of new technologies in the area of software development into the programming environment is the responsibility of the STT, Techniques Offices and center training programs. This is a major force shaping the programming environment at Level 2 of the conceptual model.

The portion of the information framework pertaining to the Techniques function is presented below:

Characteristics of the Center Techniques Office

- Mission, role, scope
- Organizational set up
- Functions
- Interface with departments

2.2.5 DMA Customer Environment

The DMA customer set adds the top level to the pyramid previously illustrated and Level 1 of the conceptual model. The DMA customers are consumers of the products and services of the DMA. These products are produced in the application/project environment which is the consumer of the services of the programming environment. Figure 2.2.5-1 illustrates the DMA customers' position in the environmental pyramid.

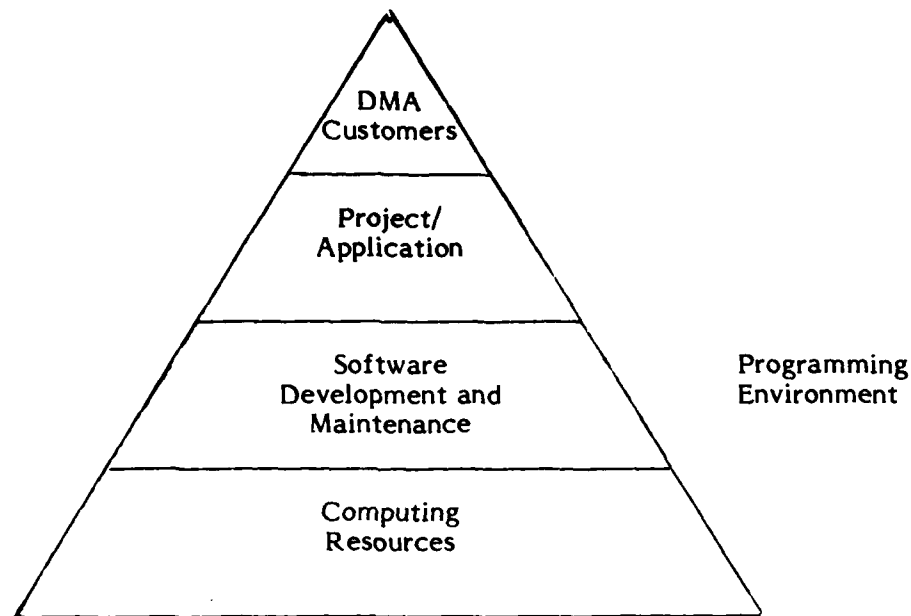


Figure 2.2.5-1 DMA Customer Environment

There is a relationship between the requirements of the DMA customers and the ultimate requirements for software (SW) development and maintenance. The character of the relationship needs to be investigated, as well as the translation process, i.e., customer requirements to SW requirements.

The information framework embodies the characteristics of this relationship which are to be investigated.

Characteristics of the DMA Customer

- Customer set
- Services required
- Software support required
- Project type, size, time frames
- Customer interfaces with DMA
- Customer interfaces with the programming environment (direct or indirect)
- Role in SW requirement specification

2.2.6 DMA Headquarters

The DMA Headquarters (HQ) manages the Research and Development (R&D) program for the DMA centers. As such, it is important that the model characterize the role that HQ plays in introducing new technologies to the centers. Headquarters is described at Level 1 of the conceptual model as an outside force which shapes the programming environment.

2.2.7 Applying the Model to the DMA Centers

A more detailed description of the organizations which are part of the programming environment model component of the DMA centers is presented in Appendixes A and B. However, Tables 2.2.7-1 and 2.2.7-2 incorporate the organizational components of the two DMA centers (HTC and AC) into the framework of the model. Organizational components of each of the centers are associated with one of the model components.

MODEL ROLES

Department	DMA Customer	Admin and Management	Techniques	Project/ Application	S/W Design Application	Computer Resources
Department of Defense	X					
Strategic Air Command	X					
National Oceanic and Atmospheric Administration	X					
Executive Office		X				
Deputy Director for Mission Support		X				
Personnel (and other offices)		X				
Comptroller		X				
Plans and Rqmts Directorate		X				
Systems & Techniques Directorate		X	X			
Programs, Production and Operations Directorate						
Geodesy & Surveys Department		X	Ref ⁵	X	Ref ¹	Ref ²
Scientific Data Department			Ref ⁵	X	Ref ³	Ref ⁴
Computer Services Department			Ref ⁵	X	Ref ¹	Ref ²
Topography Department			Ref ⁵	X	Ref ¹	Ref ²
Hydrography Department			Ref ⁵	X	Ref ¹	Ref ²
Navigation Department				X	Ref ¹	Ref ²
Field Offices Department		X				
Graphic Arts Department		X	Ref ⁵			

Table 2.2.7-1 Hydrographic/Topographic Center Department Model Role Cross-reference

MODEL ROLES

Agencies/Departments	DMA Customer	Admin and Management	Techniques	Project/ Application	S/W Design Application	Computer Resources
Strategic Air Command	X					
National Oceanic and Atmospheric Administration	X					
Department of Defense	X					
Office of the Director		X				
Other Administrative Offices - (e.g., Communications and Electronics, Public Affairs)		X				
Directorate of Civilian Personnel Comptroller		X				
Directorate of Plans & Rqmts		X				
Directorate of Programs		X				
Production and Operation		X	X			
Directorate of Systems & Techniques		X				
Directorate of Logistics		X				
Directorate of Facilities		X				
Engineering		X				
Directorate of Administration		X				
Aeronautical Information Dept				X	Ref ¹	Ref ²
Geopositional Department				X	Ref ¹	Ref ²
Aerospace Cartography Dept				X	Ref ¹	Ref ²
Scientific Data Dept - Scientific Computer Division					Ref ³	Ref ⁴
Scientific Data Dept - Other Divisions				X		Ref ²
Printing & Distribution Dept		X				

Table 2.2.7-2 Aerospace Center Department - Model Role Cross-reference

References to Tables 2.2.7-1 and 2.2.7-2

1. In Project/Application departments, certain activities may functionally lie in the realm of software development and maintenance, i.e., projects that involve open shop software development and maintenance activities (performed by staff within the department).
2. Computing Resources (independent of the Computer Services Department/HTC or the Scientific Data Department/AC) exist within many of the Project/Application departments to support open shop software related activities. These also fall into the Computing Resources component of the model.
3. All closed shop software development and maintenance activities fall into this functional category within the model.
4. The computing resources (mainframe, support hardware, software and operations staff) necessary to support closed shop development and maintenance activities and production runs, belong in the computing resources component.
5. The Project/Application departments have Techniques Offices within them. Where these offices support the open shop software activities, their functions are identified in the model.

3.0 THE TOPIC OF PRODUCTIVITY

3.1 GENERAL

While this Study has not been commissioned to determine measures of programmer productivity and software quality, the overall goals of the designed MPE (to increase productivity and quality) necessitate a brief discussion of these topics.

3.2 PRODUCTIVITY METRICS

When the Study was initiated, we set out to measure the productivity of DMA software developers and the quality of DMA software. We learned what many people at the DMA (and throughout the software management community) already knew - there are no clearly-defined metrics with which to measure productivity or quality, nor is there an agreed to level of current productivity and quality.

During the course of the study we were asked to define productivity, as it is used for commercial software management. Many DMA managers expressed a concern for explicit metrics with which to objectively evaluate their programmers and analysts. DMA managers are not alone in this concern. Throughout the software industry, many commercial and government organizations have defined productivity metrics which are meaningful to them. However, there is no widely-recognized metric of productivity which is used as an industry standard. Some metrics which are used in industry are:

1. Lines of code/time
 - debugged, documented lines of source code/year
 - bytes of new code/total man months of effort
2. Number of compilable units
3. Number of program errors
4. Computer time used
5. Pages of documentation
6. Size of internal program tables

It should be obvious that not one of these metrics can stand alone as a measure of productivity, since an understanding of each element is necessary to evaluate the activities of the software developer.

3.3 PRODUCTIVITY AND TECHNIQUES AND TOOLS

Since productivity metrics are ill-defined, it is not surprising that potential increases in productivity and quality resulting from the application of a particular development technique or tool are also ill-defined. However, there have been numerous studies which have attempted to quantify increased productivity, using one or more of the metrics previously defined.

Due to the direct and indirect costs associated with education in and implementation of modern development techniques and tools, the need for quantification of benefits is made very real.

Unfortunately, some of the costs and benefits associated with employment of software development techniques and tools are essentially unmeasurable. For example, increased programmer satisfaction and motivation are difficult (if not

impossible) to quantify. This issue has compounded the problems associated with evaluation of tools and techniques.

Another factor which has clouded the issue of measurement of productivity increase is the way in which studies have been conducted. Most studied projects have employed more than one technique or tool, therefore, the benefits attributable to a particular technique or tool are not immediately recognizable.

The issues of standardized productivity metrics, unquantifiable benefits and employment of multiple techniques and tools in productivity studies have made precise estimates of productivity improvement unobtainable. However, some studies have produced estimates which serve to define the productivity potential associated with the use of tools and techniques. The conclusions of these studies are essential to the DMA when evaluating the cost effectiveness of employing modern technologies. For this reason, some prominent study findings will be elaborated here.

Due to the lack of standardized productivity metrics and related data, these studies have employed different approaches to the evaluation of modern programming practices (MPP).

Some studies have evaluated the contribution of MPP toward solving problems (4), while others have mapped the use of MPP to achievement of project development objectives (10). The approach taken by Black (2,3) centers around validation of hypotheses which compare forecasted labor (estimated labor without application of MPP to project) to actual labor (utilizing MPP during project) in environments which utilized MPP. The approach followed by Belford (5,11) evaluated the contribution of MPP toward reducing or detecting typical software development errors. Still other studies have measured increased productivity using the previously mentioned metrics (6,7,8,9).

3.3.1 The BCS Study

The research done by Black (2,3) investigated five projects of various sizes which were utilizing various MPP. Of the five projects surveyed, the three which implemented MPP extensively experienced "significantly lower actual costs than one would expect had they used only traditional practices." Two of these three projects were 'large' and the other was considered 'small.' Each of the projects experienced greater than typical costs during the early development phases (i.e., requirements definition and design), and substantially lower than usual costs in the later phases (i.e., coding, testing, implementation). However, as Black states, "The lower costs incurred in the later phases clearly overbalanced the increased costs associated with the earlier phases."

Black's study concluded that the following practices were beneficial for both small and large projects:

- Formal assignment of tasks by the project manager
- Formal reviews
- Definition and early production of audience - specific documents
- Unit development folders
- Review prior to coding

- Coding conventions
 - interface conventions
 - code organization and comment standards

In addition to those practices just listed, the following practices were beneficial to large projects:

- Construction plan (Prepared after design, indicating activities and resources required during coding)
- Code verification
- Acceptance Testing
- Functional Testing
- Test Formalism
- Change Control Board

To further quantify the findings of the study, the three projects extensively using MPP reportedly experienced reductions in forecasted vs. actual labor costs of 57%, 80%, and 84%.

3.3.2 IBM Studies

IBM is one of few firms which has, in the past several years, collected some data about the employment of structured programming techniques (including a development support library, top down design, structured coding and chief programmer teams). Data was collected from 60 completed projects representing software development ranging from 4,000 to 467,000 source lines and 12 to 11,578 man months. The data showed a productivity improvement (using the "lines of code/time" metric, previously mentioned) averaging 70% between "little use" of new techniques and "much use." (1,12).

Another IBM study evaluated productivity (as measured by bytes of new code/total man months of effort) on projects with similar characteristics (e.g., people, experience level, management and support essentially the same). The project which employed structured programming concepts produced an average of 3756 bytes/man month. The similar project not employing all the structured programming concepts produced an average of 1161 bytes/man month. This research concluded that almost twice as much new code was produced with slightly half the effort (in spite of over 1100 formal changes made during development) (7).

3.3.3 Army Ballistic Missile Defense Program

Another study was conducted on the Army Ballistic Missile Defense program (1, 3). The program utilized the 'process design methodology' which embraced structured programming, top-down development, formal specification and analysis of requirements, simulation for evaluation of trial designs, tools to support requirements traceability, configuration management, library management, simulation control and data collection and analysis, and a management methodology to provide adequate designer/management visibility. The study concluded that

- The majority of errors were detected prior to 50% project completion.
- Productivity (as measured by machine instructions generated per hour) was considerably better than on comparable projects.

3.3.4 The TRW Study

Brown's study incorporated the knowledge and judgement of experienced practitioners in software developers within TRW. These developers were primarily responsible for software production in environments experiencing schedule and budgetary pressures similar to those observed within the DMA centers.

Brown's study of MPP (4) for TRW was two-faceted. The first part of the study surveyed the impact of 18 detailed programming standards on 30 characteristics of software or the development process. Some of the most significant results of this part of the study have been detailed below:

1. Routine size and program modularity have a strong positive relation to code auditability, understandability, maintainability, testability, operational reliability, error frequency and programmer productivity (undefined).
2. In-line commentary standards have positive or strong positive relation to code auditability, understandability and maintainability.
3. Structured coding has a positive relation to code auditability and understandability.
4. Naming conventions have positive relation to code understandability.
5. The overall weighted ratings for 18 programming standards against 30 software characteristics was 59% positive influence, 36% inconclusive, 5% negative.

The second part of the TRW Study surveyed the actual and theoretical impact of 11 MPP on 12 software problems. As Brown states, "There is strong agreement among project personnel as to the four MPP of greatest importance and impact and strong agreement on their relative ranking:

- Requirements analysis and validation
- Baselining of requirements specification
- Complete preliminary design
- Process design

The next three most highly ranked MPP were:

- Incremental development,
- Unit development folders, and
- Software development tools,

although the relative ranking of these MPP was inconclusive."

The TRW study also provided an indication of the overall impact of MPP on software quality and development cost. 85% of the survey participants felt that MPP contribute to the production of "higher than usual quality software." Almost 49% of the participants felt that MPP contribute to software development at "lower than usual lifecycle cost."

3.4 PRODUCTIVITY SUMMARY

While these studies utilized differing methodologies, employed various metrics of productivity, collected data by alternate means from different sources, and studied various combinations of MPP in use, all point to substantial increased development productivity via disciplined employment of MPP.

Most of the studies concentrated on efforts to develop new software, however, the results are also applicable to modification and upgrading which requires production of revised or new code.

After study of the DMA programming environments in relation to the conceptual Modern Programming Environment model, recommendations to implement some of the MPP discussed in this section were formulated. Whenever possible, applicable MPP with high productivity potential have been recommended. However, in the absence of firm historical records associated with some specific techniques and tools, some recommendations are based on data obtained from our experience and via interviews with DMA personnel.

In summary, the approach to recommendation of productivity enhancing MPP for the DMA was:

- familiarize study personnel with past successful productivity improvement activities
- study the DMA environment and identify unique characteristics
- determine which of the concepts underlying specific MPP are most appropriate for the DMA
- define an integrated program of activities within which the DMA can work toward systematic incorporation of modern methods and tools (i.e., an MPP evolution) and thereby avoid the possible ill effects of a too hasty MPP revolution.

4.0 RECOMMENDATIONS

This study was initiated in order to make recommendations which would enable the DMA to upgrade its present programming environment into a Modern Programming Environment (MPE). As such, it should be pointed out from the start that a true MPE possesses a multitude of characteristics which are not generally possessed by programming environments in existence today. The number of recommendations made in this section should serve to illustrate the enormity of the goal of converting any programming environment into an MPE. However, a true MPE will operate using the discipline and formality presented in these recommendations. For this reason, and because the DMA requested elaboration of tools, techniques and discipline in order to achieve a DMA MPE, recommendations were made in the eight subject areas described in this section.

In each of the subject areas, much study has been done by the software engineering community. There are techniques and disciplines in existence other than those described here, in each subject. The features described here are, to the best of our knowledge and experience, those which are the most applicable to the DMA environments.

It would be extremely unrealistic for any software development/maintenance environment to attempt to immediately implement all of the recommendations which are described in the following sections. However, in the same fashion, it would also be unrealistic for this study to result in making a few recommendations and expect that a DMA MPE would be accomplished. Therefore, a reasonably comprehensive list of modern programming practices which are felt to be applicable to the DMA as we have modeled it today has been recommended.

Section 5 deals with prioritization of the recommendations described here. Three recommendations are given highest priority because of the current characteristics of the DMA and the potential impact which they can have (if implemented as described in Section 4) on productivity and software quality. The remainder of Section 5 deals with other recommendations (as described here) which should be implemented to further improve productivity and quality and eventually achieve a DMA MPE.

4.1 SUMMARY OBSERVATIONS AND RECOMMENDATIONS

This section details the recommendations of the DMA Modern Programming Environment Study in eight main areas: Communication and Visibility, Product Quality Assurance, Facilities, Product Management and Control, Software Engineering Guidelines and Standard Practices, Tools and Software Development Aids, Technology Transfer and Training.

Each of the Recommendations is preceded by a summary of associated observations which has been extracted from Appendixes A and B. Because of the approach which was chosen to complete this study (comparing the programming environment of the DMA with a generic model of an ideal 'Modern Programming Environment'), some of the summary observations may appear to be highly critical of the DMA programming environments. This should not be negatively construed as a criticism of the DMA software environment. Many leaders in software engineering methodology possess these identical characteristics. The DMA should

be commended for recognizing that there are techniques which can be applied to modify these characteristics.

Since the recommendations incorporate many techniques and some tools, a list in condensed (outline) form has been provided in Appendix E.

The recommendations in this section often interrelate, and no attempt has been made to prioritize them, since the Implementation Plan documented in Section 5.0 illustrates interrelationships in a phased manner.

Although the recommendations do interrelate in some areas, each topic area (and recommended techniques and tools, which are subtopics) should be considered independent and thus may be implemented separately. Some of the benefits discussed in section 3 can be attained by implementation of any one of the following recommendations. No implication is being made that every recommendation should be immediately implemented. In fact, immediate, short-term and long-term implementation suggestions are discussed in Section 5.

4.1.1 Communication and Visibility

Summary Observations

At both centers, formal project reviews are seldom held, except for large projects. Standardized review procedures are not documented. When reviews are held, little advance notification of reviews is given to participants and information which will be discussed at reviews is rarely circulated in advance.

Few formal reviews or walkthroughs are held at completion of lifecycle phases or at the achievement of milestones in either closed or open shop projects.

The major formal mechanism for reporting project status at both centers is through the DMIS/P System (at HTC, a subset of DMIS/P, CSD MIS, is utilized). At both centers the reports by project are available to departmental, branch and project managers. At the AC the project manager from PP is also furnished a copy of the report.

Both centers have a monthly accounting program which categorizes and documents statistics about utilization of the Univac machines. One report produced identifies machine utilization by project.

At the AC, informal biweekly project status reports are produced for branch managers. Reports for large projects and projects with problems or unusual circumstances are furnished to the SDC division manager.

Reports of computer time expenditures at the project level are produced at both centers. However, it does not appear that these reports are systematically distributed to or reviewed by project managers. Consumption of storage space and the costs of handling, transporting and editing data are not visible and hence are not managed at the project level.

The software support organizations (CSD at HTC and SDC at AC) have little front-end, early on involvement with DMA customers in specification of computer resource requirements for major DMA products. Product requirements for software support (which enter the centers through the Program Objective Memo-

random (POM)) are generally vague. While the software support organization may be aware of an upcoming requirement for computer support, they lack visibility into the depth of support which will be required. The software support requirement for a major mapping, charting and geodesy (MC&G) product may be as generally phrased as "provide computer resources to support the implementation this mapping function."

As a result of this broad definition of future support requirements, planning and scheduling mechanisms are not as effective and accurate as they might otherwise be.

Recommendations

Recommendations pertaining to communication and project visibility are in two main areas: formal reviews and reporting. Visibility into production software support requirements is addressed through product specifications.

I. Reviews

For open and closed shop projects (except one-time, problem solving applications), reviews should be held at completion of each lifecycle phase. (The DMA software development lifecycle phases have been defined in SDCINST 5660.1 and DMAHTCINST 5660.3.) Large projects should additionally conduct reviews at the completion of each major milestone.

Review activities should begin before the review. A plan should be developed, materials needed during the review should be prepared and appropriately distributed. Documentation which is pertinent to the review should be distributed to the participants well in advance. Participants should have read the material, noting significant points and issues. The review plan should define:

1. The review objectives
2. Review participants
3. Responsibilities of the participants
4. Criteria necessary to meet the defined objectives
5. Review agenda and schedule

Review objectives should be in the form of decisions to be made as a result of a review. For example, the goal of a design review is to achieve design approval and a decision to begin coding.

Review participants should be chosen carefully. Some particularly important considerations when choosing reviewers are:

- Required technical skills
- Availability of individuals to devote time to pre-review and review activities
- Participation of individuals whose attendance is required for formal review approval

At least one individual present must represent the user/customer at each formal review. The user/customer must be aware of all major decisions, alternatives, problems, etc.

Participant responsibilities can be established by the technical skills of the individuals selected. For example, the leader of the design team is the logical choice to present the design materials during a preliminary or detailed design review.

Regardless of the purpose of the review or the technical skills involved, each review should be headed by a chairman. This individual is responsible for all pre-review activity coordination, review discussion, resolution of review problems/decisions and follow-up of review action items.

In order to successfully evaluate review decisions, supportive data must be collected and documented. All the information necessary to evaluate a decision must be made available. For example, design approval and the subsequent decision to begin coding must be based upon a number of factors. Each requirement must be addressed by the designed solution, the design must be complete, consistent, feasible, maintainable, and testable, the system parameters must not be exceeded (e.g., storage, timing), alternatives should be evaluated, etc. Normally, this information is part of the design document, and as such must be part of the set of review materials.

The review agenda and schedule should be documented and made available to participants before the review. Review agendas vary depending upon the type of review (e.g., requirements, design, etc.). A sample agenda for use during a requirements review is presented in Figure 4.1.1-1.

All review materials should be made available prior to the review. Just how long before the review these materials should be made available varies by the size of the project, the type of review and amount of materials to be studied. A rule of thumb is that the participants should be notified of the planned review at least 20 days in advance and they should receive review material from three to five days in advance. Large, complicated projects may require review material distribution as much as 20 days in advance.

During pre-review documentation perusal, individuals should note and report any major discrepancies/issues to the responsible review chairman. The chairman should have the opportunity to assign specialists to study the issue prior to the review whenever possible. A problem report form should be used to report major discrepancies noted. A sample Problem Report is illustrated in Figure 4.1.1-2.

As the sample agenda illustrates, a review does not end when presentations have been made. Most reviews uncover discrepancies and problems and some action is required for each one. It is important that all discrepancies and problems are agreed upon by the review team, documented as action items, assigned as the responsibility of an individual(s), and that corrective action is initiated. In addition, the response to each action item must be documented, evaluated and approved. Only after all action item responses have been approved can a review be considered complete.

The identification of a problem or discrepancy resulting in a review action item does not mean that other project work will be suspended until the action item

is resolved. Few action items are so major that their resolution will impact all other project activities. An action item is generally designated as the responsibility of one individual, and other project activities are given the "go-ahead" while the action item is being resolved.

The topic of reviews and associated procedures, standards and guidelines is a large one, the elaboration of which is a report in itself and will not be discussed here in greater detail. Good guidelines are available which could prove beneficial in establishment of a comprehensive and effective review program at the DMA. The following is a list of some applicable documents.

REQUIREMENTS REVIEW AGENDA

- 1) Call to order by the Chairman
- 2) Introduction of the participants
- 3) Reading of the decision making procedure and the procedure for conducting the review
- 4) Listing of the review materials
- 5) Presentation of user requirements by user requirements definition team
 - a) A summary of the problem that has led to the requirements
 - b) A description of the procedures that was used to define the requirements
 - c) A general description of the user requirements
- 6) Questions from the review team
- 7) Presentation of the requirements by requirements analysis team
 - a) An interpretation of the user requirements
 - b) The results of the computing team analysis which includes a feasibility study, preliminary estimates, etc.
 - c) A summary of the requirements
 - d) The contractor's planned approach for developing a solution to the problem
- 8) Questions from the review team
- 9) General discussion
- 10) Action items
- 11) Review team decision

Figure 4.1.1-1 Sample Requirements Review Agenda

SOFTWARE PROBLEM REPORT

NUMBER		1	
PROBLEM DISCOVERY	NAME OF FINDER	PROJECT	DATE
	2	3	4
DESCRIPTION OF SYMPTOMS			
5			
PROBLEM ANALYSIS	NAME OF ANALYST	DATE IN	DATE OUT
	6	7	8
ANALYSIS RESULTS			
9			
PROBLEM CORRECTION	NAME OF PROGRAMMER	DATE IN	DATE OUT
	10	11	12
SW FUNCTION:	SOLUTION DESCRIPTION		
☐ UTILITY ☐ COMPUTATIONAL ☐ LOGIC ☐ SIMULATION ☐ I/O ☐ EXEC ☐ OTHER	13		
	14		
NAME OF ROUTINE(S) CORRECTED		SIZE OF PROGRAM IN WHICH ERROR OCCURRED	
15		16	
TEST PHASE WHEN ERROR FOUND:	☐ MODULE ☐ INTEGRATION ☐ SYSTEMS	ERROR CATEGORY:	
S/W TOOL USED TO ISOLATE ERROR	AFTER RELEASE	☐ COMPUTATIONAL ☐ LOGIC ☐ DATA INPUT ☐ DATA OUTPUT ☐ DATA HANDLING ☐ DATA DEFINITION ☐ ROUTINE/ROUTINE INTERFACE ☐ SOFTWARE/HARDWARE INTERFACE ☐ DATA BASE ☐ OPERATION ☐ DOCUMENTATION ☐ DUPLICATE (PREVIOUS SPR# _____) ☐ OTHER (PLEASE EXPLAIN BELOW)	
HOURS TO LOCATE ERROR	COMPUTER	27	
	PERSON		
HOURS TO FIX ERROR	COMPUTER		
	PERSON		
ERROR OCCURED IN	☐ REQUIREMENTS ☐ DESIGN ☐ CODE ☐ RECODE		
CHANGES MADE TO: (CHECK ALL THAT APPLY)	☐ REQMTS ☐ DESIGN ☐ CODE		
IMPACT - LINES OF CODE CHANGED			
25			
PAGES OF DOCUMENTATION CHANGED			
26			

Figure 4.1.1-2 Sample Problem Report

DESIGNATORTITLE

MIL-STD-1521A (USAF)

Technical Reviews and Audits for Systems, Equipment and Computer Programs.

DI-A-3029

Reviews, Configuration Audits and Demonstrations

DI-E-3118

Reviews, Inspections and Audits

AF SCP 800-3

A Guide for Program Management

ESDR 27-4

Documentation for System Program Reviews

AFR 800-14
(Volume I)

Management of Computer Resources in Systems

AFM 175-118

Air Force Audit/Management System

II. Reporting

For open and closed shop projects, a regular reporting mechanism should be defined, standardized and formalized. The existing DMIS/P System is a mechanism which could be expanded and systematically applied to provide a good deal of pertinent status information.

In line with the Project Management and Control recommendation, all projects which are not of a one-time, problem solving nature should be entered into and tracked using DMIS/P. In addition, research and development projects which have an estimated duration of more than 2 man months should also be tracked in this manner.

To provide improved visibility into computer resource utilization on a project level, reports produced by the Monthly Accounting System (documenting expenditures at a project level) should be distributed to project managers. In addition, the potential for collecting and reporting minicomputer utilization statistics should be investigated. A significant improvement in minicomputer utilization can be effected if the resources are more closely monitored. While minicomputer turnaround time is not currently a problem, increased minicomputer usage is expected in the future. Anticipated turnaround problems can be avoided by a concerted effort at efficient usage now.

The project reports produced by DMIS/P should be distributed to project managers, branch chiefs, a representative of the user/customer and to a PP project manager for their scrutiny. The DMIS/P data should be systematically updated when changes occur. The updates should be monitored by the user/customer by scanning DMIS/P reports.

Individuals within CSD PMO and SDC PMO should be made responsible for distribution of the reports to the proper personnel, and effecting an updating mechanism which will take much of the burden from the project manager.

Monthly status reports should be produced by project managers (open and closed shop projects) and submitted to branch chiefs (or division managers, as

applicable), a representative of the user/customer and the responsible PP project manager. As a minimum, this report should include:

- Summary of project activities during the period
- Support personnel involved
- Problems which occurred during the period
- Person hour expenditures, actual vs. planned
- Planned activities for the next period

III. Product Support Visibility

The computer support organizations should be involved early in the specification of product requirements of which software is a part. When software is a subproduct of a major MC&G product, a representative of CSD or SDC PMO should be notified and should attend any initiation briefings or meetings which are held with the DMA customer and should also be supplied with associated documentation. The representative should identify major capabilities which the software organizations will be required to support. For example, does the plan include provisions for modification of an existing system, or will a new system be developed; does the plan require an expert(s) in a particular technology (e.g., database, networking), etc.

4.1.2 Product Quality Assurance

Summary Observations

For closed shop development at both centers, the user plays the role of independent tester and evaluator. Open shop software products are tested by the Techniques Office of the development department. Test strategies are usually informal in both open and closed shop environments, consisting of test runs with user supplied data, and test results are verified manually. The test group typically does not document test strategies in a Test Plan, except for large projects. Acceptance criteria and the results of acceptance testing are not documented.

Reviews or inspections of intermediate products (e.g., requirements specification, design document) do not take place in the open or closed shop environments at either center, except for large projects.

At the Aerospace Center, Instruction 5660.1 defines the functions of an independent SD Computer System Development Review Group (COMREV) at scheduled decision points during closed shop system development. At HTC, completed closed shop projects are specified to be reviewed by CSD PMO to assure complete documentation and adequate development testing.

Standards defining quality were not found at either center in the open or closed shop environments, consequently, QA mechanisms are centered around satisfaction of operational requirements and often do not address other critical characteristics of quality.

DoD Standard 7935.1-S provides an adequate definition of documentation qualities and has been adopted by both centers. Programs which were documented before the standard was adopted have not been revised to meet the standard and some programs recently documented do not strictly comply with the standard.

Recommendations

The centers should formulate a formal Quality Assurance Program which is applicable to both open and closed shop software development. Numerous government documents are in existence which discuss Quality Assurance in greater detail and a few such documents are listed below.

<u>DESIGNATORS</u>	<u>TITLE</u>
ESD-TR-77-255	Software Acquisition Management Guidebook: Software Quality Assurance
ESDR 74-2	ESD Quality Assurance Management
ESDM 74-1	Quality Assurance Management for Electronic Systems
AFR 74-1	Air Force Quality Assurance Program
MIL-S-42779	Software Quality Assurance Program Requirements
MIL-STD-1098	Quality Assurance Terms and Definitions
NBS Spec. Pub 500-1	Computer Software Management: A Primer for Project Management and Quality Control
MIL-Q-9858-A	Quality Program Requirements
RADC-TR-74-325	Engineering of Quality Software Systems

These documents will provide detailed information on the topic of QA, since it would be impractical and inappropriate to discuss QA in great depth in this report. However, some basic QA concepts and necessary ingredients of a thorough QA Program are discussed below.

1. Essential product qualities must be defined. Major qualities which are often included in QA program are performance, usability, and maintainability.

Performance properties address the following questions:

- Does the software meet the current specifications?
- Is it reliable? (Does it work upon all the data presented to it by the user?)
- Is it efficient? (in areas such as memory utilization and timing, relative to other similar software)
- Does the software interact well with the hardware environment of the total system? (Does it handle properly all permitted hardware actions?)

Usability properties address the following questions:

- Does the software provide a good man/machine interface?
- Can effective use of the software be achieved with a minimal learning curve?
- Can the software be used without a high degree of proficiency in some theoretical discipline?
- Is it friendly to the user? (e.g., if the user makes a mistake, can he find the source of the error readily?)

Maintenance properties are in the areas of understandability, program construction and testability:

- Is the code easy to read?
- Is the logic easy to follow?
- Is the software self-documenting?
- Was a good implementation language chosen for this type of system?
- Do the data structures provide visibility and insight into the software algorithms?
- Were good coding standards established and followed?
- Is the software functionally modular?
- Have all communication mechanisms and interactions between functions been well defined?
- Have the baseline testing requirements been defined for all software specifications?
- Does the software contain functional self-checking features?
- Does the software check the integrity of its data structures?
- Is it easy to maintain functional integrity while incorporating modifications?

2. From the list of software product qualities, those which the centers wish most to attain should be identified, formalized and standardized. The list of DMA standard product qualities should take into account the manner in which these qualities will be measured and the intended use of the software being developed or modified. For example, one-time applications may adhere to performance qualities and may not emphasize usability and maintainability issues. These software qualities should be expanded into check lists which can be used during reviews and audits to verify compliance with quality standards.

3. QA roles should be better defined at both centers. SD Instruction 5660.1 (from the Aerospace Center) provides a good beginning to a QA Program. The idea of a review group (COMREV) which performs reviews of system development at scheduled decision points makes a significant step toward assigning QA roles and responsibilities. However, participation in review activities should be expanded. Participants should include representatives of the Directorate of Programs, Production and Operations, other departments at the centers which are users of ADP services (e.g., the Geopositional Department at AC and the Geodesy Department at HTC), as well as the development departments/divisions. This QA function should be established at both centers for closed and open shop projects which are not a one-time, problem-solving application.
4. The QA function should conduct systematic reviews of all projects which are not of a one-time problem-solving nature. The reviews should be held at the conclusion of each major development phase. At both centers this should occur after initiation, definition, design (preliminary and detailed), programming, integration testing and installation testing activities. Checklists developed during earlier QA program activities should be utilized to verify compliance with established standards. Formal review procedures (as discussed in the Communication and Visibility Recommendation) should be followed to ensure maximum review effectiveness and productivity.
5. QA audits should be conducted by the QA function. These audits should be unscheduled and should occur during operational and development phases. Again, checklists should be developed and utilized for better effectiveness. Audits should be less formal than QA reviews in order to reduce disruption of development activities. An effective means of development auditing which reduces disruption of project activities centers around review of the project notebook. The project notebook, as discussed in the recommendation for Standard Engineering Practices, should contain *vital project information* and therefore, should be a good source of data for a QA audit.
6. Acceptance criteria should serve as the basis for formal project approval. These acceptance criteria should be stipulated by a team representing user, customer, and developer at the time that the system requirements are defined and documented. Each requirement should have an associated acceptance criteria which is documented in the requirements specification or Functional Description document. Figure 4.1.2-1 illustrates a Sample Requirements and Acceptance Criteria Specification. This type of representation scheme is consistent with the Functional Description Document as described in DoD 7935.1-S. In fact, Section 3 of the document as specified by the standard calls for a delineation of requirements and the associated validity and accuracy criteria.

Specification of acceptance criteria reduces the potential for ambiguous or inconsistent requirements. If a requirement does not have an associated acceptance criteria, it has been improperly stated and chances are that it will never be completely satisfied. Figures 4.1.2-1 and 4.1.2-2 illustrate Sample Requirements and Acceptance Criteria Specifications for two types of requirements. The requirement in figure 4.1.2-1 has acceptance criteria which were fairly obvious from the requirements description. The requirements description in figure 4.1.2-2 does not provide readily apparent acceptance criteria. However, the absence of acceptance criteria for the

REQUIREMENT #N DATA PURGE PERMISSION

SOURCE Statement of Work Item 1.3

DESCRIPTION

Special permission shall be required to purge any data set which has a dependent data set while these data sets are under project control, where data set B is dependent on data set A if, and only if, data set A is input to the process or processes which produce B.

ABSTRACT OF THE ACCEPTANCE CRITERIA

The data set purge permission requirement will be tested by attempting to purge data sets which have dependent data sets both with permission and without permission.

Figure 4.1.2-1 Sample Requirements and Acceptance
Criteria Specification

REQUIREMENT #N+1 FUNCTIONAL LANGUAGE

SOURCE Statement of Work Items 1.4 and 1.5

DESCRIPTION

The user languages (e.g., command, query, edit, etc.) shall be user oriented and consistent in vocabulary and syntax. Defaults shall be employed when practical and be directed towards the experienced user.

ABSTRACT OF THE ACCEPTANCE CRITERIA

A user opinion survey will be used to evaluate the user languages for all modes of operation during acceptance testing. The evaluation criteria will include: consistency in vocabulary, syntax and format, ease of changing from one user language to another, and ease of use. These criteria will first be evaluated during the user language design phase and subsequently tested concurrently with most of the other requirements, with each user completing a check list for each mode of the user language that is tested.

In addition, tests must be conducted to verify that the defaults in the user language are operational, and that they are directed to the experienced user.

Figure 4.1.2-2 Sample Requirements and Acceptance
Criteria Specification

second requirement would have made validation of the requirement impossible.

7. Discrepancy reports should be issued and monitored by the QA function to document policy violations in software products, documentation, test procedures or other anomalies encountered during QA activities. In conjunction with this, a discrepancy report log which summarizes the status of all discrepancy reports issued during software development should be kept. This log should be used to analyze the frequency and types of deviations from QA policies in order to refine and update the QA Program as needs of the centers evolve and change.
8. Configuration Management Procedures, as documented in the Product Management and Control Recommendation should be developed to support the QA Program. Adherence to configuration management procedures should be reviewed during QA reviews and audits.
9. A centralized QA file should be kept. This file should contain records of QA reviews, action items and their response, and the discrepancy reports and log.

A DMA QA Plan should be developed which addresses all the above points plus any others that are deemed necessary to assure the quality of software being produced at the DMA centers.

With the establishment of a QA program comes the question of which organization should bear the burden of the expense of maintaining QA procedures, conducting QA reviews and audits, and support of the QA program itself. Each project will benefit from the QA program by better product performance, reduction of subsequent maintenance expenditures, and increased usability resulting in greater user satisfaction. Therefore, one solution is for each project to allocate a share of its program budget at program initiation to cover QA related expenditures.

Finally, the QA program should include a provision for study of current QA techniques and automated tools which will ultimately assist program activities. For example, a tool such as PFORT (developed by Bell Labs) audits FORTRAN source code for adherence to American National Standards Institute (ANSI) standards. This tool could be incorporated into the QA program to replace manual auditing of adherence to ANSI programming standards. Other such tools have been developed which audit conformance to specific company standards. For example, TRW contracted for development of a Code Auditor for use in verifying conformance to programming standards instituted on the Site Defense Program. The DMA should consider developing or contracting for development of an auditor which can be used center-wide to check adherence to programming standards (as specified in the Software Engineering Guidelines and Standard Practices Recommendation).

4.1.3 Facilities

Summary Observations

Both centers are currently processing a large number of jobs per month (at the Aerospace Center, over 30,000, at the Hydrographic/Topographic Center, about 9,000) on the mainframe computers.

Both centers are experiencing lengthy turnaround times on the mainframe computers (although the advent of the Harris RJE Station at HTC has alleviated this problem to a degree at that center). At AC the average turnaround time is 19 hours, with estimates falling in the range from 12 to 24 hours. At HTC the average turnaround time is six hours, estimates falling in the range from four to ten hours.

At both centers, the majority of jobs are submitted via over-the-counter batch. At the time this study was initiated, there were few demand terminals (Uniscopes) available at either center. However, during the latter part of the study both centers acquired a number of terminals. This should change the mix of batch and demand job submittal. Since these terminals have only recently been acquired, data documenting the usage of these terminals and their impact on software development and job turnaround was unavailable. Also, it remains to be seen whether acquisition of these terminals will alleviate the need at both centers for demand processing.

Many of the jobs submitted access existing operational production programs. At the Aerospace center there are approximately 600 production programs being maintained by the Scientific Computer Division. At the HTC, the collective holdings of CSD libraries contain about 2,150 applications programs and associated data. The Monthly Accounting Report generated at AC during January, 1979, documents the Center's heavy usage of production programs. Thirty-two of the 600 production programs at the center accounted for 62% of the total runs during the period. The ten most frequently accessed programs accounted for 44% of job-tabulated Standard Unit of Processing (SUP) time usage during the month.

Many of the production programs are lengthy. At the Aerospace Center the average number of lines of code is about 1,000-2,000 per production program, with the most frequently accessed program containing about 7,000 lines. Most of these programs are written in FORTRAN.

Efforts have been made to optimize these production programs. At the Aerospace Center some automated tools (e.g., Boole and Babbage's UPE and FEDSIM's FORTRAN Instrumentation Package) have been used to gather program execution statistics in order to optimize programs being studied. At both centers, efforts were made to analyze the data structures being utilized by several programs to investigate potential reduction in core storage requirements. The results of one such effort was quite successful. At the AC the most frequently accessed production program (ULB-306, Terrain Processor Program) was analyzed and optimized by reducing core requirements. The optimization resulted in a core storage reduction of approximately 40%.

While the potential for program optimization (using automated or manual analysis) is high, there is not any systematic optimization program being employed at either center.

During research at the centers in February, FEDSIM had released its preliminary findings in a study of Univac jobs which errored off (failed to terminate normally). The error off rates which were reported by FEDSIM were very high, much higher than rates incurred by other agencies and within the software development community. Since these rates could potentially indicate problems within the software development environments, the methods used by FEDSIM to collect data and the data collected by that Study was requested for further research. For example, if the reports indicated that the use of a particular

practice caused abnormal job termination, then a recommendation would be in order in regard to the use of that practice.

Preliminary results of FEDSIM's study were verbally given to the MPE study team indicating that approximately one-third or 33% of HTC's production runs errored off. The figure given for AC was about 28%. Since that time, the MPE study group requested and received a copy of the Error Rate Summary Report issued for Univac jobs at the HTC during February. The report indicated that the total error rate was 25%, while departmental rates ranged from 16 to 34%. The report classified errored off runs into four categories: development, test, operational and other. Of the six departments which regularly utilize the Univac, the Topography Department had the largest percentage of total and operational errored off runs. The Geodesy and Surveys Department had the largest percentage of errored off development runs, and the Directorate of Programs, Production and Operations had the largest percentage of test runs. The Computer Services Department ranked 3rd, 5th, 6th, and 4th in development, test, operational, and other errored off runs, respectively.

Both centers were charged with investigating the causes of the high error off rates, therefore, the MPE study group did not investigate the situation more thoroughly. Univac Error-off Study results were obtained in early May by the MPE group.

Study statistics were classified into the following categories:

1. User Controllable Errors. Caused by carelessness in keypunching, bad or missing input data, intentional errored off, Executive Control Language errors, and user controllable tape errors.
2. Tape Errors. Caused by tape parities, bad tapes, noise, hardware, or unexplained.
3. Operating Procedure Errors. Errors attributed to a mistake by user or operating personnel in an operating procedure.
4. Program Errors. Errors relating to programs which are developmental or are undergoing test.
5. Unknown Errors.
6. Erroneous Normal Fins. Jobs which were reported as normally terminating, but which errored off.

The Error Off Study Group identified 686 of 2700 operational jobs (25.4%) which errored off. Table 4.1.3-1, which categorizes the errors, was extracted from the study findings.*

*Note: In May 1979, after implementing some of the recommendations of the error off study, the AC experienced an error off rate of 17.6%, a substantial improvement over the previous figure of 28%.

TABLE 4.1.3-1 SUMMARY OF ERRORED OFF RUNS

<u>CATEGORY</u>	<u>% of all Operational Job Runs</u>	<u>% of Errored Operational Runs*</u>	<u>% of Total CPU Time</u>
User	12.1	51.9	13.0
Tape	5.2	23.3	7.7
Operating Procedure	1.9	8.2	.8
Program	1.4	6.2	6.9
Unknown	2.7	11.4	1.4
Normal fin Jobs which Errored	2.0		1.5
TOTAL	25.4		31.4

*This percentage is based on total errors derived from the accounting reports. It does not include jobs which errored but terminated normally.

The Error-Off Study concluded with a number of recommendations to reduce the error off rate. These recommendations will not be repeated here, but have been reproduced in Appendix C for reference.

As Table 4.1.3-1 illustrates, 23.3% of the errored operational runs (about 40% of total errored CPU time) is attributable to tape errors. Both centers handle an enormous amount of data, almost exclusively being stored on tape. The collective holdings of CSD libraries approaches 40,000 tapes and 43 disk packs. At SDC, the number of tapes in the library is upwards of 30,000, approximately 900 of which are mounted during a typical day.

The Harris RJE terminal at HTC has been well received and is currently processing a substantial amount of central site workload. No similar facility which allows developer submittal of jobs is available at AC, although the center does have four DCT 2000 remote terminals.

Recommendations

1. A systematic plan toward production program optimization should be established at both centers to reduce mainframe utilization by production programs. This plan should incorporate a mixture of automated and manual techniques in order to achieve the maximum obtainable benefits. The plan should be formulated by a group whose primary charter is to carry out analysis and modification of the most frequently accessed production programs. For the purposes of this recommendation, this activity will be

referred to as the Resource Utilization Optimization (RUO) function. The functions of RUO should be ongoing, in support of newly developed production programs as well as existing programs. The RUO should support developers of production programs when questions arise concerning the effective operation of programs.

Most of the resource optimization tools increase the execution time of the program during the optimization analysis run by 25 to 50%. However, experience has shown that the benefits which these tools provide in terms of reduced usage of computer resources far outweigh the initial run costs. For example, the Federal Aviation Agency (FAA) used an optimization tool to free up \$471,562 worth of computer time during a two-year period (ref. 14). A data flow optimization package was utilized by NASA to free up 18 hours of computer time per day on two machines, an annual savings of about \$660,000 (ref. 14).

A Resource Utilization Optimization Plan should be developed to address the following issues:

- A. Identification of existing production programs which are accessed the most and, consequently, where the potential benefit from optimization is greatest. At AC this can be done by regularly reviewing the Utilization by Unique Program Number report produced by the Monthly Accounting Program. The Monthly Accounting Program used at HTC does not produce a similar report. Therefore, the Technical Support Division should implement the Program used by AC or study other available information to obtain data identifying frequently accessed programs.
- B. Analysis and definition of requirements for acquisition or development of automated tools or manual techniques which support program optimization. Requirements should be identified in a number of areas:
 - Source languages which will be input to the tool(s) (e.g., FORTRAN, COBOL, Assembly language)
 - Level of detail of the required analyses. For example, some tools operate at the object code level while others relate directly to the statement, paragraph or subroutine level of source code.
 - Type of analyses to be performed. For example, some tools analyze time, space, input-output transactions, file accesses, file or database organization, etc.
 - Good human engineering characteristics (e.g., easy to use, addresses a perceived problem, complements existing programming environments, tolerant of user mistakes and/or errors).
 - Good support characteristics (e.g., reliable, dependable, predictable, well documented and developer supported).
- C. Acquire or develop optimization tools and techniques which satisfy established DMA requirements (as analyzed in activity B, above). Some automated optimization tools are currently available throughout the

industry. Examples of such tools include the CAPEX Optimizer (CAPEX Corporation), PET (McDonnell Douglas Corporation), REBEL and FLEX (Advanced Computer Technology), SNOOPY (Univac UPLI) and FORTRAN Instrumenter I and II (Softool Corporation).

