1987

Software documentation for the Department of Defense using the software life cycle /

Demetria Deakos
Lehigh University

Software Documentation

For The Department Of Defense

Using the Software Life Cycle

by

Demetria Deakos

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

September 1987

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Date : September 17, 1987

Bernard Rayna

Professor in Charge

Donald J. Hillman

Head of Division

Table of Contents

Abstract

Software documentation is a major ingredient necessary for the success of a software project. The Department of Defense has definitive software documentation standards. These software documentation standards have been used to provide the basis for the documentation standards of the IEEE and can provide excellent resource material for internal standards for a software project based on software size, complexity, and management considerations.

The evolution and content of the Department of Defense software documentation standards are discussed in this thesis. The typical users and developers of the software documentation are listed. Guidelines for improving the readability and usefulness of documentation are established.

Chapter 1
Introduction

Documentation, an integral part of a software system, is the comprehensive written description of computer software in various formats and levels of detail that clearly define:

- Content
- Composition
- Design
- Performance
- Testing
- Use

The success of a large software system requires following sound documentation principles during system development and beyond. Just as a building should not be built until all plans are drawn up and agreed upon, the same principle applies to a large software system.

Two major areas of software documentation exist, development documentation and product documentation. They represent two types of documentation for different audiences: in the first case all those persons concerned with the development of the software product, and in the second case all those concerned with the use and application of the software product. Typical users of the development documentation are management, operational-design engineers, program-design engineers, programmers, program-test engineers, evaluation engineers, and on-site maintenance programmers. Typical users of product documentation are training personnel, operational end-user, maintenance

programmers, and management.

1.1 Development Documentation

Development documentation is closely related to the software life cycle. Documents needed during the development of the software system describe and specify what the user needs, i.e. the user requirements, and what the software does. Documents also deal with the specification of how programs should be constructed, and how their performance should be tested.

Typical document types needed here are requirements and functional specifications, emphasizing "what the system does", as well as design specifications, development, and test plans, emphasizing "how the system does it". It is the communications vehicle during the development process, recording technical details and key decisions for each stage of the process.

1.2 Product Documentation

Product documentation is a critical element for the use, operation, maintenance, and conversion of software systems. A software system refers to a well-tested computer program(s) which is fully documented and supported by a responsible organization.

Product documentation is prepared for the end-user to have available during normal operation, or for maintenance programmers who correct errors, or who enhance programs by adding new features based on system requirements. The end-user, who may not have a background in computers, needs to know how the program functions are to be operated, how the computer or related devices are to be operated, and what should be done if there should be a malfunction in hardware or software. The product documentation must be

prepared in a language most familiar to the specific user group it is intended for easy comprehension and use.

Additional information is required by maintenance personnel. Needed are details on the system environment such as relationships and interactions with computer installation facilities and other manual or automated data systems. This information is especially useful if programs are to transferred from one location to another.

1.3 Reasons for Documentation

Documentation provides a vehicle for communication between all individuals involved in a software project.

Documentation provides a vehicle for review of a software system. Standard review plateaus during a system development process are facilitated by timely presentation of accumulated system documentation.

Documentation provides a mechanism for monitoring project progress and evaluating personnel effectiveness by standardizing the procedures of the system development process. Since the documentation contains the breakdown of the software project into smaller parts, each part can be estimated and more accurate schedule be determined

Documentation minimizes problems with personnel absenteeism and turnover. Standards and documentation can help make any software project people-independent by not allowing dependence upon individuals for certain or all segments of a software system.

Documentation provides material by which operational people can be trained to use the system. A well written description is more satisfactory than a verbal description. This documentation can aid in the development of training

programs and be used in day-to-day reference by user personnel.

Documentation provides a mechanism for maintenance of the software system. Any modifications or updates to the software system procedures and/or programs may require three to four times the effort without good documentation.

1.4 Problems with Documentation

1.4.1 Who Writes It

Documentation has been traditionally written by the analyst/programmer. Technical writers also write documentation. There are distinct differences in the documentation produced by these two groups. Documentation written by software people is generally sloppy and disorganized, with the English resembling a programming language. However, if the documentation is less readable, it is more substantial in content. Documentation prepared by the technical writer is usually more readable, understandable, and very well organized. However, this documentation may not provide enough information since the technical writer is not software-oriented or user-oriented. If the information flow from system people to technical writers is inadequate, conflicting and noncommittal statements may be located throughout the documents, and essential ones may disappear.

1.4.2 Wrong Content

Different document types require different types of information. An example: in a document describing design, "what has been done" is not satisfactory if it lacks "why it has been done".

