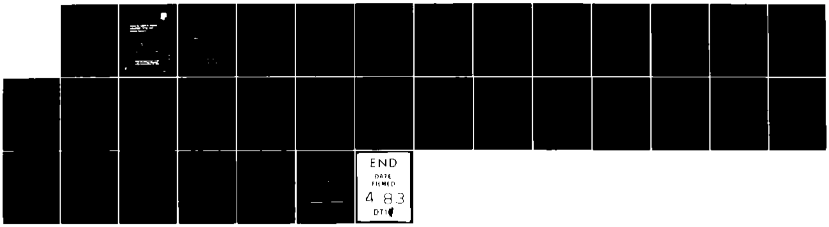
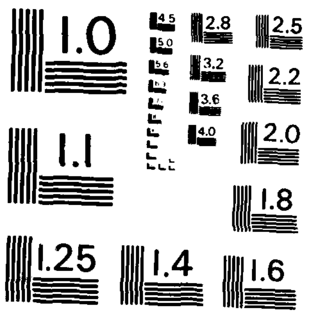
AD-A125 828 INTERACTIVE COMPUTER PROGRAM DEVELOPMENT SYSTEM STUDY VOLUME 3 FUNCTIONAL..(U) GENERAL DYNAMICS FORT WORTH TX FORT WORTH DIV H C CONN ET AL. JAN 83 DMA-2-014-VOL-3 1/1

UNCLASSIFIED RADC-TR-83-3-VOL-3 F30602-81-C-0039 F/G 9/2 NL



END
DATE
FILMED
4 83
DTIC


1.0
4.5
5.0
5.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(12)

RADC-TR-83-3, Vol III (of three)
Final Technical Report
January 1983

AD A125628

INTERACTIVE COMPUTER PROGRAM DEVELOPMENT SYSTEM STUDY Functional Description

General Dynamics Corporation

H. C. Conn, Jr, D. J. Rodjak, M. A. Goode, R. M. Sand, C. O. Anderson, R. C. Robertson

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

83 03 15 [illegible]

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-83-3, Vol III (of three) has been reviewed and is approved for publication.

APPROVED: [illegible]

ROGER B. PANARA
Project Engineer

APPROVED: [illegible]

JOHN A. [illegible], Lt Colonel, USAF
Acting Chief, Command & Control Division

FOR THE COMMANDER: [illegible]

[illegible]
Acting Chief, Plans Office

[illegible]

[illegible]

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-3, Vol III (of three)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INTERACTIVE COMPUTER PROGRAM DEVELOPMENT SYSTEM STUDY Functional Description		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 6 Jan 81 - 30 Sep 82
		6. PERFORMING ORG. REPORT NUMBER DMA-2-014
7. AUTHOR(s) H.C. Conn, Jr. R.M. Bond D.J. Rodjak C.C. Anderson M.A. Goode R.C. Robertson		8. CONTRACT OR GRANT NUMBER(s) F30602-81-C-0039
9. PERFORMING ORGANIZATION NAME AND ADDRESS General Dynamics/DSD/Central Center North Grant Lane Ft Worth TX 76108		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 63701B 32050326
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COEE) Griffiss AFB NY 13441		12. REPORT DATE January 1983
		13. NUMBER OF PAGES 36
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

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

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Roger Panara (COEE)

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

Software Engineering
programming environment
software tools

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

Vol I (of three) describes the development of the design and supporting documentation for an incremental and evolving integrated modern engineering software production environment for the Defense Mapping Agency.

Vol II is the System/Subsystem Specification.

Vol III is the Functional Description.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FUNCTIONAL DESCRIPTION
TABLE OF CONTENTS

Section	Title	Page
1.	GENERAL	4
1.1	Purpose of the Functional Description	4
1.2	Project References	4
1.3	Terms and Abbreviations	5
2.	SYSTEM SUMMARY	6
2.1	Background	8
2.2	Objectives	8
2.3	Existing Methods and Procedures	8
2.4	Proposed Methods and Procedures	11
2.4.1	Summary of Improvements	19
2.4.2	Summary of Impacts	20
2.4.2.1	Equipment Impacts	20
2.4.2.2	Software Impacts	20
2.4.2.3	Organizational Impacts	20
2.4.2.4	Operational Impacts	20
2.4.2.5	Development Impacts	21
2.5	Assumptions and Constraints	21
3.	DETAILED CHARACTERISTICS	22
3.1	Specific Performance Requirements	22
3.1.1	Accuracy and Validity	22
3.1.2	Timing	22
3.2.	System Functions	22
3.2.1	Minicomputer Hosted Tools	22
3.2.1.1	Requirements Tool	22
3.2.1.2	Design Tool	22
3.2.1.3	Coding Tool	23
3.2.1.4	Testing Tool	23
3.2.1.5	Documentation Tool	23
3.2.1.6	Configuration Control Tool	23
3.2.1.7	Project Management Tool	23
3.2.2	Support Activities	23
3.2.2.1	MPE Administrator/Toolsmiths	24
3.2.2.2	Training	24
3.2.3	Computer Links	24
3.3	Inputs-Outputs	25
3.4	Data Characteristics	28
3.5	Failure Contingencies	28
4.	ENVIRONMENT	30
4.1	Equipment Environment	30
4.2	Support Software Environment	30
4.3	Interfaces	30
4.4	Security and Privacy	30
5.	COST FACTORS	32
6.	SYSTEM DEVELOPMENT PLAN	32

FUNCTIONAL DECSRIPTION FIGURES

Number	Title	Page
2.1	Sample Modern Programming Environment	7
2.3	Existing HOL Software Development	10
2.4.1	Project Management Overview	12
2.4.2	Proposed HOL Software Development - Overview	14
2.4.3	Proposed HOL Software Development	15
3.3.1	Requirements Tool	25
3.3.2	Design Tool	25
3.3.3	Coding Tool	25
3.3.4	Testing Tool	26
3.3.5	Documentation Tool	26
3.3.6	Configuration Control Tool	26
3.3.7	Project Management Tool	27
3.3.8	Training Tool	27

A-1

SECTION 1. GENERAL

1.1 Purpose of the Functional Description. This Functional Description for the Defense Mapping Agency's (DMA) Interactive Computer Program Development System Study (ICPDSS), Contract Number F30602-81-C-0039 through Rome Air Development Center (RADC) is written to provide the system requirements of the Near-Term Modern Programming Environment (MPE). This will serve as a basis for mutual understanding between the user and developer, as well as information on preliminary design and user impacts. The description presented is generic in nature. Each of DMA's centers will have duplicates of the system described. For specific tool recommendations reference the System/Subsystem Specification.