- D. Systematically analyze identified programs by manual and automated methods. Use the optimization tools and techniques to conduct analyses of the frequently accessed production programs. Run statistics should be kept so that cost/benefit studies of program execution before and after optimization can subsequently be made. Run statistics should include elapsed SUP and CPU time, core storage requirements, auxiliary storage requirements, input-output transactions and file accesses.
 - E. Modify the programs using results of analysis. Study the results of manual and automated optimization analyses to identify changes which should be made to existing production programs. Incorporate the identified modifications into the production programs, using established configuration control methods (see Product Management and Control Recommendation).
- 2. To alleviate the high error-off problem, the centers should begin by implementing the recommendations made by the HTC Univac Error Off Study group and reproduced in Appendix C. In addition, the centers should consider changing the primary storage media from tape to disk. As documented earlier in this section and in Appendixes A and B, the centers are using tapes almost entirely as the storage medium for the vast amount of data which they are processing. Other government agencies which must process a large amount of data have converted to disk as the primary storage medium (e.g., the Social Security Administration).
 - 3. Due to the way that the Harris RJE station has been received and is being utilized at HTC, the potential for acquiring a similar station at AC should be investigated. This could alleviate some of the turnaround problems at the center, although turnaround time at HTC is due to differences in the number of jobs processed.

The acquisition of interactive terminals at the centers should make the job of the computer organizations which support maintenance activities much more productive. Interactive submittal of maintenance jobs will certainly enable the computer support organizations to be more responsive to requests for minor changes to existing software. Many commercial organizations which process a substantial amount of maintenance jobs utilize demand job submittal in conjunction with some type of software update package.

If interactive software maintenance activities at the DMA proceed similar to activities at commercial software houses (and indications are that it will), the DMA can expect increased productivity in maintenance projects. The centers should develop a centralized facility for collection of information which will enable them to justify the acquisition of additional demand terminals.

4.1.4 Product Management and Control

Summary Observations

Few formal methods to control the completed products of software development or subproducts under development exist at either center in open or closed shop environments. Document products are controlled to some degree by DoD Standard 7935.1-S.

While development of closed shop projects is divided into phases (reference SDCINST 5660.1, DMAHTCINST 5660.3), clear products of these phases are rarely identified as deliverables. Open shop projects are even less formalized, and phased development is not usually employed.

At each of the centers, production programs play a major part in daily computer usage. Each center has a program library where some documentation and software resides. Responsibility for documentation distribution is vested in the computer support organizations (SDC and CSD). However, final project products (including documentation, programs and data) are not controlled by formal mechanisms in either open or closed shop environments. Different versions of production programs sometimes exist, but are not documented or maintained as such. Changes to existing program documentation necessitated by changes or updates to operational production programs are not coordinated in a formal manner. Some production program users possess program documentation which is invalid due to changes which have been made since the documentation was originally obtained. Lists of users of particular production programs which would aid in distribution of change memorandum are not kept at either center. There does not appear to be any systematic method for controlling changes to production programs, or distributing changed documentation to program users. No function similar to that of a program librarian is employed.

During software development, formal change control mechanisms for products are not used. No framework is used to maintain the integrity of all program deliverables (key program documentation and software). Guidelines for adding, modifying and deleting items are not formalized. Maintenance and tracking functions necessary for providing management and technical visibility over evolving products are not employed. Neither open or closed shop developers utilize change control levels.

Recommendations

The importance of operational production programs to each of the centers necessitates formal change control (configuration control) procedures and delegation of change control roles and responsibilities.

Each of the centers should establish a configuration control function within the computer support organizations (SDC and CSD) and subsequently formulate a Configuration Control Plan. In addition, a centralized DMA configuration control function should be established at the Headquarters level.

The Configuration Control Plan should incorporate many concepts. Units of work to be controlled (Configuration Items or CIs) should be identified. CIs are a major piece of work, usually a specific document, unit of software or a data set.

Controlled documentation should include master copies, figures, charts and other related material. Controlled software materials should include listings, source/object code, module workbooks, test cases and test results. Guidelines should be developed to ensure that no unauthorized or unrequired modifications are made to CIs. For most large programs, these guidelines should revolve around established levels of control.

Levels which are commonly employed are: release control, project control and author control.

Release control is the most formal and applies to all software and documentation products after delivery to the user/customer or after installation. Deliverable items undergoing review or test should be placed under project control. Some non-deliverable items (e.g., support tools, project standards and procedures) may also be placed under project control. All items which are still being developed or are in a draft form should be placed under author control. Figure 4.1.4-1 depicts the typical flow of project documentation and software from Author Control through formal Release Control.

Working materials are generated by the author(s) and evolve into rough drafts of documents and/or individual software modules. Software should remain under Author Control through the module design and test phases. When documentation evolves to the form of a Review or Preliminary Copy or software progresses to system integration and test (Evaluation Phase as specified in SDCINST 5660.1 and DCSINST 5660.3), Project Control begins. Relevant material (in the form of a CI) should be turned over to the configuration control group and a notation should be made in a master file that contains necessary information for tracking the item's status. Further additions or changes must then be made in accordance with established procedures.

Once an item has been formally accepted by the user department or the customer, it should be elevated to the Release Control level. Once placed under this level of control, an impact analysis study must be performed and reviewed by the configuration control function and the project management before changes may be applied.

Three types of forms should be designed and employed to assist configuration control. The Configuration Change Request (CCR) form is used for formal submittal of controlled changes. A sample CCR is illustrated in figure 4.1.4-2 and an explanation of items on the form has been made on the following page. An Impact Analysis form (see fig. 4.1.4-3, Sample Impact Analysis Form) should be used to report the estimated impact of making the requested change in terms of modification to the product schedule, cost and design. A Software Problem Report form (see fig. 4.1.4-4, Sample Software Problem Report) should be used to report detailed information about the identification, analysis and solution of software problems. Initially, the form should contain a description of the problem and its symptoms. As the problem is analyzed and corrected, the form should be used to record relevant information.

The centers should consider establishing a Change Control Board (CCB). The CCB should be responsible for general direction during preparation of change materials, review of change materials, and recommendations to project management for change approval or rejection.

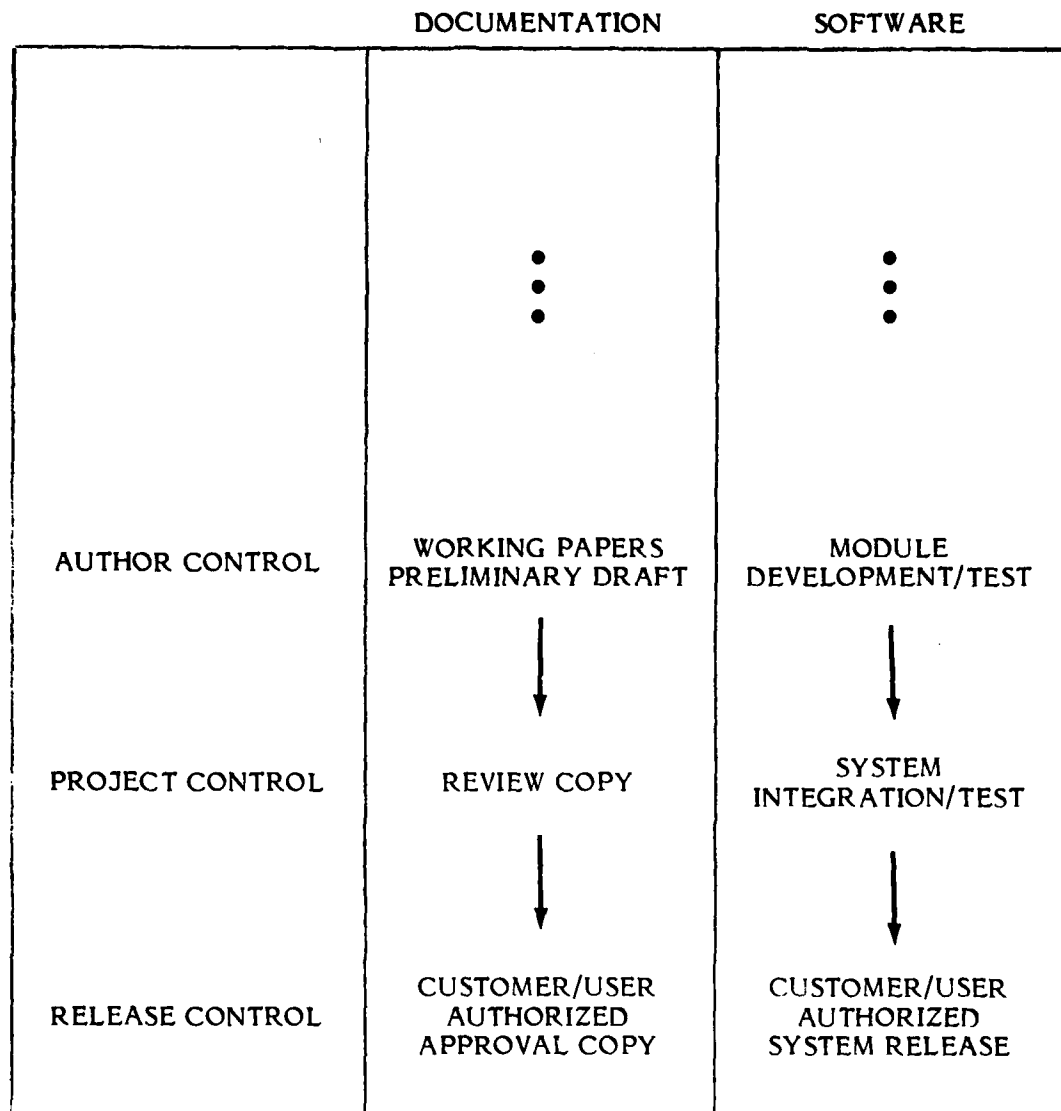


Figure 4.1.4-1 Flow of Software and Documentation Through Configuration Control Levels

CONFIGURATION CHANGE REQUEST (CCR)	
1 Originator _____ Company _____ Phone _____ Group _____ Change Request No. 9 _____ Date Originated 10 _____ Date Completed 19 _____ Approval Required by 12 _____ 11 CONTROL LEVEL * ☐ I ☐ II ☐ III	2 Change Request for (Document No./Software ID/Other): _____
3 Type of Modification: ☐ Addition ☐ Change ☐ Deletion ☐ Other	
4 Function Area _____	
5 Responsible Manager _____	
6 Requested Changes (attach sheets, forms, keypunch, etc., as required): _____ _____ _____ _____	
7 Reason for Change (use additional sheets as needed): _____ _____ _____	
8 ANTICIPATED IMPACT ☐ Major ☐ Minor _____ _____ _____	
13 IMPACT ANALYSIS _____ _____ _____	
APPROVAL 14 ☐ Responsible Manager _____ 15 ☐ CCB Chairman _____ 16 ☐ PP Project Manager _____ 17 Other _____ Signature & Date _____	18 CCR is: ☐ Authorized ☐ Rejected
Implementation of Change (Signature & Date) Configuration Control Coordinator 20 _____ Program Manager 22 _____	
Revision Name or ID 21 _____	

*Level I - Release Control
Level II - Project Control
Level III - Author Control

Figure 4.1.4-2 Configuration Change Request (CCR)

CCR Form Explanation

ORIGINATOR

1. Enter name, affiliation, telephone, group
2. Enter a description of the item for which a change request is being requested
3. Enter type of modification being requested
4. Enter function area in which change is to be made
5. Enter name of responsible manager
6. Describe in detail actual changes to be made
7. Explain reason for change
8. Enter and explain anticipated impact of change
9. Assign CCR number from CCR Tracking Log
10. Enter date received
11. Enter control level
12. Enter date approval is required by

RESPONSIBLE MANAGER

13. From Impact Analysis write a brief statement of total anticipated impact
14. Sign and date to signify concurrence

CCB CHAIRMAN

15. Sign and date to signify concurrence

PROJECT MANAGER

16. Sign and date to signify concurrence

OTHER RESPONSIBLE MANAGER

17. Sign and date to signify concurrence

CUSTOMER/USER PROJECT MANAGER (Level 1 only) AND PROJECT MANAGER

18. Indicate CCR acceptance or rejection

CONFIGURATION CONTROL FUNCTION COORDINATOR

19. Enter date change was completed
20. Enter notification that CCR has been incorporated in next release
21. Enter number or name of new release which will contain this update

PROJECT MANAGER

22. Enter notification that CCR has been incorporated in next release

IMPACT ANALYSIS FORM ① REFER TO _____ DATE _____ 	
CONTENT OF CHANGE PACKAGE ②	APPROVAL BY: Configuration Control Board name _____ date _____ PP Project Manager name _____ date _____ CONTRACT CHANGE ☐ NO ☐ YES
MANAGEMENT SUMMARY ③	
TECHNICAL IMPACT ④	
RETEST REQUIREMENTS (For Software Only) ⑤	
PROJECT SCHEDULE IMPACT ⑥	IMPLEMENTATION SCHEDULE ⑧
COST IMPACT ⑦	
⑨ _____ Program Manager	

Figure 4.1.4-3 Sample Impact Analysis Form

Impact Analysis Form Explanation

The Impact Analysis Form is to be filled out and attached to each CCR.

PROJECT MANAGER

1. Fill in CCR number that this summary refers to and the date of origination

ORIGINATOR

2. Enter brief explanation of content of the change package

RESPONSIBLE MANAGEMENT

3. Fill in management view of proposed CCR impact

RESPONSIBLE TECHNICAL MANAGER

4. Fill in technical view of impact of proposed CCR.
5. If change involves modification to software, fill in the needs for retesting the software.

OTHER RESPONSIBLE MANAGEMENT

6. Enter schedule impact of this CCR
7. Enter cost impact of this CCR

TECHNICAL MANAGER

8. Enter the schedule for implementing the requested changes

PROJECT MANAGER

9. Enter concurrence that all parts of the summary that need to be filled in are present and the report is comprehensive and legible

SOFTWARE PROBLEM REPORT

		NUMBER (1)	
PROBLEM DISCOVERY	NAME OF FINDER (2)	PROJECT (3)	DATE (4)
DESCRIPTION OF SYMPTOMS			
(5)			
PROBLEM ANALYSIS	NAME OF ANALYST (6)	DATE IN (7)	DATE OUT (8)
ANALYSIS RESULTS			
(9)			
PROBLEM CORRECTION	NAME OF PROGRAMMER (10)	DATE IN (11)	DATE OUT (12)
SW FUNCTION: (13) ☐ UTILITY ☐ COMPUTATIONAL ☐ LOGIC ☐ SIMULATION ☐ I/O ☐ EXEC ☐ OTHER	SOLUTION DESCRIPTION (14)		
NAME OF ROUTINE(S) CORRECTED (15)		SIZE OF PROGRAM IN WHICH ERROR OCCURRED (16)	
TEST PHASE WHEN ERROR FOUND: (17)	☐ MODULE ☐ INTEGRATION ☐ SYSTEMS	ERROR CATEGORY: (27) ☐ COMPUTATIONAL ☐ LOGIC ☐ DATA INPUT ☐ DATA OUTPUT ☐ DATA HANDLING ☐ DATA DEFINITION ☐ ROUTINE/ROUTINE INTERFACE ☐ SOFTWARE/HARDWARE INTERFACE ☐ DATA BASE ☐ OPERATION ☐ DOCUMENTATION ☐ DUPLICATE (PREVIOUS SPR# _____) ☐ OTHER (PLEASE EXPLAIN BELOW)	
S/W TOOL USED TO ISOLATE ERROR (18)	AFTER RELEASE		
HOURS TO LOCATE ERROR	COMPUTER (19) PERSON (20)		
HOURS TO FIX ERROR	COMPUTER (21) PERSON (22)		
ERROR OCCURED IN (23)	☐ REQUIREMENTS ☐ DESIGN ☐ CODE ☐ RECODE		
CHANGES MADE TO: (CHECK ALL THAT APPLY) (24)	☐ REQMTS ☐ DESIGN ☐ CODE		
IMPACT - LINES OF CODE CHANGED (25)			
PAGES OF DOCUMENTATION CHANGED (26)			

Figure 4.1.4-4 Sample Software Problem Report

Software Problem Report (SPR) Explanation

CONFIGURATION CONTROL FUNCTION

1. Assign an SPR number

ORIGINATOR

2. Enter name of originator
3. Enter project name
4. Enter preparation date of SPR
5. Enter a description of the problem. Include reference to specifications (e.g., Requirements or Design Documents or User's Manual) which are not being met.

ANALYST

6. Enter name of analyst
7. Enter the date analysis began
8. Enter the date analysis was concluded
9. Enter a description of what caused the problem. Include reference to design and/or code which fails to meet specification.

PROGRAMMER

10. Enter name of programmer
11. Enter date problem solution was started
12. Enter date problem solution was tested
13. Enter the type of software function performed by routine(s) corrected from the following:

Utility - general purpose library routine performing bit manipulation, trig function, sorting, etc.

Computation - arithmetic computations are primary functions performed

Logic - examples of primary functions are operator, sequential or conditional testing

I/O - primary function of software is to perform input/output function

Exec - primary function of software is to perform functions such as task scheduling, interrupt supervisor handling, input/output tasks or database manipulations

Other - include appropriate function

14. Describe the type of solution developed to solve the problem
15. List the routine(s) corrected while resolving the problem
16. Size of program (designate words, halfwords, or bytes) in error

ORIGINATOR/ANALYST

17. Enter test phase when error was discovered

ORIGINATOR

18. Enter software tool used to isolate error (environmental simulator, analyzer, data reduction program, trace program, etc.)

ANALYST

19. Enter manhours spent on computer to locate source of problem
20. Enter manhours off the computer spent to locate source of problem

PROGRAMMER

21. Enter manhours spent on the computer to develop a solution and to test it
22. Enter manhours spent off the computer to develop and test the problem solution
23. Enter software development phase where error occurred
24. Enter what was changed, software requirements, software design and/or resulting code to fix the problem
25. Enter number of lines of code changes (if any)
26. Enter number of pages of documentation changes
27. Enter a category for the type of error that occurred

The functions of a program librarian should also be addressed by the configuration control plan. These functions should include keeping track of all programs and associated documentation and tracking the status of all change requests.

Parts of this function can be supported by existing software aids. For example, dictionary and program library tools have been developed and are currently available. Existing tools should be evaluated for their suitability as the plan is being developed. IBM's Program Support Library (PSL), which is currently being installed at the DMA centers, should be included in this evaluation.

The configuration management concept, while being rather new in the field of software development, has been applied in the business world for many years. Configuration management tools have been developed at individual sites and companies throughout industry. The tools are identified and described in such sources as Data Pro and Auerbach. The DMA should institute a study of these sources to identify tools which could be applied to their environment.

4.1.5 Software Engineering Guidelines and Standard Practices

Summary Observations

SDCINST 8420.39, 'Programming Guidelines,' was published in August, 1975, at the Aerospace Center. This instruction describes the responsibilities of closed shop software developers and defines programming guidelines and standards. These development guidelines encourage top-down development, modular programming, requirements, design and coding walkthroughs, documentation during development (in compliance with DoD 7935.1-S) and team reviews. Programming guidelines restrict usage of the following:

- Use of variables in COMMON
- Data type dependencies
- Excessive passing of parameters between modules
- Assembly language programming
- Temporary work files
- Non-standard I/O interfaces
- Mixed programming languages
- Backward jumps and GO TO statements

The instruction encourages the use of:

- Ascending statement numbering
- Meaningful variable names
- Explicit control statement options
- DATA statements to initialize arrays

There is no mechanism to enforce these guidelines and, consequently, they are not followed regularly. Also, the guidelines are not specified at a sufficient level of detail to prohibit violations.

No similar instruction was found within the AC open shop environment or in either environment at the Hydrographic/Topographic Center.

Standards or guidelines addressing the following topics are not found at either center in the open or closed shop environments:

- Acceptable naming practices
- Data formatting conventions
- File formatting conventions
- Flowcharting conventions
- Preface or body commentary standards (code documentation)

At HTC, documented standard operating procedures were found in the following areas:

- Job submittal and processing
- 1108 turnaround and turnaround reporting
- Interim tape system procedures
- Restrictions in 1108 computer room
- Harris terminal usage
- Uniscope usage

In the closed shop environments at both centers, a phased development approach has been formalized and is encouraged (ref SDCINST 5660.1, DMAHTCINST 5660.3). However, there is no formal mechanism to ensure conformance to the defined approach and, consequently, it is not rigorously followed.

Neither center maintains project software notebooks or has standardized the contents of such notebooks or a project file.

A central document containing software product standards and detailing center facilities and computer procedures was not found at either center. At HTC, the Technical Support Division has documented a preliminary version of a handbook containing information about:

- Available equipment
- Available software
- Available documentation
- Procedures
- Computer output microfiche

The handbook has not yet been completed and is not currently available for use.

Few mechanisms exist to verify compliance with existing standards or procedures at either center. Present staff activities are centered around satisfying production needs and little time remains to enforce compliance with procedures or to check for achievement of standards.

The open shop environments at both centers have no formal and few informal programming guidelines and the degree of standardization, use of modern techniques and application of a phased development approach vary according to the individual in charge of the project effort.

At both centers, maintenance is difficult and time consuming. Modification of a program requires that extensive effort be spent locating information in documentation and interpreting the unique organizational characteristics of the code.

The only standard which was widely recognized (and less widely applied) was DoD 7935.1-S. While open and closed shop developers at both centers have been instructed to follow the standard, a formal method for verifying compliance with the standard does not exist. Documentation prepared before the advent of the standard has not been updated to conform with the standard, and some documentation currently being produced does not strictly adhere to the format/content requirements of the standard.

At HTC, most closed shop projects are initiated via the ADP Work Request Form (see appendix fig. B-1). Projects at AC are initiated via a letter or memo. A detailed specification of program requirements is not usually produced at either center, except for large projects. As a result, project requirements are often misunderstood and subject to extensive change throughout the development effort.

No standard methodology is used to represent program design. When design is represented, flowcharts or textual descriptions are used. Although top-down design is encouraged, there is little evidence of its actual application at the DMA.

Recommendations

This section contains recommendations for formalization of standards and procedures to be employed during software development. These recommendations provide guidelines which should be used by the DMA.

1. To support improvement of the understandability and the maintainability of the programs developed at the centers, it is recommended that a comprehensive set of standard programming practices be developed and uniformly enforced. SDCINST 8420.39 contains some of the standards which should be addressed. However, the defined programming guidelines should be more detailed to permit better interpretation, applicability, and enforcement. For example, the guidelines state that the development process should include reviews and walkthroughs. However, standards for conduct of such reviews and walkthroughs are not stated or referenced.

A new standard should be written, or Standard 8420.39 should be updated and made applicable to closed shop environments at both centers. In addition, elements of the standard should be made applicable to open shop developers. Some of the standard is not currently transferable to open shop users of minicomputers (e.g., Univac Programming Standards), yet analogous standards can and should be developed.

Whether the existing AC standard is revised or a new standard is written, the following areas should be addressed:

- Internal program-subroutine documentation (content and format)
- Code structuring and construction guidelines

- Program variable naming conventions
- Internal data structuring
- File formatting
- Program modularity
- Program parameterization
- Program portability considerations
- Core requirements/overlaying/segmenting
- Database accessing
- Input/output conventions for storage and access
- Use of programming languages
- Flowcharting conventions

The use of phased development at both of the centers should be further defined, standardized and encouraged. The existing standards (SDCINST 5660.1, DMAHTCINST 5660.3) are a step in the right direction and should be broadened and more fully supported.

Standards for Project Initiation and Definition (also called Requirements Specification and Analysis) should address the following issues:

- The representation scheme(s) to be used for specifying the requirements
- The formulation and use of acceptance criteria
- Informal and formal reviews during Initiation and Definition
- Customer and/or user involvement during the phases
- Determination of the impact of changes in requirements after Definition (e.g., on timetable, costs, etc.)
- Definition and control of all documents which will be produced to accompany the software
- Specification of the conditions which will constitute completion of the Evaluation and Definition activities
- Applicable Quality Assurance provisions

At a minimum, some condensed form of representing and specifying the requirements for software should be developed and uniformly applied. This will alleviate some problems which occur due to unstated requirements. An open shop group at the AC is currently using a one-page requirements specification worksheet. The Worksheet has been very well received and the open shop group reports fewer problems associated with specification of requirements due to employment of the mechanism.

Standards applicable during design should address the following issues:

- The method(s) and techniques to be used to represent program and module design (e.g., Program Design Language, flowcharts, etc.)
- Formal reviews and walkthroughs to be conducted during design
- The design methodology to be followed (e.g., the Jackson Methodology)
- Verification that all requirements have been addressed
- The required products of the design phases
- The method of ascertaining that design is complete and obtaining approval for coding to begin

On large efforts (new development or large maintenance efforts on existing programs) the design phase should be divided into a preliminary and a

detailed design phase. These phases address logical and implementation solutions to a problem, respectively. Ultimately, a tailored design methodology (e.g., a set of procedures, techniques, guidelines and supporting tools) should be developed (or adopted from existing methodologies) which specifically suits the DMA environment, placing major emphasis on the maintenance of existing programs, with a lesser emphasis on new software development.

Many such design methodologies and philosophies have been defined. Some of the more well known philosophies are top-down design (Mills) and stepwise refinement (Wirth). Some of the more widely recognized design methodologies and the individual(s) associated with their development are listed below:

- Structured Design (Constantine, Yourdon, Myers)
- Jackson Methodology (M. Jackson)
- Logical Construction of Programs (Warnier)
- Meta Stepwise Refinement (Ledgard, Schneiderman)
- Higher Order Software (Hamilton, Zeldin)

Some techniques for representing design are:

- Program Design Language (Caine, Farber, Gordon)
- Hierarchical Input-Process-Output (IBM)

Papers have been written which compare, contrast and evaluate these methodologies. One such article was published in the November 1977 issue of Datamation entitled "Comparing Software Design Methodologies" by Peters and Tripp.

A team should be formed in the Techniques Offices of the computer support organizations to evaluate the various methodologies and subsequently standardize one methodology for use during design. At the very least, the top-down design approach should be systematically applied at the centers, since programmers and analysts are familiar with the methodology.

Standards applicable during the programming phase are listed in part I of this recommendation.

Evaluation standards should define the difference between development test (module and integration testing) and acceptance test, stressing the importance of planning in both activities. Evaluation standards should address the following issues:

- Requirements for independent review and/or approval
- Informal reviews during integration test and formal review at acceptance test
- Means and conventions by which test cases and products are identified
- Responsibilities for preparation of test cases, submittal of test runs and preparation of test data
- Responsibilities for preparation of test plan
- Methods of validating test results

Module and integration testing (collectively called development testing) are usually less formal than acceptance testing, except for large or very complex projects. Documentation produced as a part of development testing is subject to less rigorous standards and is usually reviewed by members of the development team (not the user or customer).

Module testing should be centered around validation of module design, rather than satisfaction of requirements. Designer produced documentation should be used to verify that the code contains advertised capabilities.

There are two main issues which integration testing should address: first, the modules should execute together under normal operating conditions as documented by the designed solution; second, the entire system should also be subjected to a number of abnormal operating conditions (e.g., improperly formatted data, mixed mode data, etc.) to determine system responsiveness and flexibility.

Any problems which are uncovered during development testing should be documented via a Software Problem Report (see fig. 4.1.4-4), and a temporary resolution to the problem should be implemented. At the conclusion of development testing, all Software Problem Reports and other documents which describe the testing activities and results should be turned over to the acceptance test group. At the completion of development test, the software is stored under project control (see Product Control recommendation) and requirements should be frozen (not subjected to additional change).

Acceptance testing is much more formal than development testing and should show that all applicable requirements have been met. The test plan should be approved by the customer prior to initiation of acceptance testing. All development test problems should have been resolved, necessary modifications incorporated and retested. The development test group should turn over source and object code, all software documentation, any known discrepancies, recently completed development test cases and completed Software Problem Reports. Acceptance testing should be monitored by representatives of the Quality Assurance function (see Product Quality Assurance recommendation). The QA group also assures that Acceptance tests conform to established standards and that configuration control mechanisms (see Product Control recommendation) are being properly applied.

DoD 7935.1-S contains a detailed description of the contents of a Test Plan to be used during acceptance test. For most average to large size projects, this standard should be rigorously applied at the DMA. However, for some smaller projects, it may be economically unreasonable to require preparation of a Test Plan at that level of detail. For these small projects, the following guidelines specify items which should be a part of the Acceptance Test Plan.

- Test scope, purpose and objectives
- Test management and control procedures
- Test resources (e.g., personnel, equipment, input, output, test preparation, operator actions, prerequisite testing)
- Procedural description
- Specific testing description

- Requirement being tested
 - Module(s) or unit(s) involved
 - Data to be collected
 - Verification method
 - Acceptance criteria
- Preliminary test schedule
 - Retest requirements if test fails

3. A structured mechanism for organizing and collecting software development products, delineating schedules and responsibilities and recording decisions should be instituted. Examples of this type of mechanism are being used at a variety of different companies (e.g., Boeing's Software Notebook, TRW's Unit Development Folder). Experience indicates that it has been effective in alleviating many of the problems common to software development.

A software notebook should be prepared for each module or unit to be developed. A module is a functionally unique element of software architecture. It can usually be developed by one programmer during the given schedule and, therefore, is made the responsibility of that person. Modules fulfill testable functions and are the level of software at which requirements are traceable.

Software notebooks should be initiated as early after identification of the need for a module as is practical. This should be no later than completion of requirements analysis or initiation of preliminary design, whichever comes first.

Software notebooks usually contain:

- a. Cover Sheet and Schedule - a one-page summary, identifying sub-modules within the notebook, scheduled due dates, actual completion dates and assigned originators. It also provides space for signatures of reviewers of activities described in the notebook. A sample software notebook cover sheet is presented in figure 4.1.5-1.
- b. Requirements Specification - identifies the baseline requirements and specifies those which are to be satisfied by the module. If the module does not solely satisfy a requirement, other modules involved in satisfying the requirement are listed.
- c. Design Overview - identifies the most current design description for the module and its components. This section generally contains information suitable for preliminary design. Flowcharts are also part of this section.
- d. Interface Description - this section should describe other modules which the subject module must interact with, giving a brief description of their purpose and rationale.
- e. Data Flow Diagram - this section should describe the data which the module receives as input, manipulates and passes to other modules as output.

	DUE DATE	DATE COM- PLETED	ORIGINATOR	REVIEWER
1. REQUIREMENTS SPECIFICATION				
2. DESIGN OVERVIEW				
3. INTERFACE DESCRIPTION				
4. DATA FLOW DIAGRAM				
5. METACODE DESIGN DESCRIPTION				
6. ASSUMPTIONS AND CONSTRAINTS				
7. DESIGN TEST				
8. REVIEW OF DESIGN TEST				
9. MODULE CODE				
10. COMPONENT TEST CASE DESCRIPTIONS				
11. REVIEW OF COMPONENT TEST CASES				
12. TEST CASE RESULTS				
13. SUPPORT LIBRARY CONTROL				
14. DISCREPANCY REPORT FILE				
15. SIGN-OFF COMPLETED MODULE				

FEBRUARY 1977

Figure 4.1.5-1 Sample Software Notebook Cover Sheet

- f. Metacode Design Description - this section should contain the evolving module design, in whatever design representation scheme is chosen for the project. The end product for the section is detailed design information suitable to become a "code to" specification.
 - g. Assumptions and Constraints - any assumptions, inconsistencies, conflicts or constraints which impact the module should be stated in this section.
 - h. Design Test - this section should detail methods which will be employed and criteria which must be used to validate the design of the subject module.
 - i. Review of Design - this section should detail module design review procedures and results.
 - j. Module Code - this section should contain current source code listing for the module. When the module contains submodules, each source code listing should be physically separated by a tab or divider.
 - k. Component Test Case Descriptions - this section should describe the approach which will be used to validate the module. Information which should be included is: test inputs (by name and value), outputs, acceptance criteria and any associated software or hardware to be part of the test (e.g., drivers, plotting devices, etc.). Test cases must validate branch conditions, logic paths, data, and diagnostics as well as satisfaction of stated requirements.
 - l. Test Case Results - this section should contain a summary of all test case analyses and results. Output generated by the tests should be clearly marked and included.
 - m. Support Library Control - this section should contain all necessary information to identify the module's status within a support library. (The Project Management and Control recommendation identifies levels of control and items which should be controlled.) The current state of controlled module information should be clearly documented.
 - n. Discrepancy Report File - this section should contain copies of all problem reports generated during module development. It should also contain a log documenting the current status of problem reports.
 - o. Sign-off Completed Module - this section should contain final comments made by reviewers when the module has been completed and turned over. It may also contain any other information pertinent to the module which does not apply to previously mentioned sections.
4. The phased development standards (see #2) and the programming guidelines (#1) should be formally documented. These and other standards/guidelines which collectively define the 'DMA way' or a DMA software development methodology should be combined in a single reference document. This would then be a formal, centerwide set of software production and product standards. Throughout industry such documents exist and have been referred to as Standards Handbooks, Software Product Standards, Programmers Handbooks, etc.

The document should serve as a mechanism for orienting new programmers and analysts to the 'DMA way' of software development. The Technical Support Division Handbook being drafted at HTC is a good start in the right direction. The contents of the Handbook should be expanded to include:

- The standards and guidelines which have been described in SDCINST 8420.39.
- The additional standards and guidelines which will be formalized and adopted as a result of parts 1 and 2 of this recommendation.
- References to other applicable standards (e.g., DoD 7935.1-S, DMAHTC 5660.3, SDCINST 5660.1, etc.).
- A description of the purpose, contents and intended use of the software notebook as discussed in part 3 of this recommendation.
- Review procedures and standards.
- Procedures to be followed in using the software development libraries, and for the installation of new and modified code in a controlled environment.

The document should be available to new programmers and should serve as a project and software product standard for all development and maintenance efforts. Initial preparation of the document should be the responsibility of a technical person(s) (e.g., SDC and CSD Techniques Office representative), because of the technical nature of its contents. Likewise, periodic updates of the document should be planned and required technical support made available.

5. The DMA should incorporate all development and programming guidelines and standards into a Software Policy Document. Where the Standards Handbook will document explicit standards, the Policy document will state the DMA policy toward compliance with the standard. Justified and necessary departures from the stated policies may be authorized in response to a written request for deviation. The Software Engineering Guidelines and Standards subgroup should be responsible for controlling and maintaining the software policies. Policies in the standardized subject areas should contain the following sections.

- Policy - statement of the DMA policy in the subject area
- Requirements - references applicable standards and defines other policies which interface with the subject policy
- Definitions - defines terms which are used in the policy which otherwise may be unclear
- Responsibilities - defines any approval or review responsibilities which may be associated with the policy.

A sample policy statement on the subject of programming standards has been illustrated in figure 4.1.5-2.

The Software Policy Document should contain a chart illustrating which lifecycle activities should be governed by each procedure and associated policy statement. An example of such a chart has been developed and is presented in figure 4.1.5-3.

**DMA SOFTWARE
POLICY AND REQUIREMENTS**

Number: 2
Effective Date:
Supersedes:
Page 1 of 2
Approved:

Subject: PROGRAMMING STANDARDS

POLICY

DMA software projects shall use programming standards in order to promote uniformity, readability, understandability, maintainability, and other quality characteristics of the software products. Where applicable, such standards should also contribute to portability of software between computers and compatibility with existing and future support software. Either the programming standards given in the "DMA Software Standards Handbook" shall be used, or a project-specific programming standards document shall be produced prior to Preliminary Design Review, and shall include the standards to be followed during software design, code, and maintenance. The decision to develop project-specific standards in lieu of the "DMA Software Standards Handbook" requires approval of the Software Engineering Guidelines and Standards Subgroup. The design and code shall be periodically audited to assess adherence to the project programming standards.

REQUIREMENTS

1. The project's programming standards shall contain, as a minimum:
 - (a) The standards to be followed in describing processing flows and sequences (e.g., flowchart standards, design language standards), including definitions of symbols and conventions for their usage.
 - (b) The standards to be used for preface text and in-line comment cards.
 - (c) Higher order and assembly language coding standards for every language that is used by the project to generate code.
 - (d) Structured programming standards for every higher order language that is used by the project to generate code.
 - (e) Project-unique design standards (e.g., duty cycle, memory utilization, maximum routine size).

Figure 4.1.5-2 Sample Programming Standards Policy Statement

**DMA SOFTWARE
POLICY AND REQUIREMENTS**

Number: 2
Effective Date:
Supersedes:
Page 2 of 2
Approved:

Subject: PROGRAMMING STANDARDS

2. Periodic audits of design documentation and code shall be conducted to assess compliance with project standards. The Quality Assurance Plan establishes the procedure and frequency for these audits. Problems detected in these audits shall be identified in a written summary which is made available to the project and line supervisors responsible for the software audited.
3. The project programming standards document shall be prepared prior to PDR and kept current throughout the period of software design, code, and maintenance.

RESPONSIBILITIES

In addition to the responsibilities identified in policy 1, the Software Engineering Guidelines and Standards Subgroup must approve any decision to develop project-specific standards in lieu of the "DMA Software Standard Handbook."

Figure 4.1.5-2 (continued)

DMA POLICY OVER THE SOFTWARE LIFECYCLE

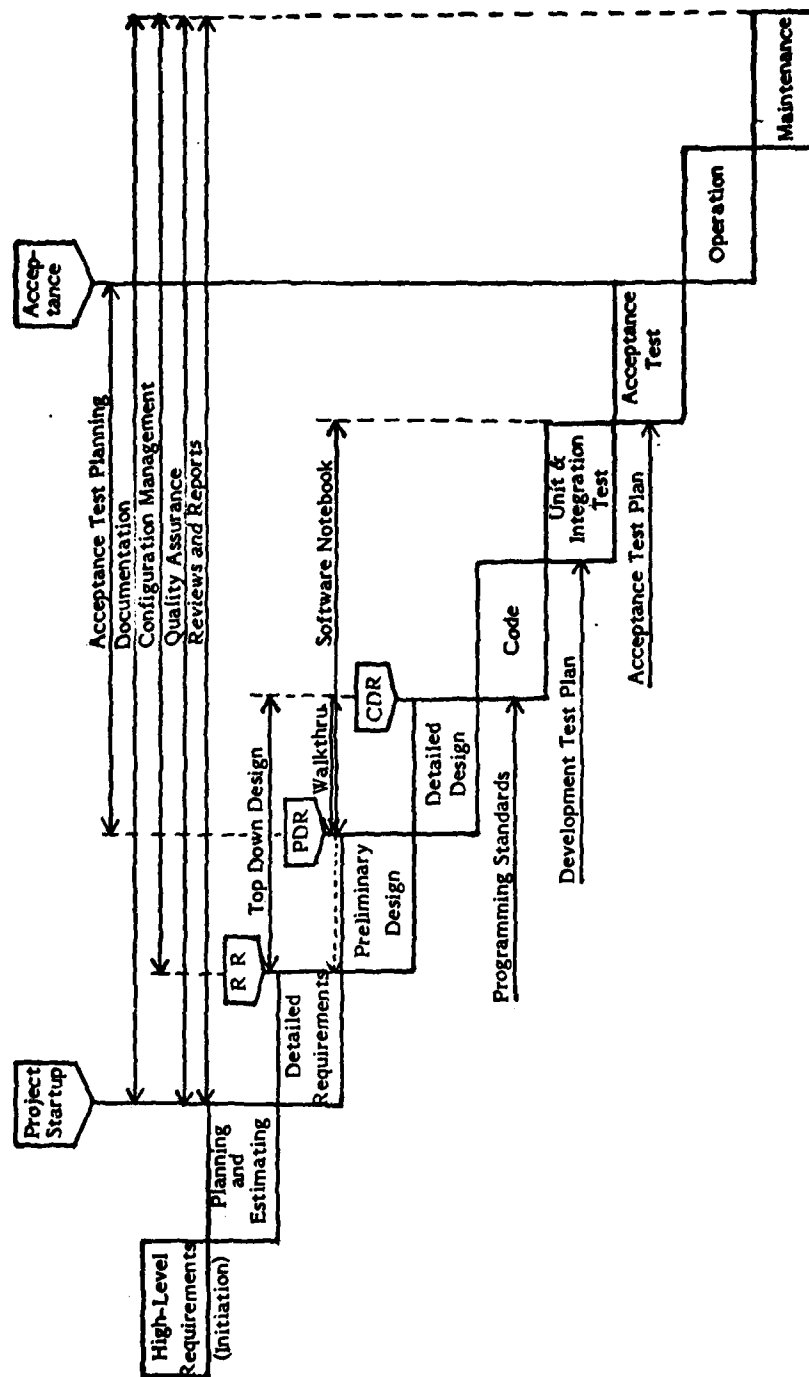


Figure 4.1.5-3 DMA Policy Over the Software Lifecycle

As the Figure illustrates, high-level requirements specification and planning and estimating occur prior to project startup. At Startup, three policies should come into effect: documentation, quality assurance, and reviews and reporting. Each of these policies should be continued throughout the lifecycle. After the specification of detailed requirement, a Requirements Review (RR) should be held and configuration management procedures and associated policies should be initiated. These Policies should remain in effect throughout the lifecycle. Top down design policies should be in effect during preliminary and detailed design, and can be validated during the Preliminary Design Review (PDR) and the Critical Design Review (CDR) by the quality assurance function and during walkthroughs by review teams. The Software Notebook must be initiated by the completion of preliminary design, however, materials for the Notebook should be available subsequent to formal approval of the system specifications by the Requirements Review team.

Programmers must utilize programming standards during the coding phase, in the same way the test teams should utilize the Development and Acceptance Test Plans during the testing phases.

4.1.6 Tools and Software Development Aids

Summary Observations

No systematic plan for identifying, evaluating, developing or acquiring state-of-the-art software engineering tools and techniques was found. STT (at the center and headquarters levels) is broadly chartered with "coordination of center technology development achieved through techniques and engineering application activities." The Center Techniques Offices are chartered with defining research and development requirements and studying ADP procedures utilized by the departments.

Center and HQ STT have observed the use of development/maintenance tools throughout the software industry. Consequently, they have recognized the need and potential for tools and software development/maintenance aids within the DMA ADP environment. They have responded to this need by acquiring some software tools and furnishing them to the centers. However, the acquisition of tools was not preceded by a thorough analysis of center ADP requirements. A plan for installation, testing, training and systematic implementation of acquired tools was not found at either center. At AC, the SDC Techniques Office is responsible for tool testing and provides contractor support during tool installation. At HTC, the Technical Support Division is responsible for installation and acceptance testing of new tools and is the primary contractor contact.

Automated tools have been introduced into the centers in three main ways: sponsored R&D activities (initiated by center STT, Headquarters STT, or center

Techniques Offices), through colleges and universities (the University of Maryland is a prime source), or through the Univac User Program Library Interchange (UPLI).

The following is a brief description of tools which are currently available or which are undergoing testing at HTC or AC:

- SNOOPY (Trace of program instructions)
- PMD (Postmortem Dump processor)
- Dynamic Dump Routines
- FLAP (Flow Analysis Program)
- INDEX (cross-reference lister)
- University of Maryland File Editor
- FILESCAN (cross-reference lister)
- FORFLOW (generates flowcharts for FORTRAN programs)
- TIDY (cleans up FORTRAN programs, rennumbers, some editing)
- REFORMATTER (reformats COBOL source decks)
- STRUCTRAN-1 (converts structured FORTRAN to conventional)
- STRUCTRAN-2 (structures unstructured FORTRAN)
- FAVS (FORTRAN Automated Verification System)
- PSL (Program Support Library)

In general, these tools are seldom used. The utility of each tool has been detailed in Appendixes A and B. The predominant reasons for lack of utilization are:

- limited utility of the tool (does not address critical problems and requirements)
- difficult to use (poor user interface or documentation)
- inoperable, or only partially operable

These tools support different software development activities, as illustrated by Figure 4.1.6-1. As the figure illustrates, most of the tools address the development activities and problems associated with coding and debugging. Only one of the tools supports activities during software definition and design.

Survey results indicated that for both centers the tool with the highest utility is the Postmortem Dump Processor (PMD). This tool was introduced through the UPLI and has been documented by Univac as a "System Utility". Documentation for the tool is adequate, all options have been defined and many examples serve to illustrate correct usage of the tool.

At the Aerospace Center, Dynamic Dump Routines were identified as having the next highest utility, followed by INDEX and SNOOPY. Each of these tools was cited as being readily available and operating at relatively low cost. The functions which these tools provide illustrate the center's need for tools which support developmental debugging.

At the HTC, the University of Maryland File Editor was identified as having the second highest utility. This contrasts with AC's evaluation of the tool's utility. The reason for this conflict occurs because of the differences between the environments at the centers. Average AC turnaround of 19 hours and limited demand processing precludes extensive use and lowers the utility of the Editor.

SW DEVELOPMENT ACTIVITIES	AVAILABLE TOOLS											
	SNOOPY	PMD DDR	FLAP	INDEX	FILE EDITOR	FILESCAN	FORFLO	TIDY	REFOR- MATTER	STRUCTRAN 1 & 2	FAVS	PSL
Analysis, Definition and Documentation of Requirements												
Preliminary and Detailed Design												X
Coding, Debugging	X	X	X	X	X	X		X	X	X	X	
Integration Testing Acceptance Testing (Evaluation)											X	
Maintenance					X			X	X	X	X	X
Documentation (of all activities)			X	X			X				X	

* PSL supports top-down program development by the mechanisms used for identifying and storing data (source code, compiled modules, test data, etc) in a hierarchical manner.

Figure 4.1.6-1 Relationship of DMA Tools to Software Activities

Both centers identified STRUCTRAN-1, STRUCTRAN-2 (DMATRAN) and FAVS as having low utility. Although the centers encourage structured programming, there is no standard enforcing the use of structured constructs at either center. Study of some production program source code did not identify employment of structured constructs.

While FAVS has been installed on the Univac computers at both the DMA centers, it has not been broadly introduced or tested within the DMA software community. The tool was contracted for through RADC and final testing was accomplished under RADC supervision. Implementation included testing on canned programs as well as on DMA production programs.

A more detailed historical description of FAVS has been documented in appendixes A and B. This description includes future DMA plans for the tool. Only general observations of the initial installation applications will be repeated here. FAVS requires a large amount of core (documented core requirements were not found in the User's Manual) and substantial execution time when analyzing large programs, thereby causing bottlenecks in already saturated job throughput. Attempts at modularizing large programs and subsequently analyzing modules with FAVS did not meet with hoped-for success. Modifications which were applied to FAVS were piecemeal, or "fixes" which were not thoroughly analyzed or applied in a disciplined manner. There is a disparity between the advertised capabilities of the tool and its reported utility at the DMA centers. The reported operational characteristics do not measure up to advertised capabilities. Successful operation of FAVS has been marred by incomplete implementation and verification of the tool.