1.4.3 Wrong Point of View

System documentation is not satisfactory if it is written from the writer's point of view, rather than the user's point of view. Information required in the maintenance phase is in many respects distinctly different from that in the development phase.

1.5 Categories of Problematic Documentation

Problematic documentation falls into three categories: no documentation, insufficient documentation, and misleading documentation.

1.5.1 No Documentation

The problem of no documentation is most critical at the management level. Documentation defines the software system to be developed. Without this definition, it is not obvious if the software has met its specified requirements.

The development of the software system becomes more people dependent. Turnover of key personnel becomes critical since their knowledge is lost. Previous efforts in the software development may be discarded if it is not understandable. New personnel and maintenance personnel need a longer learning period since necessary information is not available. Flexibility and expandability originally designed into the system may be lost increasing the maintenance effort.

The software user is also affected adversely. Many features of the software

system may not be utilized if a user has no way to learn of the software capabilities. If problems are encountered, the user may not know how to proceed.

1.5.2 Insufficient Documentation

Insufficient documentation can be seen at all levels: management, systems and programming, and users. Several factors contribute to the situation, including improper system planning, lack of budgeted funding, and insufficient time. The foremost factor is not setting of standards for documentation.

Information not documented because of its obviousness in the development stage may be extremely valuable for the maintenance personnel, and is often lost.

1.5.3 Misleading Documentation

When insufficient documentation extends to the extreme, misleading documentation develops. This is the worst type of documentation and usually is the result of oversight, neglect, and ignorance.

The problem with misleading documentation is that an unwillingness to trust any documentation occurs. A direct consequence of this is further degeneration of documentation activity. If no decisive action is taken, eventually all documentation becomes worthless.

Chapter 2
Users of Documentation

Different users need varying amounts of documentation based on the function of each document and the job function of the user. The user may use one type of documentation to develop other documentation. Descriptions of the needed documentation are based on the software life cycle concept of the next chapter.

2.1 Functional View of Documentation

The functions documentation serves are:

- Intertask Communication
- Instructional Reference
- Quality Assurance Support
- Historical Reference

2.1.1 Intertask Communication

Most software projects are divided into tasks which often are carried out by different people. Some of these tasks are:

- Analysts formulate system requirements
- Designers develop overall program design
- Programmers provide detail code
- Quality assurance specialists are concerned with methods for quality software development and overall system testing
- Auditors monitor overall system integrity
- Maintainers improve operations or provide enhancements and extensions

Intertask communications are established in a formalized way. They provide requirements to designers, designs to coders, and system specifications to auditors and maintenance people. Functional, design, test, or system specifications provide intertask communication.

2.1.2 Instructional Reference

Readily available reference materials are needed to train users in system operation. Similarly, special documentation is needed by persons concerned with maintaining software for correction of software errors and changed requirements. This documentation is provided in user, maintenance, and operator's manuals.

2.1.3 Quality Assurance Support

System documentation which tells how well a system should perform are needed by all persons concerned with system performance and quality. Requirements documents, design specifications, quality assurance plans, test plans and test procedures need to be provided, and results need to be reported.

2.1.4 Historical Reference

Capabilities, system features, and operational details should be recorded. This will facilitate the re-use of well-proven ideas and assist in transfer and conversion of programs to new system environments. It may prevent false starts by illustrating problem solutions that have proved ineffective. System specifications, a variety of manuals, and test reports provide this information.

2.2 Software Project Manager

The software project manager uses the initial proposal requesting the software system. This may be a document as formal as a MIL-STD-490 system or product specification, or an informal wish list from an end user. The purpose of reviewing the document is to determine the feasibility of the software project and the eventual resources necessary to complete it. Feasibility may be determined from management plans from previous projects sharing similar characteristics as well as input from other personnel to be involved in the project.

If the software project is determined feasible, the software project manager must receive a formalized proposal of the actual requirements of the system. This document will be used to generate a formal internal requirements document for the developing organization. The establishment of the formal requirements becomes necessary to determine design, testing, quality assurance and configuration considerations.

The project manager will then draw his own plan(s) for the software project. This will require input for other mangers in his group to coordinate schedule and resources for design, testing, quality assurance, and configuration. Having separate managers for each activity will be determined by the size of the software project and the availability of personnel to fill these functions.

2.3 Software Development Manager

The software development manager will use the formal internal requirements document to develop a software development plan concerning design and coding standards. The software development manager will also use the formal internal requirements document to determine the design of the software. After the design is established, the software development manager is responsible for the development of several design documents. These documents describe the top level design, the lower level design, and the data base consideration.