1.2 Project References. These references provide information on the history of the project, technical data collected and the collection process, and documentation concerning related projects.

a. Project Request (copy not included) - UNCL

Solicitation Number F30602-80-R-0206
Rome Air Development Center
Attn: Contracting Division (PK)
Griffiss Air F[illegible] Base, New York 13441

b. Technical Documentation previously developed:

CDRL A002 - Statement of Operation Need and System Operational Concept - UNCL
CDRL A003 - Tool Evaluation Plan - UNCL
CDRL A004 - Tool Survey - UNCL
CDRL A005 - Alternative Analysis - UNCL

c. Significant Correspondence:

CDRL A001 - Monthly Status Reports - UNCL

d. Related Projects Documentation:

FEDSIM (Federal Computer Performance Evaluation and Simulation Center) Installation Review - DMAHTC - November 1980 - UNCL
DMA Operational Concepts (1982-1990) - May 1979 - UNCL
DMA Programming Support Library (PSL) Interim Evaluation Report, IBM/FSD - November 1980 - UNCL

DMAAC/Scientific Computer Division - Software Life Cycle Standards - February 1981 - UNCL
FEDSIM Installation Review - DMAAC - August 1980 - UNCL
DMA Modern Programming Environment (MPE) - January 1980 - UNCL
FEDSIM Optimization and Error Rate Studies - February 1981 - UNCL

e. Additional Documentation

CDRL A007 - System/Subsystem Specification - UNCL
CDRL A008 - Final Report - UNCL

1.3 Terms and Abbreviations.

ANSI	AMERICAN NATIONAL STANDARDS INSTITUTE
ASCII	AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE
ADP	AUTOMATED DATA PROCESSING
CDRL	CONTRACT DATA REQUIREMENTS LIST
DMA	DEFENSE MAPPING AGENCY
DMAAC	DEFENSE MAPPING AGENCY AEROSPACE CENTER
DMAHTC	DEFENSE MAPPING AGENCY HYDROGRAPHIC/TOPOGRAPHIC CENTER
DoD	DEPARTMENT OF DEFENSE
LAN	LOCAL AREA NETWORK
FEDSIM	FEDERAL COMPUTER PERFORMANCE AND EVALUATION AND SIMULATION CENTER
HOL	HIGH ORDER LANGUAGE
ICPDSS	INTERACTIVE COMPUTER PROGRAM DEVELOPMENT SYSTEM STUDY
MPE	MODERN PROGRAMMING ENVIRONMENT
PERT	PERFORMANCE EVALUATION REVIEW TECHNIQUE
RADC	ROME AIR DEVELOPMENT CENTER
R&D	RESEARCH AND DEVELOPMENT
SIP	SOFTWARE IMPROVEMENT PROGRAM
UNCL	UNCLASSIFIED

SECTION 2. SYSTEM SUMMARY

This section provides a general description, written in non-Automated Data Processing (ADP) terminology, of the proposed DMA Modern Programming Environment (MPE). As an introduction the following paragraphs provide a brief overview of the purpose and components of a MPE.

A MPE is a means of improving the software development process, thereby improving the quality of software in terms of reliability, maintainability, and performance. This is accomplished through the use of a standard, integrated set of methodologies using automated software development tools. These tools and methodologies cover all life cycle phases of the software development process including requirements, design, coding, testing and maintenance. Capabilities outside the life cycle are project management and training support. A MPE is confronted with continually changing requirements and available tools. The MPE is upgradable as this evolving process occurs. Figure 2.1 illustrates an example of a MPE.