In light of these observations it is not surprising that the tool was rated as having low utility by those who were most familiar with its initial implementation. Since the time that these observations were made, a follow-on contract has been issued to provide enhancements and corrections to the tool. The first enhance, completed in July 1979, resulted in increased efficiency. Depending on the user options chosen, the program executes approximately 25-33% faster than the original version.

IBM's Program Support Library (PSL) was also identified as having low utility by survey participants at both centers. At the AC the tool has not been successfully installed. Installation problems have centered around inconsistencies between the current DMA Univac Executive routines and IBM assumed Univac Executive routines. The tool is currently being evaluated by the DMA and a follow-on contract for training and maintenance is planned.

Documentation for PSL was obtained and reviewed. It is comprehensive and complete, while being rather lengthy. It contains an extensive description of input requirements and messages generated by the system. The document is easy to follow and understand.

Recommendations

A group of technical individuals who are familiar with the development and maintenance of software at the DMA should be formed. Initially, this group would function within the closed shop, mainframe environment and would logically be represented by individuals from the center computer services department/division

(SDC at AC, CSD at HTC). Since the activities of the group should expand the extent of capabilities and expertise required at the centers, it should be built up gradually in the mainframe environment. In the long term, their activities should not be limited to the scope of mainframe software development, but should be center-wide.

This tools group should use the results of the Modern Programming Environment Study as a baseline characterizing current DMA software development and maintenance practices, tools and techniques and to identify areas where tools can be applied to support development activities.

The charter of the tools group should include:

1. Establish tools baselines.
 - Develop and periodically update a statement of requirements for tools to expand the DMA Modern Programming Environment (MPE) as developed by this Study.
 - Periodically inventory tools available in the DMA programming environments. This inventory should update the inventory defined by this Study. Brief descriptions of available tools should be documented in a Tools Catalog in order to effect widespread tool visibility.
 - Interact with the DMA Software Technology Task Force. (See Technology Transfer recommendation), the Center Technique's Offices, and STT (at HQ and center levels) as necessary to identify tool requirements.
2. Identify tools which have a potential to satisfy tool requirements (as defined in #1).
 - Map problems and software activities to tools which are available or which can be developed. For example, the activity of verifying adherence to standards can be automated through the use of PFORT (a Bell Labs ANSI standards auditor) or by development of a DMA specific code auditor (see Software Engineering Guidelines and Standards recommendation).
 - Interact with and support the Resource Usage Optimization function (see Facilities recommendation) as necessary for tool identification and justification.
 - Interact with FEDSIM to gain further expertise in the area of tools existing in the federal government and their availability and applicability to the DMA MPE.
3. Evaluate the tools identified in #2 to determine their suitability in the DMA MPE, based on defined characteristics and requirements.

When a number of choices exist, evaluate the identified alternatives for automation to determine the best available alternative. Evaluation can be accomplished via literature searches and interviews of users of candidate

tools. When a tool has been identified, users of the tool can often be obtained via contacting the industry or government developer and requesting a list of current users.

4. Prepare justification for acquisition or development of new tools, based upon evaluation results.
5. Prepare memos or technical bulletins in cooperation with the Software Technology Task Force (as defined in the Technology Transfer recommendation) which will serve to:
 - Describe currently available DMA tools.
 - Describe modifications or updates to DMA tools as necessary.
 - Introduce newly acquired or developed tools to the DMA.
6. Coordinate the introduction of new tools into the DMA with the Software Technology Transfer Task Force.
 - Provide and coordinate training for use of existing or newly acquired tools.
 - Provide consultation to users of tools as necessary.
 - Serve as the focal point for tool documentation.

Each new tool which is brought into the centers should have an individual(s) who will serve as a "product sponsor" for the tool. This individual should be familiar with the operation of the tool and be available as necessary for technical consultation. Product sponsors for FAVS, PSL and DMATRAN should be identified and trained (as necessary).

7. Periodically evaluate DMA tools to determine their continued suitability at the DMA.

Collect and evaluate data which can be used to perform cost/benefit analyses on tools. Examples of direct costs related to tool usage are elapsed SUP and CPU time, memory requirements, analyst time involved in preparing tool input and analyst time involved in digesting tool output. Direct benefits are provided when the time required for manual analysis is greater than the time required for automated analysis. Indirect benefits are obtained when analysis is performed which would not or could not otherwise be performed.

Of all the functions described above, the most critical to the success of the tools group is the planning and establishment of the tools requirements baseline. Tool and technique recommendations which are a part of the DMA MPE recommendations were the product of thorough requirements analysis and definition. The method used to analyze and define DMA tool and technique requirements is defined in a step-wise fashion below and should be employed by the DMA tools group when updating tool requirements.

Step 1: Define a baseline of typical activities which are performed in support of software development and maintenance. Appendixes A and B represent the baseline of current DMA characteristics which were a product of the MPE Study.

Step 2: Define typical problems which DMA software developers and maintainers are facing. Major problems which were identified by the MPE Study are a part of the characteristics and recommendations and should be used to form a baseline of problems. Examples of these problems have been extracted from study results and are listed below:

- No system for control of documentation update and updated document distribution to users.
- Some production programs are inefficiently consuming computer resources, slowing job throughput and lengthening turnaround.
- The costs of computer resources are not visible to project managers.
- No baseline of required product qualities is defined.
- Regular reviews of software subproducts (e.g., requirements specification, design documentation) are not held.
- Methodologies for requirements and design definition are not employed.
- Detailed programming standards or guidelines have not been formalized and informal guidelines are not always followed.

The MPE Study characteristics and recommendations should be examined to complete the problem baseline. The problem baseline can be expanded and updated by a tool group interaction with the Techniques Offices, STT and software developers. A good method for identifying problems is conduct of informal workshops. Participants in these workshops should be identified from all groups which utilize the mainframe computer for software development and maintenance activities.

Step 3: Examine the baselines of development activities and problems to identify potentials for automation.

Repetitive tasks or those which use an established methodology (e.g., flowchart production) are prime candidates for automation. Other tasks which require detailed analyses (e.g., dataflow analysis, analysis of functional decomposition) are also automatable. This step should incorporate a systematic search of literature (Data Pro, Auerbach and Interface are sources of tools available in the marketplace).

The sixth element of the tools group charter is also very important and deserves further elaboration. Tool introduction into the environments should be approached systematically. The following guidelines should be adopted:

- All turn-key software tools should be carefully evaluated before purchase.
- A trial, on-site use experiment should be performed (e.g., a 30-60 day contractual agreement allowing DMA test implementation).
- Specific acceptance criteria and well defined requirements should be developed and used as the basis for tool evaluation.

- The introduction of a tool into the environment should be performed in a phased manner. A limited number of applications should be chosen and closely supported, so that familiarity with the tool is established. This will serve as a knowledge base for future users, and should facilitate a smooth introduction into the environment.
- Each tool should have a product sponsor knowledgeable in the tools operation and capabilities.

FAVS

The need for functions provided by a tool like FAVS should be recognized. Survey results indicate high utilization of Postmortem and Dynamic Dump Routines. The information obtained from these tools is subsequently used for manual analyses of program code and data. Some of these manual analyses are the same as those which FAVS has been advertised to perform. For example, FAVS static analyses identify possible misuse of constants and variables in expressions, assignments and invocation.

The functions which FAVS is advertised to perform by measuring testing coverage are very useful and should be promoted. FAVS has potential for use by a QA group (see Quality Assurance recommendation) during acceptance testing.

The DOCUMENT option of FAVS is advertised to provide information which can be used by a programmer when preparing technical documentation for a program. Especially useful are: production of cross reference listings, a report which describes the subroutine calling sequence, and a report which lists all READ statements encountered. Most of this type of information is required in Technical Description or Program Maintenance Manual documents.

The functions addressed by FAVS are useful and should automate some development and maintenance activities. Furthermore, FAVS documents some program characteristics which otherwise might be omitted or undocumented. However, initial criticism of FAVS at both centers is substantial. There is a definite need for modifications to the tool. The question at this point is then whether to continue to fund modifications to FAVS or to begin looking elsewhere for a better tool.

There is currently not enough data to provide an answer to this question. The limited experiences with the tool have not been documented or substantiated. No records are readily available documenting actual core usage and execution time requirements. To our knowledge, there are no current users of FAVS at either center. Without this type of data and a user group which is systematically testing and documenting test results, it is impossible to do an in-depth evaluation of the tool.

The disparity between the advertised capabilities of the tool and reported operational characteristics should be analyzed in depth. A test group should be established within the Techniques Offices of the computer support organizations at both centers. This test group should identify acceptance criteria which specify minimum operational capabilities of the tool within the DM3 environment. Test cases which will execute every available FAVS option should be defined and implemented. The tool should be tested using small, medium and large test programs. All deviations from advertised capabilities should be noted. The SUP

AD-A082 713

PLANNED SYSTEMS INTERNATIONAL INC CAMBRIDGE MA

F/6 9/2

DMA MODERN PROGRAMMING ENVIRONMENT STUDY.(U)

JAN 80 L STUCKI, J BROWN, L HAMMOND

F30602-79-C-0022

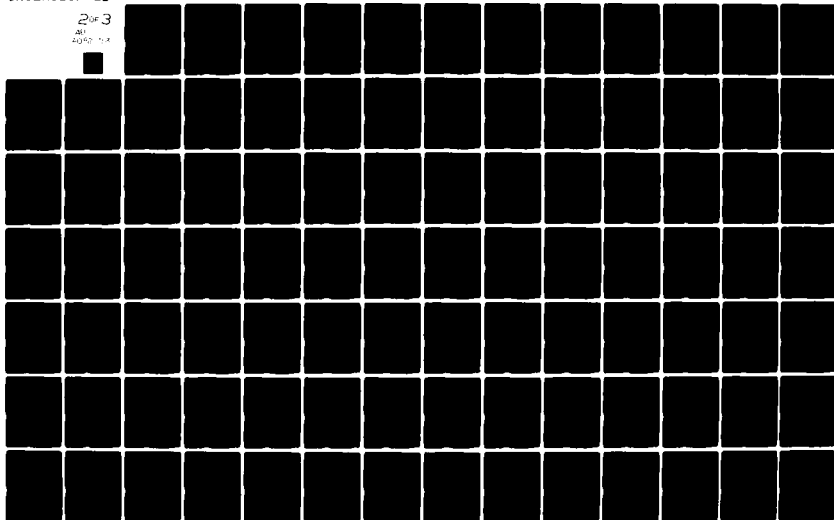
UNCLASSIFIED

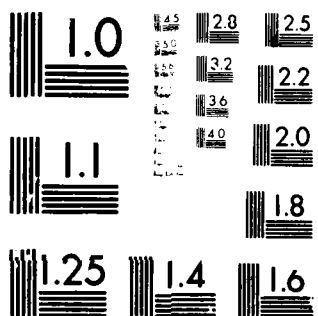
RADC -TR-79-343

NL

2 of 3

AD-A082 713





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

and CPU time consumed during FAVS execution should be documented as well as the amount of core used. In addition, previous unsatisfied users of FAVS should be queried for specific details concerning FAVS' operating characteristics.

Upon successful completion of test activities, an individual at each center should be identified as a product sponsor for FAVS. This individual should have experience utilizing FAVS and should be familiar with its operating capabilities.

Since the time that the preliminary MPE Technical Report was produced (June 1979), enhancements have been made which substantially increase the efficiency of the tool. Subsequent modifications are planned which will further increase its efficiency, correct known errors, improve the user interface and allow processing of the ASCII FORTRAN dialect. These enhancements, if successfully implemented, will substantially improve the utility of the tool. Each enhancement is planned to be followed by testing and will result in a new version of the tool.

All of the data which we collected during the initial study and observations which we have made subsequent to the production of the preliminary MPE report document the need for a tool with FAVS capabilities. The current plan for enhancements and testing is a good one. The final test for FAVS will come in May 1980, after the final enhancements and testing are completed. Therefore, it is important that implementation be carefully accomplished.

A method which has been successful at BCS during large-scale implementation of a commercial tool combines the use of pilot experiments with good user publicity, an effective training program and user consultation services.

When the tool was installed at BCS, many projects were screened for initial application. Several projects were chosen and training courses were conducted. The tool was utilized with close coordination with a knowledgeable product sponsor or consultant. Each of the chosen projects achieved varying measures of success with the tool. For the most part these benefits were intangible; increased visibility, better communication between developer and user, fewer than normal inconsistencies in the requirements, less problems when entering the design phases and few changes in requirements.

Project managers were then asked to describe their experiences with the tool via memos and seminars. These descriptions served as the best possible advertisement for the tool. The program for promotion of the tool was so successful that the responsible division was swamped with requests for training and tool application.

One of the key elements in the program was the use of the sponsor/consultant during initial tool utilization since most initial applications are plagued with questions about specific tool functions and input. These questions were generally easily answerable by the consultant. Without the services of the consultant, the user would have become discouraged and discontinued use of the tool or would have improperly used the tool, both of which would negatively impact the utility of the tool.

For these reasons, we recommend that the centers introduce the final version of FAVS in the described manner.

The potential benefits which can be obtained from successful implementation of PSL are great. The tool has received good ratings from industry users and has been shown to assist program development in most instances.

As stated earlier, the evaluation of PSL within the DMA environment revolves around documentation reviewed, since PSL has not been successfully installed. If PSL performs as advertised, some of the problems associated with program control will be alleviated. In addition, PSL encourages good program development practices (e.g., top-down development, structured coding), providing indirect benefits as a result of this encouragement. To avoid the occurrence of similar problems encountered with FAVS, the introduction of PSL to the DMA should follow the guidelines described earlier in this recommendation.

PSL should be implemented in conjunction with the configuration control practices detailed in the Product Management and Control recommendation.

STRUCTRAN-1 and STRUCTRAN-2 (DMATRAN)

Structured programming, a technique which has a direct relationship to more readable and reliable programs, programs which are easier to maintain and increased programming productivity, is not currently being applied by most DMA programmers. One consequence of this lack of application is disuse of the STRUCTRAN tools.

The structured programming technique is more readily applicable to COBOL than FORTRAN programming. This happens because FORTRAN does not incorporate the control structures promoted by the technique (e.g., DOWHILE, DOUNTIL, IFTHENELSE). As a consequence, numerous preprocessors have been developed throughout the industry which translate normal FORTRAN into structured FORTRAN (and vice versa). These preprocessors require another step in the execution process, providing yet another excuse for disuse of the technique.

However, structured programming in FORTRAN need not require the use of a preprocessor. Control structures can be incorporated through the use of comment statements, and formatting conventions which can be used in an ordinary executable FORTRAN program to display control flow structure, thereby making the program easier to understand and maintain. These conventions were developed as a part of research on a program transformation system which applies the conventions to its output, and have been documented in several University of California Technical Reports. A paper prepared for the Texas Conference on Mathematical Software in March 1978 by James M. Boyle entitled "Towards a Rational Approach to Formatting Programs to Display Their Structure" details FORTRAN conventions which may be used to structure FORTRAN without the aid of a preprocessor. The DMA should consider implementing a course in structured FORTRAN coding without the use of a precompiler (see Training recommendation) in order to reduce the computer throughput burden while still employing structured constructs.

An individual who is familiar with the structured programming techniques and the operating capabilities of DMATRAN should be identified (or trained as necessary) and should serve as a product sponsor for the tool.

Regardless of whether the DMA utilizes a precompiler (DMATRAN) or chooses to implement structured programming without a precompiler, the fundamentals of structured programming should be taught and standardized. Without this, high utilization of structured programming concepts or the DMATRAN tool will not be accomplished and the benefits of structured programming will not be achieved.

As Figure 4.1.6-1 shows, the DMA does not currently have tools which support software activities during the early phases of development and/or maintenance projects (e.g., during requirements specification and design activities). Tools and techniques are available in the marketplace which address these activities and they should be evaluated by the tools group for potential DMA acquisition.

Some well-known tools which address definition, analysis and specification of requirements are listed below along with the name of the developer.

1. Problem Statement Language/Problem Statement Analyzer (PSL/PSA) - University of Michigan
2. Structured Analysis and Design Technique (SADT)^o - Softech Corporation
3. SAMM Interactive Graphics System (SIGS) - Boeing Computer Services Company tool which supports the Systematic Activity Modeling Method (SAMM), also of Boeing
4. Requirements Engineering and Validation System (REVS) - TRW tool which supports the System Requirements Engineering Methodology (SREM), also of TRW

There are few existing tools which support design specification analysis:

1. Program Design Language (PDL) - Caine, Farber and Gordon
2. Process Design System (PDS) - Texas Instruments

However, many of the tools for requirements analysis and specification are also applicable to high-level design specification.

A myriad of techniques for design specification exist and these have been addressed in the Software Engineering Guidelines and Standard Practices recommendation.

Tools which support analysis of the effective operation of programs have been addressed in the Facilities recommendation.

4.1.7 Technology Transfer

Summary Observations

The mission of the Directorate of Systems and Techniques (ST) is to coordinate, direct and supervise the establishment of mapping, charting and geodesy (MC&G) technologies. The mission of the center Techniques Offices is to

develop, support and improve production capabilities through the application of technology. The Techniques Offices of CSD and SDC establish plans for implementation of new capabilities into Department/Division production, develop techniques and technical capabilities to satisfy objectives and provide technical advisory services.

The mission statements of each of the above organizations address technologies and techniques which relate to MC&G activities. One of these technologies is the application of automated MC&G capabilities. The centers are becoming increasingly dependent upon the computer and digital data in the fulfillment of MC&G goals. Successful application of automated MC&G capabilities is dependent upon the effective use of state-of-the-art computer disciplines, techniques and tools.

There is no group within either center whose major responsibility is the effective use of state-of-the-art computer disciplines, techniques and tools. As a result, the full benefits to be derived by using the computer to support MC&G activities are not being attained.

Past attempts to introduce computer disciplines, tools and techniques into the DMA programming environments have met with some success. STRUCTRAN-1 and -2 and FAVS are examples of tools of significant advertised power which have met with some unfavorable user response. There was not a systematic plan for testing, evaluation and implementation of the tools into the environment. The introduction of structured programming techniques has not met center-wide acceptance. Survey and interview response indicate that structured concepts (in both design and programming) are not being employed.

Some new automated concepts are being introduced into center user departments without comprehensive training programs. For example, at the time this Study was initiated, the primary method of job submittal was over-the-counter batch. Since this time, both centers have acquired additional computer terminals. The acquisition of terminals will subsequently change the method of job submittal, introducing a new automated concept. However, study research did not identify the conduct of training courses pertaining to terminal use and concepts. This could have a resultant negative effect on programmer productivity and software quality. For example, a programmer could spend time at the terminal editing or adding to an existing program file. At the end of his session, he may submit a computer run via the terminal. At the same time, he must also save or store the contents of his new program file. If he neglects to perform this storage function, the session's work must be re-done at a later date. Rework has an associated impact on productivity.

Another example is provided through the introduction of System 2000 (a commercial database system developed by MRI) at HTC. At the time of interviews at HTC, System 2000 data dictionaries and data relationships were being defined. Training courses for user departments in the use of System 2000 were planned. However, no attendance in these courses was scheduled for personnel in the Computer Services Department. Consequently, those individuals most qualified to provide database consulting and support will not be trained in System 2000. This creates a potential for substandard application of the system and resultant decreased quality of data stored in the database.

In-house training is currently identified by STT and the Techniques Offices in conjunction with input from the production departments. Courses are offered in various programming languages, EXEC-8 utilities and job control language and some file and data handling concepts. Both centers lack courses in program development methodologies, programming style, programming techniques and automated tools which are available to support program development.

Computer Aided Instruction (CAI) and visual-aided instruction courses are periodically offered in-house at both of the centers. Course tapes are obtained from various sources; a library of CAI tapes is not maintained.

At the Aerospace Center, SDC has established a method for transfer of information which is needed by programmers and analysts. The method is called Technical Information for Programmers (TIPs) and can be utilized by adding an instruction to the job runstream. TIPs output consists of current information about system utilities, library subroutines, mathematical/statistical packages, plotter operation, etc. Users seem to like the TIPs system, finding it a central source of vital information.

Recommendations

Because of the criticality of the technology transfer function and the potential for increased programmer productivity and software quality, a Software Technology Acquisition and Management Transfer Task Force should be established at each of the centers. This group should be interdepartmental, with representatives from various job classifications (e.g., branch chiefs, programmer/analysts, program managers, etc.). Representatives should be supplied from or coordinated through each department Techniques Office. In addition, center STT should also be represented on the Task Force. The activities of the two center Task Forces should be coordinated through HQ STT.

The role of the Task Forces should complement that of the techniques offices, with particular focus on the transfer of software technologies, disciplines, and tools. The mission of the groups should be to establish DMA-wide policies (tailored to center needs as necessary) in the software technology transfer area. The groups would also be responsible for administration of the software technology transfer program.

Before an elaborate combination of training courses, seminars, workshops and other methods of technology transfer are initiated, the software technology transfer program should build a general software engineering foundation at each of the centers. An effective means of building this foundation is through informing, involving, and motivating intended program recipients.

A colloquium lasting approximately half a day should be planned. The following issues should be addressed:

- The DMA-technologies of the past
- The DMA-technologies of the present and future
- Increasing reliance on automation
- The need for disciplined, systematic approach

- Classic examples of catastrophic software failures
- Benefits that increased software discipline can yield
- The role that every computer user and developer plays in the disciplined approach

Colloquium attendance should also include participants from upper management. It is essential that management as well as developers become aware of the criticality of hardware and software resources to the DMA.

The next step in providing the basis for the technical training program is to provide a comprehensive introduction to software engineering to all program participants. Concepts addressed during this introduction should include the following:

- Definition of software engineering
- Applicable DMA Standards, Instructions and Guidelines
- The importance of a continuous product QA program
- Elements of Requirements Specification
- Elements of Design
- Elements of Programming
- Elements of Testing
- Elements of Maintenance

There are a number of methods that can be used to effect technology transfer. A combination of these methods is the best way of reaching the most people, covering the widest variety of topics and ensuring that the best expertise is used to convey state-of-the-art knowledge. The list which follows describes methods of technology transfer and makes specific recommendations in the topic areas.

1. **Training Courses.** In-house training courses should be regularly scheduled and conducted. Potential courses for inclusion in the training curriculum have been addressed in the training recommendation.
2. **Computer Aided Instruction (CAI).** The centers should obtain CAI equipment and designate a location where this equipment will reside. A central library of CAI courses should be established and maintained. Courses can be obtained from a variety of sources: through commercial houses, through government agencies (e.g., the US Department of Agriculture maintains a library of CAI courses), through educational institutions, or they can be recorded using DMA specialists in selected topic areas.

The CAI facilities should be available for use by both open and closed shop software developers during regular center working hours.

3. **Conferences/Commercially Available Seminars.** Organizations like the Association for Computing Machinery (ACM), the Technical Marketing Society of America (TMSA), the Management Development Foundation (MDF), and the Institute of Electrical and Electronics Engineers (IEEE) frequently sponsor conferences and seminars on software related topics. A schedule of planned conferences and seminars should be obtained and reviewed by the HQ Task Force. An organization called TSI, which represents a number of seminar

producers in the country today, maintains quarterly schedules of seminars in computer sciences, management sciences and mathematical and statistical sciences. TSI provides in-house training or consultation and, under some circumstances, TSI can bring an unscheduled public seminar to new cities. This will allow the DMA to save the cost of transportation, food and lodging for attendees. Conferences in related subject areas should be identified. At least one member from each of the center Task Forces should be selected for attendance at the selected conference/seminar. Upon returning from the conference/seminar, the Task Forces should conduct a workshop where use of new ideas, techniques, etc., is discussed for potential inclusion into the technology transfer program.

4. Undergraduate or Graduate Education. The Aerospace Center currently has a good program which provides about 30 hours of graduate level education annually in a range of computer-related curriculum. Currently, the program sponsors from 2-5 people a year to long-term training at institutions such as the University of Maryland. This is a good program which has been well received. The program should be broadened, allowing more participants to receive training in a wider variety of software engineering topics. There is currently no systematic method for assimilating the acquired education into the centers. Recipients of long-term training should be required to give a presentation to the Technology Task Force upon completion of training. The presentation should focus on learned techniques and their potential application at the DMA.
5. Technical Lecture Series. A series of Technical Lectures should be conducted. The lectures should cover a range of topics, from general concepts and MC&G ideas to detailed algorithms, in order to advance theories and techniques. The skills and expertise of in-house experts should be utilized for conduct of the lectures in order to transfer technologies which may be local to a particular group, project or individual to the other center elements.
6. On-the-job Training. On-the-job training is a relatively inexpensive way to transfer working knowledge of techniques to a broad range of individuals. There are several ways in which an on-the-job training program can be instituted. Individuals who are new to the center, or have recently been transferred to a new organizational element should be supervised and trained by senior individuals. These senior individuals should be responsible for supplying new personnel with documents, procedures, standards and other materials which will be necessary for performance of job activities. Senior people should be available to initiate the junior people to their first project or job within the department, explaining the concepts which the junior individual will be expected to employ. This type of training should be continued for one week or until the junior individual has demonstrated a grasp for the new concepts.

Another means of on-the-job training is accomplished by pairing an individual who has recently received long-term training with a senior individual to transfer experiences and knowledge. This type of training arrangement can be mutually beneficial to the individuals involved. The senior person is trained in the latest techniques and the long-term training recipient obtains experience in relating learned concepts to the DMA environment.

7. Colocation of technical staff. For large or very complex projects closed shop software developers should whenever possible be located in the user work area. If an individual is currently involved in more than one project, or for some other reason is unable to maintain a location in the user work area, some other mechanism of daily contact should be established (e.g., brief working visits or telephone calls). Colocation allows both the user and the developer to maintain better awareness of project development activities. Colocation has proved to be a valuable technique when projects are of an R&D nature or are otherwise subject to changing requirements.
8. Workshops. Workshops should be conducted which range in duration from half a day to a week, depending on the topic to be considered. The purpose of a workshop is to gain experience in using a particular tool or technique. The emphasis should be on the "use of" rather than simply "an explanation of" the tool or technique.

Another type of workshop focuses upon the solution of a particular problem or discussion of a specific technical issue. This type of workshop should be utilized to transfer technologies acquired at seminars, lectures, conferences, educational institutions or via other methods when only a few DMA representatives were privy to technical discussions.

9. Information Exchange Meetings/Briefings. Periodic presentations of a less technical nature than seminars or workshops should be used to transfer technologies and knowledge of staff to distributed groups throughout the center. This type of briefing should be conducted on three levels: intra-department, inter-department and center-wide. They should be conducted at least once every two months. The topic of such briefings should be completed projects, ongoing projects and anticipated project activities. The briefings should be sponsored and coordinated by the Task Force, although the presentation itself should be given by a project team member. Attendance at these briefings should not be mandatory, but minutes of the briefings should be documented by a member of the Task Force.
10. Task Force Technical Consultation. A consultant (or group of consultants) should be selected from the members of the Task Force. The consultation function could be of a revolving nature. That is, members of the Task Force could share in performing the consulting function.

Establishment of a consultation function is an integral part of the Technology Transfer Program. Often, new technologies are learned, and newly trained personnel start work on a project, intending to incorporate learned technologies. They find that it is far different to work problems in class than to work problems for real. Class, seminar and lecture exercises were deliberately shortened and simplified so that they could be worked in the time allotted. If a consulting function is not established, individuals may discontinue use of a technique because they are unable to make it fit their application. When this occurs, resources already expended for training have, in essence, been wasted.

A technology consultant should be available during normal working hours at both of the centers to show people how to make technology work on their projects.

11. **Technical Information for Programmers (TIPs).** The TIPs program as instituted at AC should also be implemented at HTC. The Task Force should promote the program, emphasizing the nature of information provided by TIPs as well as the easy method used to obtain TIPs. In addition, the Task Force should solicit information for inclusion in TIPs from the computer support organizations (CSD and SDC) and the project/applications organizations.
12. **DMA Technology Newsletter.** The Task Force should assimilate information about the Technology Transfer Program into a newsletter which should be distributed at least once every two months. The newsletter should contain the following information:
 - Lists of training courses to be conducted in the next two months
 - Lists of available CAI courses and identification of the location of CAI equipment
 - A list of upcoming conferences or commercially-sponsored courses
 - Lists of undergraduate and graduate courses being offered in related disciplines at nearby educational institutions
 - Lists of technical seminars/lectures to be conducted in-house
 - The location and planned topic of scheduled workshops
 - The location and planned topics of Information Exchange Meetings/Briefings to be conducted
 - Names and phone numbers of scheduled technology consultants for the next two months

In addition, the Newsletter should report on any technologies which have been successfully introduced at the DMA. The source of this information should be the documented minutes of lectures, seminars, workshops and briefings.

Future attempts at introducing tools and/or techniques for use into the DMA programming environments should be better planned, controlled and monitored. A systematic approach should be followed. One such approach is outlined below.

- (a) It should be clear how the technique or tool is to be used, in what context, to solve what problem, by whom, when (e.g., design, testing, etc.).
- (b) When evaluating tools and techniques, acceptance criteria should be defined. For example, acceptance criteria for a tool might include a requirement that the tool execute within a specified CPU time.
- (c) For tool evaluation, test cases should be defined which test the full range of options advertised by the tool.

- (d) Initial use should be in a controlled environment on a predefined application, where the tool or technique can be closely monitored. This pilot testing should identify the strengths and weaknesses of the tool/technique.
- (e) A result of the initial application should be a substantiation of the rationale initially presented for its use, or the identification of additional rationale for its continued use.
- (f) Pilot testing should allow several people to become somewhat proficient at using the tool or technique so that they can assist subsequent users. There are at least two candidate tools for pilot testing in the near future. These are FAVS and PSL. Pilot projects for testing and evaluating these tools should be initiated.
- (g) Subsequent to successful pilot testing, an individual with knowledge of and experience with each tool and technique should be identified. This individual should serve as a product sponsor for the tool or technique, providing technical consultation and operating capability information as necessary to software developers.

4.1.8 Training

Summary Observations

The current training courses conducted in computer related topics include Univac EXEC job control language, operating system and processors and courses in programming languages. At AC, basic concepts were introduced to programmer/analysts in the following areas:

- Basics of project management
- 1100 Runhandling and operations
- 1108 Utilities
- DMS 1100
- File structures, file handling
- Data Base Management
- Documentation Standards
- Plotters
- Sorting
- Debugging

No courses are currently being conducted at either center in software development techniques, specification (or analysis) of software requirements, basics of software project planning, estimating and scheduling, testing, evaluation strategies, or standard practices to be employed during coding.

The Training and Career Development Division of the Personnel Office at HTC is responsible for coordination of the center training program. The division has a catalog of available courses in many disciplines which are offered from a variety of sources (e.g., Civil Service Commission, Department of Defense). Some of these sources maintain a consolidated list of course topics. The Training and Career Development Division annually circulates the training catalog and any lists which are available to each department. The departments then identify courses

which they would like to see conducted. Training course recommendations must be approved at various management levels. Courses which can be conducted in-house (not requiring travel) are less expensive and are therefore more likely to receive management approval.

No training in requirements analysis or specification is being conducted. The current computer throughput overload at both centers (especially at AC) can be traced, in part, to the method in which software requirements are specified. Specification of requirements at project initiation is vague and often not documented. Software developers design and code a program solution using the generalized requirements statement. As the ultimate user becomes better informed, he contacts the developer to specify a change in the program requirements. Consequently, the programmer must revise the program design and code to comply with the changed requirements. Furthermore, the project then encounters a schedule slippage or, at the very least, is under substantial pressure to meet its current schedule commitment. Computer resources which were expended in support of the original requirements have not been effectively utilized, and more resources are necessitated.

The current training program at both centers does not include a basic course(s) in project planning, estimating or scheduling. Estimating and scheduling at the centers is based on past experience and sometimes utilizes historical data available from the DMIS/P reporting system. However, study of DMIS/P reports indicated that in some cases data is not systematically updated to reflect project changes. At the HTC, much of the estimated project resources fell into the categories of "maintenance" or "other." Maintenance activities (for both updating and correcting errors in existing programs) comprise over 60% of the software activities at the centers, yet estimates of software activities in the maintenance category are not further subdivided.

Results of the survey indicate that approximately 10% of all software activities are spent in project planning. However, no techniques (e.g., Work Breakdown Structure, PERT) are currently being used to assist in or represent a project plan. At the HTC, PERT is sometimes used when planning and scheduling large projects. The training courses being conducted do not address the use of planning mechanisms such as the Work Breakdown Structure and PERT.

Development testing (which is composed of module and integration testing) is limited because many of the principles of a good testing program are not known, understood or applied. Little development testing is performed at either center in open or closed shop environments. Development testing is usually limited to programmer debug. Stand-alone (module) testing and string testing (which involves more than one module, but not all modules) are not systematically performed. Development test plans are rarely developed or documented. The results of testing, either development or acceptance, are rarely documented. In the closed shop environments at both centers, the developers inform their production offices that the program is ready for acceptance test via letter or memo. Acceptance test is usually performed by the user's Techniques Office. Acceptance testing usually centers around the satisfaction of high level requirements and rarely uses acceptance criteria for program validation.

Maintenance of programs is currently difficult at both centers. There are several reasons for this difficulty:

- Substandard documentation or lack of documentation
- Lack of program modularity
- Use of unstructured programming constructs
- Lack of internal program documentation or inadequate internal program documentation

The current training program does not address these topics, and therefore is not a factor in alleviating maintenance problems.

Each of the centers sponsor individuals in long-term training on an annual basis. This training usually accounts for about 30 hours in graduate computer science disciplines. Individuals who have been recipients of long-term training have generally liked the program and it appears to be a good way for an individual to "get ahead" at the centers.

Recommendations

Each department should identify an individual responsible for defining the annual training needs of his department. Prior to submittal of a list of annual training needs, this individual should contact division and branch managers to receive input to the list of selected courses.

The centers should evaluate the current Training and Career Development Division catalog and lists of courses to determine whether courses are available in the following areas: specification and analysis of requirements, planning, estimating and scheduling, test and evaluation, programming practices and guidelines, project management and software development techniques. Some courses are currently available which would benefit DMA project managers, programmers or analysts. For example, the DoD course in ADP project management has been identified as a valuable source of management information. When courses are currently available in the identified areas, a course outline should be obtained and studied. Available courses in the identified areas should cover topics discussed in the following five sections. When courses currently offered at the DMA do not cover the discussed topics, other course sources should be identified and similarly evaluated. Many such courses are commercially available. For example, Boeing Computer Services' Education and Training Division has courses which are available in various parts of the nation. A National training Schedule has been reproduced in Figure 4.1.8-1.

A well-known organization which serves as a clearinghouse for commercially available courses is TSI Sales and Marketing Inc. A copy of courses available through TSI has been provided in Figure 4.1.8-2.

education & training division
National Training Schedule
may, june, july, 1979

BCS

DEARBORN TRAINING CENTER
(DETROIT)

No.	Course	Days	Date
633	COBOL -ANS	5	6/4-8
649	COBOL Optimization	3	6/18-20
783	COBOL Sort and Report Writer	1	6/11
782	COBOL Table Handling	1	6/12
604	Introduction to Programming Concepts	5	7/16-20
611	Job Control Language - Advanced	5	6/25-29
610	Job Control Language - Basic	4	6/19-22
681	Management Programming Productivity Tech	2	5/22-23
718	MARK IV Basic and Extended Systems	5	5/7-11
718	MARK IV Basic and Extended Systems	5	7/16-20
738	Microcomputer Hardware	5	6/18-22
729	Microcomputer Programming	5	7/8-13
632	MVS Debugging - Advanced	4	6/5-8
727	MVS Debugging - COBOL	3	6/13-15
731	OS Utilities and IDCAMS	3	5/30-6/1
782	People To People Seminar	1	6/1
782	People To People Seminar	1	6/12
661	Presentation Techniques	3	6/13-15
688	Programming Productivity Techniques	5	5/14-18
688	Programming Productivity Techniques	5	6/25-29
653	Project Management	4	6/26-29
678	Statistical Analysis System	2	7/17-18
662	Structured Design	5	5/21-25
662	Structured Design	5	6/25-29
614	Systems Analysis	5	5/7-11
614	Systems Analysis	5	7/8-13
635	TSO (IBM)	2	5/21-22
635	TSO (IBM)	2	6/19-20
635	TSO (IBM)	2	7/24-25
640	TSO - Advanced	2	5/23-24
640	TSO - Advanced	2	6/21-22
640	TSO - Advanced	2	7/26-27

NEW YORK TRAINING CENTER
(NEW YORK)

649	COBOL Optimization	3	6/13-15
610	Job Control Language - Basic	4	5/22-25
653	Project Management	4	5/8-11

NATIONAL TRAINING CENTER
(SEATTLE)

752	APL	5	6/18-22
688	Assembler I	5	6/25-29
606	COBOL	10	7/6-20
606	COBOL - Accelerated	5	7/16-20
607	COBOL Workshop	16	7/23-8/10
689	Computer Graphics Overview	1	6/5
615	Computer Graphics Seminar	3	6/4-8
639	Computer Graphics Software	3	7/9-11
672	Computer Networks	4	6/19-22
637	Configuration Mgmt of Software Development	3	7/10-12
601	CTS (MAINSTREAM) - Basic	1	5/21
601	CTS (MAINSTREAM) - Basic	1	5/22
601	CTS (MAINSTREAM) - Basic	1	6/25
601	CTS (MAINSTREAM) - Basic	1	7/5
601	CTS (MAINSTREAM) - Basic	1	7/6
601	CTS (MAINSTREAM) - Basic	1	7/23
601	CTS (MAINSTREAM) - Basic	1	7/24
602	CTS (MAINSTREAM) - Advanced	3	5/7-9
602	CTS (MAINSTREAM) - Advanced	3	6/26-28
602	CTS (MAINSTREAM) - Advanced	3	7/16-20
600	EASYTRIEVE	2	5/29-30
711	Effective Writing Workshop	3	5/30-6/1
711	Effective Writing Workshop	3	6/6-8
678	Estimating Techniques	3	5/21-23
619	FORTRAN	8	6/7-16
619	FORTRAN	8	7/18-27
688	IMS Introduction	2	5/30-31
689	IMS/VS Date Base Programming	5	6/4-8
689	IMS/VS Physical Data Base Design	5	6/25-28
634	IMS/VS 3270 Message Format Service Overview	1	6/1
635	IMS/VS 3270 Message Format Service Prog.	3	6/11-13
788	Introduction to Data Base	2	5/29-30
788	Introduction to Data Base	2	7/16-17
683	Introduction to Data Processing	2	5/21-22
683	Introduction to Data Processing	2	6/21-22

NATIONAL TRAINING CENTER
(SEATTLE) (Continued)

583	Introduction to Data Processing	2	7/2
583	Introduction to Data Processing	2	7/5-6
583	Introduction to Data Processing	2	7/25-26
584	Introduction to Programming Concepts	5	5/7-11
584	Introduction to Programming Concepts	5	6/25-29
584	Introduction to Programming Concepts	5	7/23-27
681	Inquire*	5	6/18-22
510	Job Control Language - Basic	4	5/15-18
510	Job Control Language - Basic	4	5/29-6/1
510	Job Control Language - Basic	4	6/12-15
510	Job Control Language - Basic	4	7/17-20
511	Job Control Language - Advanced	5	5/14-18
511	Job Control Language - Advanced	5	6/4-8
511	Job Control Language - Advanced	5	7/20-8/3
681	MAINSTREAM EKS KIT	3	7/9-11
520	MAINSTREAM EKS Programming	10	5/17-31
520	MAINSTREAM EKS Programming	10	7/23-8/3
683	MAINSTREAM EKS Programming & KIT - Acc.	5	7/8-13
622	Manager in the Middle*	3	5/15-17
622	Manager in the Middle*	3	6/12-14
718	MARK IV Basic and Extended Systems	5	5/7-11
718	MARK IV Basic and Extended Systems	5	7/23-27
728	Microcomputer Introduction	2	5/3-4
759	Mini Max Timesharing	2	6/14-15
615	Microcomputer Fundamentals	3	7/2-5
632	MVS Debugging - Advanced	4	5/7-10
632	MVS Debugging - Advanced	4	7/18-23
727	MVS Debugging - COBOL	3	6/14-16
727	MVS Debugging - COBOL	3	6/11-13
727	MVS Debugging - COBOL	3	7/18-20
731	OS Utilities and IDCAMS	3	6/11-13
689	PASCAL Intermediate	3	5/30-31
782	People To People Seminar	1	5/14
782	People To People Seminar	1	5/15
782	People To People Seminar	1	5/16
782	People To People Seminar	1	6/4
782	People To People Seminar	1	7/2
782	People To People Seminar	1	7/3
782	People To People Seminar	1	7/16
782	People To People Seminar	1	7/17
688	PL/I	10	6/18-28
661	Presentation Techniques	3	5/14-16
661	Presentation Techniques	3	6/18-20
661	Presentation Techniques	3	7/18-20
636	Project II	5	6/11-15
554	PSL/PSA Computer Software	5	5/13-15
554	PSL/PSA Computer Software	3	6/20-22
559	SSDM Project Management	4	6/29-6/1
559	SSDM Project Management	4	6/19-22
584	Structured Analysis and System Specification*	5	5/7-11
584	Structured Analysis and System Specification*	5	6/25-29
584	Structured Analysis and System Specification*	5	7/23-27
582	Structured Design	5	6/14-18
582	Structured Design	5	6/18-22
582	Structured Design	5	7/16-20
586	Structured Program Development in COBOL	5	6/4-8
586	Structured Program Development in COBOL	5	7/30-8/3
587	Structured Program Development in FORTRAN	5	6/25-29
587	Structured Program Development in FORTRAN	5	7/30-8/3
614	Systems Analysis	5	7/23-27
611	System 2000 Application Analysis Methods*	3	6/20-22
614	System 2000 Data Base Design & Admin.*	2	6/18-19
636	TSO (Kent Data Center)	2	5/23-24
636	TSO (Kent Data Center)	2	6/29-30
636	TSO (Kent Data Center)	2	7/12-13
641	TSO C-List	2	5/31-6/1
641	TSO C-List	2	7/6-8
641	TSO C-List	2	7/16-20
683	Volume Management System (VMS)*	5	5/14-18
683	Volume Management System (VMS)*	5	6/11-15
683	Volume Management System (VMS)*	5	7/23-27

WASHINGTON DC TRAINING CENTER
(WASHINGTON DC)

*872	Computer Networks	4	6/26-29
559	SSDM Project Management	4	6/5-8

COURSES AVAILABLE VIA CTS
CTS COMMAND LANGUAGE
EIS INTRODUCTION
KIT INTRODUCTION

*For BCS/Boeing Employee Training Only

Figure 4.1.8-1 Boeing Computer Services Company
National Training Schedule

**TSI
SALES AND MARKETING INC.**

19 WEST 44TH STREET
NEW YORK, NEW YORK 10036
212-764-3070

Please send me Detailed Course Outlines describing the following seminars:

COMPUTER SCIENCES

PROGRAM NAME	PROGRAM NO.*
☐ Advanced Structured Techniques by Ned Chapin	KC 158
☐ Applications of Minicomputers in a Business Environment	QC 215
☐ ANS COROL Workshop	QC 199
☐ Applied Data Communications Systems	EC 101
☐ Automated Tools for Software Engineering by Edward F. Miller, Jr.	LC 182
☐ Computer Control and Audit	QC 186
☐ CMOS Microprocessors	GC 150
☐ Database Design	QC 200
☐ Database Design, Development and Implementation	EC 103
☐ Design of On-Line Systems	QC 201
☐ Designing with CMOS	GC 138
☐ Designing with Microprocessors	GC 140
☐ Digital Communications	GC 141
☐ Digital Systems Engineering	GC 139
☐ Distributed Computing Systems	QC 188
☐ Distributed Minicomputer Networks	EC 105
☐ EDP Planning for Management	QC 202
☐ EDP Quality Assurance	QC 189
☐ Effective Data Center Management	EC 212
☐ Effective Leadership of Data Processing Projects	EC 107
☐ Effective Presentations	QC 190
☐ Effective Programming Style	QC 203
☐ Effective System Design	QC 191
☐ Human Relations: Productive Working Relationships	QC 216
☐ Improved Test Methods Workshop by Ned Chapin	QC 154
☐ Information Systems Productivity	EC 207
☐ Integrated Circuits, Bipolar and MOS	GC 146
☐ Intelligent Terminals in Distributed Processing	QC 217
☐ Introduction to Data Communications	QC 194
☐ Lasers	GC 142
☐ Logic Design	GC 143
☐ Management of Structured Programming Projects by Ned Chapin	KC 155
☐ Management Overview of Changing Information Systems Technology	EC 213
☐ Microcontrollers 6800-8080	GC 145
☐ Microprocessors and Microcomputers	GC 144
☐ Microprocessor Interface	GC 206
☐ Minicomputer Systems	EC 110
☐ MSI/LSI Applications	GC 147
☐ On Line Systems Audit Controls	EC 116
☐ Performance Enhancement Technology	LC 208
☐ Prevention of Errors in Computer Applications by Edmund C. Berkeley	HC 152
☐ Project Management and Control	QC 195
☐ Reducing Software Complexity by Ned Chapin	KC 159

COMPUTER SCIENCES (cont'd)

☐ Software Costing Methods	LC 211
☐ Software Engineering Management and Technology by Peter Freeman and Tony Wasserman	LC 198
☐ Software Metrics Technology	LC 209
☐ System Reliability Technology	LC 210
☐ Software Testing Techniques by Edward F. Miller, Jr.	LC 161
☐ Structured Analysis and Design	EC 214
☐ Structured Analysis Workshop by Ned Chapin	KC 177
☐ Structured Design and Programming Workshop by Herbert Alter	FC 181
☐ Structured Design and Programming Workshop by Ned Chapin	KC 153
☐ Structured Design/Programming Workshop by David Shaw	QC 167
☐ Structured Design Workshop by Herbert Alter	FC 180
☐ Structured Design Workshop by Ned Chapin	KC 156
☐ Structured Programming Workshop by Herbert Alter	FC 179
☐ Structured Programming Workshop by Ned Chapin	KC 157
☐ Systems Analysis Workshop	QC 204
☐ User's Role in Systems Development	QC 205
☐ Z-80 Microprocessor Fundamentals and Applications	GC 149

MANAGEMENT SCIENCES

PROGRAM NAME	PROGRAM NO.*
☐ Management, Planning and Control of Word Processing	EM 121
☐ Office Automation Systems	EM 115B
☐ Theory Z	EM 120

MATHEMATICAL AND STATISTICAL SCIENCES

PROGRAM NAME	PROGRAM NO.*
☐ Analysis of Variance	ES 218
☐ International Conference on Model Building by Sir Maurice Kendall	ES 111
☐ Linear and Non-linear Model Fitting	ES 108
☐ Multivariate Analysis by Sir Maurice Kendall	ES 112
☐ Numerical Taxonomy	ES 114
☐ Regression and Correlation	ES 117
☐ Simulation Modeling and Analysis	ES 104
☐ Statistical Optimization, Design and Analysis	ES 197
☐ Time-Series Analysis by James Durbin	ES 129

Prefix E indicates programs presented by The Institute For Professional Education
Prefix F indicates programs presented by Alter Associates
Prefix G indicates programs presented by Infoscpe, Inc.
Prefix H indicates programs presented by Berkeley Enterprises, Inc.
Prefix K indicates programs presented by InfoSci, Inc.
Prefix L indicates programs presented by Software Research Associates
Prefix Q indicates programs presented by Q.E.D. Information Sciences, Inc.