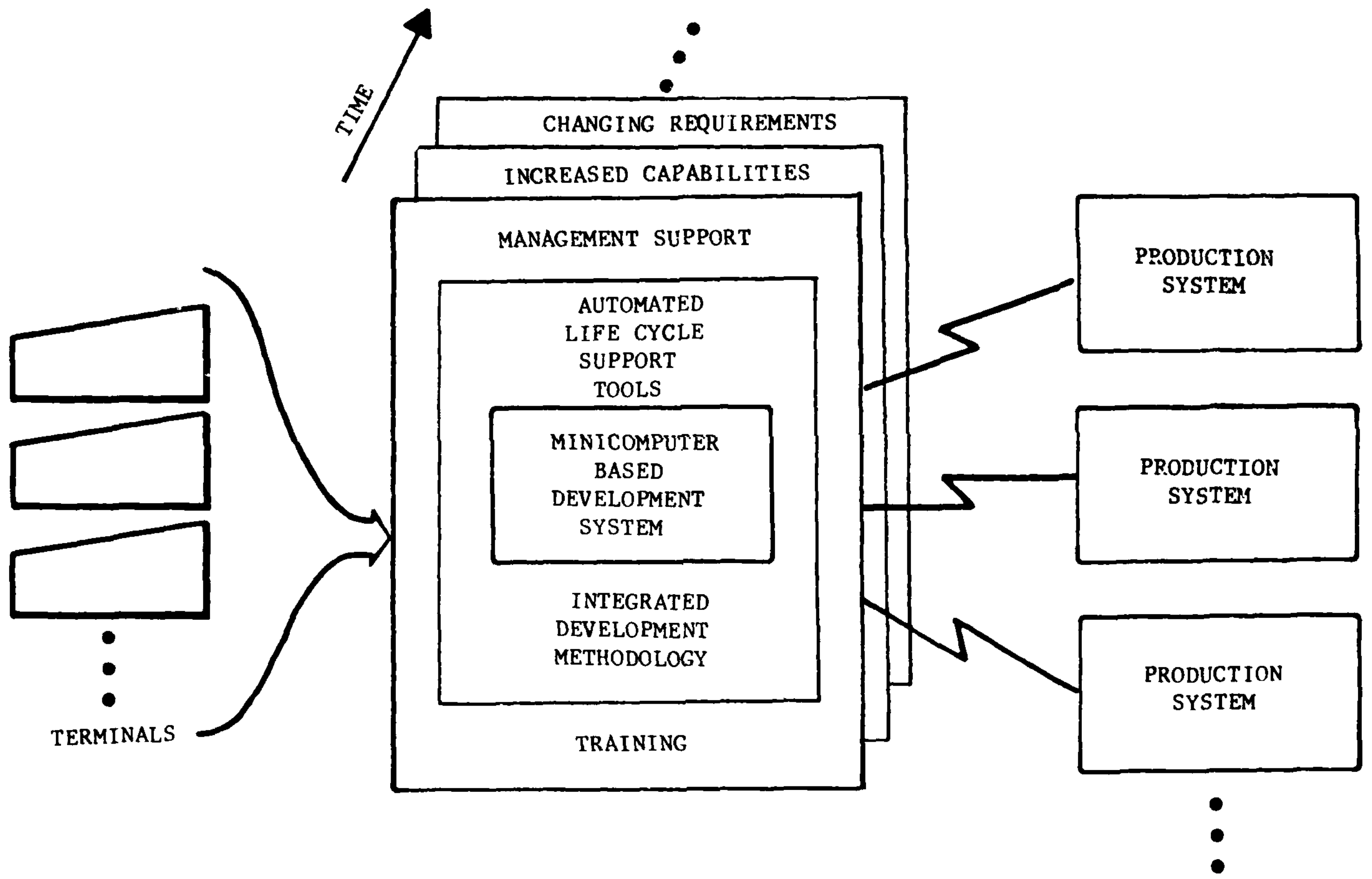


Figure 2.1 Sample Modern Programming Environment

User access to the illustrated MPE system is through interactive terminals and standard, user-friendly interfaces to the automated life cycle support tools. The development system is based on a minicomputer thus removing development activities from the production machines and forming a common, standard environment for software development. Additional provisions are early error detection through the use of requirement and design tools and the generation of a more complete and standard documentation produced through the use of the automated life cycle support tools. These benefits in turn provide for more easily maintainable, modifiable software systems. One final requirement of a MPE is that it can be easily modified and/or upgraded as the needs of its users change.

2.1 Background. The Near-Term MPE design was developed to provide DMA with the capability to meet its software development needs in 1985 and to provide a baseline for a system to meet DMA's 1987 needs. The Final Report, Section 2.0, provides information concerning the generation of the near-term and far-term needs. The specific research accomplished to identify solutions to DMA's needs is described in the Final Report, Sections 8 through 15.

2.2 Objectives. The Near-Term MPE specification incorporates the design and supporting documentation for an incremental and evolving integrated modern engineering software production environment for DMA. The period of concern is 1985 to 1987. Realization of the MPE will lead to the establishment of a comprehensive and coherent framework for specifying, designing, programming, testing and maintaining software in a highly visible, traceable and cost effective manner. The Final Report identifies R&D which must be accomplished and changes in the system which must occur to evolve from Near-Term to Far-Term MPE.

2.3 Existing Methods and Procedures. Software activities at DMA fall into three major categories: 1) development of new software, 2) addition of new capabilities to existing software, 3) detection and correction of errors in existing programs. Programs are also developed by outside vendors. Most of the software developed is written in dialects of FORTRAN and COBOL. Assembly language is also used but is not addressed in this document. Multiple software life cycle definitions are utilized; but in general all are generic to the requirements, design, programming, testing, and maintenance phased development process.

There is no formal method of specifying software requirements, although some customized methods do exist. The design of programs is not formalized; but some organizations do document their efforts through the use of program specifications. The programming phase is labor intensive with some system support utilities available to help automate the process. Most automation has been developed for the testing phase. Code auditing is automated but is not in general use throughout DMA. The maintenance function relates to the second and third categories of development previously mentioned. The revision of software is a major effort at DMA; but the current configuration management systems are not automated or strictly enforced. Currently standards are being developed and implemented to formalize many activities and methodologies in the area of software development which will enhance existing techniques. These standards include content of documentation, utilization of personnel, and use of tools and techniques to support each phase.

The management of software development projects is accomplished with no use of automated tools. Some projects are managed with manual methods such as PERT, but this is not generally done.

Figure 2.3 illustrates the current software development and maintenance procedures at DMA.

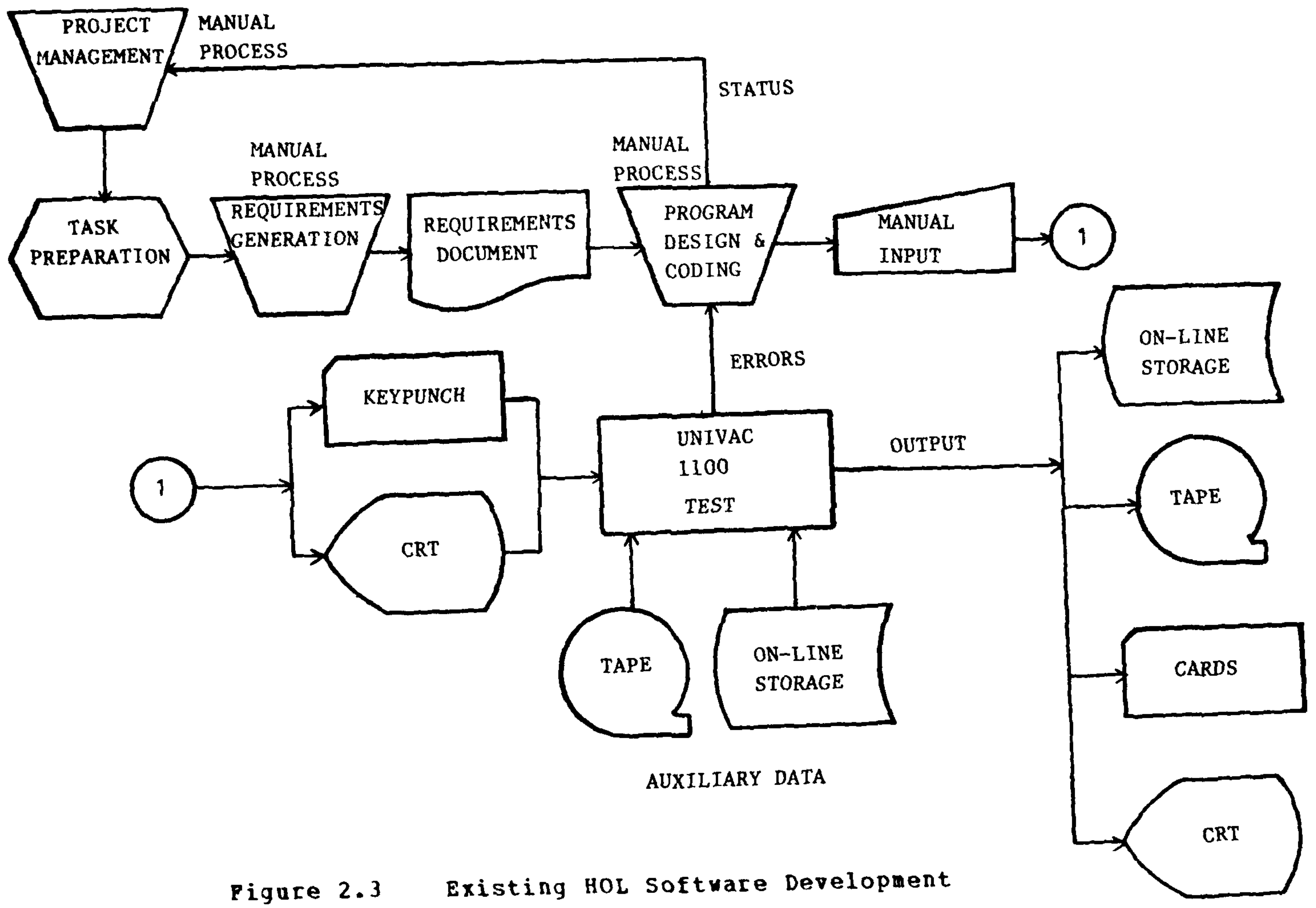


Figure 2.3 Existing HOL Software Development

2.4 Proposed Methods and Procedures. The near-term system was selected to meet the immediate needs of DMA. As defined, the system has a high probability for improving productivity.

A set of software tools residing on a minicomputer will be utilized for the requirements, design, programming, testing and maintenance functions of the software development life cycle. The specific configuration is described in the System/Subsystem Specification. For clarification, 'Maintenance functions' is defined as post production software development activity requiring work in one or more phases of the life cycle: requirements, design, programming, testing. These would include activities such as the correction of software errors discovered in production programs and modifications or upgrades to programs already on production status.

All software developed is monitored through the use of a project management tool. Examples of inputs and ouputs of the project management system are demonstrated in Figure 2.4.1. Upon receiving a job request, the project management tool is initiated for the job and at various points in the scenarios, the project management system is updated to reflect pertinent decisions and actions.

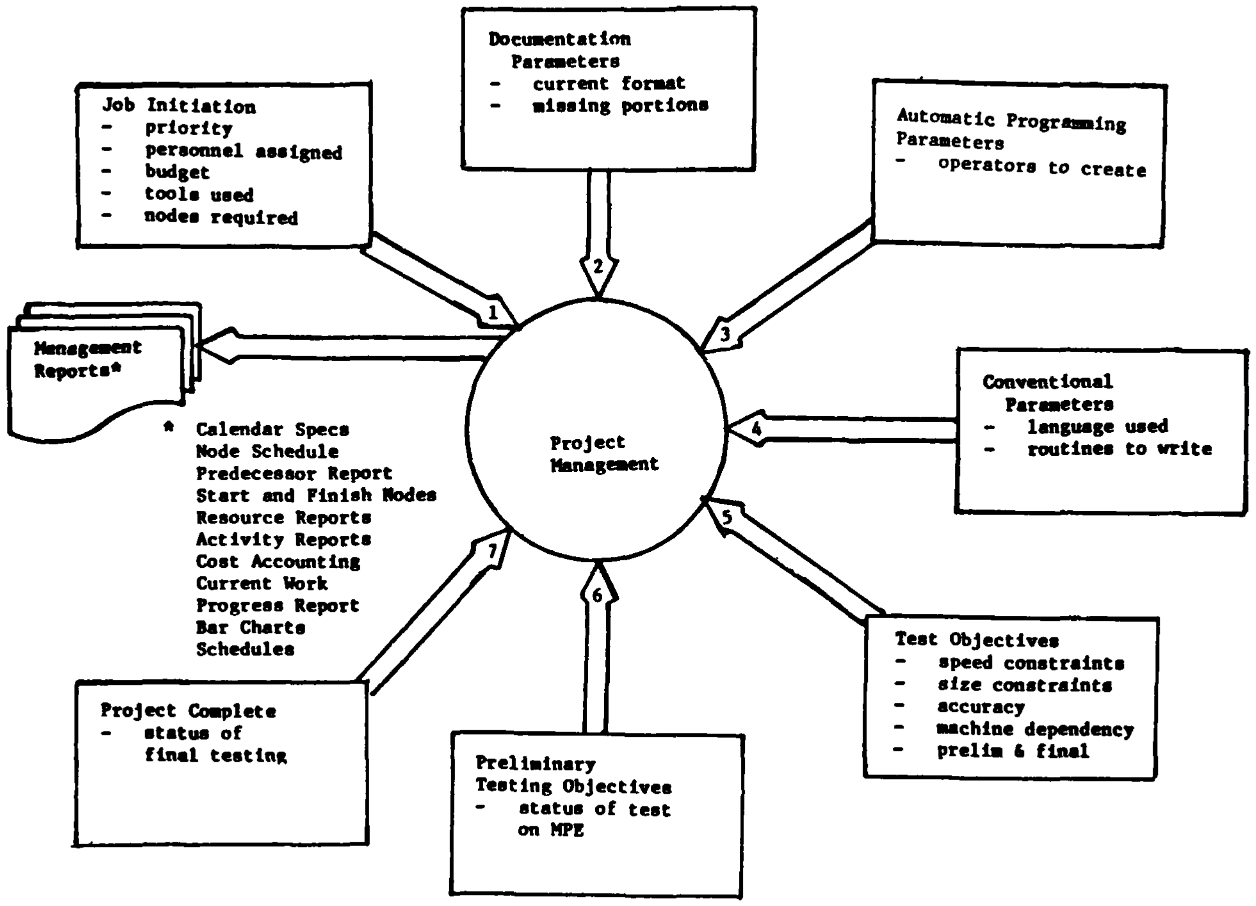


Figure 2.4.1 Project Management Overview

For purposes of discussion, scenarios will be considered for the following categories of software development:

1. maintenance to existing software which has not been upgraded through the Software Improvement Program (SIP), (Part of this program consists of an effort to improve existing UNIVAC software.)

2. maintenance to existing software which has been SIP upgraded,

3. software under development for which standards were specified,

4. new software to be developed by DMA for which standards are to be specified, and

5. new software to be developed by contractor for which standards are to be specified.

The techniques discussed are intended to demonstrate the applicability of the recommended tools to the various scenarios. Specific usage methodologies will be developed during the MPE system implementation as outlined in Section 19.1 of the Final Report.

The application of the MPE tools to the DMA environment is illustrated in Figures 2.4.2 and 2.4.3.