NAME & TITLE _____
ORGANIZATION NAME _____
DIVISION/BRANCH _____
STREET ADDRESS _____
CITY _____ STATE _____ ZIP _____
TELEPHONE (AREA CODE) _____

Figure 4.1.8-2
Courses Available Through TSI

1. Requirements Training. Whether provided through existing sources or acquired via new sources, a training course(s) in requirements definition is a must for both DMA centers. The course(s) should address the following activities:*

- (a) Context Analysis: Analysis of why the system or problem is being proposed. This analysis should take an in-depth look at the way the functions to be automated are currently being fulfilled and any problems which are inherent in the existing system. The contextual analysis should also address any other disciplines which are available to effect a system solution and should identify alterables (elements in the system subject to change). Any technical, operational and economic characteristics of the present method(s) must be defined, analyzed and documented. The final result of context analysis should be a sound understanding of the reasons for the proposed system and why certain criteria must be made part of the system.
- (b) Functional Specification: Specification of what functions the system must perform. The functional specification should partition the system solution into relevant elements whenever possible and identify interactions between these elements. The technical, operational and economic criteria defined during context analysis must be incorporated. Any new functions or limitations must also be introduced.
- (c) Design Constraint Analysis: Specification of how the proposed system is to be built and implemented. These design constraints should also incorporate associated criteria to be used to trace and validate the stated requirements.

In addition, the following topics should be discussed:

- (a) Requirements Documentation: Graphic as well as textual means to portray context, functional and design specification. Some existing means of specification have already been defined in the Software Engineering Guidelines and Standard Practices recommendation.
- (b) Requirements Team: The importance of a cohesive team to the definition and analysis effort. The requirements team consists of people from diverse backgrounds who view the system solution from different perspectives. For example, typical requirements team members include representatives of the customer, user, developer and management. Each of these representatives have differing perspectives which must be blended into the requirements specification.

The importance of requirements definition and analysis cannot be over emphasized. Slipped schedules and project rework have an associated effect on productivity and costs. This effect can be alleviated by a systematic approach to requirements definition as developed by a thorough requirements training program.

*Ross, D.T., and Schoman, K.E., Structured Analysis for Requirements Definition, IEEE Transactions on Software Engineering, January 1977.

2. Planning, Scheduling and Estimating Training. A comprehensive course(s) in planning, scheduling and estimating should be conducted. The following topics should be addressed:

- (a) The current DMA approach to estimating project costs and schedules and any associated mechanisms being applied to assist in the approach.
- (b) The critical elements in planning, estimating and scheduling and the manner in which these elements interact and can be defined.
- (c) The items which must be included in a project plan (e.g., project overview, tangible and discrete milestones, task breakdown, activity networks, budgets, schedules, project interface with external world, review process, key personnel).
- (d) Available cost and schedule models and methods (e.g., top-down, bottom-up, ratio estimating, work breakdown structure, PERT).
- (e) Typical resource expenditures during the system lifecycle.

A seminar on software cost estimating has been put together by L. H. Putnam and S. W. Wolverton. This seminar provides good background for the government internal project justification and approval processes, and should be applicable to the DMA. The Seminar is available through the Technical Marketing Society of America.

3. Test and Evaluation Training. This training course should define the difference between development test (composed of unit and integration test) and acceptance test, stressing the importance of planning in both activities. The following topics should be addressed:

- (a) Selection of test team members.
- (b) Establishment of test procedures.
- (c) Contents of test plans (both development and acceptance test plans).
- (d) Responsibilities during development and acceptance testing (e.g., for preparation of plans, reporting of problems, submittal of test data, etc.).
- (e) Roles of Quality Assurance and Configuration Control in the test process.

These topics have been discussed in greater detail in Section 4.1.5, Software Engineering Guidelines and Standard Practices.

4. Programming Practices and Guidelines Training. Section 4.1.5 describes standard programming practices which should be formalized and adopted within the centers. To introduce these standards within the centers, a comprehensive training course should be designed and conducted.

An effective method for introducing and encouraging modern programming practices is the use of a case-study approach. Two programs which address the same problem in different programming styles should be studied. One program does not employ modern practices (e.g., structured code, modularity, comment and documentation conventions, naming and file formatting conventions, etc.) and the other adheres to established programming practices. An exercise should be designed which illustrates the ease with which modifications can be applied to the program which employs modern programming practices.

In particular, a course(s) in structured programming concepts and techniques should be given. Methods of employing structured concepts in FORTRAN without the use of a preprocessor should be introduced.

Another course which should be provided as soon as possible would detail the essentials of interactive programming/software development and maintenance. With the acquisition of numerous demand terminals by the centers, this type of a course is certainly justified and required.

5. Project Management/Development Techniques. This course should identify and define the phased development approach. Roles of groups external to the project should be related to the approach. For example, the Quality Assurance and Configuration Management functions must interact with the development group, and this interface should be explained.

This course should also address the topics of reviews and reporting as defined in Section 4.1.1, Communication and Visibility.

The concept of a software notebook and its impact on the development effort (see Software Engineering Guidelines and Standard Practices recommendation) should be introduced. The notebook concept can be effectively illustrated by its use on a sample project.

Training courses have been discussed in five main areas where we have identified the greatest need. Each of these areas has been discussed in a previous recommendation within Section 4 of this report. Therefore, when identifying and evaluating available training courses, these other sections should be cross-referenced when greater detail is required.

5.0 PHASED IMPLEMENTATION PLAN

This section documents a phased plan for implementing the recommendations described in Section 4. The plan includes a sample Work Breakdown Structure (WBS), an activity network and text describing the interrelationships between the recommended activities. Since the recommendations have been discussed in detail in the previous section, they will only be included by reference in this section.

The WBS and associated activity network contain estimates of the manhours which should be expended to implement the recommended activities. In the WBS, these manhour estimates have been made in two categories; flowtime or planned duration of the activity, and a recommended manloading estimate. The flowtime estimate defines the amount of calendar time required to complete the task. The manloading estimate provides an estimate of the total effort or level of effort required for the task.

These estimates must be considered preliminary, since the experience of individuals assigned to the tasks is a factor which is somewhat unpredictable. However, the estimates have been included to provide a frame of reference for the implementation plan.

As stated earlier, to convert the DMA programming environment into an MPE, a number of modern programming practices and much more formal development/maintenance discipline need to be implemented. The recommendations made in Section 4 describe a fairly comprehensive set of characteristics which would need to be employed to develop a DMA MPE. Obviously, implementation of the full set of these recommendations will be a costly and time consuming effort. However, if the DMA is to achieve a true DMA MPE, certain of these recommendations are critical.

Because we recognize that the DMA (as would be the case with most other programming environments) will not be able to immediately achieve an MPE, we have prioritized the comprehensive set of recommendations detailed in Section 4. In addition, we have identified three of the recommendations which we feel will have the greatest immediate impact on DMA productivity and quality, given the characteristics described in Appendixes A and B.

Very little quantitative data on the potential increase in productivity and quality which could be attained by implementing these recommendations has been collected. The problems associated with quoting the potential benefit associated with one recommendation have been discussed in Section 3. For these reasons, it is impossible to quote with any great accuracy the potential benefits of a particular recommendation. It is possible, however, to estimate benefits which may be accrued, based upon our knowledge and experience and extrapolating from the little data which is available. It is also possible to state the qualitative benefits which should result from systematic, disciplined employment of these recommendations.

In order to provide the DMA with a means to justify the costs of implementing the three most critical recommendations made as a result of this study, we have provided estimates of quantitative benefits which we believe may result upon successful implementation. In addition, where quantitative benefits data has been found in the software development community, we have furnished it in this section

for the three most critical recommendations. Qualitative benefits generally associated with these three most critical recommendations have also been specified.

The remainder of Section 5 deals with a phased plan for implementing the recommendations made in Section 4. Section 5.1.1 describes the three recommendations which are felt to be the most critical and which will address some of the immediate DMA software problems, based upon the characteristics detailed in Appendixes A and B. Section 5.1.2 also lists those recommendations which are felt to be "immediate" and which should be implemented within two years, while Section 5.1.3 is devoted to those recommendations which are least critical (but still necessary to achieve a true DMA MPE) and should be implemented after the others.

5.1 IMMEDIATE, SHORT-TERM, AND LONG-TERM IMPLEMENTATION

After reading the recommendations in Section 4, the reader will probably be overwhelmed by the number and detail of activities to be initiated. In addition, the estimates of effort required to complete the recommended activities total many man years of work. DMA software developers, whose workload is already saturated, should not be expected to implement all of the recommendations at one time.

Another constraint which restricts immediate implementation of every recommendation is the impact which implementation will have on the DMA "production pipeline." If the current staff of DMA software personnel were to be assigned half or full-time responsibility for implementation of the recommendations, a severe bottleneck in DMA production would be imposed.

For these reasons, the first part of this section will deal with a plan for immediate, short-term and long-term implementation of the recommendations.

Each of the lists is ordered based upon our estimation of their criticality to the DMA and the relative ease with which they may be employed. Those recommendations which are at the beginning of the lists are either highly critical or may be implemented at low cost with little impact on current production of software (or both).

Preceding the list of recommendations suggested for immediate implementation is a more detailed discussion of the qualitative benefits which can be expected as a result of implementation of the three most critical recommendations. Where quantitative evidence of benefits has been found within the community, this is also given. When no quantitative data is available we have provided our best estimate of benefits which may be accrued.

The latter part of the section will then provide labor estimates, recommend formation of groups to implement the recommendations, and define the interrelationships between the recommendations.

5.1.1 Recommendations For Immediate Implementation

The most critical need which faces DMA software developers today is for the establishment and enforcement of programming and development standards. At

the present time the programming environments at the DMA are primarily maintenance oriented. Approximately 70% of the programming activities are in support of changes, updates, and corrections to existing production programs. This fact in itself calls attention to the need for the implementation of a rigorous standards program.

In a study conducted by the University of Maryland, research indicates that the employment of a disciplined methodology by a programming team reduces the average costs, both machine and human, of software development, relative to individual programmers and programming teams not employing the methodology by a magnitude on the order of 2 to 1 (i.e., 50%), or better. (See reference 16 for details of the research.)

We have not seen quantitative data which relates the establishment of programming standards to programmer productivity, however, it is a well-known fact that standards must be established as the foundation of cost-effective maintenance and quality program development. A significant reduction in maintenance programmer/analyst time to implement changes can be expected from such a standards program. While we have no data from a well-designed and documented experiment which solely employed programming standards, we conservatively estimate that maintenance programmer time can be reduced by 25-30% if programs are constructed and documented in accordance to standards. Therefore, the following recommendations are made:

1. The programming and software development standards which have been documented at each center (SDCINST 5660.1, SDCINST 8420.39, DMAHTCINST 5660.3) should be incorporated in a preliminary version of a Standards Handbook. The Handbook should also contain references to DoD Standard 7935.1-S.
2. Existing programming standards should be further detailed and then incorporated into the Standards Handbook. New programming standards (as identified in Section 4.1.5) should be defined, formalized and documented in the Standards Handbook. In addition, descriptions of recommended practices and techniques which should be employed in conjunction with phased development (i.e., review and reporting procedures, documentation production, use of a software notebook, etc.) should be incorporated into the Standards Handbook.
3. The reports described in Section 4.1.6 concerning the employment of structured programming without the use of a precompiler should be obtained and studied.

Regardless of whether structured programming is accomplished via support of a precompiler, it should be adopted as a DMA Standard. As described in Section 3.3.2, IBM reported that almost 50% more new code was produced with slightly half the effort for a project utilizing structured programming.

4. Peer code reviews should be conducted prior to final approval of the coding activities. This is necessary in order to verify compliance to established

standards. Even the most well-defined set of standards will not result in increased programmer productivity if they are not rigorously employed. Peer reviews are the least-costly manual method of monitoring compliance with standards.

5. After the standards have been defined, documented, and are being employed, the DMA should define the requirements for a Code Auditor tool to be used as an automated means to verify compliance with standards. Depending on the depth of the manual effort to verify standards compliance, varying levels of cost can be incurred. TRW, which long ago established and began to formally employ and verify compliance with standards, developed a Code Auditor of this type. Prior to use of the Auditor, code review was accomplished via peer reviews and by an independent QA review. They estimated cost savings of \$150,000 in the first year of the Auditor's employment to monitor standards compliance.

Another critical problem which is occurring at both DMA centers is the availability of the computer. Turnaround time on the mainframe computers was expressed as the number one problem in our interviews with DMA developers and managers. Since the DMA is primarily a maintenance environment, it is reasonable to conclude that a substantial benefit can be incurred by systematically optimizing the most frequently used production programs. This can be accomplished by either manual or automated analyses, followed by program modification. Because of the current shortage of software engineering manpower to conduct manual analyses, we recommend the use of automated optimization aids.

Some data is available (see reference 15) which supports the use of optimization tools:

1. The Federal Aviation Agency (FAA) utilized a set of tools to analyze programs for efficiency and was able to free up \$471,562 worth of computer resources during 1976-1978.
2. NASA used a data flow optimization package to free up 18 hours of computer time per day on two machines (annual savings \$660,000).

While the situations associated with the FAA and NASA cannot be duplicated, it is obvious that a significant amount of cost savings and increased computer availability is likely to result from employment of optimization tools. Therefore, the following recommendations are made:

1. The production programs which are the biggest consumers of computer resources should be identified. At the Aerospace Center, the Monthly Accounting Program Utilization by Unique Program Number report can be used for this purpose. The HTC should determine whether an equivalent report is produced at the center. If not, the AC Program should be implemented at HTC for identification of frequently utilized programs.
2. Production programs which are big consumers of computer resources should undergo optimization analyses evaluation, and subsequent associated modification.

Another critical need which was identified during the MPE Study centers around training of software development/maintenance personnel. This criticality

has occurred for a number of reasons. First, software development has always been more of an "art" than a "science." Only recently have development techniques been employed, and few courses which taught the science of software engineering have been available in the past. Second, most DMA software developers have a background in mathematics, geography and cartography. As such they have not been privy to much software engineering training. Third, the criticality of the production of maps and associated data has imposed a severe constraint in the attendance of training courses.

In the interest of a more effective, efficient and modern programming environment, these situations must be overcome and a formal training program which emphasizes the science of software engineering must be implemented.

No data is available from controlled, well-documented experiments which relates software engineering training to improved productivity and quality. Obviously, any such data (if it were available) would be dependent upon many uncontrollable and virtually unmeasurable factors; individual receptivity to training, preconceived attitudes and biases, effectivity of instructional methods, etc. However, it should also be obvious that achievement of an MPE cannot be realized without familiarity with modern disciplines and concepts. Therefore, the following recommendations are made and should be implemented in the following order:

1. An individual from each HTC department should be identified as a training focal point. Individuals currently performing this function at AC should receive greater visibility in order to better identify the training needs and thereby represent the community of software developers.
2. A catalog of currently available DMA training courses should be obtained and reviewed by each department training representative.
3. While training courses available through DODC1, CSC and AMETA are identified annually, other training courses which are commercially available should be identified. Courses should address management of software development, software development techniques, testing techniques and strategies, specification and analysis of requirements and planning, scheduling and estimating techniques. Computer and Visual-Aided Instruction courses and commercially available seminars which address these topics should also be identified.
4. A seminar which will serve to introduce software engineering to DMA management and technical staff should be developed and conducted.
5. A basic course(s) should be identified (or developed) and conducted at each center concerning the use of interactive programming.
6. Training courses in requirements analysis and specification, DMA programming practices, and project management/development methodologies should be conducted.
7. Training courses in test and evaluation methodology and planning, estimating and scheduling techniques should be conducted.

5.1.2 Recommendations for Short-Term Application

The recommendations made in Section 5.1.1 are the most critical in terms of modifying some existing DMA characteristics (i.e., long turnaround time), reducing time spent in maintenance, and educating developers in the state-of-the-art of software development. After these recommendations have been implemented, several other recommendations should be addressed.

Software notebooks are being increasingly employed throughout the industry. No quantitative benefits have been associated with their use. However, they promote disciplined development and are therefore subject to some of the benefits as described by Basili (reference 16). Qualitatively, many benefits are associated with their use. A sample of these are listed below:

- Increased maintainability.
- Increased reliability.
- Increased development status visibility.
- Increased modularity.
- Increased understandability.
- Reduction in development errors.

For these reasons, the following recommendation is made:

Software development/maintenance activities which require production of more than a few lines of code should employ the use of a software notebook. The notebooks should be initiated before the completion of preliminary design activities.

The use of software notebooks in conjunction with enforced programming and development standards can be expected to significantly increase programmer productivity (of both development and maintenance programmers), product quality and reliability while providing status visibility to the project manager at any time during the systems lifecycle. Another element of status visibility is that which is provided to upper level management. DMA managers, much like managers within other government agencies and commercial development shops, are responsible for a large number of development projects. They need a mechanism which will enable them to quickly determine the status (and current problems or other significant development information) of each project for which they are responsible. Benefits specifically associated with use of this type of a mechanism are not easily quantifiable. However, any manager who has been asked by his superior about activities of a project within his jurisdiction will immediately be able to see the benefit of such a mechanism. There is nothing more embarrassing than being the last one to be informed of a problem which is one's direct responsibility for correction.

In addition, use of the software notebook tends to bring a problem to the surface early in the system's lifecycle, enabling correction before costly side-effects proliferate.

Data has been collected for experienced reductions in forecasted vs. actual labor costs when formal reviews have been conducted along with the use of other practices (see Section 3.3.1). This data suggests that the labor cost reduction can amount to between 57 and 84%. For these reasons, the following recommendations are being made:

1. Project managers (open and closed shops) should produce brief Monthly Status Reports describing activities performed and resources expended during the period, as well as planned activities and expenditures during the next period. These reports should be distributed to department PMOs and to a user/customer representative within PP.
2. The Monthly Computer Utilization Reports (by project ID) which are produced by DMIS/P should be systematically distributed to project managers.
3. Open shop projects should be tracked utilizing DMIS/P or a similar program. A system should be devised to track utilization of mini computer resources.
4. For all average to large size development and maintenance projects, reviews should be scheduled and conducted at the completion of each major lifecycle phase (requirements definition, preliminary and detailed design, construction, and operation). Reviews for small projects should be conducted following requirements definition, detailed design, construction and operation. Representatives of an independent quality assurance organization/function and the upper level manager responsible for the project should participate in these reviews.

The DMA, which is maintenance oriented, does not have a formal system for incorporating and controlling approved changes to production programs. The DMAHTC does not have an established library of production programs. These characteristics cause a number of side effects:

- Known problems may not be corrected.
- Temporary fixes may cause other problems.
- User documentation may not be up to date.
- A problem-ridden system may be discarded.
- Duplicate effort may be expended to develop a new program to serve the same functions as a discarded problem-ridden system.
- Bad data or incorrect results may be produced for one user by a system which is known (by another user) to contain errors.
- Multiple versions of a program may exist.

Each of these side effects can be very costly and are counter productive. During interviews we recorded instances of each of these side effects. The DMA needs a methodology to record, analyze, and correct all detected program errors and update all associated software and documentation when program change is necessary. We estimate that such a methodology, if formally applied, could reduce the number of production program runs producing erroneous results by more than 50%. Because of the side effects described above and the potential for increased quality of production programs, the following recommendations are being made:

1. An individual should be identified and trained (as necessary) to fulfill the functions of a program librarian at each of the centers. As a minimum, an individual who will be responsible for production program documentation and associated change requests should be identified at each center.

This individual should collect statistics on the number and nature of changes to production programs in order to better define configuration management requirements at the DMA.

While a librarian function is provided at AC, it appears that the resources dedicated to library management preclude establishment of the full range of program library and configuration management activities.

2. A centralized discrepancy reporting system and file should be initiated and maintained by the Quality Assurance organization/function. The file will serve as a means of identifying the types and frequency of errors which occur during software development. Identification of typical errors will provide input to future requirements for tools for use during software development.
3. Configuration items should be defined for software subproducts (with complexity as defined in Section 6), permitting management, tracking, and control of each program module and all documentation.
4. For all average to large (see complexity levels as described in Section 6) software development and maintenance projects, configuration control levels and forms (as defined in Section 4.1.4) should be employed. Responsibility for control of the forms and correct employment of the levels should be vested in a Change Control Board. Information from these forms and data needed to employ change levels should be incorporated into requirements for a change control tool. Statistics which have been collected pertaining to the type and frequency of changes to production programs should also be evaluated to determine the requirements for a configuration management tool. These requirements can then be used to evaluate existing tools or as a baseline when defining the requirements for development of a tool.

5.1.3 Other Recommendations

A number of other recommendations have been formulated which do not pertain to introduction of a particular technique or discipline, but which resulted from observations made during interviews or were otherwise deemed to be necessary. No contribution to productivity or quality has been quantified or estimated as a result of implementation of these recommendations. However, from our experience these recommendations should make the programmer's job easier and will have a positive effect on morale and therefore should be implemented. These recommendations are listed in order of our perception of their potential contribution to the DMA:

1. Acceptance criteria should be defined for each software product requirement. Sample acceptance criteria specifications have been illustrated in Section 4.1.2.
2. Individuals should be identified and trained (as necessary) to provide support and technical consultation about the use of software development techniques on an as-needed basis.
3. A series of technical lectures and workshops should be scheduled and conducted to transfer new technologies throughout the centers. Some topics which could be discussed during early sessions are the employment of standards and procedures, the use of a software notebook, and techniques for specification of requirements and design data.
4. A brief technology newsletter should be drafted to include notice of pending training courses, workshops, and seminars. The newsletter should make

reference to any new tools and techniques introduced to the DMA, specifying a Center point-of-contact for further information.

5. The software qualities discussed in Section 4.1.2 should be studied. A subset of these qualities which represents important DMA product qualities should be defined as a guide to future development of a comprehensive software Quality Assurance program.
6. The recommendations made as a result of the Error-Off Study performed by FEDSIM should be implemented.
7. The tools identified by the MPE Study should be described in a Tools Catalog.
8. The Technical Information for Programmers (TIPs) mechanism of technology and information transfer should be implemented at HTC.
9. All tools which are acquired or developed should be implemented within the centers in accordance with the plan specified in Section 4.1.7 or a revision thereof.
10. Tools and techniques for use in the early phases of software development (some of which were identified in Section 4.1.6) should be evaluated for possible inclusion in the DMA tools library.

5.1.4 Recommendations for Long-Term Implementation

As the DMA becomes increasingly oriented toward use of the computer to satisfy its production needs, it will also develop much more formal development, legislation and enforcement policies. We have seen this transition occur in many development shops. In order to ensure quality, reliability, and maintainability of the software product and cost-effective application of all computer resources, rules must be standardized, delegated and enforced. Individuals rarely like this formalism, contending that it promotes excess paperwork and red tape. However, experience has shown that the best way to ensure quality is to specify how it is to be measured, monitored and controlled. The following recommendations are being made to assist the DMA in production of a quality software product. We have no quantitative data which will relate these recommendations (either positively or negatively) to productivity. Indeed, we can only qualitatively relate these recommendations to increased quality. However, we have seen these practices used by software houses which produce high-quality, dependable software and we are therefore recommending them to the DMA as a step toward a DMA MPE. As these practices continue to be employed in the industry and within the government, more quantitative data will be acquired. Other practices, in turn, will be defined and applied. Therefore, we suggest that the DMA use these recommendations as guidelines for long-term implementation, and seek additional quantitative data as it becomes available.

1. Each of the standards which have been adopted at the centers and incorporated into the Product Standards Handbook should be documented in a DMA Software Policy manual or handbook. Issues which should be part of DMA policy include programming standards, review and reporting procedures, change procedures, use of tools and techniques during the software lifecycle, and procedures to be employed during requirements definition, design and

coding. The Policy Handbook should be documented to facilitate the use of standard practices and procedures during the software lifecycle (as illustrated in figure 4.1.5-3).

2. An independent quality assurance group should conduct unscheduled audits of average to large development and maintenance projects at the centers. Auditing for conformance to established programming and phased development standards should be performed. Software notebooks should be studied to evaluate completeness of contents periodically throughout the development/maintenance activity.

5.1.5 Tools Recommendations

5.1.5.1 FAVS

Information recently obtained concerning the current status of FAVS points to increased positive opinion about the status of the tool and the functions it performs. In addition, modifications and enhancements are being systematically applied, and execution time has been substantially reduced. Data originally obtained while at the centers indicated that there was a need for the tool.

Study of FAVS manuals indicates that reports which it produces will significantly aid in debugging and testing. In addition, these reports will fulfill some of the program documentation needs, automating a function which, because it is uninteresting and cumbersome, is not currently being adequately performed.

The DMA should continue to support the upgrade and subsequent use of FAVS. However, introduction of the tool to the DMA community should be carefully planned in order to ensure favorable reception to its use.

The following are suggestions concerning FAVS:

1. Modifications to the system should be accompanied by a descriptive set of acceptance criteria to ensure that they are implemented correctly.
2. A contract with a minimum duration of one year should be entered into with the contractor, General Research Corporation, (or other contractor with experience in automated verification), which will provide maintenance, consulting, technical and training support. Specifically, services provided should include:
 - a) Participation in selection of at least two pilot projects within the DMA which will employ FAVS during development activities.
 - b) Training of personnel from the pilot projects pertaining to FAVS utilization.
 - c) Technical support during initial application of FAVS by the pilot projects. This should include job set-up, report interpretation, collection of statistics on run time, number and type of errors detected by static analysis number and type of fixes made as a result of program instrumentation, analysis and subsequent program modification, increased program efficiency resulting from implementation of instrumentation-generated fixes and documentation of depth of testing

(number of source lines executed during testing) resulting from FAVS use.

- d) Consulting during FAVS application by the pilot projects on an as-necessary basis.
 - e) Impact analysis, modification/update of code and documentation, and subsequent retesting of FAVS upon detection of any errors or ambiguities by the pilot projects.
 - f) Development of solid evidence using the statistics collected during activity c) above which will serve to introduce, justify, and "sell" FAVS to other DMA software developers. This evidence will in essence illustrate the cost and benefits of using FAVS.
 - g) Conduct of training courses in the centers introducing FAVS by presenting the evidence prior to detailed training.
 - h) Providing technical support and consulting during application of FAVS by the DMA communities.
3. An individual should be identified and trained (by the maintenance contractor as described in No. 2 above) to serve as an in-house technical consultant for FAVS.

These suggestions should increase the chances for successful introduction and employment of FAVS by the DMA.

5.1.5.2 PSL

Due to the size of many of the development applications destined to become DMA production programs, the functions provided by a development support library have a significant value. However, implementation of PSL has not been successful at the DMA. We have not been privy to information which documents the reasons for its (thus far unsuccessful implementation), receiving only nominal comments about the problems being faced in the interface with the Univac executive. Therefore, we cannot make detailed recommendations about the tool. However, the DMA should be aware of the following information. The Naval Weapons center in China Lake conducted a study of the development support library concept and its implications on a programming environment. The study, which resulted in production of a paper entitled An EXEC 8 Programming Support Library, contends that most of the functions accomplished by PSL are currently available through application of a set of EXEC 8 utilities. Since the report has only recently come to our attention, we have not had time to perform a thorough analysis of it or the implications which it may have for the DMA. However, most of the utilities discussed are available on the DMA Univac operating system and so the research should be applicable to the DMA.

Since the DMA has already purchased PSL, it will not be possible to integrate this information into a buy/no-buy decision. However, the study results should be incorporated into future decisions which are made in regard to PSL follow on efforts. Also, in light of the impending hardware upgrade at the DMA, contracts given to support the continued development/modification of the Univac version of PSL should be seriously contemplated prior to issue.

DMA implementation of PSL should incorporate the suggestions made for FAVS implementation in Section 5.1.5.1.

5.1.5.3 DMATRAN

Research supports the use of structured programming concepts and techniques to increase productivity and maintainability. However, as indicated earlier in this report, structured programming in FORTRAN can be accomplished without the use of a precompiler such as DMATRAN (STRUCTRAN-1 and -2).

The impact which use of a precompiler has on computer utilization (i.e., as a result of additional steps during compilation) could be substantial, considering the current job and computer saturation at the DMA. For this reason we do not currently recommend that the DMA adopt a standard mandating the use of DMATRAN on all FORTRAN project. However, we are convinced that structured programming concepts and constructs should be employed at the DMA. The DMA should establish and enforce a standard which requires the use of structured programming on all projects of complexity levels 2-5 (see Section 6). This standard should allow the use of either a structured programming precompiler or alternate constructs which can be employed without a precompiler.

5.2 IMPLEMENTATION PLAN - TEXTUAL DESCRIPTION

The centers should establish a Task Force with responsibility for acquisition, management and transfer of state-of-the-art tools and techniques. This Task Force should be centralized at the Headquarters level, and should have representatives from both centers in Levels 2, 3 and 4 of the conceptual model (see Section 2, Figure 2.1-1). The group should be organized such that members report at a high enough level within the DMA so that Task Force recommendations have the necessary influence.

At the start of this activity, a kernel Technology Acquisition, Management and Transfer (TAMT) group should be defined. Kernel group individuals should be familiar with software development and maintenance activities at the centers and should also have knowledge of other individuals in the centers who have related technical expertise. This kernel group should be responsible for initiating and managing activities in four major areas; Software Engineering Guidelines and Standards, Tools, Facilities and Technology Transfer and Training. The group should also study the recommended charter of the Task Force (see Section 4.1.7) to determine individuals from the centers who will best be suited to implement the recommendations. The group should be expanded as necessary to fulfill activities as defined in the charter. This task will also involve determination of leadership for each of the four subgroups. These leaders will have responsibility for refining and directing the activities of the subgroups.

The Task Force should be responsible for guiding the activities of the subgroups and for assisting the implementation of approved recommendations. This task includes TAMT participation in subgroup activities and coordination of requirements efforts in the subgroups. TAMT should also play a part in the decision of which subgroup activities should be conducted in parallel, and how all activities should be prioritized. TAMT will have responsibility for approval of all subgroup activities and planned efforts.

When the activities of the subgroups interrelate, TAMT will be responsible for coordination of the activities. This means that TAMT will assure that interrelated activities are discussed by both subgroups, that any decisions which result from these discussions are approved by both subgroups, and any disagreements which occur between the subgroups are resolved.

TAMT should also conduct periodic reviews with the members of the subgroups to ensure that activities are proceeding as planned. Periodic, unscheduled audits of the activities and products of the subgroups should also be conducted.

5.2.1 Software Engineering Guidelines and Standards Subgroup

The Software Engineering Guidelines and Standards subgroup should be established under the direction of the TAMT Task Force. The charter of the subgroup should include the recommendations documented in Section 4.1.5. The first activity of the subgroup should be to refine and enhance existing software development standards and guidelines. They should establish firm policies to ensure that phased development is practiced and that clear products of phases are defined. The existing set of programming guidelines should be better defined and should be expanded to incorporate the guidelines documented in 4.1.5. These

programming standards should be documented in a Standards Handbook which should be furnished to all DMA programmers and analysts. Project management procedures and guidelines (see Sections 4.1.4 and 4.1.5) should be defined and policies in this area should be formalized. Quality assurance standards as documented in Section 4.1.2 should be formalized. Standards for conduct of reviews and submittal and receipt of regular status reports (see Section 4.1.1) should be incorporated into DMA policy. Program configuration control mechanisms should be defined, incorporating the recommended mechanisms for requesting, evaluating, managing and implementing program changes.

References to all of the procedures, guidelines, and standards should be centralized in a document (DMA Software Policy document) which defines the policy of the DMA during the software development and maintenance lifecycle.

The subgroup should also be responsible for ensuring that software developers receive adequate training and support in the topic areas which are addressed by the policies and procedures. Personnel must understand the policies and procedures before they can be expected to implement them. This support/training responsibility should be coordinated with the Technology Transfer and Training subgroup.

The Standards subgroup should periodically review, revise and update the formulated guidelines and policies in conjunction with the Task Force to ensure that they remain consistent with software activities at the DMA.

The policies which are part of the DMA Software Policy document should be combined with the Standards Handbook and the Tools Catalog (see Section 5.2.3) into a DMA Software Development and Maintenance (D&M) Handbook which will serve as a reference for all individuals operating within levels 2, 3 and 4 of the conceptual model (see Section 2). The Handbook should also include standard operating procedures for submittal of jobs.

5.2.2 Technology Transfer and Training Subgroup

The Technology Transfer and Training subgroup charter should include those items which have been discussed in Section 4.1.7. A baseline of training recommendations has been defined in Section 4.1.8. However, before an elaborate combination of training and technology transfer mechanisms is initiated, this subgroup should develop a seminar to be attended by DMA individuals at levels 2, 3 and 4 of the conceptual model. This seminar should be designed to promote awareness of and provide the rationale for software engineering principles and discipline.

This subgroup should evaluate current DMA training curricula in an effort to provide the recommended training courses (see Section 4.1.8). Training/technology transfer needs which cannot be met via in-house sources should be identified in order that they may be met from external sources. A technology transfer and training consultant function should be instituted to answer questions which may come up when individuals begin to implement new technologies within the DMA.

This subgroup should develop and implement a Technology Transfer and Training Plan which expands the current training program in order to satisfy identified needs. Alternate external sources of training in new technologies should be evaluated based upon cost, availability and subject matter. Specific technology

transfer and training sources which will fulfill the identified requirements should be a part of the Plan.

5.2.3 Tools Subgroup

A recommended Tools subgroup charter has been defined in Section 4.1.6. Section 4.1.6 also defines the current inventory of tools at the centers. This baseline should be expanded through interaction with the production departments and computer service organizations. The tools inventory should be documented in a Tools Catalog (which is available to all programmers) in order to facilitate tool usage. The Tools Catalog should define which software activities each tool is designed to support, serving as a "user's guide" to developers. Each of the identified tools should be accompanied by current documentation. If documentation is unavailable, it should be produced.

The subgroup should participate in all tools study efforts which are currently being performed (and those which are planned) at the DMA. This task will consist of identification of study efforts, definition of problems/activities which the tools address, evaluation of the tools in a controlled environment (as discussed in Section 4.1.6), and documentation of the results of tool evaluation.

The subgroup should play a major role in the definition of requirements for tools at the centers. This requirements statement should incorporate the tools recommendations which are part of Section 4. Requirements should be defined by identifying software activities and development/maintenance problems for potential automation. User groups should be involved when identifying activities and problems. Automatable activities and problem solutions should be qualified by the expected benefits which can be derived via automation. The requirements definition activity should also define the potential for automating the function of checking for compliance with standards defined in the Standards Handbook.

From the identified tools requirements, a DMA system of support tools should be designed. Identified development/maintenance problems and software activities should be mapped to the functions of existing DMA tools. A literature search should be conducted and FEDSIM should be contacted to identify tools which are externally available. Any commercial or industry contacts which are utilizing tools in a similar environment should be contacted. The Tools subgroup should also support the identification of available tools which address program optimization. Activities not addressed by the functions of available DMA tools should be mapped to these externally available tools.

All identified tools should be evaluated (as described in Section 4.1.6) and the Tools subgroup should submit acquisition recommendations to the TAMT Task Force. After Task Force approval of tools recommendations, the Tools subgroup should be responsible for preparation of justification materials.

Upon receipt of new tools (or upon completion of modifications to existing tools), the Tools subgroup should be responsible for introduction of the tools into the DMA environment. This task will involve the development of controlled pilot experiments within which all advertised functions of the tools can be tested and its cost of usage can be determined.

The subgroup should be continually involved in updating the requirements for tools as the software environment at the centers changes. All problems which are

encountered with tools at the DMA should be directed to the subgroup so that periodic evaluation of DMA tools may take place. In this way, tools may be evaluated for their continued suitability to DMA requirements and future tool acquisitions may be initiated in a timely fashion.

5.2.4 Facilities Subgroup

The Facilities subgroup should be responsible for development of a plan for optimization of frequently used production programs. This activity will begin with identification to frequently accessed programs through the use of existing monthly accounting programs. Automated and manual means of analyzing programs for efficiency should be studied. The Tools subgroup should be involved in identification of tools which would support this analysis. Justification for identified tools should be accomplished via interaction with the Tools subgroup. Acquired tools and manual methods should be used to determine program inefficiencies. Programs should be modified to incorporate more efficient constructs. This modification should be performed in accordance with recommended change control procedures.

The Facilities subgroup should initiate other investigations of efficient utilization of DMA resources. Areas where the potential for increased efficiency have already been identified are in the utilization of alternate storage media and a remote job entry station.

5.3 IMPLEMENTATION PLAN - WORK BREAKDOWN STRUCTURE (WBS)

The WBS describes five main tasks which must be performed, incorporating the recommendations described in Section 4. These tasks have some interrelationships which have been defined within the activity network presented in Section 5.4. The WBS does not incorporate task interrelationships. Tasks at the second level (those with three digit identifiers) are followed by flowtime estimates and a manloading recommendation.

01 Establish and initiate the Technology Acquisition, Management and Transfer (TAMT) Task Force

011 Set up the TAMT Task Force

- 0111 Establish kernel TAMT group
- 0112 Define the charter, functions, and operating guidelines of TAMT
- 0113 Expand kernel TAMT to full Task Force

Expected Flowtime for Task 011 - 4 months
Recommended manloading - 4 man months

012 Establish charters for and organize subgroups

- 0121 Establish Software Engineering Guidelines and Standards Subgroup
- 0122 Establish Technology Transfer and Training Subgroup
- 0123 Establish Tools Subgroup
- 0124 Establish Facilities Management Subgroup

Expected Flowtime for Task 012 - 2 months
Recommended manloading - 2 man months

013 Guide subgroup activities

014 Coordinate Subgroup activities

Tasks 013 and 014 should be on-going activities which should be conducted at a half-person level of effort

015 Review Subgroup activities

016 Audit Subgroup activities

Tasks 015 and 016 should be on-going activities which should be conducted as needed, at about a half-person level of effort.

02 Operate the Software Engineering Guidelines and Standards Subgroup

- 021 Define and document guidelines and formulate policies for phased software development

- 0211 Define project management guidelines and procedures
- 0212 Expand and detail the existing programming guidelines, formulate new guidelines in recommended areas, furnish each programmer with the documented guidelines.
- 0213 Refine and enhance existing software Quality Assurance Guidelines and procedures to ensure that QA elements, mechanisms and responsibilities are clearly defined.
- 0214 Broaden the distribution of currently available project reports, formalize explicit procedures for conducting project reviews and reporting project activities and status.
- 0215 Formalize the current methods of program change control, incorporating recommended mechanisms for requesting, evaluating, managing and implementing program changes.
- 0216 Develop aids for practical application of tools and techniques during software development and maintenance (e.g., Standards Handbook, DMA Software Policy document).

Expected Flowtime for Task 021 - 26 months
 Recommended manloading - 52 man months

Task 021 should possibly be contracted out to take advantage of substantial industry experience.

022 Provide standards and guidelines training and support

Task 022 should be an on-going activity which should be conducted at a half-person level of effort. Training and support should be provided in-house when available, with seminars conducted as necessary.

023 Develop DMA Software Development and Maintenance (D&M) Handbook

- 0251 Document use/operation procedures for programmers
- 0252 Tailor DoD documentation standards as necessary
- 0253 Include recommended guidelines and procedures

Expected Flowtime for Task 023 - 6 months
 Recommended manloading - 6 man months

024 Periodically review, revise and update development/maintenance and programming guidelines and policies in conjunction with the Task Force to ensure that they remain consistent with DMA software activities.

Task 024 should be an on-going activity which should be conducted at a half-person level of effort.

03 Operate the Technology Transfer and Training Subgroup

031 Investigate the training needs of center personnel involved in software development/maintenance, including technical and management staff.

- 0311 Define a baseline of training needs by incorporating recommended training requirements
- 0312 Increase awareness of and provide the rationale for software engineering principles and discipline

0313 Investigate alternate mechanisms for training to introduce new skills or update existing skills as necessary to meet training requirements

03131 Training (internal and external sources)

03132 Seminars (e.g., BCS' Advanced Technology Seminar),
Conferences (e.g., IEEE, ACM)

03133 Undergraduate or graduate education (e.g., University of Maryland)

03134 Computer Aided Instruction

03135 On-the-job training

0314 Investigate methods for facilitating communication and technical awareness within the DMA

03141 In-house seminars

03142 Information Exchange meetings

03143 Technical memorandum

03144 Technical lecture series

03145 Colocation of technical staff

03146 Technical Information for Programmer (TIPs)

03147 Workshops

03148 DMA Newsletter

0315 Define requirements for a technology transfer and training consulting function

Expected Flowtime for Task 031 - 12 months

Recommended manloading - 18 man months

032 Develop a Technology Transfer and Training Plan

0321 Identify alternative methods to facilitate technology transfer and to expand current training program in order to satisfy identified needs

03211 Incorporate the recommendations of this study

03212 Expand upon the recommendations of this study as new technologies are identified

03213 Identify courses listed in the Education and Career Development Division catalog which will satisfy training requirements

03214 Check other sources (e.g., BCS' Education and Training Division courses) as necessary to satisfy the full list of identified training requirements

0322 Evaluate all identified alternatives based on cost, availability, and subject matter.

0323 Recommend specific technology transfer and training sources

0324 Prioritize technology transfer and training from the recommended sources

Expected Flowtime for Task 032 - 6 months

Recommended manloading - 9 man months

033 Implement the Technology Transfer and Training Plan

Task 033 should be an on-going activity which should be conducted at a one man level of effort, supplementing this level of effort with seminars and other training mechanisms as necessary.

04 Operate the Tools Subgroup

041 Inventory current tools at the DMA

- 0411 Identify/document baseline from the MPE study
- 0412 Expand through interaction with production departments and computer services organizations
- 0413 Develop and regularly update a catalog of tools, and make documentation available and current as necessary

Expected Flowtime for Task 041 - 4 months

Recommended manloading - 4 man months

042 Participate in all study efforts (e.g., FAVS, PSL) currently being performed on tools at the DMA

- 0421 Identify study efforts being conducted
- 0422 Define problems/activities which these tools address
- 0423 Evaluate tools in a controlled environment
- 0424 Document tool evaluation

Task 042 should be an on-going activity which should be conducted at a half-person level of effort.

043 Define requirements for tools

- 0431 Initiate requirements statement by incorporating tools recommendations from this study
- 0432 Identify activities and development/maintenance problems as defined by this study for potential automation
- 0433 Involve intended user groups in tool need identification to identify other critical needs
- 0434 Define and document additional rationale for and expected benefits to be derived through automation of development/maintenance activities or identified problem solutions
- 0435 Use information from Standards Handbook to identify possible automatable functions. Needs should be identified in all phases from requirements to operations and also to support management functions (e.g., reporting)

Expected Flowtime for Task 043 - 6 months

Recommended manloading - 6 man months

044 Design a system of tools to satisfy identified tool requirements using current inventory and evaluation as a baseline

- 0441 Map problems and software development/maintenance activities to existing tools in light of evaluation results

- 0442 Conduct a literature search to identify other externally available tools (e.g., Auerbach, Data Pro) which have potential to solve problems or automate activities
- 0443 Interact with FEDSIM to identify tools available in the federal government which have potential to solve problems or automate activities
- 0444 Support the Facilities subgroup when identifying tools to satisfy resource utilization optimization requirements
- 0445 Identify and approach other commercial and industrial contacts
- 0446 Map problems and software development/maintenance activities to known externally-available tools which have potential to solve problems and automate activities (e.g., PFORT as a standards analyzer)

Expected Flowtime for Task 044 - 18 months

Recommended manloading - 18 man months

- 045 Evaluate and make recommendations for acquisition of available (new) tools. Recommendations must be coordinated through and approved by TAMT.

- 046 Prepare justification for tool acquisition as necessary

Tasks 045 and 046 should be on-going activities which should be conducted at a half-person level of effort.

- 047 Coordinate the introduction of tools into the DMA

- 048 Periodically update the tools requirements and inventory and continually evaluate the suitability of tools in use at the DMA

Tasks 047 and 048 should be on-going activities which should be conducted at a half-person level of effort.

- 05 Operate the Facilities Management Subgroup

- 051 Develop production program optimization plan

- 0511 Identify existing production programs for optimization
- 0512 Identify ways in which programs can be optimized and streamlined
- 0513 Identify tools or other manual methods to aid in optimizing/streamlining of production programs in coordination with TAMT
- 0514 Develop justification for acquiring tools in coordination with TAMT
- 0515 Acquire tools or techniques for optimization activities
- 0516 Use the acquired tools and techniques to analyze the identified programs

- 0517 Modify the programs using the results of the optimization analysis
- 0518 Support efforts to implement configuration management and the program librarian function

Expected Flowtime for Task 051 - 6 months
Recommended manloading - 9 man months

Task 051 should possibly be contracted out to take advantage of substantial industry experience. FEDSIM should be another source of effort in this task.

- 052 Investigate the need for an RJE facility at AC
Study, evaluate and recommend alternatives for the RJE station

Expected Flowtime for Task 052 - 2 months
Recommended manloading - 2 man months

Task 052 should possibly be contracted out to take advantage of substantial industry experience. FEDSIM should be another source of effort in this task.

- 053 Investigate the potential for incorporating other storage media (besides tape) for selected applications (as discussed in the Facilities recommendation) into the DMA

Expected Flowtime for Task 053 - 4 months
Recommended manloading - 4 man months

Task 053 should possibly be contracted out to take advantage of substantial industry experience. FEDSIM should be another source of effort in this task.

- 054 Make recommendations in the Facilities area, incorporating the results of conducted analyses

Expected Flowtime for Task 054 - 2 months
Recommended manloading - 1 man month

- 055 Forward the recommendations to the Task Force for approval and implementation

The expected flowtime and associated manloading estimates will vary for this task, depending on the recommendations which are made as a result of the analyses.

5.4 ACTIVITY NETWORK

Figure 5.4-1 illustrates the planned activity network for implementing the recommendations made in Section 4. The activities at level 2 of the WBS (the three digit tasks) have been incorporated into the network. The resource estimates which appear before the activity node represent the expected flowtime associated with the activity.

The first WBS task, that of establishing and initiating the charter of TAMT, serves as the foundation for all the other tasks. Therefore, it constitutes the start of the Modern Programming Environment upgrading activities. The activities which are part of this task are not included in the activity network, since most of them are on-going. The flowtime defined with the first task is expended in setting up the Task Force, establishing and organizing the four subgroups.