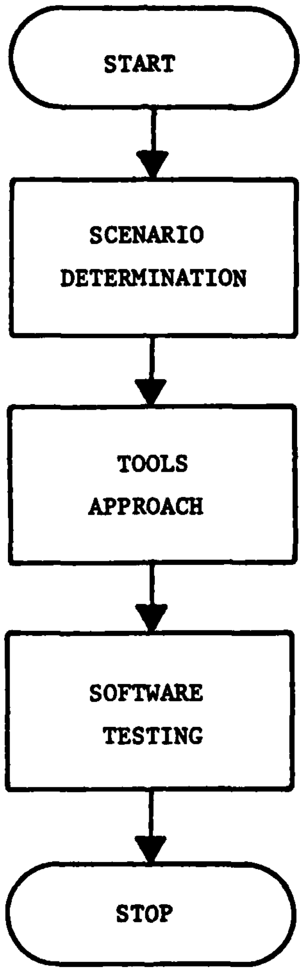
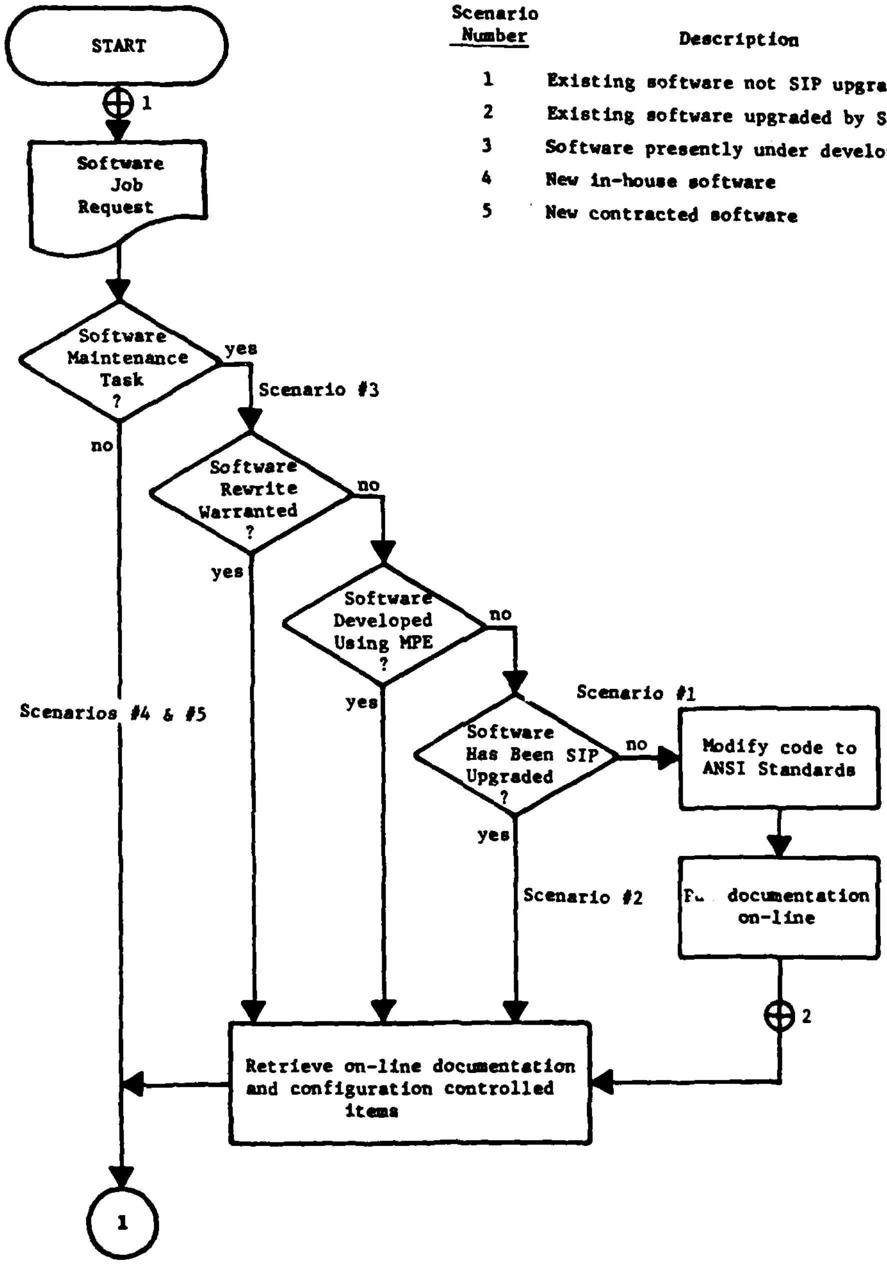


Figure 2.4.2 Proposed HOL Software Development - Overview

Scenario Number	Description
1	Existing software not SIP upgraded
2	Existing software upgraded by SIP
3	Software presently under development
4	New in-house software
5	New contracted software



START
1
Software Job Request
Software Maintenance Task ?
yes
Scenario #3
no
Software Rewrite Warranted ?
no
yes
Software Developed Using MPE ?
no
yes
Scenarios #4 & #5
Scenario #1
Software Has Been SIP Upgraded ?
no
yes
Modify code to ANSI Standards
Scenario #2
F[illegible] documentation on-line
2
Retrieve on-line documentation and configuration controlled items
1

Figure 2.4.3 Proposed HOL Software Development (page 1 of 2)

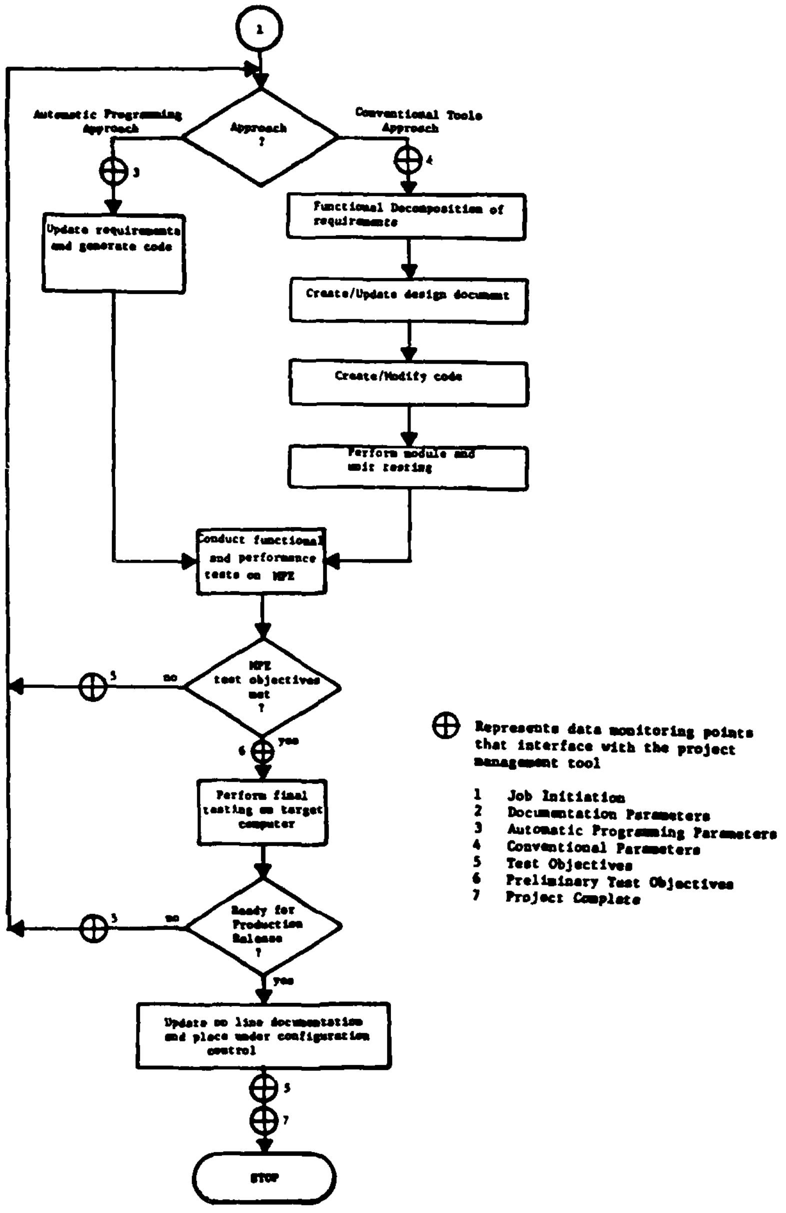

1
Approach ?
Automatic Programming Approach
Conventional Tools Approach
3
4
Update requirements and generate code
Functional Decomposition of requirements
Create/Update design document
Create/Modify code
Perform module and unit testing
Conduct functional and performance tests on MPE
MPE test objectives met ?
no
5
yes
6
Perform final testing on target computer
Ready for Production Release ?
no
5
yes
Update on line documentation and place under configuration control
5
7
STOP
⊕ Represents data monitoring points that interface with the project management tool
1 Job Initiation
2 Documentation Parameters
3 Automatic Programming Parameters
4 Conventional Parameters
5 Test Objectives
6 Preliminary Test Objectives
7 Project Complete

Figure 2.4.3 Proposed HOL Software Development (page 2 of 2)

Within the defined scenarios, one of two basic tool approaches will be followed.

The first, referred to as the "automatic programming approach", will make repeated use of the subsets of an automatic programming tool until performance criteria are achieved. The usage of the various subsets is as follows:

- the graphics editor is used to enter program structures (control maps) to functionally decompose requirements and design specifications as well as changes, if any, which are required as a result of performance testing,

- an analyzer verifies consistency and interfaces,

- source code is automatically produced from requirements definition,

- the source code is compiled and linked, and

- the system is performance tested to determine acceptability.

Failure to pass performance testing results in repetition of these steps until criteria are satisfied.

There appears to be no restriction on the size of system which may be developed with such a tool. As systems are developed, generic operations are developed, documented and can be placed in a library for use as building blocks on subsequent systems.

On occasions, there may be circumstances which dictate the use of an approach other than automatic programming. Reasons to utilize such an approach include systems which indicate the use of COBOL, time critical applications, and applications for which automatic programming is not cost effective.

The second approach, referred to as the "conventional tools approach", will make use of a requirements language, design language, structured FORTRAN or COBOL, testing and documentation tools through the life cycle. Utilization of tools in the "conventional method" consists of repeated application of the following procedures until performance criteria are achieved.

- A requirements language is used for functional decomposition of requirements specifications, and interface and data flow analysis on the resulting program model.

- A design language is used to originate the design or make design changes, if any, which were mandated as a result of performance testing.

- Source code is used to implement the original design or modified to reflect changes brought about by design changes, or performance testing results.

- A testing tool is used to detect syntax errors, perform static analysis, and perform execution analysis.

- Performance testing is evaluated to establish the acceptability of the system. Failure to pass performance testing results in repeating the process.

One of these tool application approaches is followed until the preliminary test objectives are met. At this time, the source is transmitted via data link to the target host for final testing.

While testing on the target host, the project management system is apprised of the test status. Upon successful completion of final test objectives, job completion data is processed by the project management system. This action prevents the system status from being obscured from control and insures a match between production software and the associated documentation. Target host test objectives will verify proper usage of machine dependent devices, software and techniques. Once final testing is completed and the system is ready for production status, on-line documentation such as requirements and design documents, source code and test data should be updated and placed under configuration control.

All coding will be accomplished in structured FORTRAN or COBOL. Additionally, systems supporting documentation, text editing, testing, configuration control and project management will reside on the minicomputer.

The MPE administrator and toolsmith functions will support the project management function as well as system management; and a tool for building presentations and interactive lessons will be utilized for training purposes.

The minicomputer and production mainframe will need to be connected through a communications link. To support MPE users in a timely manner and to provide adequate access, multiple minicomputers will be required.

2.4.1 Summary of Improvements. In Section 2.3, deficiencies were identified. The solutions provided by the near-term MPE are described in the following paragraphs.

The lack of a formal method of specifying requirements is resolved through the use of an automated requirements tool, which must be formalized through adoption into standards being developed. Design also is to be formalized through the use of an automated tool by a similar process. The use of both of these tools represents functional improvements in the existing software development process. Additional functional improvements are the use of an automated configuration management system, and the use of a project management tool.

The programming phase is upgraded by the addition of new capabilities and is an improvement of degree rather than function. The addition of new terminals allows for more software development and maintenance to be accomplished interactively rather than in batch mode. The system design tool utilization decreases the labor intensive effort by partially automating the process. Another improvement of degree is the generation of project review documentation. New documentation will be generated automatically as part of the output of the software development tools in the requirements, design, project management, and configuration control activities.

By removing the software development process from the production machines, and by supporting interactive development through additional terminals, the time span required for software development will be reduced. There will be no competition with large production jobs for computer time and turnaround time will be significantly improved by the increased availability of interactive terminals. Additionally, the increased documentation associated with development; the standardization of the documentation; and the configuration control of all on-line documentation should result in improved quality and productivity in the software maintenance phase. The documentation referred to here includes on-line requirements specification and design documents, source code and test data.