When an activity is related to another activity, a dashed line with an arrow indicating the flow of activity has been used to represent this interrelationship. For example, the Facilities subgroup analyzes the requirements for automated tools which would support optimization activities. These requirements should be coordinated with the Tools subgroup before final recommendations are formulated.

Each of the four activity lines is terminated by a dotted line which directs the flow of activities back to the initiation and coordination activity. The subgroups should periodically revise their activities to remain consistent with the changing DMA environment. Revision of activities must be coordinated and approved by the TAMT Task Force (illustrated by the two-directional arrow).

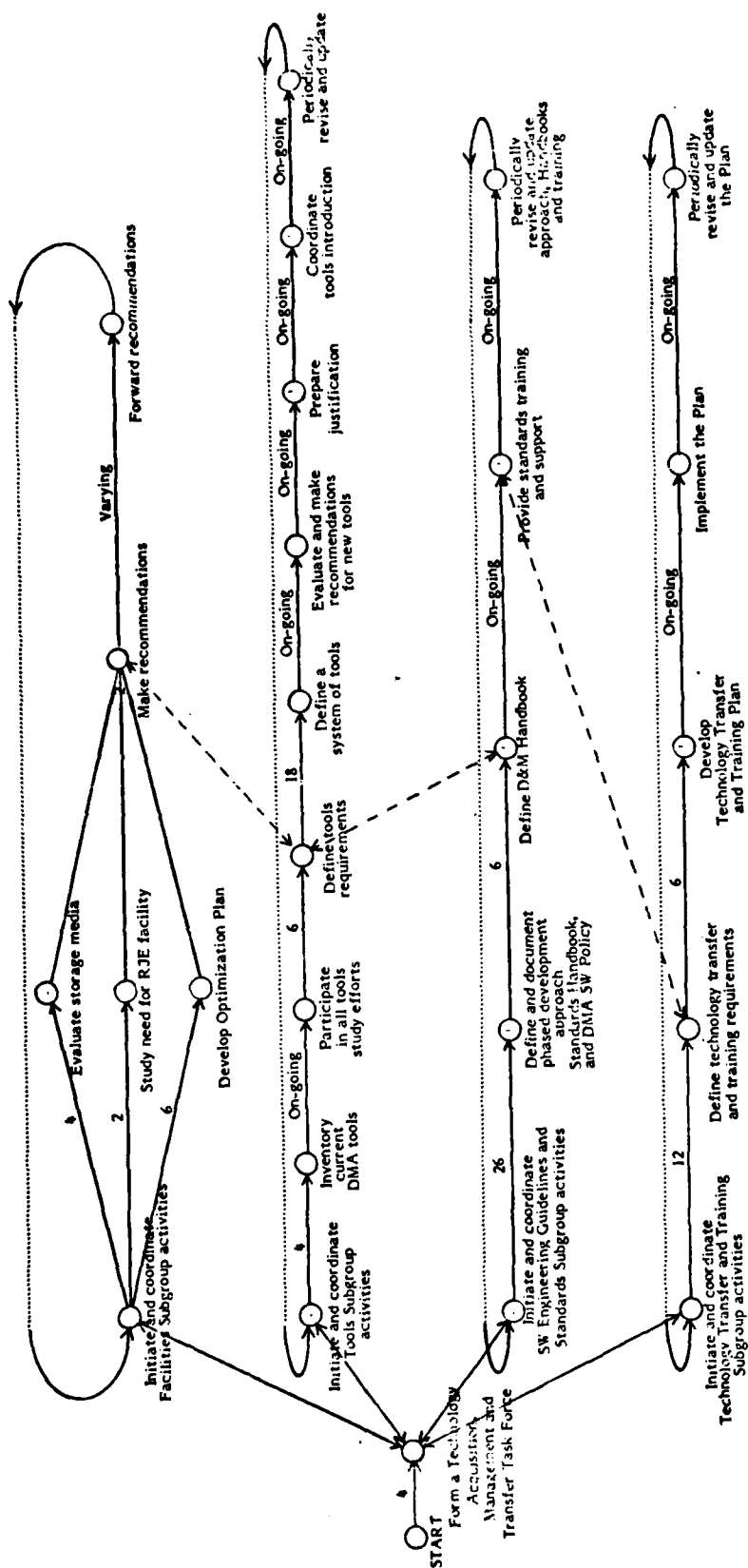


Figure 5.4-1 Implementation Activity Network

6.0 SELECTED APPLICATION OF MPE DISCIPLINES

This section will provide a framework for selection of discipline, techniques and tools for application to DMA programming development and maintenance projects. The framework was derived to be consistent with that which is presented in DoD 7935.1-S, Automated Data Systems Documentation Standards.

6.1 ADP PROJECT COMPLEXITY

DoD 7935.1-S provides guidelines for documentation of ADP projects based upon the determined complexity and formality of the project. Five levels of complexity have been incorporated into the framework. Basically, twelve factors are assigned a criticality level between 1 and 5, depending on the characterization of each project.

Figure 6.1-1 is a reproduction of the framework presented in DoD 7935.1-S. The twelve project factors are presented along the vertical axis, while five levels of complexity are listed on the horizontal axis. Each element represents the complexity level assigned to a particular factor characteristic. For example, a project where 5-10 people are assigned would be given a complexity level of 3 for factor 6.

This framework can be used to obtain an overall criticality (between 1 and 5) for ADP projects in the following manner:

1. Consider each factor and record the appropriate factor level value (1-5).
2. Total the factors recorded for each of the twelve items, divide that total by the number of factors used (twelve, if all twelve are used) and round your answer to the nearest whole number. The resulting answer will be a 1, 2, 3, 4, or 5. This value is the generalized complexity level of the project.

The computed generalized complexity level can then be used when determining which techniques, disciplines and tools should be applied during the project development/maintenance lifecycle.

6.2 COMPLEXITY/MPE RELATIONSHIP

From the recommendations listed in Section 5, those which represent the application of techniques, disciplines and tools have been extracted. Table 6.2-1 illustrates the suggested relationship between these software engineering practices and the computed generalized project levels (described in 6.1). This Table should be used as a guideline for application of the recommended practices to different types of DMA projects.

The table uses two categories of required practice: formal and informal. Any practice which is designated as F (formal) or I (informal) is required. The formality distinction should ultimately be the responsibility of the project manager. However, as a general rule of thumb, practices which are informally required:

- will not necessitate the full-time contribution of a project member,
- can often be documented by hand (as opposed to typewritten or word processed work),
- need not be reviewed by upper management, and
- are permitted exceptions to rigorous standards when justified by time and money constraints.

The following examples will serve to illustrate the use of Table 6.2-1.

A project manager has computed a generalized complexity factor of 3 for his project. After looking at Table 6.2-1, he knows that he must formally:

- plan and document his schedule, task breakdown, budget, and labor requirements,
- define and enforce the use of programming and documentation standards,
- define acceptance criteria for every functional requirement.

Every other practice must be informally done, the rigor which is used will vary by practice. For example, he must conduct a review at the end of each lifecycle phase, the visibility and attendance at which will be subject to his discretion. This differs from a level 4 complexity project because lifecycle reviews in that category have prescribed attendance and agenda.

He must employ a computer utilization analysis routine on tested code, however, the results produced by the analyses will be incorporated at his discretion. This differs from a project in complexity category 2 which need not employ utilization analyses, and a project in category 4, which must employ the analyses and must provide justification for noncompliance with utilization analysis results.

FACTORS		1		2		3		4		5	
COMPLEXITY:		1		2		3		4		5	
1. ORIGINALITY REQUIRED		NONE: REPROGRAM ON DIFFERENT EQUIPMENT		MINIMUM: MORE STRINGENT REQUIREMENTS		LIMITED: MORE ENVIRONMENT, NEW INTERFACES		CONSIDERABLE: APPLY EXISTING STATE OF ART TO ENVIRONMENT		EXTENSIVE: REQUIRES ADVANCE IN STATE OF THE ART	
2. DEGREE OF GENERALITY		HIGHLY RESTRICTED, SINGLE PURPOSE.		RESTRICTED: PARAMETERIZED FOR A RANGE OF CAPACITIES		LIMITED FLEXIBILITY: ALLOWS SOME CHANGE IN FORMAT		MULTI-PURPOSE/FLEXIBLE FORMAT, RANGE OF SUBJECTS.		VERY FLEXIBLE: ABLE TO HANDLE A BROAD RANGE OF SUBJECT MATTER ON DIFFERENT EQUIPMENT	
3. SPAN OF OPERATION		LOCAL OR UTILITY		COMPONENT COMMAND		SINGLE COMMAND		MULTI-COMMAND		DEFENSE DEPARTMENT, WORLD-WIDE	
4. CHANGE IN SCOPE AND OBJECTIVE		NONE		INFREQUENT		OCCASIONAL		FREQUENT		CONTINUOUS	
5. EQUIPMENT COMPLEXITY		SINGLE MACHINE ROUTINE PROCESSING		SINGLE MACHINE, ROUTINE PROCESSING, EXTENDED PERIPHERAL SYSTEM		MULTI-COMPUTER, STANDARD PERIPHERAL SYSTEM		MULTI-COMPUTER, ADVANCED PROGRAMMING, COMPLEX PERIPHERAL SYSTEM		MASTER CONTROL SYSTEM, MULTI-COMPUTER, AUTO INPUT-OUTPUT AND DISPLAY EQUIPMENT	
6. PERSONNEL ASSIGNED		1-2		3-5		5-10		10-15		15 AND OVER	
7. DEVELOPMENTAL COST		1-10 K		10-50 K		50-200 K		200-500 K		OVER 500 K	
8. CRITICALITY		DATA PROCESSING		ROUTINE OPERATIONS		PERSONNEL SAFETY		UNIT SURVIVAL		NATIONAL DEFENSE	
9. AVERAGE RESPONSE TIME TO PROGRAM CHANGES		2 OR MORE WEEKS		1-2 WEEKS		3-7 DAYS		1-3 DAYS		1-24 HOURS	
10. AVERAGE RESPONSE TIME TO DATA INPUTS		2 OR MORE WEEKS		1-2 WEEKS		1-7 DAYS		1-24 HOURS		0-60 MINUTES	
11. PROGRAMMING LANGUAGES		HIGH LEVEL LANGUAGE		HIGH LEVEL AND LIMITED ASSEMBLY LANGUAGE		HIGH LEVEL AND EXTENSIVE ASSEMBLY LANGUAGE		ASSEMBLY LANGUAGE		MACHINE LANGUAGE	
12. CONCURRENT SOFTWARE DEVELOPMENT		NONE		LIMITED		MODERATE		EXTENSIVE		EXHAUSTIVE	
TOTALS		X1*		X2*		X3*		X4*		X5*	
		* COMPLEXITY TOTAL:									

Figure 6.1-1
Level of Project Complexity

Level Recommended Practice	1	2	3	4	5
Internal reviews	O	O	I	F	F
Lifecycle reviews	O	I	I	F	F
Computer Utilization Analyses	O	O	I	F	F
Programming Standards	O	I	F	F	F
Development Standards	O	I	I	F	F
Acceptance Criteria	I	F	F	F	F
Documentation Standard	I*	I*	F*	F*	F*
Software Notebook	O	I	I	F	F
Problem Reporting System	O	I	I	F	F
Configuration Items	O	O	I	F	F
Configuration Levels	O	O	I	F	F
Configuration Controls	O	O	I	F	F
QA Audits	O	I	I	F	F
Planning/Scheduling	I	I	F	F	F
Monthly Status Reporting	O	I	I	F	F
Tool Application	O	O	I	I	F
F = required formal I = required informal O = optional					

*As defined in DoD 7935.1-S

Table 6.2-1
Application of Disciplines

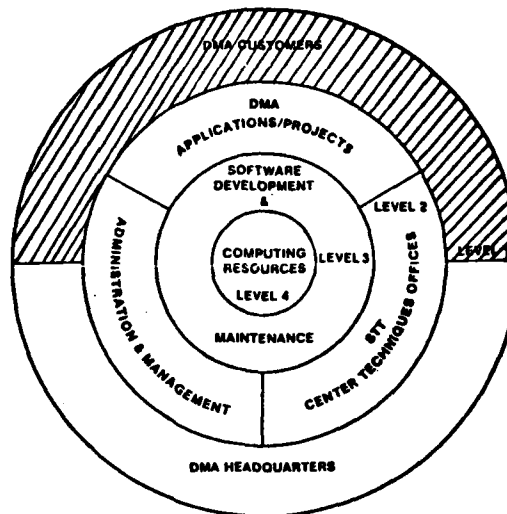
References

- 1) Myers, Ware; "The Need for Software Engineering," Computer Magazine, February, 1978.
- 2) Black, Rachael K. E., "Effects of Modern Programming Practices on Software Development Costs," Digest of Papers, COMPCON, Fall, 1977, pp 250-253.
- 3) Black, R. K. E., Katz, R., Gray, M. D., and Curnow, R. P., BCS Software Production Data Final Technical Report. Contract No. F30602-76-C-0174, Report No. RADC-TR-77-116, March, 1977.
- 4) Brown, J. R., "Modern Programming Practices in Large Scale Software Development," Digest of Papers, COMPCON, Fall, 1977, pp 254-258.
- 5) Belford, P. C., Donahoo, J. D., Heard, W. J., "An Evaluation of the Effectiveness of Software Engineering Techniques," Digest of Papers COMPCON, Fall, 1977, pp 258-267.
- 6) Williams, R. D., "Managing the Development of Reliable Software," Proceedings of the International Conference on Reliable Software, April, 1975, pp 3-8.
- 7) Baker, F. T., "Structured Programming in a Production Programming Environment," Proceedings of the International Conference on Reliable Software, April, 1975, pp 172-185.
- 8) Mujamoto, Issao., "Some Considerations in Database Application Programming," Proceedings of the Second International Conference in Software Engineering, October, 1976, pp 545-555.
- 9) Reiter, Robert W., Jr., "Investigating Software Development Approaches: A Synopsis," Proceedings of the Third Summer Software Engineering Workshop, September, 1979, pp 74-81.
- 10) Brown, J. R., and Hammond, L. S., Management Tools Case Study, Final Technical Report for Rome Air Development Center, Contract number F30602-78-C-0044.
- 11) Donahoo, J., Carter, S., Hurt, J., and Farquhar, R., Software Production Data, RADC-TR-77-177, Huntsville, Alabama: Computer Sciences Corporation, 1977.
- 12) Walston, C. E., and Felix, C. P., "A Method for Programming Measurement and Evaluation," IBM Systems Journal, No. 1, 1977, pp 54-73.
- 13) Gaulding, S. N., and Lawson, J. D., "Process Design Engineering: A Methodology for Real-Time Software Development," Proceedings, Second International Conference on Software Engineering, October, 1976, pp 80-85.
- 14) Lucas, L. W., An EXEC 8 Programming Support Library; Naval Weapons Center, China Lake, March, 1977.

- 15) Reifer, D. J., Documentation from the 1979 Software Tools Seminars, July, 1979.
- 16) Basili, V. R., and Reiter, R. W., Investigating Software Development Approaches; University of Maryland. Technical Report TR-688, August, 1978.

APPENDIX A - DMA AEROSPACE CENTER

I. CHARACTERISTICS OF THE DMAAC CUSTOMER ENVIRONMENT



A. Who are the customers?

The mission of the Aerospace Center is to provide aerospace mapping, charting and geodetic products, data, and services to the U.S. Armed Forces and to other DoD and federal agencies.

DMAAC Departments support the mission of the DMA customers by providing a variety of products. Software is often a subproduct of these products. As such, the following departments are users of computer resources and support:

- Geopositional Department
- Aeronautical Information Department
- Aerospace Cartography Department
- and parts of the Scientific Data Department

B. What are the services provided?

Through upgrading existing software, development of new support software and user/customer consultation, the following kinds of end products and services are supported:

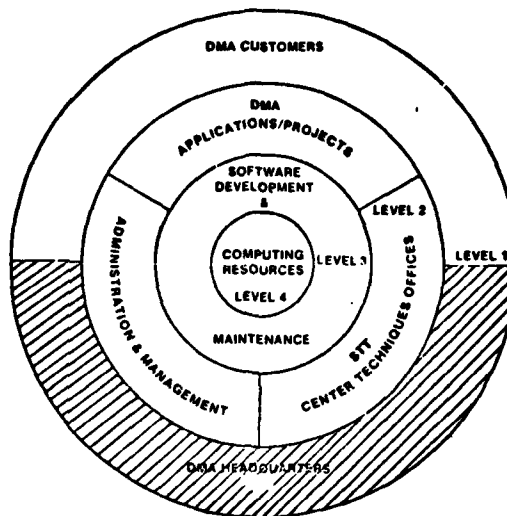
- aeronautical, extraterrestrial and astronautical charts
- air target materials
- operation and maintenance of centralized DoD libraries
- geodetic and geophysical studies and data
- digital and point positioning data bases
- cultural and terrain information

Software subproducts are used to produce, maintain, store, and manipulate data, to drive mapping and charting equipment, produce and validate mathematical models, generate data in a digital format, and other functions which support the end products of the DMA.

C. How do the customers interface with the DMAAC?

- 1) DMA project requirements are generally part of the five-year Program Objective Memorandum (POM) and are reprogrammed annually. These requirements first enter each of the centers through the Directorate of Programs, Production and Operations (PP). At the Aerospace Center, the Scientific Data Department (SD) provides Automatic Data Processing (ADP) software support for all mainframe applications. The Scientific Computing Division (SDC) of SD is contacted by PP when project requirements include provisions for software support.
- 2) Software support is provided by other departments or through a contractual arrangement with a commercial contractor when one or more of the following conditions are met:
 - a) The programming requirement is for problem-solving, one-time applications.
 - b) When unusual circumstances exist, such as a short-time requirement for which SDC does not have the resources to meet the time schedule.
 - c) When programming is being done by an organization outside DMAAC (e.g., Air Force Standard Finance Systems).
- 3) The initial request for software support which comes into the DMAAC via PP is usually stated in very general terms. SDC individuals are assigned to provide a liaison with the customer. Through this liaison, software requirements are analyzed and specified at a greater detail.
- 4) SDC Techniques Office serves as a focal point for all customers concerned about an on going or proposed project.

II. CHARACTERISTICS OF DMA HEADQUARTERS

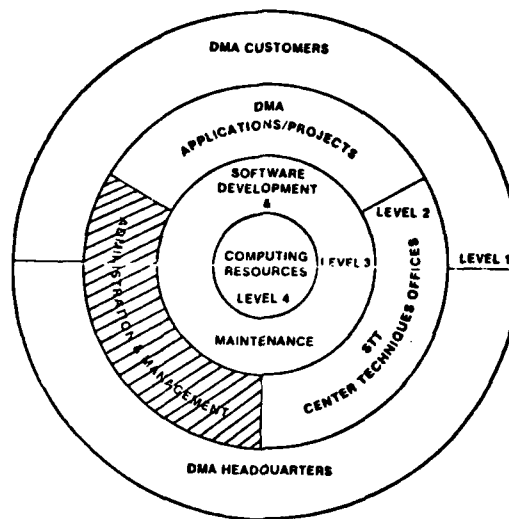


The planning, programming and budgeting (PPB) system provides broad guidance to defense efforts in the United States. The system begins at the level of the Joint Chiefs of Staff, filters to the Department of Defense and then to DMA Headquarters, (HQ). At the highest level (Joint Chiefs of Staff) a requirement is specified in a very general form. As this requirement filters down to the DoD and subsequently to DMA HQ, it is further defined and incorporated into PPB documents which are periodically (every 2-5 years) published. The DMA PPB documents which are periodically published are:

- DMA 8060.1 - DMA Programming Manual
- DMA 8040.1 - DMA Planning Manual
- DMA 7000.1 - DMA Budgeting Manual

These documents serve as formal input to the planning, programming and budgeting systems at the DMA centers.

III. CHARACTERISTICS OF DMAAC ADMINISTRATION AND MANAGEMENT



- A. Requirements for software products (whether for new software or maintenance to existing software) come through the Directorate of Programs, Production and Operations (hereafter referred to as PP). Center production requirements are planned every 5 years through a mechanism called the Program Objective Memorandum (POM). The POM incorporates elements of manpower, contract dollars and investment procurement dollars to accommodate the stated production requirements. Input to the POM comes from various sources. Every department submits written input of requirements which is based upon validated production activities. When these requirements are summed up, the total production requirements generally exceed the present center capabilities. When this situation occurs, the present capabilities are less flexible than the production program products themselves. That is, when requirements exceed capabilities, the requirements are generally decreased to meet the existing level of capabilities. Capabilities are rarely increased to meet validated needs. There seems to be resistance to changes in the operating budget.

Requirements are decreased by reassessment of needs and by establishing priorities for each of the production requirements. In this way, needs which have been reevaluated may be modified or may disappear. Low-priority requirements may be deleted from the POM to accommodate high-priority requirements.

- B. The POM is reprogrammed by the PP every year in the time frame from January to April. Input comes from the same sources as the 5-year plan. As could be expected, the yearly POM contains a much more accurate description of production product requirements and present capabilities which will satisfy these requirements.
- C. The software which is required to support production program requirements is part of the POM (far-term and near-term). Generally, estimates of the resources required to support software development and upgrade

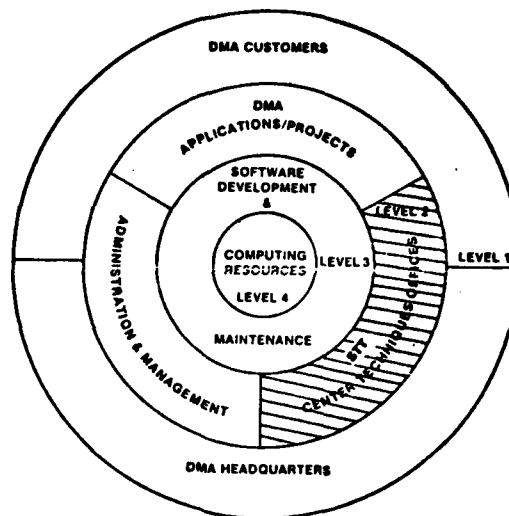
are based on past experience. Historical data concerning similar software activities is obtained and analyzed. This data is then used to project future requirements. Therefore, the accuracy of historical data plays a critical part in projection of requirements.

- D. Historical data is obtained from two major sources: the DMA Project Management Information System (DMIS/P) and the production departments (those departments which produced similar production software in the past). The DMIS/P system input is in the form of estimates as well as actual resource expenditures. This input is generated by the production departments at project initiation and actual expenditures are recorded on a weekly basis. This means that the actual expenditures are as reliable as the individual who records them. For this reason, the PP also contacts the production departments on a personal basis when obtaining historical data. In this way, they are able to obtain an informal assessment about the accuracy of the DMIS/P historical data.

By projection of actuals against estimates, experience can be gained which will make future estimates more reliable. At the current time there appears to be no automated way of comparing actual expenditures for software production to original estimates. A manual system for projecting the history of actual expenditures upon estimated expenditures is being employed.

- E. Development of each software production program goes through four phases: preliminary analysis, design, coding and test and acceptance. Each production program is assigned a production program manager from the PP. This program manager is responsible for the total production program system. He contributes to input to the POM in respect to the production program. The program manager is not a computer programming expert, he has general familiarity with the technical details and problems of computer systems. The software requirements are levied upon the production departments themselves. The program manager obtains status visibility via bi-weekly reports generated through DMIS/P and by maintaining a close interface with the production departments themselves. The PP program manager has a good perspective of when intermediate software products should be completed and he uses this knowledge to maintain status control of the production program. The PP program manager generally relies more heavily on these informal status control mechanisms than on the reports produced by DMIS/P.
- F. Comments were made that the DMIS/P system was cumbersome to use for status control because the system does not produce statistical information. Currently, the manual system of comparing actual to estimated activity detracts from the value of DMIS/P information.

IV. CHARACTERISTICS OF CENTER TECHNIQUES OFFICES AND STT



- A. The following departments at AC have Techniques Offices and Production Management Offices which serve staff functions for their departments:

1. Geopositional Department
2. Aeronautical Information Department
3. Printing and Distribution Department
4. Aerospace Cartography Department
5. Scientific Data Department

Techniques Offices and Production Management Offices also exist at the division level within some of the larger departments.

- B. The "Techniques Office Concept" has been in existence at AC for several years. The Concept is an attempt to bring together previous decentralized efforts to studying and developing better procedures and methods for overall operations (including ADP operations).

- C. The functions of the SD Techniques Office are:

1. Formulate Department technical objectives, evaluate current capabilities and direct and monitor Department technical activities.
2. Perform analysis and planning to ensure the proper integration of current and planned Department production processes into an effective and efficient production system. Included is the definition of resources, manpower, and training associated with the technical objectives.
3. Research, evaluate and direct Department technical studies in the fields of computing and information processing (including data base management), image processing, sensor simulation, products/source assessment and library science and photographic science for application to Department production.
4. Provide technical liaison and support development projects with other Center organization elements, DMA and the MC&G

community; perform staff studies and evaluate technical proposals for the department.

5. Direct the Department related DMA research and development program, including developing requirements statements for hardware, software and studies; coordinating on Department related research and development program items, developing requirement specifications, participating in in-process reviews and directing test and evaluation.
6. Direct the development of systems design specifications for new currently available capabilities and direct the certification of newly acquired or developed Department production systems software/equipment performance.

D. The functions of the SDC Techniques Office are:

1. Develop Division technical objectives in coordination with the Department; develop techniques and technical capabilities to satisfy objectives; plan for research and development required to provide technical capability.
2. Perform technical development required to support Division production assignments; develop system design and specifications for new capabilities; design plans for the direct acceptance and evaluation testing of hardware and software; establish plans for implementation of new capabilities into Division production; provide technical advisory services.
3. Research, evaluate and perform Division technical studies in the fields of computing and information processing; which includes such things as computer architecture, graphics displays and telecommunications for application to Division production and Center scientific and technical computing.
4. Assist other Department/Division techniques offices as required in technical development relating to computer science.

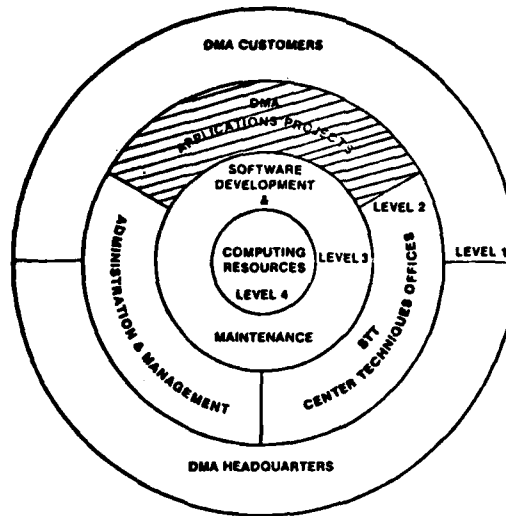
E. Directorate Mission

1. The Directorate of Systems and Techniques (ST) is composed of two divisions whose mission is to coordinate, direct and supervise the establishment of mapping, charting and geodesy technologies. This responsibility includes:
 - DMAAC staff activities pertaining to DMA research, development, test and evaluation programs.
 - Management and execution of assigned DMA special project programs for national level data acquisition system development.
 - Preparation, management and maintenance of Center technical management information system
 - Coordination of Center technology development achieved through techniques and engineering application activities.
2. ST assists the Directorate of Personnel in the selection of candidates for advanced technical training programs.

F. ST Advanced Technology Division functions

1. Act as a focal point for proposed research, development, test and evaluation (R,D,T & E) activities.
2. Provide direction to Directorate and Division personnel, DMAAC elements, external organizations and contractors.
3. Coordinate, prioritize and direct inter and intra-departmental techniques office activities.
4. Initiate and coordinate special studies of ongoing production processes in order to guide development of capabilities which will support mapping, charting and geodesy (MC & G) objectives.
5. Prepare projections of RDT & E impact on manpower, training, equipment and budget to be included in DMAAC planning and resource documents.
6. At the Center and Headquarters level, participate in MC & G requirements analysis studies leading to development of new and/or advanced technology in support of emerging advanced weapon systems.
7. Provide direct assistance to DMAAC/PR and PP as requested to enhance MC & G product development in support of advanced weapon systems.
8. Assist Directorate of Personnel in all phases of selection of DMAAC candidates for advanced technical training. This activity includes defining needed training, validation of curricula and selection of candidates.
9. Provide representation as required for special studies and for conduct and coordination of research, development, test and evaluation programs and interrelated activities with DoD and other U.S. Government agencies and RDT & E laboratories.
10. Implement agreements established by DMA for conduct and coordination of RDT & E programs and interrelated activities with DoD RDT & E laboratories and other U.S. Government agencies.
11. Maintain continuous contact with the Acquisition Systems Development Division of ST and other related working groups to assure adequacy and continued exploitation of all source data and materials available to satisfy present and anticipated MC & G requirements.
12. Represent DMAAC in meetings and conferences with other government agencies, contractors, research groups and scientific societies in order to maintain and broaden DMAAC knowledges and to ensure that DMAAC is fully aware of current "state-of-the-art" in all areas related to advanced MC & G technology.

V. CHARACTERISTICS OF THE DMAAC APPLICATIONS/PROJECT ENVIRONMENT



- A. The users of computer facilities for software development/maintenance at the Aerospace Center fall into two broad categories:
- Those who use SDC services to satisfy their requirements (closed shop).
 - Those who do not use SDC services, but instead do their own programming (open shop), or obtain programming support from non-SDC sources. Users who perform open shop programming either use the mainframe computer or their own minicomputers.
- B. Regardless of whether open or closed shop programming is performed, all software projects are initiated due to a perceived need. This need is usually postulated in the Program Objective Memorandum (POM) (see section III.), and then becomes the responsibility of a particular department. The user of closed shop programming services participates mostly during the first and last phases of development - requirements definition and test and acceptance (T&A).
- C. The Scientific Data Department Computer Division (SDC) receives requests for closed shop programming support from the responsible Department through the Directorate of Programs, Production and Operations (PP). These requests are usually in the form of a letter or memo which describes the requirements in a high-level, generalized manner. The next step in the project life cycle depends on the size and complexity of the software development effort.
- D. If the programming effort is very small, requiring changes to a few lines of an existing production program, the user and SDC jointly determine how the change(s) should be made. SDC then reviews the effort and gives an estimate of when the changes can be applied. Once these changes have been applied, SDC turns the project over to the responsible department for Test and Acceptance (T&A). Testing is

performed using existing data and if the changed program passes T&A, the production copy of the program is updated.

Efforts which are larger require a more in-depth analysis. Generally, the responsible department performs a functional analysis of the requirements and subsequently writes a Functional Description document. This document is usually written in conformance to the DoD Standard for Documentation of ADP programs (Standard 7935.1-S).

- E. The completed Functional Description is forwarded to SDC. SDC studies the requirement and generates an analysis of the programming requirements. This programming analysis specifies such items as data base and computational needs. The programming analysis is reviewed by an individual in the user department who has a familiarity with the project or experience with a similar project.

The programming requirements are either accepted or respecified and sent back to SDC. For smaller projects (i.e., 1- or 2-person effort) the acceptance is usually informally given; larger projects may require that an acceptance review meeting take place.

- F. During the design and coding phases the user plays only a minor role, unless the requirements for the software change. Participation in regular reviews (i.e., biweekly, monthly) is not required, and the design methodology is usually transparent to the user.
- G. Most of the users interviewed indicated that requirements were often subject to change during the development cycle. This appeared to be the case due to the state-of-the-art nature of projects. Most users view the development cycle as a learning process and as the user and/or developer becomes better informed, the requirements are subject to change.

When project requirements change, the user contacts SDC and an informal review of the perceived change occurs. This review may result in revised project estimates.

- H. The divisions which perform open shop programming have a slightly different development cycle, however, their characteristics fall into the same broad categories as discussed. One such open shop group is in the Geopositional Department and uses the mainframe computer.

As with closed shop programming requirements, the project is initiated through the PP. At this point, a translation of the high-level need into project specifications occurs. Translation involves development of input and output specifications, algorithms and plans for a mathematical model(s) which will satisfy the project requirements. An informal review or discussion is held and an estimate of resources to be allocated is made. System development begins with design of the mathematical model. The model is then translated into a programming language. Once coded, the system is tested to ensure satisfaction of requirements. If the system passes acceptance tests, a letter or memo documenting this fact is sent to the departmental production office as well as to PP. If the system does not pass acceptance tests, problems

or exceptions are documented for review. Algorithms, input and output specifications and the mathematical model are verified and the program is changed accordingly.

Open shop programming in the organization just described is done in a research setting and is performed by one or two mathematicians who have had a few courses in programming. As such, techniques are not applied with much rigor and the application of standards is dependent on the individual doing the programming.

- I. As stated earlier, there are many diverse minicomputers being employed at the DMAAC. Programming on these minicomputers is being performed in an open shop environment. The largest minicomputer support group is now operating as part of the Geopositional Department. The group was originally a branch of the Scientific Data Department until the center was reorganized several years ago. This minicomputer group was originally dependent upon the mainframe machine for processing of data. However, new equipment has been added which handles the transfer of data between the minicomputers.

This group supports minicomputer applications within the Geopositional Department. Specifically, minicomputer services are provided to the Positional Data Division (GDD) and the Photogrammetric Data Division (GDM). Each of these divisions has its own Techniques and Production offices. The Production Offices submit information to the POM and are responsible for assignment of work within their divisions. The Techniques Offices are responsible for developing requirements specifications.

When GDD or GDC has a requirement for support by the minicomputer group (situated within GDM), the request is channeled through the production offices. The division level production offices submit a request letter to the department level production office. The department production office in turn forwards the letter to the division production office within GDM.

Once the minicomputer group has been notified that a request for its services has been submitted, they conduct an informal meeting with the user. During this meeting, detailed requirements are requested, often using the Computer Program Specification Worksheet illustrated in Figure A-1. After the user has defined the necessary information, the minicomputer group makes an estimate of planned resources needed to complete the requirements. The user either rejects or accepts the estimate. Upon acceptance, programming/analyst staff are selected and assigned to the project. The staff conducts worksessions during design and the user in this environment often participates in review and approval of the completed design.

- J. The user of open shop support plays a minor part during coding, resuming a major role during Test and Acceptance (T&A). The users' Techniques Office is responsible for test planning. The minicomputer shop sends a letter to its own divisional Production Office stating that the product is ready for T&A. The developer's division Production

COMPUTER PROGRAM SPECIFICATION WORKSHEET

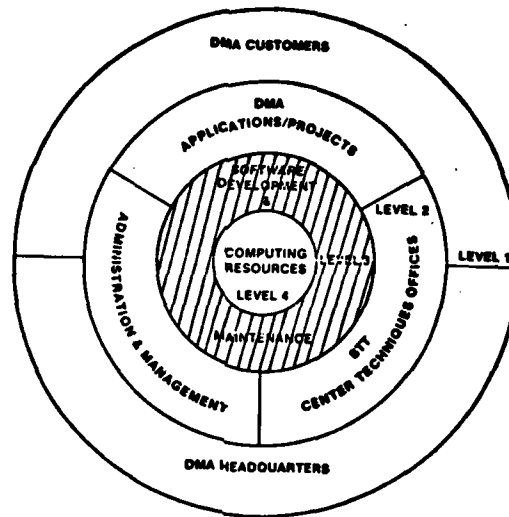
- I. Program Name (Functional, Descriptive)
- II. Program Description (Requirements or Purpose, Brief Description, Options Desired)
- III. Input Specifications (From what equipment, Media, Format, From what program, Security of Input Data, etc.)
- IV. Data Specifications
 - Units (Degrees, Inches, Radians, Etc.)
 - Max Sizes (amounts of data)
 - Minimum Sizes (amounts of data)
 - Normal Size (amounts of data)
- V. Frequency (How often will the program be run)
- VI. Timing Requirements (expected run time, expected turnaround)
- VII. Input Editing Requirements (Logical Checks, Blunder Checks)
- VIII. Math Model (Security Classification, References, Limitations, Variable Definitions)
- IX. Output Specifications
 - Accuracy Requirements
 - Displays
 - Media
 - Format
 - Input Into What Program
 - Print Format
 - Graphic Requirements
 - Data To Be Saved - What Media
 - Security Requirements
 - Units
 - Precision
 - Special Output For T&A
- X. Test Data To Be Provided
- XI. Contact Person
- XII. What Will Program T&A Be Looking For To Provide Acceptance Criteria?
- XIII. Other
 - Hardware Button Layout
 - Etc.

Figure A-1 Computer Program Specification Worksheet

Office next forwards the letter to the department Production Office which in turn forwards it to the users' divisional Production Office.

- K. One minicomputer group has initiated a policy whereby a review with the user is held prior to the start of T&A. The review is in the form of a demonstration of the software product and aids the users' Techniques Office during the testing process.

VI. CHARACTERISTICS OF THE DMAAC SOFTWARE DEVELOPMENT AND MAINTENANCE ENVIRONMENT



A. Project Types, Outcome and Size

1. Time is a key factor in the operation of the center. The center is under severe time constraints for the delivery of their products. This causes a direct impact in the delivery of software. In fact, the demands for products has an even greater effect on the software people than it does on the product people.
2. The software development activities of the center fall into three major groups:
 - addition of new capabilities to existing software
 - development of new software
 - detection/correction of errors in existing programs
3. Software development activities can also be differentiated by whether the effort is concerned with a production program (the development and/or maintenance of a program to be used by multiple users) or an effort to develop/refine/test a concept or algorithm.
4. Most of the production software developed is written in FORTRAN, some is written in COBOL, with a small amount in assembly language (only in special applications).
5. Digital data has become extremely important. Much effort is spent in producing, maintaining and storing digital data. Outcomes/products of projects are changing from 'paper' to digitized data, e.g., from chart to terminal or digital outputs.

6. Software production/maintenance is becoming much more important. With respect to this survey results indicated that:
50% of effort is spent in adding new capabilities to existing software
30% of effort is spent in development of new software
20% of effort is spent in finding/correcting errors.
7. Most software development tasks last from 3-6 manmonths in duration. These tasks usually involve a small number of people, 1-3.

B. Project Organization

1. Groups

a) Roles, Responsibilities, 'Team Members'

Comments were made indicating that there was often not enough involvement on the part of both management and users, and that the roles of these people were not specified in a formal manner.

b) Comments were also made indicating that, for small open shop projects, there often was not enough technical leadership/input from senior staff to the junior staff who often were responsible for the small efforts.

c) On small (1-2 person) projects in both open and closed shop environments, staff are expected to function in many different roles (e.g., requirements analyst, system analyst, programmer, customer liaison, tester).

2. User Involvement

a) The user plays the major role in test and acceptance (testing and acceptance usually indicates project completion). The user usually plays the Quality Assurance (QA) role. When test plans are developed, they are usually developed by the user, and these test plans are generally aimed at satisfaction of requirements. The test plan appears to be an acceptance test plan and not a development test plan.

b) The user participates in requirements analysis and specification. Although an informal agreement is often the outcome of this interface, some users develop a Functional Description using DoD Standard 7935.1-S.

c) The closed shop user sometimes participates in an informal review of the design.

d) The open shop user participates in resource planning.

e) The closed shop user receives resource estimates made by SDC.

- f) User needs often influence project scheduling.
- g) The user is involved in formal reviews during software development about half the time. Formal reviews are more often held at the beginning and end of a project and seldom in the middle.

3. Project Manager/Reporting Relationships

- a) There is usually a project manager (even if he is the only member of the team). Most people felt that the project manager had enough authority (with the exception of staffing and determining computer priority).
- b) Project leadership for large open and closed shop projects was often subject to change during the project lifetime.
- c) Generally, it was observed that senior people were assigned management of the larger projects and junior people the smaller projects. It was felt that the smaller projects suffered as a result of this policy.

C. Project Planning and Initiation

1. Source of Requirements

- a) The Directorate of Plans and Requirements (PR) is the source of external product (e.g., SAC product) requests.
- b) The request for SDC program development/maintenance services generally comes through the PP Directorate. However, this channel is sometimes by-passed.
- c) A request for SDC program development/maintenance services is generally in the form of a memo or letter. This request is generally reviewed by an SDC Branch Chief or Senior Analyst and a more formal programming analysis results.
- d) Some of the users who develop their own software on a mini-computer have produced their own requirements specification form (reference Computer Program Specification Worksheet Figure A-1).

2. User Participation in Requirements Definition

Requirements for computing support by the SDC are developed by the user (when he knows precisely what he wants) or jointly by the user and SDC (when the requirements are vague).

3. Estimating of Schedules, Costs and Resources

- a) Cost and schedule estimating usually involves comparison with similar projects and is usually done by individuals with estimating experience (generally a Senior Analyst or Senior Pro

grammer). Formulas are seldom used to estimate costs and schedules.

- b) Resource planning (for manpower and computing resources) is currently done on an informal basis, the accuracy of the current planning has been questioned, and the need for better resource planning and estimating procedures was expressed.
- c) Schedules are sometimes dictated by the user. Unrealistic completion dates contribute to lack of planning and inaccurate estimates.
- d) Unstable, ambiguous or inconsistent requirements contribute to inaccurate planning and estimating.
- e) People felt that the R&D (state-of-the-art) nature of projects complicates the planning and estimating processes and leads to inaccurate results.

4. Planning of Deliverables and Milestones

Project Milestones are defined about half the time. When they are defined, they are not always used to monitor project progress. It appears that when milestones are set, they are at a very high level (e.g., at the life cycle or phase level). Generally, larger projects define milestones more formally than smaller projects.

Documentation is rarely identified as a deliverable.

5. Project Plan Documentation

A Project Plan is seldom developed (formally defined and documented).

6. Staffing

- a) The need for special skills is recognized and addressed in the planning phase most of the time.
- b) Project staff (for open and closed shop) are assigned based upon experience and availability.
- c) SDC staff often transfer to other departments, serving as a pool of experienced personnel.

D. Production Software Life Cycle

1. General

- a) Results of the survey indicate that the following percent of effort is spent in the life cycle phases:

Requirements Specification = 14%
Planning effort = 11%
Software Design = 17%

Coding = 23%
Testing = 16%
Maintenance = 19%

- b) The percent of total effort devoted to adding new capabilities to existing programs was 49%, while effort expended in producing a new program from scratch was 31% and detecting and correcting errors in existing production programs accounted for 20% of effort.
- c) Maintenance activities were divided as follows:
- 1) analysis and respecification
of requirements 27%
 - 2) redesign 19%
 - 3) coding 34%
 - 4) retesting 20%

We assume that this distribution applies mainly to activities when adding new capabilities.

- d) SDCINST 5660.1, 'Management of Automatic Data Processing (ADP) System Development Projects', formally establishes procedures and responsibilities governing the management of ADP system development projects in the SD (all closed shop projects).

The instruction describes process flow during the development life cycle. The lifecycle has been divided as follows:

- Initiation
- Development
 - Definition
 - Design
 - Programming
- Evaluation
- Operation
- Maintenance

This lifecycle is similar to that of HTC, except that HTC views maintenance and revised operation as part of the Operational phase.

The instruction defines reviews which should be held during the lifecycle. These reviews are to be conducted by the SD Computer System Development Review Group (COMREV) at scheduled decision points to provide recommendations to the Department Chief and subsequently to PP. Members of COMREV are specified to include:

- SD Assistant Department Chief
- Chiefs of SD PMO, SD Techniques Office, SDC
- Chief of the organization which will ultimately manage the project.

Reviews are specified to be held:

- during Initiation to approve the Feasibility Study
- during Definition to approve the Functional Description
- during Design to approve the Program Specification
- during Evaluation to approve acceptance testing

During study at the centers, comments were made that the reviews specified by instruction 5660.1 were not always held due to critical project deadlines and lack of available review resources.

2. Requirements Phase

- a) Some people are aware of and use procedures when specifying requirements, while others do not know that such procedures exist. Our observation is that the most widely followed standard for contents of a requirements specification is DoD Standard 7935.1-S. Some of the open shop environments have defined the contents of their requirements specifications via worksheets.
- b) The outcome of the requirements definition effort in both open and closed shop environments is usually an informal agreement rather than a formal document. In either case, the requirements are defined via informal working sessions which are held during initiation (software specification). In the closed shop environment, a Functional Requirements Description document is produced more than half of the time.
- c) There is rarely a walkthrough during requirements analysis. Formal reviews of requirements (as specified by SDCINST 5660.1 during the Definition phase) are conducted less than half of the time. In the closed shop environment, the SDC prepares a programming analysis which is reviewed by the user. In the open shop environment requirements are translated into input-output specifications, algorithms and models which are informally reviewed.
- d) The most likely reasons for changes in requirements during the development effort are an ambiguous, incomplete or inconsistent specification and changing user needs/project scope.
- e) A number of comments were made that more time and effort should be spent in this phase, that there should be more participation by senior analysts and the user in order to fix requirements.
- f) Comments were made that the R&D nature of projects (e.g., dealing with state-of-the-art methodologies) was a major factor in the loose (informal) definition of requirements.
- g) A project file or notebook was informally kept for about half

of the projects (mostly large projects). When notebooks were kept, requirements were usually part of the notebook.

3. Design Phases

Design is usually documented by closed shop developers using the Program Specification. Open shop developers rarely formally document program design. System/Subsystem Specification documents are not usually produced in the closed shop environment. Data Base Specifications are produced about half of the time. However, this specification is being written more and more often.

4. Coding*

- a) Informal working sessions during coding are performed less than half the time. Techniques such as peer review of code are rarely used.
- b) Walkthroughs and formal reviews are rarely held.
- c) An informally defined version of a project notebook is sometimes kept by closed shop developers. Open shop developers do not use a similar mechanism to maintain program visibility. When the notebook is used by closed shop developers, it sometimes contains source code listings.

* Note: Standards, procedures and techniques used during this phase are addressed in a later section.

5. Testing

- a) The developer performs testing as a part of the coding effort and delivers the product to the user for more formal testing. Developer testing often includes informal working sessions. Many comments were made on the adequacy of development testing. Standards to be applied during development testing have not been formalized.
- b) Formal test planning is usually done by the users and often centers on satisfaction of defined requirements. Poorly-stated or undocumented requirements were generally felt to hinder the test effort. The user generally defines test cases and sometimes furnishes test data. Test Plans for large projects were often developed by the user, but documentation of the results of testing is not usually performed.
- c) A review of the results of acceptance testing is held about half the time. Completion of acceptance testing is the most likely measure of project completion.
- d) When project notebooks/files are kept in the closed shop environments, they sometimes contain documentation of test cases and results.

6. Maintenance

- a) Software is generally moderately difficult to maintain (e.g., correct errors, upgrade, etc). Reasons given for this difficulty were poor documentation, lack of internal commenting, lack of design documentation and the non-existence of a program maintenance manual.
- b) More effort is spent in adding to or modifying existing programs than developing new programs from scratch.