The only area where tasks are to be eliminated is when the software development/maintenance effort warrants the exclusive use of the automated programming software tool for the development of a program. These programs will be automatically generated from the requirements specification, eliminating all manual activity associated with design, programming, and testing. In this area configuration control would be at the requirements specification level only.

<u>2.4.2 Summary of Impacts.</u> The anticipated impact on the existing equipment at DMA involves only the addition of hardware. A communications interface will be necessary between an existing mainframe and the proposed minicomputer. With the exception of communications software, no new software will be required on existing hardware. Software development will be reduced on existing systems and moved to new equipment, requiring the users to learn additional aspects of computer access and software development. For the costs associated with this system reference Section 20.0 of the Final Report. Personnel will be required to support operation of the new computers and the MPE administrator and the toolsmith functions.

<u>2.4.2.1 Equipment Impacts.</u> One or more mainframe communication ports will require configuration to minicomputer access. The ports/channels selected must operate at a high baud rate. The specific number of ports/channels to be dedicated will vary as multiple minicomputer systems are installed.

<u>2.4.2.2 Software Impacts.</u> The communications configuration software will need modification to support the system hardware changes described in Section 2.4.2.1 of this report. It is anticipated that no other existing software will be added to or modified.

<u>2.4.2.3 Organizational Impacts.</u> The positional responsibilities of personnel will not need modification, but an addition will be required. One organization must control the system configuration of the multiple minicomputers to maintain intersystem software compatibility. A group should be responsible for the configuration management of all production software once a baseline has been achieved. Personnel will be required to support the operation of the minicomputers and the MPE administrator and toolsmith functions.

<u>2.4.2.4 Operational Impacts.</u> The programming standards of DMA will require modification/extension to support and enforce the new aspects of software development. These include methods and tools to be used, documentation to be produced, project review management procedures, and configuration management techniques. System users will require training to utilize the new minicomputers as well as the hosted tools within the defined standards.

2.4.2.5 Development Impacts. Since it is recommended the proposed system first be implemented under an experimental system configuration, no effort will be required by the user community prior to system development. System development would be aided by identifying a core of personnel to perform the system analysis first rather than providing access to the general populace.

2.5 Assumptions and Constraints. The assumptions associated with this description include: (a) the capability of communicating over a link between a minicomputer using a standard interface and a production mainframe; (b) that physical space is available for the minicomputers and terminals; (c) workload will increase; and (d) skill level of personnel will be upgraded.

Constraints assumed to be applied to this description include: (a) security and (b) standards.

SECTION 3. DETAILED CHARACTERISTICS

3.1 Specific Performance Requirements. The specific performance requirements of the system based upon the DMA needs as described in the Final Report are described qualitatively in the following paragraph.

The system shall provide users with an interactive access capability to software development hardware and support tools. This capability will provide for improved software development/maintenance productivity by providing automated support, quicker access, and improved response time.

3.1.1 Accuracy and Validity. Not applicable

3.1.2 Timing. Not applicable

3.2 System Functions. The Near-Term MPE functions as a tool to provide life cycle support to the software development process. In this section the individual functions of each major element will be described as well as the function of the aggregate.

3.2.1 Minicomputer Hosted Tools. A large minicomputer will be utilized to host the MPE including requirements specifications, design, coding, testing, documentation, configuration control and project management tools. The computer should support multiple terminals distributed according to functional responsibility.

3.2.1.1 Requirements Tool. The specification and documentation of the requirements of a computer program should be partially automated through the use of a software support tool. This tool should allow the interactive development of a requirements specification document using a defined methodology, and analysis of the specification for data flow and control sequences. When the program specified can be categorized to fit within certain constraints, the requirements tool should be able to directly generate a high order language (HOL) program to accomplish the specified task.

3.2.1.2 Design Tool. The design of certain categories of programs will be accomplished utilizing design support software. Whether the task is accomplished by an individual or a team, the tool will provide precise, accurate and orderly transitions between requirements, design and coding activities as well as intra-design activities. The tool will provide, through a prescribed methodology, the capability to describe the design in simple, understandable constructs that are easy to code; allow for checking of the design constructs; and translate the design into a readable design document.

3.2.1.3 Coding Tool. Data entry will be performed interactively when generating new code or documentation. This activity should be supported by state-of-the-art word processing and text editing capabilities. The HOL(s) used for coding should be ANSI standard and fully compatible (without considering device dependent extensions) with the target production machine's compiler(s) to which completed, tested programs will be sent for final compilation and production status. A precompiler may be used as necessary to produce this standard code and allow the use of structured programming constructs. The use of these constructs increases the readability of the code and therefore the maintainability of the resulting program.

3.2.1.4 Testing Tool. The testing support tool should provide static and dynamic analysis of the specified HOL source code including usage, path flow and coverage statistics. Additional capabilities to enhance documentation, such as the output of cross-reference tables and summary data or pretty-printing the source input should also be included.

3.2.1.5 Documentation Tool. Program developers should be able to create and maintain documentation on-line through the use of a word processor. This would also allow for configuration management of the documents associated with a program along with other on-line documentation as described in the following sections.

3.2.1.6 Configuration Control Tool. Configuration control will be supported through the use of a data control system. The configuration management of textual material, for example, HOL code, documentation, including on-line requirements and design definitions, and test data, should be provided.

3.2.1.7 Project Management Tool. Project management should be supported through the use of interactive tools that perform resource allocation and analysis, time and cost analysis, and report processing.

3.2.2 Support Activities. Due to the complexity of the proposed Near-Term environment, the evolutionary process required to achieve the Far-Term environment, and a need for a focal point for identification/resolution of problems, support activities must be provided to supplement the development/maintenance environment.

3.2.2.1 MPE Administrator/Toolsmiths. These would be support positions which would primarily serve as the focal point for management to observe the system activities and as an information source for MPE training. Personnel involved with this function would be knowledgeable in the current tools and methodologies contained in the MPE as well as the minicomputer environment. Specifically, the MPE administrator would be responsible for an overall understanding of the MPE and its use. Toolsmiths would aid the MPE administrator by each having a thorough knowledge of a particular component of the MPE system. Tasks would include performing error rate studies, helping users with software development problems and the identification of needs not satisfied within the user/management communities.

3.2.2.2 Training. A microcomputer based system for the development and delivery of lecture material should be used as part of a comprehensive training program to provide low cost, self-paced training to personnel outside the production environment.