E. Project and Product Management

1. Product Control

- a) Change control procedures appear to be informal.
- b) Most user departments have a set of production programs which they frequently use. Closed shop requests for changes to these programs are generally made via letter to SDC. Once changes have been applied they are documented via a memo after completion of test and acceptance. No lists are kept which document particular users of library production programs.
- c) Closed shop users of data base support by SDC generally deal with SDC programmers who are familiar with their applications. No lists are kept that document which user is responsible for a particular data base.

2. Personnel Management

- a) Personnel are evaluated annually at the Aerospace Center. SDC programmers and analysts are evaluated based on the following factors:
 - timeliness of work completion
 - number and complexity of projects involved in
 - quality of work as judged by users
 - demonstrated knowledge of subject matter
 - number of subsequent revisions to project work
- b) Project staff sometimes changes during the development cycle (especially on large projects). Most people are involved in more than one project at the same time. Two observations were made in this area:
 - the most competent people were "spread too thin"
 - it was hard to manage individuals who were involved in simultaneous efforts

3. Project Management

- a) Monitoring and Control of Milestones and Software Phases

- 1) SDC instruction 5660.1, "Management of Automatic Data Processing (ADP) System Development Projects", establishes procedures and responsibilities governing management of ADP projects in SD. The instruction contains a development process flow description which specifies roles to be played by key individuals, actions to be performed, phases which development passes through and reviews which should be held. Five main phases are defined: initiation, development, evaluation, operation and maintenance. The instruction applies many sound concepts currently in use by software developers throughout the industry.

There does not appear to be any mechanism in existence to assure compliance to the instruction, consequently, adherence to the standard is not as good as could be expected.

- 2) Informal estimation of the effort spent in the closed shop life cycle is input to DMIS/P. However, this information is not usually used to monitor progress. Documentation of the DMIS/P system was obtained and it appears that it could be used at the project level for monitoring and control. However, reports from this system are currently supplied to SDC branch chiefs (one level above the project management) and the PP program manager and not local project managers.
- 3) When milestones were developed during the planning activity, they were generally not used for control.
- 4) Control of the evolving product through the software life cycle is difficult because clear products of phases are not produced.
- 5) Costs on a project level were not visible and did not appear to play a major part in the management of a project.

b) Communication and Visibility

- 1) Visibility is provided at the branch chief level (SDC) through DMIS/P reports. These reports are also furnished to the PP program manager.
- 2) For closed shop projects, SDC produces biweekly reports. These reports are informally made to the branch chiefs by the local project managers. The branch chiefs informally report status to the Department Manager of SDC. The reports to the Department Manager of SDC are sometimes in the form of exception reporting due to other critical demands for the time of the department manager and the branch chiefs.

- 3) Most people felt that they were not currently overburdened by review participation and that reviews were extremely necessary.

SDC developers felt that more reviews should be held and these should include the user to a greater degree. These additional reviews were requested to be held during development (not just at acceptance testing and requirements specification).

- 4) Comments were made that reviews were informal and should be more formal and structured.

Agendas for reviews are seldom formalized, and the responsibilities of participants are not often clearly understood.

Management feedback to these informal mechanisms was not substantial.

- 5) When requirements were changed, the communication mechanisms between the user and the developer were informal and this sometimes led to misunderstandings or lack of awareness.

c) Project Development/Maintenance Guidelines and Standards

- 1) There do not appear to be any overall center-wide procedures or standards for the way software maintenance or software development projects progress. The only standard which was recognized by all departments was DoD Standard 7935.1 "ADP Documentation Standard."

Project guidelines (which would provide help to developers when scheduling and conducting reviews, controlling the evolving product and documentation, etc.) are not standardized throughout the center. SDCINST 5660.1 documents the recommended approach to management of closed shop ADP system development projects. Deviations from the standard are permitted depending on project complexity. This instruction is not always followed for closed shop applications. Open shop developers do not have a similar standard and are generally unaware of instruction 5660.1.

- 2) SDCINST 8420.39, "Programming Guidelines", presents guidelines for development and maintenance of applications programs. It establishes SDC personnel responsibilities for conduct of reviews, walk-throughs and evaluation of project sub-products (e.g., documents). It also specifies guidelines to be followed by developers. These guidelines restrict:

- Use of variables in COMMON
- Data type dependencies
- Excessive passing of parameter between modules
- Assembly language programming
- Use of work files
- Use of non-standard Input/Output interfaces
- Use of mixed programming languages
- Use of backward jumps and GO TO statements

The instruction encourages the use of:

- Ascending statement numbering
- Meaningful variable names
- Explicit control statement options
- DATA statements to initialize arrays

The established guidelines are not detailed to a level which would permit unqualified interpretation, applicability and enforcement.

Programming standards and guidelines (e.g., coding standards) are recognized as being needed to sustain consistency throughout the center. This lack of consistency inhibits program maintenance. Even where guidelines exist, we have found no positive mechanisms which act as an incentive for their use.

Naming conventions are used as a means of producing more readable code. (However, no documentation of any standard naming conventions was found.)

Some data formatting and file formatting conventions are followed. (Documented guidelines or standards were not found.)

- 3) The time factor has a significant impact on the use of guidelines at the center. Most products are desired "yesterday" or in a hurry and short cuts are used so that the product can get out as soon as possible. The connection between guidelines and potential benefits is not made.
- 4) The multiple software development and maintenance environments that exist at the DMA will impact the future development/enforcement of guidelines and standards. The applicability of certain practices will depend upon the type of software development activity (e.g., maintenance, new development, algorithm experimentation).

F. Evaluation

1. Quality Assurance

- a) There are few existing standards which define quality, or against which code and/or documentation can be compared (other than DoD Standard 7935.1-S, SDCINST 8420.39 and SDCINST 5660.1).

However, attention is paid to the quality of software products, most staff feel a great responsibility to produce correctly working programs.

- b) The lack of sufficient standards affects the long range reliability of software and its maintainability. The informal testing standards that now exist don't assure that a program is adequately tested.

This lack of formality of the procedures directly impacts the enforceability of any standard operating procedures.

- c) The Test and Acceptance activities performed by the user in conjunction with the developer comprise the formal testing phase for most programs. Most of the testing performed by the developer is less formal and more geared toward debugging.

Procedures which are now followed during developer testing are not documented. Documentation of these would provide a basis for their review and revision (if necessary).

Programs are often used before T&A activities have been completed, and/or T&A may only test the major execution options/sections of the code and not detect errors existing in less often executed portions of the programs.

G. Documentation

- 1) DoD Standard 7935.1-S is used as the programming documentation standard for the center. This standard presents a method for determining which types of documents are to be produced depending upon project complexity. The standard also addresses in detail the content of each type of document. The standard permits some flexibility in compliance depending upon the complexity of the project.

Though the standard exists, we observed that it was only nominally followed by most open shop programmers and that adherence to the standard was not systematically enforced in either open or closed shop environments.

Some existing programs were documented before the standard was written and, consequently, do not adhere to the standard. A systematic method for updating older, non-conforming documentation has not been instituted.

- 2) Modules are internally documented most of the time, though the adequacy of the documentation can be improved. Apparently no guidelines exist that specify content or format for documentation internal to the software.
- 3) Within SDC, developers are encouraged to document programs as they evolve. However, software documentation is generally produced after the development of the software has been completed.

Some open shop developers which program on the mainframe computer do not appear to document the program(s) until acceptance.

H. Tools and Techniques

- 1) Tools are generally acquired through R&D activities, colleges and universities, or through the Univac Use Program Library Interchange (UPLI) program.
- 2) Structured coding (e.g., use of IF-THEN-ELSE, DO-WHILE) and Top-down program development are not widely used. Structran 1 and 2, tools to assist in structured coding in FORTRAN, are not utilized. (This can partially be attributed to the incorporation of STRUCTRAN 1 and 2 into FAVS)
- 3) Top-down design techniques are not used widely. No particular design methodology seems to be followed.
- 4) Project files are kept for most projects, but the contents of these files is not standard across the center. These do not appear to be a working file (e.g., used to document daily decisions, and monitor progress), but rather an archival file.
- 5) Only a limited number of automated (computer based) tools are available for use during software development and maintenance.

In general, those that are available are seldom used (of course, this varies by tool). The predominant reasons for this lack of utilization are:

- limited utility of the tool
 - difficult to use
 - inoperable, or only partially operable.
- 6) The following charts list the tools that have been identified as being available for use at DMAAC. The charts tally and summarize (using a weighted average) the part 1 survey results concerning the utility, cost, and availability of the tools.

	<u>Utility</u>			<u>Cost</u>			<u>Availability</u>		
	<u>H</u>	<u>M</u>	<u>L</u>	<u>H</u>	<u>M</u>	<u>L</u>	<u>H</u>	<u>M</u>	<u>L</u>
SNOOPY (Trace of program instructions)	4	2	4	0	4	4	8	1	1
PMD (Postmortem Dump processor)	11	5	0	0	0	12	13	0	0
Dynamic Dump Routines	6	1	1	1	1	4	2	3	1
FLAP (Flow Analysis Program)	0	1	2	0	1	2	0	0	2
INDEX (cross-reference lister)	4	3	3	1	1	5	8	2	0
University of Maryland File Editor	2	0	2	1	0	3	1	1	2
FILESCAN (cross-reference lister)	2	3	1	0	3	3	4	3	0
FORFLO (generates flowcharts for FORTRAN programs)	3	1	6	2	1	5	5	2	1
TIDY (cleans up FORTRAN programs, rennumbers, some editing)	3	6	2	0	2	6	8	2	0
REFORMATTER (reformats COBOL source decks)	1	1	2	1	0	2	0	3	0
STRUCTRAN-1 (converts structured FORTRAN to conventional)	0	0	5	3	0	1	1	2	1
STRUCTRAN-2 (structures unstructured FORTRAN)	0	1	4	3	0	1	1	2	1
FAVS (FORTRAN Automated Verification System)	1	1	6	4	2	1	0	4	4
PSL (Program Support Library)	1	0	4	3	0	1	0	1	4

	<u>Utility</u>	<u>Cost</u>	<u>Availability</u>
SNOOPY	3	2	4.4
PMD	4.4	1	5
DDR	4.25	2	3.3
FLAP	1.6	1.6	1
INDEX	3.2	1.8	4.6
U of M Ed	3	2	2.5
FILESCAN	3.3	2	4.1
FORFLOW	2.4	2.3	4
TIDY	3.18	1.5	4.6
REFORM	2.5	2.3	3
S-1	1	4	3
S-2	1.4	4	3
FAVS	1.7	3.9	2
PSL	1.8	4.0	1.4

Key:

for Utility

5 = High utility
3 = Moderate utility
1 = Low utility

for Cost

5 = High cost
3 = Moderate cost
1 = Low cost

for Availability

5 = Readily available
3 = Moderately available
1 = Relatively unavailable

- 7) The following chart gives more in-depth information about four of the tools in particular.

	<u>FAVS</u>		<u>PSL</u>		<u>STR-1</u>		<u>STR-2</u>	
	Yes	No	Yes	No	Yes	No	Yes	No
Training	7	7	0	12	3	9	3	9
Substantial Benefits	4	8	1	6	1	7	1	7
Complex inputs	8	2	3	1	2	3	2	3
Clear, concise, informative output	3	7	1	4	2	4	2	4
Supported by developer	3	7	1	4	2	3	2	3
Adequate doc	5	5	2	2	1	4	2	3
Available	10	2	3	5	5	2	6	1
I am inadeq. informed	7	5	11	3	11	3	10	4

- 8) The following tables tally and summarize the survey information for documentation.

	<u>Utility</u>			<u>Cost</u>			<u>Availability</u>		
	<u>H</u>	<u>M</u>	<u>L</u>	<u>H</u>	<u>M</u>	<u>L</u>	<u>H</u>	<u>M</u>	<u>L</u>
COBOL Debug Manual	2	1	3	0	2	2	2	2	0
FTN-PAPER (ASCII FORTRAN Guide)	6	1	1	0	2	4	4	2	0
RUNSTREAM (User Intro to Runstream)	9	3	0	0	0	9	7	3	0
GOULD (supplement to GOULD programming manual)	4	8	2	0	3	6	3	3	4
PLOT (SCD Plotter Programmer Manual)	3	4	1	0	3	3	2	3	2
PRGMAN (SCD Plotter Programmer Manual)	1	4	1	0	2	2	1	2	1
SORT (Information about 1108 sorting)	5	4	2	0	2	4	5	2	1

- 9) The utility of FAVS which is reported above does not incorporate data obtained after enhancements were made to the tool in July 1979. The version of FAVS which was utilized on a limited scale prior to these enhancements was inefficient, had several errors, did not have a good user interface and required a large amount of core memory. Since this time, four enhancements have been planned, each of which is to be followed by testing at the DMA centers. Telephone conversations with the RADC contract monitor and DMA contacts have revealed that enhancements completed as of September 15th have been almost entirely successful and have served to increase the perceived utility of the tool.

I. Project Support

1) Clerical/Secretarial Support

Secretarial support for software development and maintenance activities appears to be nearly adequate. A desire for more support was expressed by some, but lack of support was not indicated by a majority.

2) Computer Aid

A desire for more support from "computer aids" was expressed. Apparently there are tasks being performed by programming staff that do not require a high level of skill and could be performed by non-technical or less skilled personnel.

Key punching was specifically mentioned as a task that could be performed by staff other than the programming staff and where adequate support was currently lacking.

3) Management/Administrative

Management support was observed to be lacking in the amount of review and involvement. Some felt that there was a lack of management understanding of the problems encountered in and the complexity of ADP.

4) Technical Consulting

Technical consulting is handled in an informal manner. When a problem is encountered, peer group support is used to solve the problem or provide the desired information. A desire was expressed for increased availability of technical consulting through a more formal mechanism. Technical support for information/technology transfer was deemed very important and, when available, beneficial to software projects.

For some specific areas, technical consulting was lacking (e.g., tools, vendor support for minis, and systems programming).

J. Developers

A. Educational Background and Training

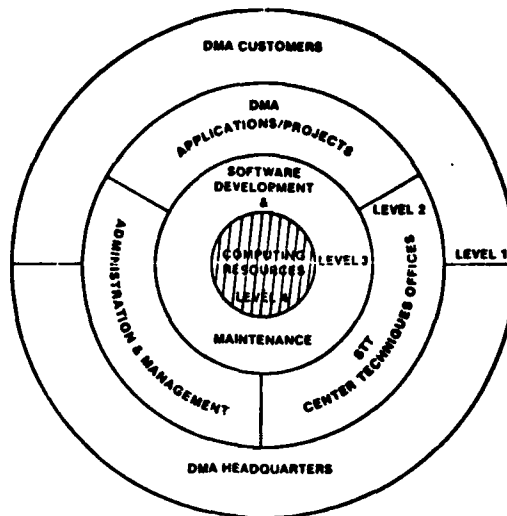
- 1) The predominant educational background of the staff is in either mathematics, physical sciences or geography.
- 2) Most staff receive training from the DMA, they attend an eight-week orientation program in cartography.
- 3) The emphasis in the past has been on cartography and not on computer/software skills.

- 4) SDC does provide training for selected areas relating to software development and maintenance. The courses cover basic areas such as FORTRAN, COBOL, ASSEMBLER, 1108 EXEX runstream, 1108 OS, database management, some Univac utilities, plotting, etc. There is little or no training in the areas relating to software engineering concepts, techniques and practices, and not much offered relating to software project management.
- 5) Training provided by the other departments does not emphasize ADP techniques and skills.
- 6) The center sponsors 2-5 people a year in long-term training. This usually accounts for about 30 hours in graduate computer science disciplines and curricula.
- 7) Computer Aided Instruction (CAI) courses in EXEC-8, DMS-1100 and data base technologies are periodically offered in-house.

B. Turnover

- 1) Staff movement between projects appears to be relatively high. Reassignment of staff is sometimes the cause of subsequent development problems.
- 2) Staff transfers between divisions in some cases can result in a significant loss of skills for the department from which the staff member is leaving. This is a particular problem for SDC when staff technically trained and competent in both computer and cartography transfer out and cannot easily be replaced.

VII. CHARACTERISTICS OF COMPUTING AND COMPUTING SUPPORT RESOURCES



A. HARDWARE CONFIGURATION

- 1) The hardware used is a distributed and diverse set of equipment. The environments in which software is developed and upgraded are not uniform; there is a mainframe computer and an abundance of minicomputers. The number of minis being employed for production programs is growing at a fast pace and will continue to grow in the future.
- 2) Most open shop users of computer resources felt that long turnaround time was having a negative impact on productivity and software quality. Given better computer turnaround on the mainframe machine, they felt that test thoroughness would improve.
- 3) Most of the users of minicomputers felt that they had good turnaround.
- 4) The mainframe computer(s) are
 - a Univac 1108 (MP)
 - a Univac 1100/42

The capacity (Dec '78) is 1340 CPU hours per month. Plans for equipment augmentation in the first quarter of fiscal year 1979 would add an additional 320 CPU hours (131 K of core).

- 5) Associated equipment as reported in December, 1978 includes
 - 4 DCT 2000 Remote terminals (card input and card and print output)

2 Uniscope 100 Terminals
1 Teletype terminal
1 IMLAC Graphic terminal

6) Other support equipment includes

IBM 1401
Flatbed, Drum and Gould plotters
PCAM Equipment
Versitic Electrostatic Plotter

7) A variety of minicomputers are in abundance. The following represents a list of dedicated minicomputer equipment by functional group which was presented to us during DMAAC briefings during December, 1978.

AEROSPACE CARTOGRAPHY INSTRUMENTATION

- 1 PDP 8 (Automated type placement system)
- 1 PDP 15 (Lineal input system)
- 1 HP 2100 (Gerber plotter)
- 1 PDP 11/45 (Gerber CRT print head)
- 1 Data General (CALAMA)
- 1 PDP 11/45 (Raster plotter scanner)
- 1 PDP 11/45 (Textual type placement system)
- 1 PDP 11/60 (Automated graphic digitizing system)
- 1-2 PDP 11/34
- 2-4 PDP 11/70
- 1 PDP 11/45 (DIMIAS)

INTEGRATED PHOTOGRAMMETRIC INSTRUMENTATION NETWORK

- 3 MODCOMP 11/45 (Central processor)
- 32 MODCOMP 11/25 (Device controllers)
- 3 PDP 15

ANALYTICAL PHOTOGRAMMETRIC POSITIONING SYSTEM

- 1HP 9810 (deployed in 1971)
- 1HP 9825

OFF-LINE ORTHOPHOTO PRINTER SYSTEM

- MODCOMP 11/25

DMA MANAGEMENT INFORMATION SYSTEM

- Burroughs 3500

SCENE SIMULATION

- LIS 2

PHOTO RECTIFICATION

- NOVA

AAIPS

- 2 DATA GENERAL S230
- 1 DATA GENERAL C330
- 1 DATA GENERAL M600

8. In addition, the Geopositional Department has several mini computers and associated equipment. A list of available equipment was not obtained.

B. COMMON JOB ATTRIBUTES

1. DMAAC has a Monthly Accounting Program which is used to categorize and document statistics about the utilization of the Univac 1108.

A copy of the output from the Monthly Accounting Program for the month of January 1979 and documentation which describes the Program was obtained. The Program produces 10 reports. These reports are described below.

- a. 1108 Utilization by PROMACS Job Identification

This report lists the total Standard Unit of Processing (SUP) time, the total Central Processing Unit (CPU) time and the number of runs for each unique PROMACS number.

- b. 1108 Utilization by Organization

This report lists program usage by branch, division, department and center. SUP time, CPU time and number of runs are shown by program, by PROMACS number and by organization. Summaries by run type usage (i.e., production, debug, rerun) are shown for each organization level.

- c. Site of Submission Report

This report lists SUP time, number of runs, number of cards read, number of pages printed and number of cards punched at the central site and all other submission sites (i.e., terminals, Uniscopes).

- d. Security Classification Workload Report

This report lists SUP time, number of runs, number of cards read, number of pages printed and number of cards punched at each security level.

- e. Data Systems Designator (DSD) Report

This report lists for each DSD number the SUP time and number of runs for each of the following run types: Productive, debug, DMAAC Rerun, and Univac Rerun.

- f. Utilization by Unique Program Number

This report lists for each unique program number the following: number of runs, run rank, SUP time SUP rank, accrued percentage of total SUP time, average SUP time, average SUP rank average elapsed time, the ratio of total elapsed time to total SUP time, CPU time and the ratio of CPU time to SUP time. A second part of this report ranks program usage by run frequency and by SUP usage.

- g. Run Frequency by Time of Day

This report is a bar chart portraying the total number of runs occurring within each one hour period.

- h. Run Frequency by SUP Time

This report is a bar chart portraying the total number of runs within various SUP time ranges.

i. Run Frequency by Active Time

This report is a bar chart portraying the total number of runs within various active time ranges.

j. Inactive Time report

This report lists idle time, down time, preventive maintenance time and systems test time.

These reports were studied and a few general statistics were gathered:

1. The total number of runs submitted during the period was 33,923.
2. Total SUP time elapsed was 2,776 hours, 17 minutes.
3. Total CPU time elapsed was 1,276 hours, 48 minutes.
4. Average SUP time per run was approximately 5 minutes.
5. Average CPU time per run was approximately 2 minutes, 15 seconds.

2. DMAAC runs are submitted using an identifier to classify run type. Run types are: productive, debug, Univac rerun and DMAAC rerun. A productive run uses a library program to produce a product or part of a product. A debug run is the type of run submitted during program development or maintenance. A run classified as Univac rerun was resubmitted because of a hardware or system problem which occurred during the initial submission. Generally, the run-stream does not change between submittals. A DMAAC rerun is a run which is resubmitted due to an operation or job handling error (i.e., failure to mount a tape on the initial submittal, poor deck handling causing a deck to be dropped).

Of the 33,923 runs processed during the period, 26,778 (79%) were productive, 6,851 (20%) were debug, 164 (.5 %) were Univac rerun and 130 (.4 %) were DMAAC rerun.

3. A more detailed breakdown (by department) is presented in Table A-1. As the Table illustrates, the Geopositional Department is the biggest user of the mainframe (Univac 1108) hardware resources, accounting for 53% of the total elapsed SUP time and 60% of the total runs submitted during the period.

The two divisions which use the Univac the most are the Photogrammetric Data Division (Geopositional Department) and the Scientific Computer Division (Scientific Data Department), contributing 14% and 28%, respectively, of the monthly runs submitted.

About 95% of the runs submitted in the Geopositional Department are classified as production jobs, while 58% of the Scientific Data Department's jobs are production. As these figures illustrate, the amount of production jobs submitted is substantial. Production jobs have a great impact upon existing computer resources.

4. The Utilization by Unique Program Number Report was the source of the following information:

Department Job Characteristic	Geopositional	Scientific Data	Aerospace Cartography	Aeronautical Information
Production Jobs				
Number Runs	15,912	7,486	2,896	393
SUP Hours	1,260	778	271	73
Debug Jobs				
Number Runs	795	5,242	587	127
SUP Hours	56	229	49	16
DMAAC Rerun Jobs				
Number Runs	39	64	14	13
SUP Hours	3	12	3	3
Univac Rerun Jobs				
Number Runs	9	133	10	12
SUP Hours	1	10	1	4

Table A-1

Sample Monthly Tabulation of 1108 Utilization by Department

- a. 32 programs were each accessed by 300 or more runs during the period. These 32 programs accounted for 21,079 runs (62% of total runs) during the month.
- b. 32 additional programs were each accessed by between 100 and 300 runs during the month (14%).
- c. 194 programs were each accessed by between 11 and 99 runs during the month (22 %).
- d. The remaining 211 programs were accessed by 10 or less runs during the month (2 %).
- e. The 10 most frequently accessed programs accounted for 44 % of the SUP usage during the month.

The above information further documents the burden placed upon the existing computer resources for production jobs.

5. The Run Frequency by SUP Time Report was the source of the data in Table A-2. As the table illustrates, most jobs elapse between 20 seconds and 30 minutes, with the largest percentage of jobs elapsing between 1 and 2 minutes.

<u>Elapsed SUP Time</u>	<u># Runs</u>
4 - 11 hours	12
2 - 4 hours	17
1 - 2 hours	143
30 - 60 minutes	928
10 - 30 minutes	3434
5 - 10 minutes	3211
3 - 5 minutes	3110
2 - 3 minutes	2304
1 - 2 minutes	4761
40 - 60 seconds	2869
20 - 40 seconds	4115
15 - 20 seconds	1561
10 - 15 seconds	1678
8 - 10 seconds	682
6 - 8 seconds	763
4 - 6 seconds	2008
2 - 4 seconds	910
1 - 2 seconds	357
.5 - 1 second	94
0 - .5 seconds	96

Table A-2
Run Frequency by SUP Time

C. USER SUPPORT/CONSULTATION

1. User support at the DMAAC is informal. Open shop users of the central site obtain help with hardware and software from two sources: SDC programmers/analysts and other open shop users. These users generally talk with other open shop users before contacting SDC. When they do consult SDC, they will generally approach someone who they are familiar with and trust. We did not observe a centralized consulting function within the SDC or the user shop areas.

Most open shop mainframe users felt that adequate consulting was available, although a few felt that a centralized consulting function would be beneficial.

2. Open shop users of mini computers obtain help from two sources: other open shop users and mini computer vendors. These users informally receive help among their peers and consult vendors when other alternatives have failed. Most of this type of user felt that vendor support of mini computer software was inadequate.
3. Closed shop users (SDC programmers and analysts) receive support informally through their peers. Most of them felt that this was adequate.

D. SYSTEMS SUPPORT/SYSTEMS PROGRAMMING

1. Closed shop operating systems programming is the responsibility of the Scientific Data Department Systems Programming Branch. The branch is also chartered with developing systems specifications, models and documentation for the mini computers within the Department as well as developing information storage and retrieval systems software for cartographic and intelligence data bases.
2. Each of the Departments which perform open shop programming on minicomputers are responsible for maintenance of their own hardware and development of their own systems software. Some systems software is furnished by the mini computer vendor.
3. Systems programs for the mini and mainframe hardware are almost exclusively written in Assembly language.

E. OPERATIONS/JOB HANDLING

1. The mainframe computer is run by the ADP Operations Branch of the Scientific Data Department. They are also chartered with monitoring the performance of the mainframe and associated plotting equipment.
- 2) Computer operations for the most part were judged to be "as good as can be expected." The high volume of jobs was cited as making operations/job handling tasks difficult.

- 3) Table A-3 illustrates run frequency by time of day for the month of January. As the table shows, most jobs are run in the period from 1:00 - 2:00 PM, the least jobs are run from 12:00 - 1:00 AM.

<u>Time of Day</u>	<u>Number of Runs</u>
12:00 - 1:00 AM	922
1:00 - 2:00	1143
2:00 - 3:00	972
3:00 - 4:00	1083
4:00 - 5:00	1239
5:00 - 6:00	1192
6:00 - 7:00	1154
7:00 - 8:00	1110
8:00 - 9:00	1268
9:00 - 10:00	1665
10:00 - 11:00	1744
11:00 - 12:00	1615
12:00 - 1:00 PM	1710
1:00 - 2:00	2287
2:00 - 3:00	1963
3:00 - 4:00	1807
4:00 - 5:00	1492
5:00 - 6:00	1484
6:00 - 7:00	1616
7:00 - 8:00	1402
8:00 - 9:00	1284
9:00 - 10:00	1231
10:00 - 11:00	1263
11:00 - 12:00	1277

Table A-3

Run Frequency by Time of Day

4. The predominant secondary storage device is magnetic tape. These tapes are generally owned by a particular user and are either stored in the SDC tape library or at the user's facilities. Observations point to increased storage at the user's facilities due to the length of time required by SDC to locate, retrieve and mount a particular tape requested by the user. While no data other than individual comments was available to support this claim, the number of tapes stored in the library (upwards of 30,000) and the number of tapes mounted in a day (during the first five months of the year, an average of 1800 per day), attests to the facilities required to support the use of this storage medium.
5. Idle time, down time, preventive maintenance time and systems test time all represent periods of time when computer resources are not processing either development or production work. The report which documents the frequency of this type of inactive time for the month of January has been reproduced in Table A-4.

As the table illustrates, most of the inactive time which occurs lasts for less than 1 minute. There were only three occurrences of inactive time over one hour.

<u>Range</u>	<u>Frequency</u>
0 - 1 minute	52
1 - 5 minutes	26
5 - 10 minutes	11
10 - 20 minutes	18
20 - 30 minutes	2
30 - 60 minutes	6
1 - 2 hours	2
2 - 4 hours	0
4 - 8 hours	0
Over 8 hours	1

Table A-4

Idle Time, Down Time, Preventive Maintenance Time
and Systems Test Time

F. UTILITIES AND PROCESSORS

1. A number of utilities and processors are currently available on the mainframe machine. Most of these utilities were obtained through the Univac Program Library Exchange or the University of Maryland. Below is a list of some utilities which are part of the SDC tool file and a brief description of the function of each.

- a. COMPILE - Reads a temporary program file Table of Contents and generates a heading card and processor call card for each symbolic element. These cards are then added to the run-stream and executed.
- b. DOWNDATER - Compares two symbolic elements and generates correction cards needed to change one element into the other.
- c. LOGPRINT - Prints SYSS\$SYSTEM\$LOG\$ entries in field data and octal.
- d. PFCK - Verifies the integrity of mass storage files when the integrity is questionable due to hardware or software failure.
- e. PROFILE - It calls the appropriate processor and produces headings and listings for each source element in a program file.
- f. RLIST - Makes specified absolute elements available for use by LINKS, aiding in checkout of re-entrant programs.
- g. UNDO - Takes a specified absolute element and attempts to reconstruct a symbolic element from it.
- h. FLIST - Creates a control stream which contains EXEC 8 control cards to produce a file listing.
- i. TDUMP - Tape dump routine for any 7 or 9 Track tape.
- j. DISCS - Lists information about 8440 and 8414 disks.

The results of the survey indicate that other utilities are being used, however, documentation was unavailable at the time of this Study.

- 2. Utility documentation (which is part of the SDC Tools File) is not user orientated or detailed. Our observation is that most of the utilities are not widely used because of poor documentation, limited applicability and lack of awareness of what is available. Of the ten utilities listed above, the most frequently used utility is FLIST. A need for better documented utilities with greater applicability was expressed by open and closed shop users alike.
- 3. Few utilities are available to open shop users of mini computers. These utilities vary by machine and are generally provided through the vendor.

G. TOOLS

- 1. Users of the mainframe computer (open and closed shop) have access to the SDC Tools File. This file contains a mixture of utilities and tools, the documentation of which can be generated via a computer run. The tools documentation file also contains manuals which are useful to applications programmers.

There is a fine line between what is termed a "tool" and a "utility." For the purposes of this study, the following definitions will be used:

- A utility is not used for program development, to validate a program or to aid in debugging. Utilities are often used in systems applications (i.e., changing storage medium, verifying absolute or relocatable elements, etc).

- A tool is used during or after program development to assist the programmer in debug, checkout and validation. Recently, tools have been developed to aid in the early phases of program development (e.g., requirements, design).
2. A brief description of the tools which are available from the SDC tools file follows:
 - a. File Editor - Used to make changes to an existing program file.
 - b. FILESCAN - Generates cross-reference tables from the relocatable elements in a program file.
 - c. FORFLO - Generates flow charts for FORTRAN programs.
 - d. INDEX - Generates cross-reference tables for FORTRAN source programs.
 - e. TIDY - Cleans up existing FORTRAN programs (renumbers statements, indents logic, etc).
 - f. REFORMATTER - Reformats COBOL source programs.
 - g. SNOOPY - Program trace routine designed for use with assembly programs. It provides an account of every instruction executed and its effect.
 - h. Postmortem Dump Processor (PMD) - Writes the final contents of the program's main storage areas into a diagnostic file. The PMD processor then edits and prints the information.
 - i. Dynamic Dump Routines - Collect, edit and print information from the storage areas used by a program (usually written in assembly language). The routines are used during program execution.
 - j. FLAP - Flow analysis program.
 - k. Structran-1 - Converts structured FORTRAN source programs to conventional FORTRAN.
 - l. Structran-2 - Converts unstructured, conventional FORTRAN to structured FORTRAN.
 - m. Fortran Automated Verification System (FAVS).

Each of these tools is the responsibility of the Scientific Computer Division. The Division is the source of all documentation requests and user support.

3. Earlier in this Appendix survey results of the usefulness, cost and availability of each of these tools was documented. Rather than duplicate that material, a few significant points will be extracted here.
 - a. The tool which was ranked the highest in overall usefulness was PMD. This tool was acquired through the Univac - User Program Library Interchange (UPLI) program and has been formally documented as a Univac "Systems Utility." The documentation is adequate, all options have been defined and many examples serve to illustrate the correct usage of the tool.
 - b. Dynamic Dump Routines, also acquired through UPLI, were also frequently used and were thought to be very useful.

Again, documentation for the routines is good and is significantly user-oriented.

- c. The two tools which were ranked as having the least usefulness are Structran-1 and Structran-2. Adequate documentation for both of the tools exists. The tools introduced a concept (structured programming) which would require additional development effort to apply, thereby lengthening the already too-short schedule for software development. The fact that structured programming makes programs easier to read, understand and modify was not generally accepted by programmers.
- d. Another tool which was rated as having low usefulness was FAVS. The documentation for this tool is adequate and a training course was employed to introduce it when it was first brought to the DMAAC. Comments were made to the effect that actual implementation of FAVS did not live up to the promises made during the training courses. This disillusioned many users who sincerely believed in its merit. Some individuals felt that the tool was brought out too early, that it should have been used by a test group before it was introduced to the center software developers in mass.

The utility of FAVS which is reported above does not incorporate data obtained after enhancements were made to the tool in July 1979. The version of FAVS which was utilized on a limited scale prior to these enhancements was inefficient, had several errors, did not have a good user interface and required a large amount of core memory accounting for its poor rating during our center visits. Since this time, four enhancements have been planned, each of which is to be followed by testing at the DMA centers. Telephone conversations with the RADC contract monitor and DMA contacts have revealed that enhancements completed as of September 15th have been almost entirely successful and have served to increase the perceived utility of the tool.

- e. User evaluation of the Program Support Library (PSL) is incomplete due to the current status of the tool. Difficulties arose when the developers of PSL sought to implement it at the Aerospace Center. Therefore, while a number of people felt that the tool would be extremely valuable, it is essentially untried in the Aerospace Center environment.

H. TYPES OF SERVICES

- 1. At the time of this Study, submittal of jobs to the Univac machines occurred in three ways: central site (over the counter), terminal or

via Uniscope. In December there were three terminals and two Uniscopes.

A copy of the Site of Submission Report for February has been reproduced in Figure A-2. As the Figure shows, all but 3,760 runs (approximately 221 hours of total elapsed SUP time) were submitted at the central site. 3,312 jobs were submitted at terminals 1A, 1D, 3C and 4B. 448 jobs were submitted at the Uniscope CRTs.

2. The predominant method of job submittal (batch) is illustrated by the number of cards which were read during the period. 12,418,588 cards, or a daily average of 413,953, were read in during the period.
3. During the month, 1,392,216 pages of output were generated. Converted to daily and run averages, 46,407 and 41 pages were generated, respectively.
4. Jobs with different security classifications are processed at the center. Four classifications are used: unclassified, confidential, secret and top secret. When confidential, secret or top secret jobs are processed, the machine cannot be used to process unclassified jobs. The core memory must be clear of all other jobs. Generally, the Univac 1108 (referred to as System B) is used to process all classified jobs. When possible, the confidential, secret and top secret jobs are processed during non-prime computing hours (late at night or early in the morning). When the demand for classified processing is low, system B processes collaterally with System A (the 1140/42).

A representative Security Classification Workload Report was obtained and studied. The following data was extracted from this report. Unclassified runs accounted for 22,804 runs or 67% of the workload during January. Confidential, secret and top secret runs accounted for 750, 3,280 and 6,876 runs (2%, 10% and 20%), respectively, during the same time period. This data illustrates the burden (32% of total runs) placed on the Univac systems to process classified jobs. While confidential and secret jobs are normally processed during non-prime hours, several comments were made which document prime time submittal of high-priority secret jobs, interrupting normal, unclassified job submittal. No data was available to document the frequency of such occurrences.

SITE OF SUBMISSION REPORT SYSTEM A AND B

Central Site

Total SUP Time	=	2555 Hr 26 Min
Total Runs	=	30163
Total Cards Read	=	11063610
Total Printed Pages	=	1263272
Total Punched Cards	=	750022

Terminal 1A

Total SUP Time	=	66 Hr 29 Min
Total Runs	=	1173
Total Cards Read	=	669717
Total Printed Pages	=	24372
Total Punched Cards	=	6339

Terminal 1D

Total SUP Time	=	101 Hr 26 Min
Total Runs	=	1959
Total Cards Read	=	340938
Total Printed Pages	=	80177
Total Punched Cards	=	11236

Terminal 3C

Total SUP Time	=	0 Hr 52 Min
Total Runs	=	10
Total Cards Read	=	3638
Total Printed Pages	=	228
Total Punched Cards	=	0

Terminal 4B

Total SUP Time	=	6 Hr 04 Min
Total Runs	=	170
Total Cards Read	=	13074
Total Printed Pages	=	3302
Total Punched Cards	=	320

Uniscope 1D

Total SUP Time	=	29 Hr 41 Min
Total Runs	=	168
Total Cards Read	=	312703
Total Printed Pages	=	17338
Total Punched Cards	=	12698

Figure A-2 Sample Site of Submission Report

I. ACCESS

1. Job priorities are ultimately the responsibility of the Directorate of Programs, Production and Operations (PP). Section IV details this function further. As software production is authorized, the project is given a priority number which determines how quickly the project is initiated. A Univac scheduling algorithm is used to determine order of processing for jobs submitted to the central site. Jobs submitted to the central site fall into one of three priority classes: standard, express, and high priority. Most medium-to-large size jobs fall into the standard priority classification. Small jobs can be given an express priority. High priority jobs are usually given a code identifier which is written on the run card. The operator then keys in a high-priority code and the job is processed before express and standard jobs.
2. In the past, jobs submitted by SDC were given a higher priority than jobs submitted by other departments. However, SDC jobs are not currently given higher priority than jobs submitted by the open shop programmers.
3. Survey results indicated that the average turnaround time for jobs submitted to the mainframe computer is 19 hours. Most responses fell in the range from 12 to 24 hours.
4. Computer support/computer access for software development and maintenance was thought to be poor. Two criticisms were expressed across the board:
 - very poor turnaround time was thought to be extremely detrimental to programmer productivity, especially during development and testing, and
 - lack of terminal access to the computer, (e.g., the almost exclusively batch environment) was a contributing factor to poor turnaround and poor computer access.
5. The job scheduling/priority system was criticized and cited as being a factor contributing to poor turnaround.

J. PROGRAM LIBRARY

1. A library of production programs is maintained by the Scientific Computer Division. As of December, 1978, the number of library programs was approximately 600. About 30 of these programs are used more frequently than others.

The number of lines of code in these programs varies, the average size being 1,000-2,000 lines. Some of the programs are very large, the most frequently used program contains about 7,000 lines. Most of these programs are revisions of earlier programs and, therefore, have been in use for several years. The documentation for these programs does not always conform with DOD Standard 7935.1, since many of them were written before the standard went into effect.

2. An effort was made to optimize some of the most frequently used production programs in the past. The Boole and Babbage Univac Program Evaluator (UPE) was used, as well as a tool called the Fortran Instrumentation Package (FIP) which was supplied by FEDSIM. To our knowledge, neither of these optimizers is currently being employed.

The Boole and Babbage UPE was used for a trial time period several years ago. Although it was successfully used on some programs, it failed on others. Consequently, DMAAC did not purchase the tool. To our knowledge, the tool is no longer supported or marketed by Boole and Babbage.

3. The Utilization by Unique Program Number Report documents the use of specific programs (production and non-production) during the month of January. The most frequently used program was accessed 2539 times. The program is a Terrain Processor program which is designed for digital data file maintenance. Data is stored in 2 dimensional (121 x 121) matrices and is used and manipulated in various ways.

A project file was requested and obtained. The package included:

- Programmer's Manual
- User's Manual
- History Folder
- Source Code of the Program

The manuals were studied and several observations were made:

- a. The User's Manual outline and format does not conform to DOD Standard 7935.1-S (ADS Documentation Standards). However, most of those items in the Standard are contained within the document (in a different format).
- b. The User's Manual lacks a general, descriptive section defining the purpose of the program.
- c. The User's Manual provides an extensive description of the cards required for input submission.
- d. The User's Manual lacks sample output from the program. However, sample plotter output is referenced as being "available in the SDC Program Library".
- e. The Programmer's Manual contains information which would normally be present in a Program Maintenance Manual *as defined by DOD 7935.1-5). It defines the common areas, files used and the overriding philosophy which was used when generating the program. This Manual does not contain general description of the program or a list of error conditions (there may not be any). The description of output files is brief and assumes a familiarity with various other programs. The

Manual does not contain procedures which would assist program maintenance (reference DOD 7935.1-S, Figure 3-08, Program Maintenance Manual, Section 4). Such procedures would describe use of Conventions, modification verification procedures, and data base maintenance requirements.

- f. Data which was available documenting the timing of this program varied depending on the size of input files and functions which were performed. The Monthly Accounting Report ranks this program as number 1 in total SUP time usage (15% of total SUP usage).
- g. The core usage requirement of ULB306 is 36K. Core requirements were reduced from 60K through the use of a packing mechanism or library function to reduce record size.
- h. The program accepts input from some combination (depending on application) of compacted tapes, old master file tapes, update tapes and a cataloged packed master file disc. During processing, a secondary disk may be required, and data may be off-loaded onto a drum for plotting purposes. During the program's output phase, various packed, new-master tapes are built.

It is our observation that the heavy usage of tapes as a storage device by this program is also typical of many other production programs.

- i. ULB306 is a revision and optimization of another program, ULB177-5. The revision was instituted to provide flexibility and ease of usage.
- j. Source code was studied and the following paragraph describes the program from that standpoint.

The program is composed of over 80 subroutines which vary in length from 10 to about 400 lines of source code. Some of the subroutines have very good internal documentation (comments), while a few do not include comments at all. The percent of subroutine lines which are comments range from 0-31%. This percentage does not appear to vary by the size of the subroutine. Programming style varies from subroutine to subroutine, emphasizing the manner in which the program evolved over several years. Some of the subroutines were written before the advent of programming guidelines or by a programmer who chose not to adhere to the guidelines, SDC Instruction 8420.39 (Programming Guidelines) is not strictly followed.

- 4. Programs developed in the open shop environment are incorporated into the SDC program library if documentation has been produced and the developers are willing to turn maintenance functions over to SDC. Open shop programs which will

not meet these stipulations are maintained by open shop users. As a result, the code and the documentation are not controlled using formal mechanisms and may not meet quality standards.

AD-A002 713

PLANNED SYSTEMS INTERNATIONAL INC CAMBRIDGE MA
DMA MODERN PROGRAMMING ENVIRONMENT STUDY.(U)
JAN 80 L STUCKI, J BROWN, L HAMMOND

F/6 9/2

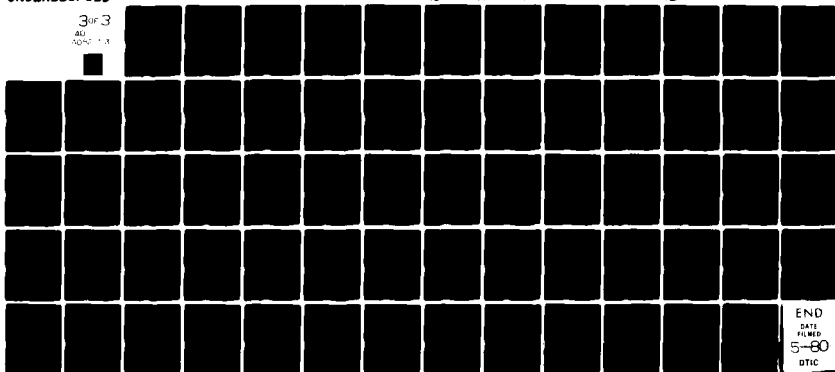
F30602-79-C-0022

UNCLASSIFIED

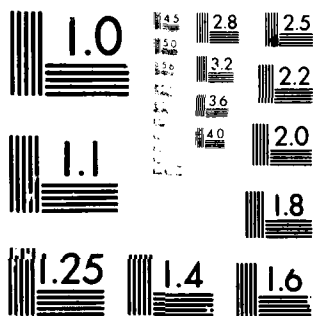
RADC -TR-79-343

NL

3 of 3
AU
2000



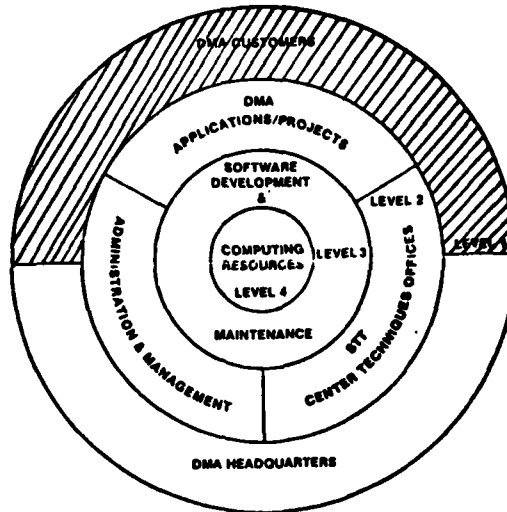
END
DATE
FILMED
5-80
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Appendix B DMA HYDROGRAPHIC/TOPOGRAPHIC CENTER

I. CHARACTERISTICS OF THE DMAHTC CUSTOMER ENVIRONMENT



A. Who are the major DMAHTC customers?

The mission of the Hydrographic/Topographic Center is to provide terrain mapping, charting and geodetic products, data and services to the U.S. Armed Forces and to other DoD and federal agencies.

DMAHTC departments support the mission of the DMA customers by providing a variety of products. Software is often a subproduct of these products. The following departments are major users of computer resources and support:

- Geodesy and Surveys Department
- Scientific Data Department
- Computer Services Department
- Topography Department
- Navigation Department

B. What types of services do they require? What type of software support do they require?

Through upgrading existing software, development of new support software and user/customer consultation, the following kinds of end products and services are supported:

- hydrographical and terrain charts
- land target materials
- geodetic and geophysical studies and data
- digital and point positioning data bases
- cultural and terrain information

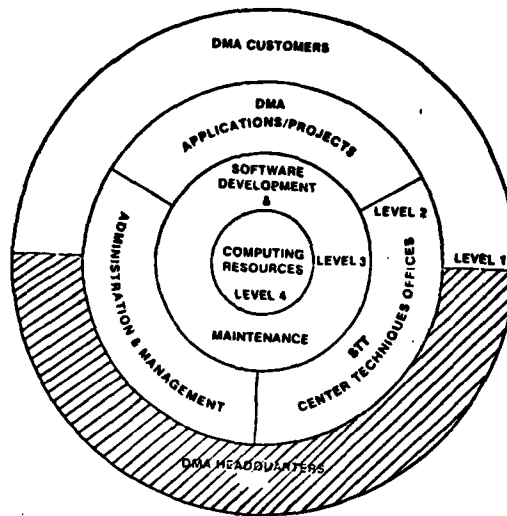
Software subproducts are used to produce, maintain, store and manipulate data, to drive mapping and charting equipment, produce and validate

mathematical models, generate data in a digital format, and other functions which support the end products of the DMA.

C. How do the customers interface with the DMAHTC? With the programming environment? What is their role in the specification of software requirements?

- 1) DMAHTC project requirements are generally part of the five-year Program Objective Memorandum (POM) and are reprogrammed annually. These requirements usually enter each of the centers through the Directorate of Programs, Production and Operations (PP). However, some requirements are submitted to PP by the production departments. At the Hydrographic/Topographic Center, the Computer Services Department (CSD) provides Automatic Data Processing (ADP) software support for all central site applications. The Production Management Office (PMO) of the CSD is contacted by PP when project requirements include provisions for central site software support.
- 2) Software development support is provided by other departments or through a contractual arrangement with a commercial contractor when one or more of the following conditions are met:
 - a) The programming requirement is for problem-solving, one-time applications.
 - b) When unusual circumstances exist, such as a short-time requirement for which CSD does not have the resources to meet the time schedule.
 - c) When programming is being done by an organization outside DMAHTC (e.g., Air Force Standard Finance Systems).
 - d) When CSD does not have functional knowledge in the application area.
 - e) When the application is being developed on non-mainframe equipment.
- 3) The initial request for software support which comes into the DMAHTC via PP is usually stated in general terms. CSD individuals are assigned to provide a liaison with the customer. Through this liaison, software requirements are analyzed and specified in greater detail.
- 4) CSD PMO serves as a focal point for all customers concerned about an ongoing or proposed project's interface with central site mainframe processing.

II. CHARACTERISTICS OF DMA HEADQUARTERS

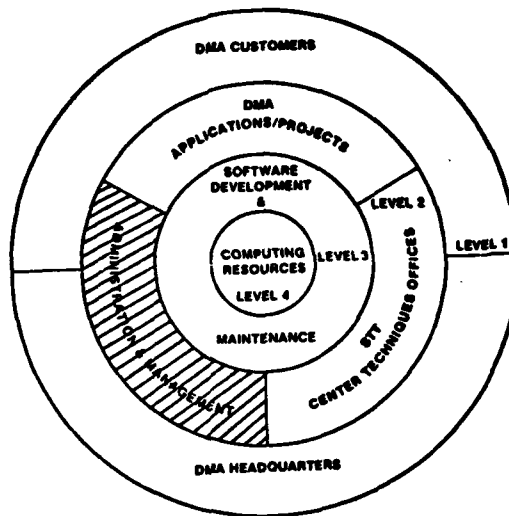


The planning, programming and budgeting (PPB) system provides broad guidance to defense efforts in the United States. The system begins at the level of the Joint Chiefs of Staff, filters to the Department of Defense and then to DMA Headquarters, (HQ). At the highest level (Joint Chiefs of Staff) a requirement is specified in a very general form. As this requirement filters down to the DoD and subsequently to DMA HQ, it is further defined and incorporated into PPB documents which are periodically (every 2-5 years) published. The DMA PPB documents which are periodically published are:

- DMA 8060.1 - DMA Programming Manual
- DMA 8040.1 - DMA Planning Manual
- DMA 7000.1 - DMA Budgeting Manual

These documents serve as formal input to the planning, programming and budgeting systems at the DMA centers.

III. CHARACTERISTICS OF THE DMAHTC ADMINISTRATION AND MANAGEMENT



A. Administration's Role Within the Center

1. Customer Interface

- a) The resources for software development are initially provided by the Directorate of Programs, Production and Operations (PP). Once the responsibility for software development has been assigned to a particular department within the DMAHTC, the management and control for software development is provided by the PMO of the responsible department.
- b) PP arbitrates and assigns software development resources based upon existing requirements and anticipated workloads.
- c) PP determines the necessity for contracting computer software to commercial contractors based upon existing and anticipated workload. The contracting arrangement must be approved by the Comptroller.

2. Responsibilities of the Comptroller

- a) The Data Automation Division (DAD) of the Comptroller serves as the senior ADP policy office and is responsible for establishing and updating software development policies. CSD works closely with DAD in many ADP areas, especially in monitoring the newly implemented ADP policies.
- b) CSD and DAD work together to assess functional area processing workload and resulting justifications for new ADP equipment which would interface with the central site processor.
- c) DAD functions include:

1. Review of ADP-related instructions annually to assure relevance to current DMA needs.
2. Serve as a focal point for DMAHTC software policy interpretation and enforcement.
3. Review of requests to grant waivers to existing policies, as necessary.

3. Personnel and Resource Acquisition

- a) CSD PMO serves as the fulcrum between functional operational requirements of central site data processing and the DAD. PMO reviews functional requirements for acquiring ADP equipment and recommends updates to the POM. ADP justifications are reviewed by PMO to determine the extent that the desired ADP capabilities would impact central site ADP. The justification is then forwarded to DAD for necessary support actions. When ADP acquisition involves complex technologies, PMO arranges consultation with CSD experts and representatives from interested production elements.
- b) The PMOs for each department usually manage the development and coordination of the departmental planning, programming and monitoring efforts. They are responsible for various administrative services (e.g., overtime, vendor training, contract support).
- c) New ADP capabilities (software, hardware and commercial services) are acquired in the following manner:
 - 1) Subject to DMA HQ approval of the updated POM, DAD requests the proponent Department to identify the individual responsible for the project ADP justification.
 - 2) A schedule of events for the project is developed by this responsible individual and representatives of PP and PR. General requirements are discussed and major questions are resolved. The justification request approach is agreed upon and an appropriate checklist is provided.
 - 3) The Department representative follows the justification format and completes each item on the checklist.
 - 4) The justification is reviewed by the appropriate Departmental ADP Coordinator and is submitted to DAD.
 - 5) DAD reviews the justification for completeness and content.
 - 6) PP, PR and ST individuals review the justification for content and adherence to work load projections and planning.
 - 7) DMA Headquarters informally reviews the justification.

- 8) DAD formally coordinates the justification within DMAHTC and transmits it to Headquarters.

4. Training

- a. CSD is responsible for training user personnel in the use of new systems which the Department is responsible for developing. Usually a course is offered after a major production system has been tested and is operational.
- b. Software developed by a commercial contractor is sometimes accompanied by developer training courses. However, the selection of individuals to attend these courses is not systematic. For example, a commercial database system (System 2000, developed by MRI Systems Corporation) is currently being implemented without CSD coordination.
- c. Some training courses are offered to all central site users at the center by CSD personnel. These courses address such subject matter as programming languages and job control language. Usually, a limited number of positions (openings) are available.
- d. CSD personnel receive general and specific training in software techniques sporadically throughout their careers. Attendance in scheduled training courses is often superseded by the demand for CSD skills. New CSD employees receive 4-6 weeks of training before specific duties are assigned.
- e. Although the Center has a number of terminals in use which interface with the central site processor, formal training courses have not been scheduled in fundamental techniques of terminal use.
- f. A Univac training coordinator works with the CSD PMO. He furnishes a monthly schedule from Univac listing available courses at the Univac training facility. Individuals from user departments can attend if proper forms are submitted and approved. There is no systematic means of informing production department users of these training courses.

5. Standards, Procedures and Guidelines

- a. Some DMAHTC ADP Standards have been formulated and documented by DAD with close coordination with CSD (reference DMAHTCINST 5660.3, Programming and Production Operations of Computer Applications Software) in an attempt to define Departmental responsibilities in software management. Such instructions are reviewed by DAD (see Section 2, Responsibilities of the Comptroller).
- b. Other DMA standards have been introduced from the DoD (reference DoD Standard 7935.1-S, ADS Documentation Standards).

- c. *There does not appear to be a rigorous method or a central responsibility for ensuring that standards are followed. Some informal methods are being employed, however, adherence to the existing standards is not as good as it could be. Control over open shop software development methodologies is very hard to attain due to the distance from CSD departmental control. Open shop developers have resisted strict compliance to existing standards due to the perceived short-term effect on the productivity pipeline.*

6. Tools and Techniques - Administrative Aids

Closed shop project lifecycle data is entered into the CSD Management Information System (MIS) upon project approval and after development estimates have been made utilizing the ADP Work Request Estimate form. This form has been reproduced in Figure B-2. There are nine categories, referred to as lifecycle milestones, into which development work is classified: Estimate, Development of alternate definitions, Design/revision Integration, Programming, Evaluation, Operation, Non-project estimate, Project overhead, Maintenance and Other. Automated reports documenting actual and scheduled progress in the milestones are produced weekly. These reports are furnished to the PMO Resource Manager.

B. Management's Role within the Center

1. Project support

- a. CSD PMO provides control over the many security facets of ADP operations at HTC. Management of these security functions is delegated to the Security Office at the center.
- b. CSD PMO is responsible for assuring that departmental policies of quality assurance are adhered to by line divisions.

2. Decision Making

- a. PP determines priority and resource level for projects.
- b. If problems develop during project life, PMO personnel initiate corrective action, adjust times and dates, communicate with customers and PP about status, problems, undefined requirements additional needs, etc.

3. Management Tools and Techniques - Project Monitoring and Control

- a. Solicitations for software development are directed to CSD via completion of DMAHTC Form 5600-5, which has been reproduced in Figure B-1.
- b. The ADP Work Request Estimate Form (reproduced in Figure B-2) is used by CSD PMO to provide estimates of man-hours to be expended in phases of the system lifecycle. Information on

the form is used as input to the CSD MIS which introduces a validated workload for the accounting and reporting system.

c. The CSD MIS generates various weekly reports. The following is a partial list of available reports.

1. A report by manager for each projects receiving services. Original estimated due dates, current estimated due dates, estimated hours, hours expended in the reporting period, hours to date, and percent of resources expended are detailed for each activity within major milestone category.
2. Report of systems under development by division and organization. The same type of information is reported as given in Report 1 (above).
3. Job time reported by organization. Lists (by badge number) regular and overtime hours, milestone number, and activity control number.
4. Invalid Job Report lists information which is invalid for one or more of the following reasons:
 - Invalid Control Number
 - Division not scheduled
 - Milestone not scheduled
 - Job is scheduled complete
 - Job not open in CSD MIS
5. Listing of valid and invalid data sorted by control code. Information listed is activity identifier, milestone, scheduled completion date, scheduled man-hours, organization and transaction codes. Invalid data could occur for one of the following reasons:
 - Invalid milestone
 - Invalid Transaction Code
 - Line or block number not in file
 - Line or block number not assigned to CSD
 - Invalid function or organization code
 - Invalid schedule date
 - Missing input card
6. Report of the contents of the CSD MIS Schedule File.

Other reports may be produced by the system, however, documentation was unavailable at the time of the Study.

d. Some large projects are currently using the Program Evaluation and Review Technique (PERT) for scheduling, monitoring and control. Periodic reviews are also used during the lifecycle of these large projects. For example, the DMIS/P system is

currently undergoing revision. For this revision, a Task Work Plan was distributed at a redesign meeting, and target milestone dates were defined. Once interrelationships between tasks were defined, a PERT network was constructed to assist in control of the project.

TO: CSM (PMO)		A D P WORK REQUEST		CUSTOMER CONTROL NO.	
				DATE OF REQUEST	
JOB NAME		DMIS LINE NO.	BLOCK NO.	CSD CONTROL NO.	
REQUESTED BY		ORGANIZATION CODE	EXTENSION	LOCAL CONTROL	
DATE DELIVERY REQUESTED	APPROVED BY			DATE APPROVED	
SPECIFICATIONS Lengthy specifications may be attached; however, a brief description of the work required <i>MUST</i> be shown in this space.					
FOR USE OF COMPUTER SERVICES DEPARTMENT ONLY					
	DIVISION ASSIGNED	COMPLETION DUE DATE	ESTIMATED MANHOURS	ESTIMATED COMPUTER HOURS	GROUP ORG. CODE
CSM	Production Management				
CSS	Technical Support				
CSI	Information Systems & Retrieval				
CSC	Scientific Computing				
CSE	Equipment Operations				
CSO	Techniques Office				
REMARKS					
ADP WORK REQUEST REQUIREMENTS COMPLETED WITHOUT EXCEPTION				SIGNATURE	
DATE COMPLETED		CHECKED BY			

DMATC FORM 5600-S
NOV 78

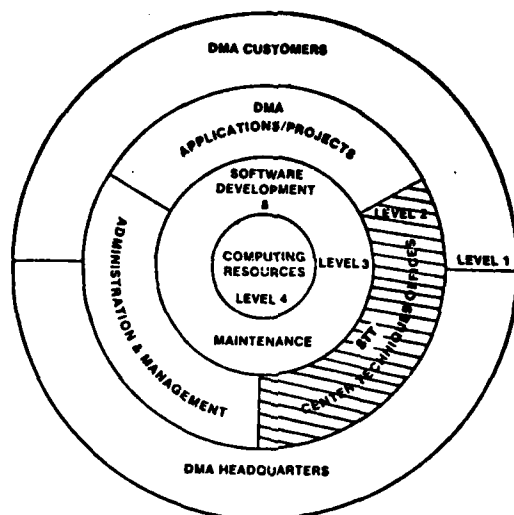
SUPERSEDES DMATC FORM 5600-S, JUN 73, WHICH IS OBSOLETE

Figure B-1 Sample ADP Work Request Form

TO:		ADP WORK REQUEST ESTIMATE		Date of Request	
Job Name		ID No.		GDIP No.	
Requested By		Organization Code		Local Control	
Date Delivery Requested		Approved By		Date Approved	
DCS Point of Contact:		Number of Programmers:			
System Cycle		Cycle M/H	+	Project Overhead	=
				Total M/H	=
				Non Project M/H	-
					Target Date
A0 ESTIMATE A1 Development/Alt Definitions A2 Design/Revision Integration A3 Programming A4 Evaluation A5 Operation A6 Non-Proj. Est. A7 Project Overhead A8 Maintenance A9 Other Totals and Completion Date (Sum of 0-5)					
COMMENTS:					
Estimate Developed By:					
Date Completed		Checked By			

Figure B-2 Sample ADP Work Request Estimate Form

IV. CHARACTERISTICS OF THE CENTER TECHNIQUES OFFICES



- A. The following departments at HTC have Techniques Offices and Production Management Offices (PMOs) which serve staff functions for their departments.

1. Geodesy and Surveys Department
2. Scientific Data Department
3. Computer Services Department
4. Topography Department
5. Hydrographic Department
6. Graphic Arts Department

Also, Techniques Offices and PMOs exist at the division level within some of the larger depts.

- B. The "Techniques Office Concept" is just getting underway at the DMAHTC. Techniques Offices were instituted less than a year ago at HTC, while they have been in existence at DMAAC for a few years. The Techniques Office concept is an attempt to bring together previous decentralized efforts of studying and developing better procedures and methods for overall operations.

- C. The functions of the CSD Techniques Office are:

1. Manage the CSD Techniques Improvement Program (TIP).
2. Prepare or review the preparation of all proposed departmental techniques improvement projects.
3. Conduct or monitor progress of all departmental techniques studies.
4. Coordinate the Department's TIP with other departments, staff offices, and other DMA elements.
5. Conduct analytical studies regarding ADP procedures utilized by the department; prepare implementing plans when studies conclude that implementation of a new procedure will result in improved operations.
6. Administer the CSD effectiveness/productivity program.

7. Develop and staff CSD research and development requirements; monitor external R&D efforts of interest to the department.
 8. Manage the preparation and maintenance of standard operating procedures for all departmental operations.
 9. Accomplish mathematical investigations to develop new generalized computer solutions to problems.
- D. At HTC, CSD Techniques Office proposals for operational improvement are in the following main areas:
1. Operational analysis support addressing improvement of overall CSD programming and mainframe computer operations.
 - a. Quality assurance procedures
 - b. Standard Operating Procedures
 - c. Effectiveness/productivity efforts
 - d. Analysis of the mainframe operational environment
 - e. Analysis of statistical production data dealing with error-off rates and other aspects of qualitative mainframe users performance
 2. Miscellaneous internal CSD support
 - a. Evaluation of off-the-shelf hardware and software
 - b. Preparation of ADP contract documentation
 - c. Database related support (including System 2000)
 - d. Coordinator of CSD's User Assistance Program
 3. Support to advanced weapons systems and other applications - related efforts
 - a. Perspective view
 - b. Pershing II
 - c. Firefinder
 - d. GIST/SACARTS
 - e. Cruise missile
 4. Support to miscellaneous DMA HQ funded R & D ADP efforts requiring project monitorship, on-site contractor support and evaluation/implementation of related efforts.
 - a. MPE Study
 - b. PSL Implementation Contract
 - c. FAVS
 - d. COBOL Automated Verification System (CAVS)
 - e. Monitorship of miscellaneous contractual efforts supporting establishment/operation of the Simulation and Emulation Facility (SAEF) at RADC.
 5. Support to special projects/studies
 - a. Interim computer upgrade
 - b. Phase II replacement computer effort
 - c. FEDSIM Study follow-on
 - d. Pilot Digital Operations Support

All major operational changes which would effect the department coordinated with the Department PMO. Any changes proposed by the CSD Techniques Office which will effect turnaround or production are coordinated with ADP coordinators within user departments through PMO.

- E. STT at center level has overall functional responsibility for the Techniques Improvement Project (TIP) at the center. The Techniques Offices receive the TIP assignments as a result of both internal and external department generated TIPS. The TIP provides the means for test and evaluation of hardware and software acquired through the DMA Research, Development, Test, and Evaluation (RDT&E) Program or via the Investment Procurement Program. It is being used to plan and execute product improvement procedures for existing products material in inventory and to develop production and maintenance procedures for new production.

For example, TIP projects can address improved procedures in connection with the mainframe computers, or improved data handling methodologies. They are generally short-term projects (less than 6 man-months) and have some specific goal which determines project completion.

A request for a TIP, which is referred to as a Process Improvement Proposal (PIP), is prepared/coordinated by the center Techniques Offices and forwarded to STT. STT reviews the PIP and either approves or rejects it. If approved, the PIP is then redesignated as a TIP and is forwarded to PP for manpower/resource allocation.

The Directorate of Programs, Production and Operations (PP) schedules manpower and funding support for TIP assignments and assigns a job number for resource accounting purposes. ST programs resources in the POM to support the TIP and is responsible for monitoring the TIP. TIP assignment is then forwarded to particular department for assignment and execution.

The Techniques Offices (TO) can either play a lead or support role where TIPs are concerned. When the TO's receive manpower authorizations and are responsible for management of the TIP, they play a lead role. When the TO's support the improvement of an existing system, they play a support role. For example, the TO's may support weapons system requirements.

- F. The departmental Techniques Offices interact with each other via a number of methods. Each month the center STT office presents a briefing which describes the status of various Techniques Office TIPS. A representative from each departmental Techniques Office is present at the briefing.

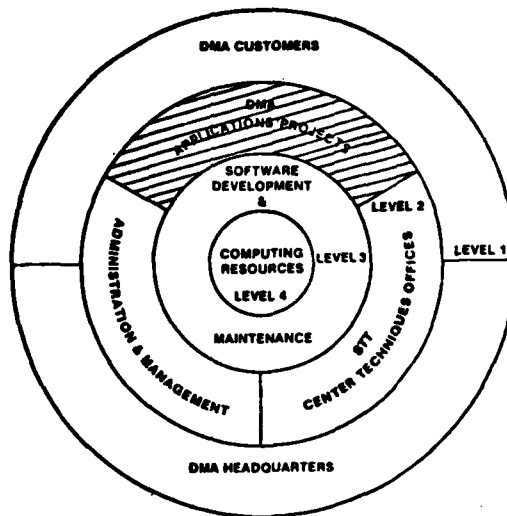
Other briefings on special projects are held periodically and these also serve as a vehicle for communication. Other forms of interaction include technical memos, program reviews and individual contacts.

- G. DMA Headquarters (HQ) Directorate of Systems and Techniques, Advanced Technology Division (STT) manages the Research and Development

(R&D) program for the DMA. The R&D program is broken into technical disciplines (e.g., cartography, photogrammetry, computer science, etc.), and each discipline is the prime responsibility of a HQ STT individual. In addition, each center has an individual within ST which interfaces with HQ STT.

- H. The centers submit an R&D requirements statement for products to HQ STT. STT reviews the statements and either approves modifies or rejects them. If the statements are approved, R&D funding is allocated and the requirements are 'farmed out' to various laboratories (e.g., Rome Air Development Center, Engineering Topographic Laboratory). The laboratories satisfy the requirements through commercial contracts or via an in-house project.

V. CHARACTERISTICS OF THE DMAHTC APPLICATION PROJECT ENVIRONMENT



- A. The users of computer facilities for software development/maintenance at the Hydrographic/Topographic Center fall into two broad categories:
 - Those who use CSD services to satisfy their requirements (closed shop).
 - Those who do not use CSD services, but instead do their own programming (open shop), or obtain programming support from non-CSD sources. Users who perform open shop programming either use the mainframe computer or their own minicomputers.
- B. Regardless of whether open or closed shop programming is performed, all software projects are initiated due to a perceived need. This need is usually postulated in the Program Objective Memorandum (POM) (see section III.), and then becomes the responsibility of a particular department. The user of closed shop programming services participates mostly during the first and last phases of development - requirements definition and test and acceptance (T&A).
- C. CSD usually receives requests for closed shop programming support from the responsible Department through the Directorate of Programs, Production and Operations (PP). These requests are usually in the form of a letter or memo which describes the requirements in a high-level, generalized manner. The next step in the project life cycle depends on the size and complexity of the software development effort.
- D. If the programming effort is very small, requiring changes to a few lines of an existing production program, the user and CSD jointly determine how the change(s) should be made. CSD then reviews the effort and gives an estimate of when the changes can be applied. Once these changes have been applied, CSD turns the project over to the responsible department for Test and Acceptance (T&A). Testing is performed using existing data and if the changed program passes T&A, the production copy of the program is updated.

Efforts which are larger require a more in-depth analysis. Generally, the responsible department performs a functional analysis of the requirements and subsequently writes a Functional Description document. This document is usually written in conformance to the DoD Standard for Documentation of ADP programs (Standard 7935.1-S).

- E. The completed Functional Description is forwarded to CSD, which studies the requirement and generates an analysis of the programming requirements. This programming analysis specifies such items as data base and computational needs. The programming analysis is reviewed by an individual in the user department who has a familiarity with the project or experience with a similar project.

The programming requirements are either accepted or respecified and sent back to CSD. For smaller projects (i.e., 1- or 2-person effort) the acceptance is usually informally given; larger projects may require that an acceptance review meeting take place.

- F. During the design and coding phases the user plays only a minor role, unless the requirements for the software change. Participation in regular reviews (i.e., biweekly, monthly) is not required, and the design methodology is usually transparent to the user.
- G. Most of the users interviewed indicated that requirements were often subject to change during the development cycle. This appeared to be the case due to the state-of-the-art nature of projects. Most users view the development cycle as a learning process and as the user and/or developer becomes better informed, the requirements are subject to change.

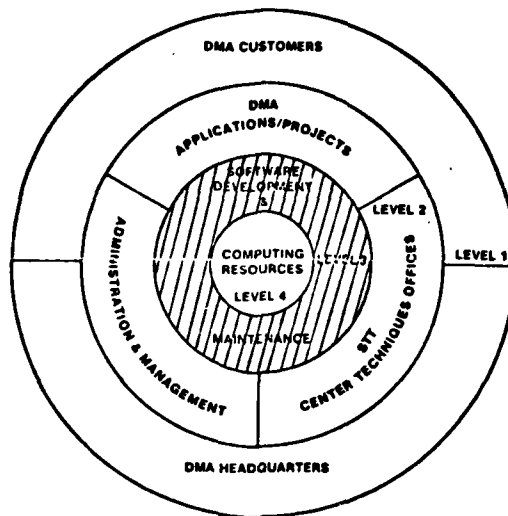
When project requirements change, the user contacts CSD and an informal review of the perceived change occurs. This review may result in revised project estimates.

- H. The divisions which perform open shop programming have a slightly different development cycle, however, their characteristics fall into the same broad categories as discussed.

Open shop programming is done in a research setting and is performed by one or two mathematicians or cartographers who have had a few courses in programming. As such, techniques are not applied with much rigor and the application of standards is dependent on the individual doing the programming.

- I. The user of open shop support plays a minor part during coding, resuming a major role during Test and Acceptance (T&A). The users' Techniques Office is responsible for test planning. The open shop developer sends a letter to its own divisional Production Office stating that the product is ready for T&A. The developer's division Production Office next forwards the letter to the department Production Office which in turn forwards it to the users' divisional Production Office.

VI. CHARACTERISTICS OF THE DMAHTC SOFTWARE DEVELOPMENT AND MAINTENANCE ENVIRONMENT



A. PROJECT TYPES, OUTCOME AND SIZE

1. At both centers (AC and HTC), digital data has become extremely important. Much effort is spent in producing, maintaining and storing digital data.

Outcomes/products of projects are changing from 'paper' to digitized data, e.g., from chart to terminal or digital outputs. The DMA's increased dependence on ADP for production is resulting in much more emphasis being placed on software development/maintenance.

2. Software production/maintenance is becoming much more important. With respect to this, results of the survey indicate that approximately:

55% of effort is spent in adding new capabilities to existing software

23% of effort is spent in developing new software

22% of effort is spent in finding/correcting errors

(These percentages are very close to those at AC. Those were 50%, 30%, 20%, respectively.)

3. Most software development tasks last from 3-6 manmonths in duration. These tasks usually involve a small number of people, 1-3.
4. Most software related activities in the Technical Support Division (TSD) of CSD are directed to maintaining system software integrity.

5. Programming by CSD is referred to as closed shop programming. CSD is normally responsible for programming and contracting for applications software (SW) which is designed to run on the mainframe computers. The following are exceptions to this policy:

- a) Problem-solving, one-time applications.
- b) Unusual circumstances (e.g., short-time requirement for which neither CSD nor user department has resources) when CSD and user will review and recommend action to PP and CSD.
- c) When programming is being done by an organization outside HTC (e.g., Air Force Standard Finance Systems).
- d) When CSD does not have functional knowledge in the application area.
- e) When the application is being developed on non-mainframe equipment.

The user and CSD jointly make a recommendation as to whether the work should be accomplished by open or closed shop. They consider time and complexity of the task and produce manhour estimates. The user specifies the required time frame and the impact of not meeting that time. CSD schedules development and provides impact statements to PP. Final approval of open shop development must come from PP, based upon recommendations from the user and CSD and reviewed by CMD.

6. Upon receipt of a request for SW development or maintenance, CSD searches the SW library to determine whether existing SW will perform the required function. This search is performed within one working day.
7. Application SW is developed using FORTRAN or COBOL. Exceptions to this standard must be approved by CMD.
8. Systems vary in size. DMIS/P reported project size (January reports) by total estimated manhours ranges from 234 to 5367 hours.

B. PROJECT ORGANIZATION

1. Groups

- a) Users participate in requirements specification and in acceptance testing. User participation during project planning (milestone identification), project monitoring (keeping track of project progress, milestone completion), and project review (e.g., design reviews) is minimal.

- b) Management participation is limited in reviews and guidance during the intermediate phases of the project.
- c) Most developers are multifunctional, playing different roles during the software life cycle.

2. User Involvement

- a) The user plays an important role in requirements specification by issuing a memorandum. The outcome of this exercise is usually a document which loosely states the requirements and serves as an informal agreement between CSD and user (the customer).
- b) The user plays an important role in test and acceptance. Testing and acceptance using live data will require user participation in the form of feedback related to the project.
- c) The user often modifies the requirements, as his understanding of the problem and its solution is increased during development.
- d) The user often plays a significant role in resource planning and cost and schedule preparation/estimation.
- e) Formal project reviews are not held, except for large projects. Informal review participation is centered around requirements specification, acceptance testing/product review, and occasionally design review. Larger projects tend to have more interaction between user and CSD than average size projects.

3. Project Manager/Reporting Relationships

- a) More often than not, a project will have an assigned project manager (about 60% of the time). Project managers usually have the necessary authority to carry out their responsibilities.
- b) Some of the closed shop groups are organized under the "group leader" concept, whereby an individual who reports to the division manager may be responsible for several projects. This concept permits management flexibility in moving employees from one group to another, based on workload demand or occurrence of a crisis situation. The concept encourages transfer of on-the-job experience.
- c) The production departments, which are responsible for generating a product, must rely on a support department (CSD) which has many prescheduled tasks with established priorities. Production responsibilities often conflict with established support priorities.

C. PROJECT PLANNING AND INITIATION

1. Source of Requirements

- a) Closed shop projects are initiated when CSD incorporates an accepted ADP Work Request Form into its workload. Initial requirements for open or closed shop development of SW come from the production element to PP. PP issues a Program Assignment Memorandum (PAM) which identifies effected center elements, responsibilities and level of effort.

The PAM is reviewed by CSD and the user element. From this review, an ADP work request is used to determine resource estimates resulting in a formal requirements agreement.

- b) Some requests to CSD groups are informal. A phone call or memo may precede the ADP Work Request form to determine project or timeframe feasibility.

2. User Participation in Requirements Definition

- a) Most closed shop projects promote user and developer participation in an informal work session after CSD receives the ADP Work Request form. The session serves to research resources and calendar time required to satisfy the requirements. Estimates resulting from the research are submitted to the CSD PMO for review. If approved, individuals from CSD are assigned to the project and informal work sessions are used to define requirements in greater detail. During larger projects, both development and maintenance oriented, a review is sometimes held between user and developer which results in a well-defined requirements agreement.
- b) Some CSD software developers commented that interfaces with intelligence agencies may result in requirements or requirements changes for which little advance notice is given.
- c) CSD software developers are not involved early in the development cycle. Major DMA programs (of which software is a component) may be brought in-house by any staff or production element without CSD knowledge. Requirements for software are often vague, giving the impression that the software development process will be a short and uncomplicated activity. When the software development group is consulted in the early stages of major program development, the analysis process surfaces critical background information to define problems leading to solutions.

3. Estimating of Schedules, Costs and Resources

- a) The CSD PMO enters resource estimates and schedules for approved work requests into CSD MIS. These estimates are usually made by an individual in a software development group with similar project experience or estimating expertise. Standardized estimating guidelines or methods (e.g., the use of formulas) are not used. Schedules are often dictated by the user groups (due to production commitments) and conflicts between available CSD resources and schedule commitments often cause open shop development of software.
- b) CSD software project development is broken down into major categories which are referred to as milestones (see Figure B-2, Sample ADP Work Request Estimate Form). At project initiation, manhours planned to be expended in each of these categories are defined and input to the CSD MIS System.
- c) The R&D nature of sophisticated, complex development projects complicates the planning and estimating processes and leads to unexpected project delays and increased cost.

4. Planning of Deliverables and Milestones

- a) These milestones (see 3.b above) correspond to what are often referred to as system lifecycle phases in the software community.
- b) Major documentation subproducts (i.e., *Users' Manual*, *Maintenance Manual*, etc.) are not planned at project inception as deliverables. Resources to be expended in producing these subproducts are not allocated as a separate item.
- c) The management aid EZPERT is sometimes used to support project planning.

5. Project Plan Documentation

- a) Project planning is generally informal in both closed shop (note exceptions in 3.b and 4.a above) and open shop SW development or maintenance activities. Project plans are rarely produced, except for large projects.
- b) Results of the survey indicate that 5-10% of total project effort is spent in planning.

6. Staffing

- a) The need for special skills (e.g., data base experts, assembly language programmers, etc.) is usually recognized during the planning process.

- b) Project staff (open and closed shop) are assigned based upon two factors: experience/qualification for task and availability.

D. SOFTWARE LIFECYCLE

1. The following figure depicts a typical Automated Data Systems (ADS) Development life cycle and responsibilities involved for SW developed within CSD as documented in instruction 5660.3.

A TYPICAL ADS DEVELOPMENT LIFE CYCLE

INITIATION	DEVELOPMENT			EVALUATION	OPERATION	
No. 1 Initiation	No. 2 Definition	No. 3 Design	No. 4 Programming	No. 5 Integration Test Installation	No. 6 Maintenance	No. 7 Revised Operation

- No. 1 - Total responsibility of the using department.
- No. 2 - Responsibility of using department but must coordinate with CSD.
- No. 3 - Responsibility of CSD with close user coordination.
- No. 4 - Responsibility of CSD with user cognizance.
- No. 5 - Responsibility of CSD with user being very active in prescribing test data.
- No. 6 - Normally CSD responsibility with close coordination and definition of required change by the user.
- No. 7 - Responsibility of the user with consulting and troubleshooting services by CSD.

The initiation phase consists of submittal of a project request in the form of a memo, letter or ADP Work Request Form. Definition consists of requirements analysis and detailed specification. Preliminary and detail design activities are not separate activities, but are both performed during Step #3. At the completion of the programming step, a letter or memo is sent to the user department stating that the program(s) is ready for test and evaluation. Integration and test installation generally consists of the user running the program(s) with existing data and manually verifying that tests produce anticipated results. Maintenance consists of activities to revise, upgrade or correct the operational program.

- 2) The life cycle just described also applies when an outside contractor is used to produce the design and programming, however, the responsibilities differ slightly and are listed as follows:

No. 1-Total responsibility of the using department.

No. 2-Responsibility of using department with close coordination with CSD. CSD serves as the contracting officer's representative (COR).

No. 3-Responsibility of contractor with CSD serving as COR and using Department giving close technical review.

No. 4-Same as No. 3.

No. 5-Same as No. 3 but with constant participation by CSD and user. User responsible for prescribing test data.

No. 6-Normally CSD responsibility with close coordination and definition of required change by the user. CSD should consider using contractor support in this area.

No. 7-Responsibility of the user with consulting and troubleshooting services available from CSD and/or contractor.

- 3) Results of the survey indicated that software development activities were divided in the following manner:

●	Requirements Analysis and Design-	8%
●	Planning -	10%
●	Design -	18%
●	Coding -	23%
●	Testing -	21%
●	Maintenance -	20%

- 4) Results of the survey indicated that maintenance activities were divided in the following manner:

●	Requirements Analysis-	26%
●	Redesign -	19%
●	Coding -	33%
●	Retesting -	22%

- 5) The outcome of the requirements specification procedure is usually a document which loosely defines the requirements and acts as an informal agreement with the customer/user.

- 6) Requirements are subject to change during the project life cycle. The most likely reasons for a requirements change are changed user/customer needs and the user/customer becoming better informed.

- 7) Worksessions involving the user and developer are often held during

requirements definition, while formal reviews or walkthroughs are not held, except for large projects.

- 8) Design is usually represented using flowcharts or English-like text. Working sessions are often held during design, while formal walkthroughs and formal reviews of design are seldom held. Design completion is signified by user approval of program design.

Design documentation, if produced, usually consists of a program specification document.

- 9) Coding worksessions are sometimes held, while walkthroughs and formal reviews are rarely held.
- 10) Test plans and acceptance criteria are seldom specified and documented.
- 11) Programs are informally tested at the module, integration and system levels. Test strategies are informal, the major testing efforts are by the development staff and the user (acceptance testing).
- 12) Informal worksessions between the developer and user are often held during testing, however, documentation of test results is seldom produced.

E. PROJECT AND PRODUCT MANAGEMENT

1. Product Control

- a) Lists of users of production programs are not kept.
- b) Change control levels are not employed.
- c) Different versions of production programs sometimes exist, but are not documented or maintained as such.
- d) There does not appear to be any systematic method for controlling changes to production program documentation or distributing change documentation to program users.

2. Personnel Management

- a) Center productivity can be measured by number of maps or digital products per time period. However, there is no similar baseline method of evaluating programmer productivity.
- b) Lack of detailed planning in mechanisms such as the POM exists. As a result, development organizations are often unable to determine future staffing, hardware, and software requirements.

- c) The current ADP job classifications and grades at the center do not provide an incentive for an individual to acquire specialized computer skills.

3. Project Management

- a) Monitoring and control of milestones and software phases.

- (1) The DMA Program Management Information System for Production (DMIS/P) is designed to display total man-year and O&M and Military Pay funding resources. These resources are projected against mission and mission support requirements and programs at levels of detail and frequencies that are consistent with DMA operating concepts and useful in the decision-making processes at successive levels of DMA management. DMIS/P provides the information required to support the following functions:

- i. Establish and maintain policy
- ii. Review policy execution
- iii. Program formulation
- iv. Resources allocation
- v. Program execution review
- vi. Satisfy external information requirements

Reports for CSD projects are produced by the CSD MIS system on a weekly and monthly basis. The CSD MIS is a subset of DMIS/P which accounts for scheduled resources. These reports are available to project branch, division, and department managers. The CSD PMO is responsible for distributing these reports and also uses the reports to ascertain project status. Project managers are responsible for submitting updated MIS information to CSD PMO resource managers.

- (2) Sample CSD MIS reports were obtained and some observations may be made:
 - The manager's report by system contains scheduled completion dates for project tasks. In some cases, the scheduled completion date has passed. It appears that CSD MIS data is not being updated regularly.
 - For some reported systems, only milestone categories A8 and A9, "maintenance" and "other," respectively, (Refer to Figure B-2, ADP Work Request Estimate Form) are frequently or solely used. Activities in other categories have not been identified. Information was not available in order to make a decision of whether the categories are being improperly used, however, it does appear that a more thorough job of estimating could be performed.

- The reported systems range in total estimated man-hours from 234 to 5,367.
 - Some managers have only one system for which they are responsible; the largest reported number of systems per manager is 12.
- (3) Some large projects are employing PERT to assist in planning and control.
 - (4) Costs on a project level (for both computer resources and labor) were not visible and did not appear to play a major part in the management of a project.
 - (5) Instruction 5660.3 defines a typical closed shop ADS Development Cycle and has been discussed in Section D, Software Life Cycle.
 - (6) Some CSD sources indicated that the formality or discipline afforded to project management is determined by the visibility which the project has at the center and HQ.
 - (7) Clearly defined products produced during each phase of the SW life cycle are not identified.
- b) Project Communication and Visibility
- (1) Closed shop reviews conducted by CSD are held after the ADP request is received and after the system specification has been analyzed and defined.
 - (2) Review procedures (e.g., definition of individuals who should be involved, topics to be covered, checklists, etc.) are not documented in either open or closed shop environments.
 - (3) User and customer generated status requests for closed shop projects are channeled through the CSD PMO.
 - (4) Reviews are held more often in the closed shop environment than in the open shop environment. The number of reviews held in either environment is not adequate to maintain project visibility.
- c) Project and Software Development Guidelines and Standards
- (1) Programming guidelines (e.g., commenting standards, naming conventions) are not formally documented. There do not appear to be informal programming guidelines which are widely recognized.

- (2) Standard Operating Procedures exist for the following topics:
 - Job Submission and Processing
 - Univac 1108 Turnaround and Turnaround Reporting
 - Interim Tape System Procedures
 - Restrictions in 1108 Computer Room
 - Harris Terminal usage
 - Uniscope usage
- (3) DMAHTC standard programming languages are FORTRAN and COBOL. Assembly language is used by system programmers.
- (4) Informal programming guidelines are only nominally followed.

F. PROJECT AND PRODUCT DOCUMENTATION

- 1) DOD Standard 7935.1-S defines ADS Documentation Standards to be followed by DMA open and closed shop programming environments. Some programs were documented before the standard was written and, therefore, do not adhere to it.
- 2) Documentation is not generally identified as a deliverable, resources are not allocated at project initiation for document production.
- 3) Documentation usually does not evolve as programs are developed, rather, it is produced at project completion.
- 4) Documentation of production programs is stored and maintained by CSD.
- 5) Documentation of some of the most frequently used production programs was obtained and studied. The documentation reviewed was:
 - MILREF, User's Manual, Preliminary Report dated June 1977
 - CELEST Computer Program for Computing Satellite Orbits, Technical Report, NSWC/DL TR-3565, Oct 1976
 - CELEST Operations Manual, Preliminary DMATC Copy, Feb 1979

From this review, a number of observations were made:

- a) None of the documentation received was a User's Manual, as defined by DOD Standard 7935.1-S. Some of the documentation was in the form of a Final Technical Report, produced by a commercial contractor.
- b) The first document lacked information on job card setup and

data files, an applications programmer would be forced to seek other information sources before attempting a run.

- c) Item 2 was written from a detailed, technical point of view, containing the algorithms used by the CELEST program. It would serve to explain the programmer intent if modifications to the program were being considered.
- d) The third document contains input card formats, a list of files and control card setup. The appendix contains options and input card formats for file handling programs. The document lacks material which relates the input deck to the task of computing satellite orbits. Applications users already familiar with the CELEST program would find the document a good reference for checking the input cards. However, a new CELEST user, having only documents 2 and 3 for reference would find use of CELEST difficult.

6. Survey results indicate how often the following types of project documents are produced for new development projects.*

Project plan	-Often
Functional Requirement Specification	-Seldom
Data Requirements Doc	-Often
System/Subsystem Specification	-Seldom
Program Specification	-Sometimes
Data Base Specification	-Seldom
User's Manual	-Often
Operation Manual	-Seldom
Maintenance Manual	-Seldom
Test Plan	-Seldom
Test Analysis Report	-Sometimes

*Often is over 60% of the time, Sometimes is 40-60% of the time, Seldom is less than 40% of the time.

G. EVALUATION - QUALITY ASSURANCE

- 1. Independent testing of SW is performed by the user departments.
- 2. Attention is paid to the quality of SW produced, but formal documented quality assurance guidelines were not available.
- 3. A little over one-half of the respondents to the survey said that informal QA guidelines do exist. These guidelines applied to and were followed as indicated below.

Requirements specification-	Fairly rigidly
Design specification-	Nominally
Coding-	Nominally
Documentation-	Nominally
Testing-	Fairly rigidly
Maintenance-	Fairly rigidly
Redesign, Recode, and Test-	Nominally

4. The CSD PMO is charged with verifying CSD compliance with QA policies and assuring that adequate reviews occur during development.

H. TOOLS AND TECHNIQUES

- 1) Structured coding (e.g., use of IF-THEN-ELSE, DO-WHILE) and Top-down program development are seldom used. Structran 1 and 2 (also called DMATRAN), tools to assist in structured coding in FORTRAN, are not widely utilized. No information was available to document use of the tools.
- 2) No particular design methodology seems to be followed.
- 3) The contents of project files is not standard across the center. These do not appear to be a working file (e.g., used to document daily decisions and monitor progress), but rather an archival file.
- 4) A limited number of automated (computer based) tools are available for use during software development and maintenance.

In general, those that are available are seldom used (of course, this varies by tool). The predominant reasons for this lack of utilization are:

- limited utility of the tool
 - difficult to use
 - inoperable, or only partially operable.
- 5) The following charts list the tools that have been identified as being available for use at DMAHTC. They tally and summarize (using a weighted average) the part I survey results concerning the utility, cost, and availability of the tools.

	<u>Utility</u>			<u>Cost</u>			<u>Availability</u>		
	<u>H</u>	<u>M</u>	<u>L</u>	<u>H</u>	<u>M</u>	<u>L</u>	<u>H</u>	<u>M</u>	<u>L</u>
SNOOPY (Trace of program instructions)	1	1	1	0	1	0	1	0	0
PMD (Postmortem Dump processor)	8	0	2	0	1	1	7	0	0
Dynamic Dump Routines	1	0	1	1	0	0	2	0	0
FLAP (Flow Analysis Program)	0	0	3	0	0	0	1	0	0
INDEX (cross-reference lister)	1	1	0	1	0	0	2	0	0
University of Maryland File Editor	5	2	0	0	0	1	4	0	0
FILESCAN (cross-reference lister)	0	1	2	0	0	0	2	0	0
FORFLO (generates flowcharts for FORTRAN programs)	2	0	6	1	1	0	3	1	1
TIDY (cleans up FORTRAN programs, renumbers, some editing)	2	2	2	0	1	1	3	1	0
REFORMATTER (reformats COBOL source decks)	1	0	0	0	0	0	0	0	0
STRUCTRAN-1 (converts structured FORTRAN to conventional)	0	0	1	0	0	0	0	0	0
STRUCTRAN-2 (structures unstructured FORTRAN)	0	0	1	0	0	0	0	0	0
FAVS (FORTRAN Automated Verification System)	1	1	2	0	0	0	2	0	0
PSL (Program Support Library)	1	0	0	0	0	0	1	0	0

	<u>Utility</u>	<u>Cost</u>	<u>Availability</u>
SNOOPY	3.0	3.0	5.0
PMD	4.2	2.0	5.0
DDR	3.0	5.0	5.0
FLAP	1.0	-	5.0
INDEX	4.0	5.0	5.0
U of M Ed	4.4	1.0	5.0
FILESCAN	1.6	-	5.0
FORFLOW	2.0	4.0	3.8
TIDY	3.0	2.0	4.5
REFORM	5.0	-	-
S-1	1.0	-	-
S-2	1.0	-	-
FAVS	2.5	-	5.0
PSL	5.0	-	5.0

Key:

for Utility

5 = High utility
3 = Moderate utility
1 = Low utility

for Cost

5 = High cost
3 = Moderate cost
1 = Low cost

for Availability

5 = Readily available
3 = Moderately available
1 = Relatively unavailable

6. The FORTRAN Automated Verification System (FAVS), which incorporates DMATRAN is not currently being used except for evaluation purposes at HTC. The most frequently given reasons for lack of usage are complex input requirements and inadequate developer support and training. Comments were made to the effect that the core requirements for the system are high and that FAVS does not analyze large programs. The utility of FAVS which is reported above does not incorporate data obtained after enhancements were made to the tool in July 1979. The version of FAVS which was utilized on a limited scale prior to these enhancements was inefficient, had several errors, did not have a good user interface and required a large amount of core memory accounting for its poor rating during our center visits. Since this time, four enhancements have been planned, each of which is to be followed by testing at the DMA centers. Telephone conversations with the RADC contract monitor and DMA contacts have revealed that enhancements completed as of September 15th have been almost entirely successful and have served to increase the perceived utility of the tool.
7. Some general purpose reference papers and manuals have been written at DMAHTC or have been acquired from outside sources. Most of these manuals are widely accepted and used. The manuals and papers which were surveyed to have the highest utility are
 - o RUNSTREAM - User introduction to Univac Runstream
 - o SORT - Information about 1108 sorting
8. Poor survey response to questions about tools and reference manuals/papers may indicate a lack of awareness among programmers about what is available.
9. Tools and techniques which have been implemented have not shown claimed benefits.