3.2.4 Computer Links. Communication links must be established among the host minicomputers and between the minicomputers and production mainframe. This is necessary in order to efficiently utilize the proposed environment. The primary function of the minicomputer/mainframe link will be to transfer completed, tested systems to the mainframe for production use. The minicomputers will be connected through a local area network (LAN) providing communication and backup capabilities thus improving system reliability.

3.3 Inputs-Outputs. The following figures present the inputs and outputs of the major functional components of the Near-Term MPE.

```
______INPUTS______      _____PROCESSING_____     ________OUTPUT_________

                        ********************
                        *                  *
DATA-----------------> * DATA ANALYSIS    *     -----> ERROR REPORTS
CONTROL STRUCTURE---> * CONTROL ANALYSIS *     -----> SYSTEM MODEL
DEFINITIONS---------> * LOGIC ANALYSIS   *     -----> GRAPHICS DISPLAY
FUNCTION------------> * FORTRAN SOURCE   *     -----> FORTRAN SOURCE
                        *     GENERATION   *
                        *                  *
                        ********************
```

Figure 3.3.1 Requirements Tool

```
______INPUTS______         _____PROCESSING_____   ________OUTPUT_________

                           ********************
                           *                  *
STRUCTURE DEFINITION---> * LOGIC ANALYSIS   *  ----> DESIGN DOCUMENT
OPTIONS----------------> * FORMATTING       *  ----> CUSTOMIZED REFERENCE TABLES
DESIGN-----------------> * FLOW ANALYSIS    *  ----> INVOCATION HIERARCHY
MANAGEMENT INFORMATION-> * SUMMARIZING      *  ----> DESIGN STATISTICS SUMMARY
                           *                  *
                           ********************
```

Figure 3.3.2 Design Tool

```
______INPUTS______      _____PROCESSING_____     ________OUTPUT_________

                        ********************
                        *                  *     -----> GENERATED DATA
                        * COMPILATION      *     -----> EXECUTABLE MODULE
PROGRAM SOURCE------> * LINKING          *     -----> OBJECT MODULES
OPTIONS-------------> * LOADING          *     -----> SOURCE LISTING
                        *                  *     -----> COMPILATION ERRORS
                        ********************
```

Figure 3.3.3 Coding Tool

```
______INPUTS______      _____PROCESSING_____    _______OUTPUT_________

                       ************************
                       *                      *
FORTRAN/COBOL CODE--> * STATIC ANALYSIS      *    -----> STATIC ANALYSIS REPORTS
TEST OPTIONS--------> * CODE INSTRUMENTATION *    -----> INSTRUMENTED CODE
                       * EXECUTION ANALYSIS   *    -----> EXECUTION ANALYSIS REPORT
                       * SOURCE CODE          *
                       *                      *
                       ************************
```

Figure 3.3.4 Testing Tool

```
______INPUTS______      _____PROCESSING_____    _______OUTPUT_________

                       **************************
                       *                        *
                       * REFORMATTING           *
ENGLISH TEXT--------> * MERGING OF TEXT        * -----> FORMATTED TEXT
MULTIPLE FILES------> * GLOBAL/SELECTIVE CHANGE*
PROCESSING COMMANDS-> * AUTOMATIC PAGINATION   *
                       * SPELLING CHECKING      *
                       *                        *
                       **************************
```

Figure 3.3.5 Documentation Tool

```
______INPUTS______      _____PROCESSING_____    _______OUTPUT_________

                         *************************
                         *                       *
                         * SELECTIVE COMPILATION *  -----> LATEST VERSION
DOCUMENTS-----------> * FILE GENERATION       *  -----> PREVIOUS VERSIONS
PROGRAMS------------> * FILE SECURITY         *  -----> UPDATED MODULES
COMMANDS------------> * FILE MANIPULATION     *  -----> PROGRAM HISTORY
                         * DATA COLLECTION       *
                         *                       *
                         *************************
```

Figure 3.3.6 Configuration Control Tool

```
______INPUTS______     ______PROCESSING______      _______OUTPUT_________

                       ************************
CALENDARS-----------> *                      *     -----> NETWORK CHARTS
EVENTS--------------> * TIME ANALYSIS        *     -----> BAR CHARTS
RESOURCES-----------> * RESOURCE ALLOCATION  *     -----> TAILORED REPORTS
UNIT COSTS----------> * COST CONTROL         *     -----> HISTORICAL DATA
                       *                      *     -----> NETWORKS
                       ************************
```

Figure 3.3.7 Project Management Tool

```
______INPUTS______     _____PROCESSING_____     _______OUTPUT_________

                       ******************
LESSON MATERIAL-----> *                *     -----> FORMATTED LESSONS
COMMANDS------------> * DEVELOPMENT    *     -----> LESSON FILES
LESSON FILES--------> * DELIVERY       *
USER----------------> * SUPPORT        *
AUTHOR--------------> * TRAINING       *
                       ******************
```

Figure 3.3.8 Training Tool

<u>3.4 Data Characteristics.</u> Not applicable

<u>3.5 Failure Contingencies.</u> Potential failures could occur in any of the software tools described on the minicomputer.

a. <u>Back-up.</u> Redundancy in the minicomputer hardware and software will be available through the use of multiple, identically configured systems connected through a LAN.

b. <u>Fallback.</u> All systems can be simulated through manual processes or deferred in the case of massive system failure with the exception of the language processors. These particular processors will be redundant, one per host computer.

c. <u>Restart.</u> Not applicable

SECTION 4. ENVIRONMENT

The "system" being described is an environment. The environment "surrounding" the Near-Term environment proposed is the production facilities of DMA. The software, interfaces and security of the production environment are beyond the scope of this document. The interface between the Near-Term MPE minicomputer systems and target production systems will depend upon DMA decisions in developing the planned local networks.

4.1 Equipment Environment.
Not applicable

4.2 Support Software Environment.
Not applicable

4.3 Interfaces.
Not applicable

4.4 Security and Privacy.
Not applicable

SECTION 5. COST FACTORS

The proposed system represents only the first stage in a process to introduce a modern programming environment (MPE) into DMA. This system is a base from which a 1985 MPE will evolve using methodologies and tools now being developed by DoD and industry. The growing digital product line of DMA will require an increase in the quality and quantity of application software which cannot be met strictly with staffing methods. Alternatives to this system have been evaluated and the methods and data are presented in the Final Report.

SECTION 6. SYSTEM DEVELOPMENT PLAN

This section is not applicable under the current contract. A generalized plan for development is presented in the Final Report.