I. PROJECT SUPPORT

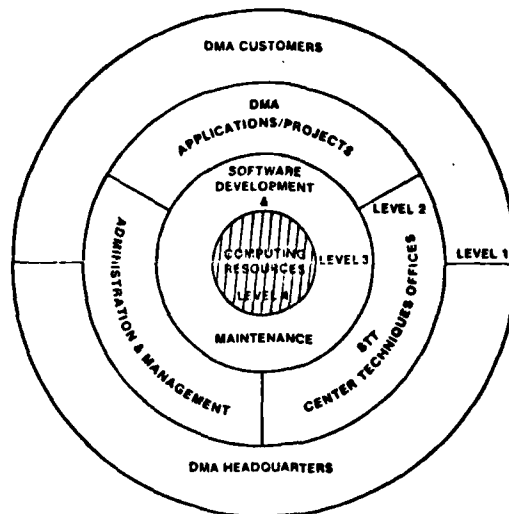
1. The secretarial support for software development and maintenance activities is adequate.
2. A desire for more support from "computer aids" was expressed. Apparently there are tasks being performed by programming staff that do not require a high level of skill and could be performed by non-technical or less skilled personnel.
3. Key punching was specifically mentioned as a task that could be performed by other than the programming staff. Key punching for larger jobs is contracted out.
4. Technical consulting is handled in an informal manner. When a problem is encountered, peer group support in most cases is used to solve the problem or provide the desired information. A desire was expressed for a more formal mechanism.

5. Management support is lacking in the amount of review/involvement. Some felt that there was a lack of understanding of the problems encountered in and the complexity of ADP.

J. PEOPLE

1. The predominant educational background of the CSD staff is either in mathematics, computer science or management science. The predominant educational background of center users is in the physical sciences (e.g., math, geodesy).
2. Some open shop developers are sent to long-term training and also receive on-the-job training. Comments were made that this training was not always directly related to departmental objectives or needs.
3. Formal (long term) training has a built-in disincentive because of pay grade inflexibility for computer specialists. The scale does not provide for promotion opportunities when specialized computer training is acquired.
4. There are some handicapped people who would benefit from special training in SW skills, if it were available to them.
5. Because of a continuing crisis situation under which some CSD individuals work their scheduled participation in a training program is often canceled.

VII. CHARACTERISTICS OF COMPUTING AND COMPUTING SUPPORT RESOURCES



A. HARDWARE CONFIGURATION

1. The hardware used is a distributed and diverse set of equipment. The environments in which software is developed and upgraded are not uniform; there is a mainframe computer and an abundance of minicomputers. The number of minis being employed for production programs is growing at a fast pace and will continue to grow in the future.
2. Long turnaround time on the mainframe machine is having a negative impact on productivity, software quality, and test thoroughness.
3. Most of the users of minicomputers felt that they had good turnaround.
4. The general purpose mainframe computers are:

Univac 1108
Univac 1100/42 (These machines are multiprocessors)

Burroughs 3500 (unit processor)

a) The 1100/42 has:

- Main memory storage
- Drum storage
- Disk storage

Total storage capacity over 1 billion 400 million characters. Also in configuration: 16 tape drives (eleven 9-track, five 7-track), two printers, one punch, two card readers.

b) The 1108 has:

- Main memory storage
- Drum storage
- Disk storage

Total storage capacity close to 1 billion characters. Also in configuration: 12 tape drives (eight 9-track, four 7-track), two printers, one punch, two card readers.

c) The Burroughs 3500 has:

- Main memory
- Disk

Total storage capacity exceeds 660 million characters. There are: seven tape drives (six 9-track, one 7-track), one card reader, one card punch, one printer.

5. Remote input-output devices as of January 1979 used for SW development/maintenance work on the Univac included seven uniscopes (four unclassified) and a Harris Terminal. Output can be sent to either Harris or mainframe computer printers. The Harris RJE station currently handles about one-fourth of CSD workload. CSD also operates a Farrington 3030 Optical Character Reader.
6. There are approximately 40 minicomputers in various departments throughout the center.

B. COMMON JOB ATTRIBUTES

1. Monthly computer accounting reports for the period of January, 1979, were obtained. The following information was summarized from those reports:
 - a) Total number of runs submitted in period: 8,896
 - b) Total cards read: 3,595,871 (approximately 404 per run)
 - c) Total cards punched: 58,860
 - d) Total pages: 835,687
 - e) Total tapes used: 7,690
 - f) Total CPU minutes: 1,361
 - g) Total SUP minutes: 40,788
 - h) Average CPU minutes per run: 1.5
 - i) Average SUP minutes per run: 4.59
 - j) Total jobs having more than one error termination: 1,621
 - k) Total active wall clock time: 135,535 minutes
 - l) Total times FORTRAN compiler called: 10,801
 - m) Total times COBOL compiler called: 81
 - n) Total times assembler called: 282
 - o) Total times MAP processor called: 2,961
 - p) The organization which called the COBOL compiler most was the Inquiries and Support Branch of the Scientific Data Department

- q) The organization which called the FORTRAN compiler most was Techniques Office of the Geodesy Department
 - r) The organization which called the assembler most was Technical Support Division of the Computer Services Department
2. Figure B-3 summarizes computer utilization information for the four HTC organizations which are the biggest users of the central site machines. The organization referred to as "NONE" was a general category for jobs during the period which did not specify an organizational identifier. The large number of jobs which were classified in this category (1,094) indicates that some effort must be made in the future to eliminate uncategorized jobs if adequate statistics are to be kept.

ORGANIZATION 1108 USAGE CHARACTERISTIC	Computer Service Department	Geodesy Department	Topography Department	NONE
	Technical Support Division	Satellite Computation Branch	Digital Products Branch	
Total cards in	316,195	885,141	35,019	369,254
Total cards out	2,838	18,279	930	1,954
Total pages	258,785	61,410	26,030	81,286
Tapes used	1,599	449	813	1,115
Total CPU minutes	702	1,590	2,793	2,639
Total SUP minutes	6,066	3,678	6,316	6,014
Total number of jobs	1,965	861	379	1,094
Jobs with +1 error terminations	217	245	55	257
Total Clock Time	18,248	14,115	12,202	16,131
Average CPU minutes	.2 - .6	1.85	7.3	2.4
Average SUP minutes	1.9 - 5.2	4.27	16.7	5.5
Average tapes used	1	0 - 1	2	1
Average cards submitted	160	1,028	90	340
Average pages generated	130	71	69	75
% jobs with +1 error	11	28	14.5	23.5
% total clock time	13.5	10.4	9	12
% total SUP time	14.9	9	15.5	14.7
% total jobs	22.1	9.7	4.3	12.3
Times COBOL compiler called	1	0	0	8
Times FORTRAN compiler called	860	1,537	187	485
Times assembler called	113	1	0	14
Times MAP processor called	504	423	57	206

Figure B-3 Top Four Organizational Users of Univac 1108

C. USER SUPPORT/CONSULTATION

1. CSD PMO is responsible for the "file management" function (controlling and assisting users in procuring adequate mass storage).
2. CSD PMO is responsible for assuring that the functional users have been adequately trained in the use of ADP applications SW which was developed in CSD under contract. Any center production or staff element may serve as the representative of the contracting officer for applications SW being developed under contract. Technical coordination with CSD during the lifecycle of the project is limited.
3. CSD provides general computing consulting services and expertise for hardware applications and telecommunications.

D. SYSTEMS SUPPORT/SYSTEMS PROGRAMMING

1. The Technical Support Division (TSD) handles systems programming, library functions, and utilities requests for both open and closed shop applications on the mainframe computers.
2. TSD plays a supportive role at the direction of the Techniques Office with commercial SW systems like FAVS and PSL. They are responsible for installation and acceptance testing of such systems. After installation of such systems, TSD is the primary contact when modifications are requested.
3. TSD jobs receive a higher run priority when submitted to the mainframe machines.
4. TSD supports the executive software and associated libraries which may require modification or scrutiny.
5. TSD develops technical specifications required for site planning and equipment installation.
6. TSD reviews computer capacity factors and recommends and justifies specifications for acquisition of new equipment.

E. OPERATIONS/JOB HANDLING

1. CSD's Equipment Operations Division (EOD) manages, operates and controls HTC's three general purpose computers, centralized key-punch capability, electronic accounting machines, microfilm service, Xerox services, tape libraries, staging areas and remote terminals to mainframes.
2. The Univac 1100/42 is used primarily in support of unclassified and collateral classified workloads. The 1108 supports classified and Special Activities Area (SAA) work.
3. Computer operations maintains a three shift, 5-day work week with an average of three shifts of overtime on weekends (if necessary).

4. Checkpoints and restarts are used on large programs only.
5. EOD maintains an active mix of ten or more jobs concurrently.
6. EOD operates a tape maintenance program whereby new tapes are cleaned, certified and released or scratch tapes are cleaned before reentry to the system. There seems to be a tape usage problem causing abnormal terminations. EOD has set up a procedure to control resubmittal of bad tapes by requiring that a label be put on a tape by operations when a tape job abnormally terminates. After three failures operations will reject the tape.
7. The high volume of jobs was cited as making operations/job handling tasks difficult.

F. UTILITIES

A list of system software (utilities, processors and compilers) available on the Univac machines in November, 1978, follows.

- | | |
|--|---|
| 1. EXEC-8 - OS Executive | 19. LIST - Lists elements, files |
| 2. RLIB\$ | 20. MAP - Map collector/processor |
| 3. ACOB - ASCII COBOL | 21. PMD - Postmortem dump routines |
| 4. ALG - ALGOL processor | 22. SECURE - File maintenance program (for backup purposes) |
| 5. ANSI - ANSI COBOL | 23. SORT - Sort routines |
| 6. ASM - Assembler processor | 24. SORT/MERGE - Sort routines |
| 7. BASIC - Univac Basic | 25. SSG - Symbolic stream generator |
| 8. CFCR | 26. TSS |
| 9. COB - Fielddata COBOL | 27. ASGA - Reloads archival files |
| 10. CTS - interactive interface for demand users | 28. CATALOG - lists command options available |
| 11. CULL - Cross reference lister | 29. DISCS - Lists disk information |
| 12. DATA - Enter data processor | 30. EDM - COM Text Editor |
| 13. DOC - Document processor | 31. FLIST - File lister |
| 14. ED - Text Editor | 32. GETFIL |
| 15. ELT - Element processor | 33. MFD |
| 16. FOR - FORTRAN V | 34. SORTF - Assigns sort files |
| 17. FLUSH | 35. SUSPEND/RESUME |
| 18. FURPUR - File and program utility routines | |

Miscellaneous processors and routines currently available are briefly described below.

1. NCRIN - Converts print file for input to Quantor COM recorder
2. SORTSDF - University of Maryland processor for sorting SDF files
3. LIB*PLOT - DMAAC printer plot
4. LIB*NBS - Miscellaneous NBS processors
5. MATH*STAT - Mathematical statistical package
6. COBOL FLOWCHART - produces flowcharts from COBOL programs
7. LIB*TAPE - Tape utility package
8. RINFRO - Retrieve information about a run
9. VTRAN - A variable block size I/O routine
10. LABEL - Creates a skeleton tape label file
11. FORSORT - FORTRAN sort linkage
12. SALVAGE
13. BPL - Breakpoint labeler
14. PATCH - Processor to make corrections to a Level 3 FORTRAN absolute element
15. MAPCH - Translates, maps character sets
16. CPDMPH - Prints, duplicates, compares tapes

G. TOOLS

1. The tools which are available or are currently undergoing implementation, testing or enhancement have been listed in Section H.
2. Both centers identified FAVS, STRUCTRAN-1 and STRUCTRAN-2 (DMATRAN) as having low utility. This evaluation is clarified by a number of observations. Although the centers encourage structured programming, there is no standard requiring use of structured constructs at either center. Study of a limited number of production programs does not indicate that programmers are employing structured constructs. The use of STRUCTRAN-1 or -2 adds another step to already tight production schedules (that of conversion to and from the structured code). STRUCTRAN-1 and -2 have been incorporated into FAVS. It is unclear whether separate versions of STRUCTRAN-1 and -2 are currently maintained at either center. When FAVS was installed at the centers, a number of operating problems resulted. Although these problems were eventually corrected, many comments were made questioning the methods of correction. Evidently the corrections were piecemeal, or "fixes" which were not thoroughly analyzed or applied in a disciplined manner.

The initial version of FAVS requires a large amount of core (documented core requirements were not found in the User's Manual) and substantial execution time when analyzing large programs, thereby causing bottlenecks in the already saturated throughput of jobs. Attempts at modularizing large programs and subsequently analyzing modules with FAVS did not meet with hoped-for success.

FAVS-generated program diagnostics for these modules were inaccurate, causing user disillusionment.

Documentation of FAVS was obtained and reviewed. The advertised capabilities of FAVS are impressive and include; syntax and structural analysis of source programs, static analysis to detect inconsistencies in program structure or in use of variables, automated documentation, source code instrumentation, test coverage analysis, retesting guidance and transformation of unstructured FORTRAN code to structured code.

There is a disparity between the advertised capabilities of the initial version of the tool and its reported utility at the DMA centers. The reported operational characteristics do not measure up to advertised capabilities. Successful operation of FAVS had been marred by incomplete implementation and verification of the tool.

The utility of FAVS, which is described in this report, does not incorporate data obtained after enhancements were made to the tool in July 1979. The version of FAVS, which was utilized on a limited scale prior to these enhancements, was inefficient, had several errors, did not have a good user interface and required a large amount of core memory. Since this time, four enhancements have been planned, each of which is to be followed by testing at the DMA centers. Telephone conversations with the RADC CONTRACT MONITOR AND DMA contracts have revealed that enhancements completed as of September 15 have been almost entirely successful and have served to increase the perceived utility of the tool.

3. IBM's Program Support Library (PSL) was also identified as having low utility by survey participants at both centers. At the AC the tool has not been successfully installed. Installation problems have centered around inconsistencies between the current DMA Univac Executive routines and IBM assumed Univac Executive routines. The tool is currently being evaluated by the DMA and a follow-on contract for implementation, training and maintenance is planned.

Documentation for PSL was obtained and reviewed. It is comprehensive and complete, while being rather lengthy. It contains an extensive description of input requirements and messages generated by the system. The document is easy to follow and understand.

H. TYPES OF SERVICES

1. A color coding procedure is used to identify classified data and jobs and formal, documented procedures for handling classified computer material exist.
2. The DMA HTC Univac computers run separately. The 1108 is used for SAA jobs during prime hours and runs in classified collateral mode during non-prime time hours and when the demand for classified processing is high. When classified jobs are run during prime time subsequent to SAA mode, the 1108 is completely purged. The 1100/42 is used for collateral (unclassified) jobs during non-

prime time hours. Classified collateral processing is run during non-prime time hours with benefit of terminal interface. SAA terminals are available on the 1108 during prime time as are collateral terminals to the 1100/42.

3. 7,065 jobs during January (79% of total jobs) were submitted via card readers (at the Harris RJE and the central site), 1,675 jobs were submitted via terminals or uniscopes.
4. The center will be acquiring additional computer terminals in the near future. No training courses in use of the terminals appear to be planned.
5. The data base technology concept is beginning to be employed at DMAHTC. Data elements are being standardized and a Data Dictionary is being constructed for use in conjunction with the commercial data base System 2000. At the time of interviews at the center, no CSD training in the use of System 2000 was planned.

I. ACCESS

1. Computer access priorities are determined by:
 - batch or demand processing mode
 - project required start time
 - project completion deadlines
2. A scheduling algorithm (called CSLOTS) has been developed by TSD and is currently being used to schedule Univac resources in accordance with HTC program priorities.

J. PROGRAM LIBRARY

1. CSD establishes and maintains the program library, including procedures for recording, indexing, cataloging and controlling ADP applications SW. This library includes SW developed by CSD, open shop, and through contract.
2. The collective holdings of CSD libraries approach 40,000 tapes and 83 disk packs. Approximately 18,000 of these are unclassified (collateral), while about 17,000 (47%) are classified. About 3,000 of the total tapes are used on the Burroughs machine.
3. In December, 1978, CSD was maintaining a catalog of approximately 2,150 applications programs and associated data.
4. There is little documentation of what is available in the program library, who uses which program, or how to control changes to existing programs.

APPENDIX C

Error Off Study Recommendations at HTC

A. Recommendations to Reduce User Errors

1. Keypunches at the Center should have standardized keyboards. Additionally, keyboard characters should correspond to their associated fielddata keypunch code.
2. Users should get an 80-80 listing of card decks to aid in desk checking before processing jobs.
3. All users should be educated in keypunch operations and UNIVAC's Executive Control Language.
4. Users should develop better record keeping systems for tapes which include tape density, track size (7 vs. 9), data stored, format, etc.
5. Operational programs should be written to terminate normally.
6. Users should be provided with better documentation.
7. Canned runstreams should be used whenever possible.
8. Departments should periodically evaluate their quality control techniques.
9. Notify divisions that monthly accounting reports are available for their review.

B. Recommendations to Reduce Tape Errors

1. The recommendations contained in the MRF, 13 June 1978, subject: Tape Error Analysis, (reproduced later in this Appendix) should be implemented.
2. Users should be educated in proper methods of handling and transporting tapes.
3. The tape library should have a better method for keeping a tape's history including:
 - a. Date entered system
 - b. Certified density
 - c. Track size (7 vs. 9)
 - d. Date cleaned
4. A group should be formed to further investigate the Centers tape related problems and the most effective ways to solve them.

C. Recommendations to Reduce Operating Procedure Errors

1. Operators should make use of the checkpoint/restart Exec system feature when converting to classified mode.
2. More extensive use should be made of 'read only' labels on tapes when applicable.
3. Operators should be more descriptive in their messages explaining why a job has been killed.
4. Review the staging area's method of keeping track of tapes sent to and returning from the computer room.
5. Operators should have complete and up-to-date descriptions of special operating procedures for large production systems (i.e., what jobs can be run together, what to do if one of a number of sequential jobs errors, etc.).

D. Recommendations to Improve the Definition of Operational Runs

1. Users should be provided with a more detailed description of what jobs should be classified as operational runs.
2. Operating system support jobs should not be classified as operational.

1. MFR: TAPE ERROR ANALYSIS

An analysis of tape errors was performed on the site 1 1100/42 during the period 15-26 May. Although the analysis was not as thorough as anticipated, the results are felt to be a reasonable indication of the causes of unrecoverable tape errors.

The following are the inspection results from the 40 tapes which encountered tape errors:

--CAUSE--	--QUANTITY--
1) Scratched	18 tapes or 45%
2) Crimped	10 tapes or 25%
3) Tape defect	6 tapes or 15%
4) 9-track tape on 7-track drive	2 tapes or 5%
5) Even parity tape being read as odd	2 tapes or 5%
6) Nothing detected	2 tapes or 5%

NOTES:

- A) Sample category 3) items include missing oxide & pit marks.
- B) Categories 4) and 5) are noted as programmer errors.
- C) One additional programmer problem was detected: digital jobs that failed to finish writing on a tape would then rewind, read it, and get a runaway. (This problem will soon be resolved.)
- D) In one instances there were problems with digital runs because all the tape drives had leaders that were incorrectly threaded.

The results of the test have provided a basis for the following recommendations:

- 1) The tape drives should be cleaned a minimum of once every 8 hours (also recommended by Univac). This, in conjunction with the periodic cleaning of tapes and the cleaning of new tapes should minimize the wear of our tapes. Scratched tapes are felt to be caused by dirty recording heads on the tape drives.

REQUIRED: EOD operations training.
EOD manpower designated for tape maintenance.

- 2) EOD personnel need formal training by Univac on proper handling of tapes, including the mounting of tapes. This should resolve some of the problems with crimped tapes.

Also, it is noted that operators now place a card on users' decks if the cards have been inadvertently reshuffled. A new card, 'NOTE - your tape has been dropped and may be damaged' seems a logical addition.

REQUIRED: EOD operations training.

- 3) The results gained from the inspection of tapes when an unrecoverable error occurs are appreciable, especially from the user's viewpoint. Currently, users

can only guess at the cause. When a second run with the same tape succeeds they mistakenly attribute the problem to marginal hardware.

It is to the users' benefit if the procedures used during the 2-week test are implemented on a permanent basis. The benefits are:

- A. Users learn the cause of tape problems quicker, eliminating re-runs.
 - B. Operators learn of dirty tape heads quicker, allowing subsequent users to successfully use the drive.
- 4) The number of old tapes being used at DMATC is judged to be excessive, especially in view of the new, higher density tape drives that were recently installed. Some of the tapes inspected during this test were certified at 556 and 800 BPI while the new drives have 1600 BPI capability. Unfortunately, the user has no easy way to judge the age or quality of the tapes he uses.

An ongoing survey of our tape library should be begun in conjunction with additional record-keeping of new tapes. The certification density should be kept in the tape library system and included in the listing each user receives. In short, the tape library system needs to be updated to better meet the user's needs. Its present reference to 1/2-inch and 3/4-inch tapes indicates its lack of relevance to the user.

REQUIRED: EOD manpower designated to survey the tape library.
ISR manpower designated to update the tape library system.

- 5) An investigation should be made to assess the minimum quality of tape required for our operation and how this compares with the quality of GSA tapes we now receive.

APPENDIX D

DMA Survey - Parts 1 and 2

NAME _____ (optional)
DEPARTMENT _____

DMA SURVEY--Part I

Answer each question as it pertains to software which is developed in your department. When you don't feel qualified to respond to a question, please indicate this in the space left for comments. The comment space should also be used for any additional information that you feel is pertinent to the question.

RESPONDENT AND JOB CHARACTERISTICS

1. From what point(s) of view are you responding to this survey?
____ senior analyst
____ functional analyst (cartographer/mathematician)
____ programmer/analyst
____ user
____ project manager
____ technical manager (or branch chief)
____ Other: _____

Comments _____

2. Typical turnaround time for batch jobs submitted to the mainframe computer is _____ hours.

3. Fill in the blanks with characteristics which apply to a typical software product developed in your department.

- a. Require about _____ K words of central memory
b. Execute in approximately _____ CPU seconds
c. Require _____ (number of) secondary storage devices (i.e., magnetic tapes, disk, drum)
d. Contain _____ executable lines of code
e. Documentation produced:
- | | Always | Often | Seldom | Never |
|----------------------------|--------|-------|--------|-------|
| Project Plan | _____ | _____ | _____ | _____ |
| Functional rqmts. desc. | _____ | _____ | _____ | _____ |
| Data requirements document | _____ | _____ | _____ | _____ |
| System/subsystem spec. | _____ | _____ | _____ | _____ |
| Program specification | _____ | _____ | _____ | _____ |
| Data base specification | _____ | _____ | _____ | _____ |
| User's manual | _____ | _____ | _____ | _____ |
| Computer Operations Manual | _____ | _____ | _____ | _____ |
| Program Maintenance Manual | _____ | _____ | _____ | _____ |
| Test Plan | _____ | _____ | _____ | _____ |
| Test Analysis Report | _____ | _____ | _____ | _____ |

Comments _____

4. Software is generally developed
____ for a one-time application by a single user
____ for application by a few users
____ for application by many users (production software)
____ for application by few users, yet in reality is used by many

Comments _____

5. Are jobs submitted in stages (e.g., multi-part submittal) where results of one stage are necessary for the next stage?

____ Usually ____ Sometimes ____ Seldom ____ Never

Comments _____

PROJECT MANAGEMENT AND SUPPORT

6. Does each project have a designated project manager or team leader with responsibility for project completion?

____ Always ____ Often ____ Seldom ____ Never

- a. If there is a project manager or team leader with project completion responsibility does he also have the authority necessary to get the job done?

____ Always ____ Often ____ Seldom ____ Never

Comments _____

7. How much of the activities performed by programmers/analysts in software development could be performed by competent secretarial or clerical personnel if they were available (e.g., documentation)?

____ None; adequate support is being received
____ Less than 20%
____ 20% - 50%
____ Over 50%

Comments _____

8. How much of the activities performed by programmer/analysts in software development could be performed by a computer aid if one were available (e.g., keypunching, data entry)?

____ None; adequate support is being received
____ Less than 20%
____ 20% to 50%
____ Over 50%

Comments _____

9. When problems or questions arise during the software development process, is adequate consulting support available?

____ Always ____ Usually ____ Seldom ____ Never

Comments _____

SOFTWARE LIFE CYCLE

10. Estimate the percent of effort spent during the following phases of software development:

____ Specification and formalization of requirements
____ Project Planning
____ Design of the software
____ Coding of the software
____ Testing (by developers)
____ Maintenance of the software

- a. During program maintenance estimate the percent of effort spent in each of the following:

____ Analysis and respecification of requirements
____ Redesign
____ Coding
____ Retesting (by developers)

Comments _____

11. Check those items which are likely to serve as a measurement of project completion and rank them by likelihood (1 = Most Likely, 6 = Least Likely).

____ Completion of software documentation --Rank _____
____ Completion of coding --Rank _____
____ Completion of debug/checkout --Rank _____
____ Satisfaction of acceptance tests --Rank _____
____ Consumption of project monies --Rank _____
____ Reaching project completion date --Rank _____
____ OTHER: _____ --Rank _____

Comments _____

12. When is software documentation generally produced:

____ As the software is being developed
____ After the software has been developed
____ Only when required by the project plan

Comments _____

13. During the software planning process are intermediate goals (project milestones) defined? ____ YES ____ NO

a. If YES, are these milestones used to monitor project progress?
____ Always ____ Often ____ Seldom ____ Never

Comments _____

14. Formal (documented) procedures to be followed during requirements specification

____ do not exist
____ exist and are followed
____ exist but are not followed

Comments _____

15. The outcome of the requirements definition effort is:

	Always	Often	Seldom	Never
a. a formal, documented and approved requirements specification	_____	_____	_____	_____
b. an informal agreement with the user/customer	_____	_____	_____	_____
c. a loosely defined set of requirements which is subject to change during project development	_____	_____	_____	_____
d. Other: _____	_____	_____	_____	_____

Comments _____

16. Check those items which are likely reasons for rewrite or change of the software requirements during the development effort and rank them by likelihood (e.g., 1 = Most Likely, 7 = Least Likely).

_____ specification errors	--Rank _____
_____ ambiguous, incomplete or inconsistent specifications	--Rank _____
_____ change in project scope	--Rank _____
_____ change in project requirements	--Rank _____
_____ change in project hardware	--Rank _____
_____ normal way of doing business	--Rank _____
_____ user and/or developer become better informed	--Rank _____
_____ Other: _____	

Comments _____

17. The design effort is complete when:

	Always	Often	Seldom	Never
a. assurance is given that all re-requirements have been addressed	_____	_____	_____	_____
b. the scheduled due date for design completion is reached	_____	_____	_____	_____
c. the next-lower level of design would result in implementation decisions	_____	_____	_____	_____
d. the user has reviewed and approved the design	_____	_____	_____	_____

Comments _____

18. Check the statements that apply to resource planning:

	Always	Often	Seldom	Never
a. Guidelines used in plan preparation	_____	_____	_____	_____
b. Customer/user participation	_____	_____	_____	_____
c. Use of Work Breakdown Structure	_____	_____	_____	_____
d. Resources allocated by project phase	_____	_____	_____	_____
e. No formal planning (guesstimates)	_____	_____	_____	_____

Comments _____

19. Is the need for special skills/expertise recognized and addressed in the project plan?

_____ Always _____ Often _____ Seldom _____ Never

Comments _____

20. Check the statements that apply to the way cost and schedule estimates are derived:

	Always	Often	Seldom	Never
a. via formula(s)	_____	_____	_____	_____
b. via comparison to similar projects	_____	_____	_____	_____
c. via individuals with estimating expertise	_____	_____	_____	_____
d. cost and schedule are dictated by user	_____	_____	_____	_____
e. simulation	_____	_____	_____	_____
f. no formal estimation (guesstimates)	_____	_____	_____	_____
g. Other: _____	_____	_____	_____	_____

Comments _____

SOFTWARE TESTING, MONITORING AND CONTROL

21. Do any documented guidelines for software/documentation change control exist? _____ YES _____ NO

a. Are there formal procedures for implementing these guidelines? _____ YES _____ NO

b. Are these guidelines and/or procedures followed for software _____ Always _____ Most of the time _____ Seldom _____ Never
for documentation

_____ Always _____ Most of the time _____ Seldom _____ Never

Comments _____

22. Do quality assurance procedures or guidelines exist?

_____ YES _____ NO

a. Check the activities to which they apply and the degree to which they are followed:

	Rigidly	Nominally	Not at all
_____ Requirements specification	_____	_____	_____
_____ Design specification	_____	_____	_____
_____ Coding	_____	_____	_____
_____ Documentation	_____	_____	_____
_____ Testing	_____	_____	_____
_____ Maintenance	_____	_____	_____
_____ Redesign, code, retest, etc.	_____	_____	_____

Comments _____

23. Does completed software undergo quality assurance testing by a group which is independent of the development team?

_____ Always _____ Most of the time _____ Sometimes _____ Never

Comments _____

24. Are formal test plans/test strategies developed and documented?

_____ Always _____ Most of the time _____ Seldom _____ Never

a. Do these include testing to insure that all the requirements have been met?

_____ Always _____ Most of the time _____ Seldom _____ Never

Comments _____

25. At which levels do software testing and evaluation occur?

_____ Module level

_____ At module integration (integration testing)

_____ At system testing (acceptance testing)

_____ Other: _____

Comments _____

26. Are informal team work sessions held:

Always Often Seldom Never

During software specification _____

During software design _____

During coding _____

During testing _____

Comments _____

27. Are formal walkthroughs (attended by management, prepared for in advance, etc.) held:

Always Often Seldom Never

During software specification _____

During software design _____

During coding _____

Comments _____

28. Which of the following formal reviews are held during software development:

Always Often Seldom Never

Feasibility Study Review _____

Functional Description Review
(Requirements Review) _____

Program Specification Review
(Preliminary Design Review) _____

Detailed Design Review _____

Software Code Review _____

Test Results Review _____

OTHER: _____

Comments _____

29. Indicate the most appropriate answer in regard to formal reviews:

Always Often Seldom Never

a. Advance notice of review given _____

b. Take place on schedule _____

c. Attendance by management _____

d. Is a project review plan prepared _____

e. Attendance by user/customer _____

f. Independent review team _____

Comments _____

30. Describe the reporting/communication mechanisms commonly used within projects
- | Report/Memo/Other | Formal/
Informal | Frequency* | From** | To** |
|-------------------|---------------------|------------|--------|------|
|-------------------|---------------------|------------|--------|------|

Weekly/monthly activity report

Project/milestone progress report

Technical memos

Minutes of Meetings

*weekly, monthly, as requested, etc.

**the responsible party for preparing the report, and the party to whom it is sent.

Comments _____

31. Are automated tools used to monitor project schedule and costs?

_____ Always _____ Often _____ Seldom _____ Never

Comments _____

TOOLS AND TECHNIQUES

32. Check the techniques which are addressed by documented standards or guidelines and indicate the degree to which they are followed.

	Always	Often	Seldom	Never
_____ a. Structured coding (e.g., use of IF-THEN-ELSE, DO-WHILE)	_____	_____	_____	_____
_____ b. Top-down design	_____	_____	_____	_____
_____ c. Top-down coding	_____	_____	_____	_____
_____ d. Naming conventions	_____	_____	_____	_____
_____ e. Module internal documentation (e.g., comments)	_____	_____	_____	_____
_____ f. Data formatting conventions	_____	_____	_____	_____
_____ g. File formatting conventions	_____	_____	_____	_____

Comments _____

33. Indicate the techniques which are used for design representation:

	Always	Often	Seldom	Never
a. Flow charts	_____	_____	_____	_____
b. Program Design Language (PDL)	_____	_____	_____	_____
c. HIERARCHY plus INPUT-PROCESS-OUTPUT(HIPO)	_____	_____	_____	_____
d. Other: _____	_____	_____	_____	_____

Comments _____

34. From the list below check the tools which are used during software development and indicate your opinion of their usefulness, cost and availability (H = high, M = medium, L = low)

USE	Utility	Cost	Availability
☐ SNOOPY (Trace of program instructions)	☐	☐	☐
☐ PMD (Postmortem Dump processor)	☐	☐	☐
☐ Dynamic Dump Routines	☐	☐	☐
☐ FLAP (Flow Analysis Program)	☐	☐	☐
☐ INDEX (cross-reference lister)	☐	☐	☐
☐ University of Maryland File Editor	☐	☐	☐
☐ FILESCAN (cross-reference lister)	☐	☐	☐
☐ FORFLO (generates flowcharts for FORTRAN programs)	☐	☐	☐
☐ TIDY (cleans up FORTRAN programs, rennumbers, some editing)	☐	☐	☐
☐ REFORMATTER (reformats COBOL source decks)	☐	☐	☐
☐ STRUCTRAN-1 (converts structured FORTRAN to conventional)	☐	☐	☐
☐ STRUCTRAN-2 (structures unstructured FORTRAN)	☐	☐	☐
☐ FAVS (FORTRAN Automated Verification System)	☐	☐	☐
☐ PSL (Program Support Library)	☐	☐	☐
☐ Other: _____	☐	☐	☐
☐ Other: _____	☐	☐	☐
Comments _____			

35. In your opinion, are the tools and techniques named in question 34:
- ☐ adequately supported (e.g., consultation)
- ☐ inadequately supported
- ☐ adequately documented
- ☐ inadequately documented

Comments _____

36. List the five operating system utilities (i.e., DOWNDATER, PFCK, UNDO, TDUMP, SORTSDF, FLIST, LABEL, VTRAN, etc.) most frequently used in your department.

Comments _____

37. From the following list, check the manuals which are used and indicate your opinion of their usefulness, cost and availability (H = high, M = medium, L = low)

USE	Utility	Cost	Availability
☐ COBOL Debug Manual	☐	☐	☐
☐ FTN-PAPER (ASCII FORTRAN Guide)	☐	☐	☐
☐ RUNSTREAM (User Intro. to Runstream)	☐	☐	☐
☐ GOULD (supplement to GOULD programming manual)	☐	☐	☐
☐ PLOT (SCD Plotter Programmer Manual)	☐	☐	☐
☐ PRGMAN (SCD Plotter Programmer Manual)	☐	☐	☐
☐ SORT (Information about 1108 Sorting)	☐	☐	☐
☐ Other: _____	☐	☐	☐
☐ Other: _____	☐	☐	☐
Comments _____			

38. How frequently do you use TIP (Technical Information for Programmers)?

☐ Always ☐ Often ☐ Seldom ☐ Never

Comments _____

SOFTWARE DEVELOPMENT OBJECTIVES

	Always	Often	Seldom	Never
39. Do you develop software for varied hardware configurations?	☐	☐	☐	☐
40. Do you encounter software transfer problems?	☐	☐	☐	☐
41. Do you design software to be portable?	☐	☐	☐	☐

Comments _____

42. Software developed for application by many users is generally
☐ easy to maintain, upgrade, extend and adapt
☐ difficult to maintain, upgrade, extend and adapt

Comments _____

43. Describe the frequency of errors discovered in operational software:

☐ Very low ☐ Low ☐ Moderately low
☐ Very high ☐ High ☐ Moderately high

Comments _____

44. Estimate the percent of total effort devoted to
☐ Adding new capabilities to existing programs
☐ Starting from scratch to produce a new program
☐ Detection/correction of errors in existing production programs

Comments _____

45. Indicate whether the following statements are true or false as they relate to FAVS, PSL, STRUCTRAN-1 and STRUCTRAN-2, by circling T or F.

	<u>FAVS</u>	<u>PSL</u>	<u>STR-1</u>	<u>STR-2</u>
a. comprehensive in-house (DMA) training program	T F	T F	T F	T F
b. provide substantial benefits to the user	T F	T F	T F	T F
c. complex input requirements	T F	T F	T F	T F
d. output is clear, concise and informative	T F	T F	T F	T F
e. supported by developer	T F	T F	T F	T F
f. adequately documented	T F	T F	T F	T F
g. available for use at DMA	T F	T F	T F	T F
h. I am unaware of or inadequately informed on the tool	T F	T F	T F	T F

Comments _____

46. Are project notebooks originated and maintained for each project?
_____ Always _____ Often _____ Seldom _____ Never

a. If project notebooks are used, check the information which is generally maintained in the notebook:

- _____ Requirements Specification
- _____ Design Overview
- _____ Module Interface Description
- _____ Data Flow Diagrams
- _____ Assumptions and Constraints
- _____ Module Code Listings
- _____ Test Case Descriptions
- _____ Test Case Results
- _____ Support Library Control
- _____ Discrepancy Report File
- _____ Minutes of Reviews

Comments _____

NAME _____
DEPARTMENT _____

DMA SURVEY--Part 2

PROJECT PLANNING

- 1) Comment on the adequacy of project planning (e.g., milestone identification, schedule development, cost and resource estimation, including people) and the contribution of planning to the success/failure of DMA software products.

- 2) Identify specific areas for improvement and possible mechanisms for facilitating these improvements.

PHASES & OUTCOMES OF PHASES

- 3) Do you feel that emphasis should be redirected to the earlier phases of software development (e.g., less time during coding, more time for requirements design & test specification, etc.)? In what way would this shift in emphasis address existing problems? (Specify the problems)

- 4) Comment on testing and validation of software (e.g., satisfaction of requirements, consistency between requirements & design, module integration & system testing).

USER INVOLVEMENT

- 5) Discuss the end user's involvement in the software development effort (e.g., requirements analysis, design validation, review participation). Is this adequate? How should it be improved?

PROJECT ORGANIZATION/MANAGEMENT

- 6) If you could change one thing about the way that projects are organized and/or managed (e.g., project teams, leadership, etc.) what would you do? How would this improve productivity? Product quality?

REVIEWS/REPORTING

- 7) Comment on the overall productivity during reviews (e.g., are participants prepared, are pertinent discussions held, are potential problems addressed, etc.).
- 8) Do you think that more or fewer reviews should be held during software development? What are the potential effects of conducting more or fewer reviews on product reliability, maintainability, usability and user satisfaction?
- 9) Do you feel that you make less progress than you might because you are burdened with producing progress reports or participating in reviews? If so, what change should be made (e.g. verbal vs. written progress reports, less frequent reports, shorter reviews/reports)?

MAINTENANCE

- 10) In your environment is program maintenance hindered by lack of adequate documentation (internal program comments, requirements/design specifications, user/operation/maintenance manuals)?

CHANGE CONTROL

- 11) Do you experience problems during documentation or software development due to current change control procedures or mechanisms (e.g. control of changes to existing programs or documentation)?

STANDARDS/PROCEDURES

- 12) Comment on the relationship between guidelines and programming productivity and product quality. Take into consideration the use of guidelines during all development phases (e.g., requirements specification, design, testing, etc. as well as coding).

TECHNIQUES/TOOLS

- 13) What do you feel is the contribution of programming techniques (e.g., structured programming, top-down development) to programming productivity and software quality?
- 14) What do you feel is the contribution of tools (e.g., FAVS, PSL, STRUCTRAN-2) to programming productivity and software quality?
- 15) Do you have any suggestions for acquisition of tools/application of techniques? In what area would you most like to see research and development of tools & techniques initiated?

PROJECT SUPPORT

- 16) Comment on the various project support activities/roles (i.e., adequacy of) & their effect upon project completion/failure:
Secretarial

Computer Aid (e.g. key punching, data entry)

Technical Consulting (e.g. tool support, programming problems, systems programming)

Computer Facilities (e.g. turn around time, machine access)

Computer Operations (e.g. job scheduling, job handling)

Administrative (e.g. contracts, finance)

Management (e.g. planning, reviews, decision-making)

QUALITY ASSURANCE

- 17) Do you feel that adequate attention is paid to product quality within your environment? Do adequate guidelines exist for all development activities? Are products sufficiently reviewed and tested?
- 18) Which qualities of DMA software are most in need of improvement (e.g. portability, modifiability, understandability)?

PRODUCTIVITY

- 19) At what percent of your total capacity do you feel you are presently working? Comment on the productivity level of your fellow workers.
- 20) How do you feel that your productivity could be improved (e.g. change in working environment, training programs, etc)?

HUMAN FACTORS

- 21) Are you ever asked to perform tasks in the area of software development for which you feel unqualified (e.g. requirements specification, design, coding, testing, documentation)?
- 22) Is adequate attention paid to the ease of use/application of procedures, techniques and tools at the DMA? Give examples and suggested improvements.
- 23) What do you feel are the most important objectives to be achieved during software development (e.g. within budget, on schedule, extensive user involvement, easy to maintain, etc)? Can you identify mechanisms that will assist in achieving these objectives?

24) Indicate whether the following issues are problems in your environment and suggest ways that the most critical problems could be alleviated:

- a. project completion at planned cost
- b. project completion within planned schedule
- c. software maintainability
- d. software reliability
- e. software quality
- f. programming productivity
- g. project management
- h. documentation
- i. standards/procedures/guidelines
- j. other

25) General comments on this survey. (Part I and/or Part II).

Appendix E

Outline of Recommendation Content

- I. Communication and Visibility
 - A. Formally planned and conducted lifecycle reviews
 - B. Regular project status and progress reporting
 - C. Early developer visibility into required project support
- II. Product Quality Assurance (QA)
 - A. Define essential DMA Software product qualities
 - B. Define roles and responsibilities for QA
 - C. Conduct QA Reviews
 - D. Conduct QA Audits
 - E. Define acceptance criteria
 - F. Systematize discrepancy reporting
- III. Facilities
 - A. Define and implement program optimization plan
 - B. Implement Error-Off Study Recommendations
 - C. Investigate Harris RJE at AC
 - D. Continue transition from batch to demand mode
 - E. Investigate I/A Capability
- IV. Product Management and Control
 - A. Establish configuration control function
 - B. Define and control configuration items
 - C. Employ change control levels
 - D. Employ other change control mechanisms
 - 1. Configuration Change Request
 - 2. Impact Analysis

- 3. Problem Reports
 - E. Establish Change Control Board
 - F. Establish a Program Librarian
 - G. Evaluate existing tools
 - H. Acquire or develop DMA Specific Tools
- V. Software Engineering Guidelines and Standard Practice
 - A. Define standard programming practices
 - B. Standardize phased development
 - C. Employ Software Notebook
 - D. Develop Standards Handbook
 - E. Develop Software Policy Handbook
- VI. Tools and Software Development Aids
 - A. Establish and Maintain inventory of tools
 - B. Define requirements for new tools
 - C. Identify new tools
 - D. Evaluate new tools
 - E. Coordinate introduction of new tools
 - F. Periodically evaluate DMA tools
 - G. FAVS
 - 1. Establish test group
 - 2. Define test plan
 - 3. Define acceptance criteria
 - 4. Evaluate NBS FORTRAN ANALYZER, MNF Compiler
 - H. PSL-Implement using Study guidelines
 - I. STRUCTRAN-1 and STRUCTRAN-2 (DMATRAN)
 - 1. Encourage use of structured concepts

2. Implement structured concepts through comments
3. Develop or acquire tools for use in early phases of software development

VII. Technology Transfer

- A. Sponsor introductory software engineering seminar
- B. Sponsor series on software engineering
- C. Technology Transfer techniques
 1. Training
 2. CAI, VAI
 3. Commercially-sponsored conferences/Seminars
 4. Undergraduate and graduate education
 5. Technical lecture series
 6. On-the-job training
 7. Colocation of technical staff
 8. Workshops
 9. Information Exchange Meetings/Briefings
 10. Technical consultation
 11. Technical Information for Programmers (TIPS)
 12. DMA Technology Newsletter
- D. Tools & Techniques Introduction Plan

VIII. Training

- A. Define departmental training responsibility
- B. Evaluate currently available courses (within centers)
- C. Evaluate commercially-available courses
- D. Conduct courses
 1. Requirements definition and analyses
 2. Planning, scheduling & estimating

3. Test and evaluation
4. Programming practices and guidelines
5. Project Management/Development techniques

General Survey Information

1. DMAAC Survey

Conducted: 2-13-79 and 2-14-79
Survey sample size 28

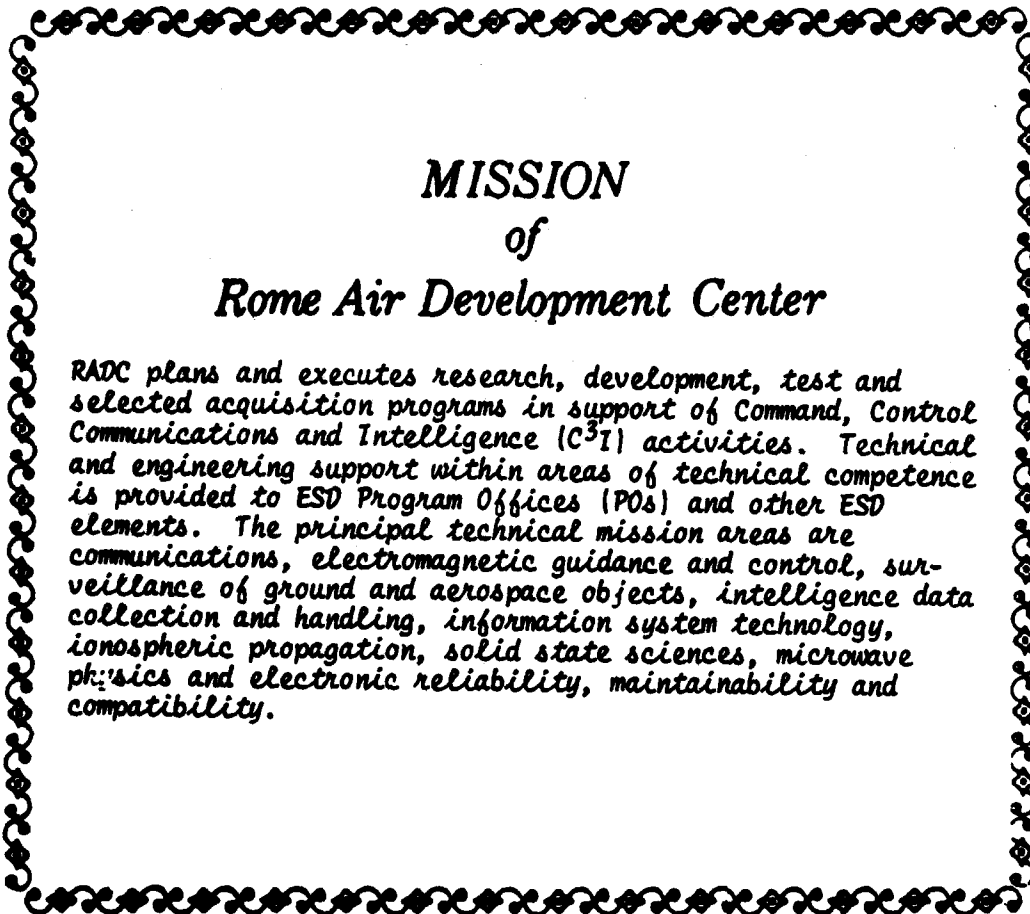
2. DMAHTC Survey

Conducted: 2-22-79 and 2-23-79

Some surveys were returned to BCS by mail due to unavailability of participant at the time of the survey.

Survey Sample Size 15

Sample size was small due to coincidence of BCS trip with severe snowstorm which necessitated closure of government offices. Phone calls after survey conduct supplemented original data.



MISSION of Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C³I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.