After the code is developed, the software development manager is responsible for the development of the support documentation. This documentation contains the practical information necessary to create, modify, and install the source code and data.

2.4 Software Testing Manager

The software testing manager will use the formal internal requirements document to develop a software test plan concerning testing. The software test manager will also use the formal internal requirements document to determine the specifications to be tested. After the testing methodology is established, the software testing manager is responsible for the development of several test documents. These documents describe the test specifications, the test procedures, testing problems, and the report of the final test results.

2.5 Software Quality Control Manager

The software quality control manager will use the formal internal requirements document to develop a software quality control plan. The software quality control manager will also use the formal internal requirements document to verify that the software is reliable, effective and developed according to software development plan. After the design is established, the software quality assurance manager is responsible for documentation of discrepancies.

2.6 Software Configuration Manager

The software configuration manager will use all documentation available on a software project. The software configuration manager is responsible for the identification and control of the documentation, the establishment of baselines or versions of documentation, the status accounting, the auditing of the documentation, and the orderly process of filing and storage.

2.7 Software Design Engineer/Programmer

The software design engineer/programmer will use the formal internal requirements document to determine the design of the software. After the design is established, several design documents are developed. These documents describe the top level design, the lower level design, and the data base consideration. These documents are used to produce the code.

2.8 Software Engineer/Programmer

The software engineer/programmer will use the design documents, which describe the top level design, the lower level design, and the data base consideration. These documents are used to produce the code.

After the code is developed, the software engineer/programmer will generate the support documentation. This documentation contains the practical information necessary to create, modify, and install the source code and data.

2.9 Software Maintenance Engineer/Programmer

The software maintenance engineer/programmer will use the internal requirements documents, the design documents, and the documentation in the source code to design the changes to a software system. A new baseline of documents may be created. If a change in requirements is implemented, the requirements documents and design documents may be updated. If an error in software is corrected, the design documentation may be updated.

The software engineer will use the support documentation to be able to modify and reinstall the source code. This documentation may also be updated.

2.10 Software Test Engineer

The software test manager will use the formal internal requirements document to determine the specifications to be tested. After the testing methodology is established, several test documents are developed. These documents describe the test specifications, the test procedures, testing problems, and the report of the final test results.

2.11 Software Quality Assurance Engineer

The software quality control manager will use the formal internal requirements document to verify that the software is reliable, effective and developed according to software development plan. After the design is established, the software quality assurance engineer documents the discrepancies.

2.12 Software Configuration Engineer

The software configuration engineer will use all documentation available on a software project. The software configuration engineer identifies and controls the documentation, establishes the baselines or versions of documentation, provides the status accounting, provides the auditing of the documentation, and performs the filing and storage of documentation.

2.13 User's Systems Engineer/Analyst

The user's system engineer/analyst will determine the needs of the user. He will generate the original system documentation. He will evaluate the developer's formal internal requirements documents, operations manuals, and test report for satisfaction of his requirements. He will be responsible for approving requests for changes to the software system, and have his documentation updated.

2.14 System User/Operator

The system user/operator will use the manual necessary for the operation of the hardware hosting the software system. This information should include the loading of the program, any necessary adaptation data, and requirements that are site dependent.

2.15 Software User/Operator

The software user/operator will use the manual for the operation of the software. This information should include program features, limitations, error conditions, and recovery procedures for the software.

2.16 Software Training Personnel

The software training personnel will use the manuals describing the operation of the system and the operation of the software. The operation of the system concern the loading of the software program into the hardware. The operation of the software concern the input/output requirement of the software program. An elementary training program would include examples of the more common features. A more advanced training manual would includes examples of features not commonly encountered.

Chapter 3
Software Life Cycle

There exists a time period between the formulation of the idea about the software system and the time the software system ends. The software life cycle divides the time of software development into manageable parts and provides a framework of milestones, which monitor progress and make decisions about direction and control of the software project. These parts cover such different activities as initiation, requirements analysis, design, programming, testing, operation and maintenance. Figure 3-1 on page 17 shows the different phases of the software life cycle.

3.1 System Life Cycle

The software life cycle is part of the system life cycle. A system includes the combination of hardware and software. The system life cycle is broken into four phases:

- Concept Exploration
- Demonstration and Validation
- Full Scale Development
- Production and Deployment

The Concept Exploration phase is the initial planning period. The technical, strategic, and economic bases are established through comprehensive studies, experimental development, and concept evaluation. This effort may be directed toward refining solutions or developing alternative concepts to satisfy a required operational capability.

The Demonstration and Validation phase is the period when major system

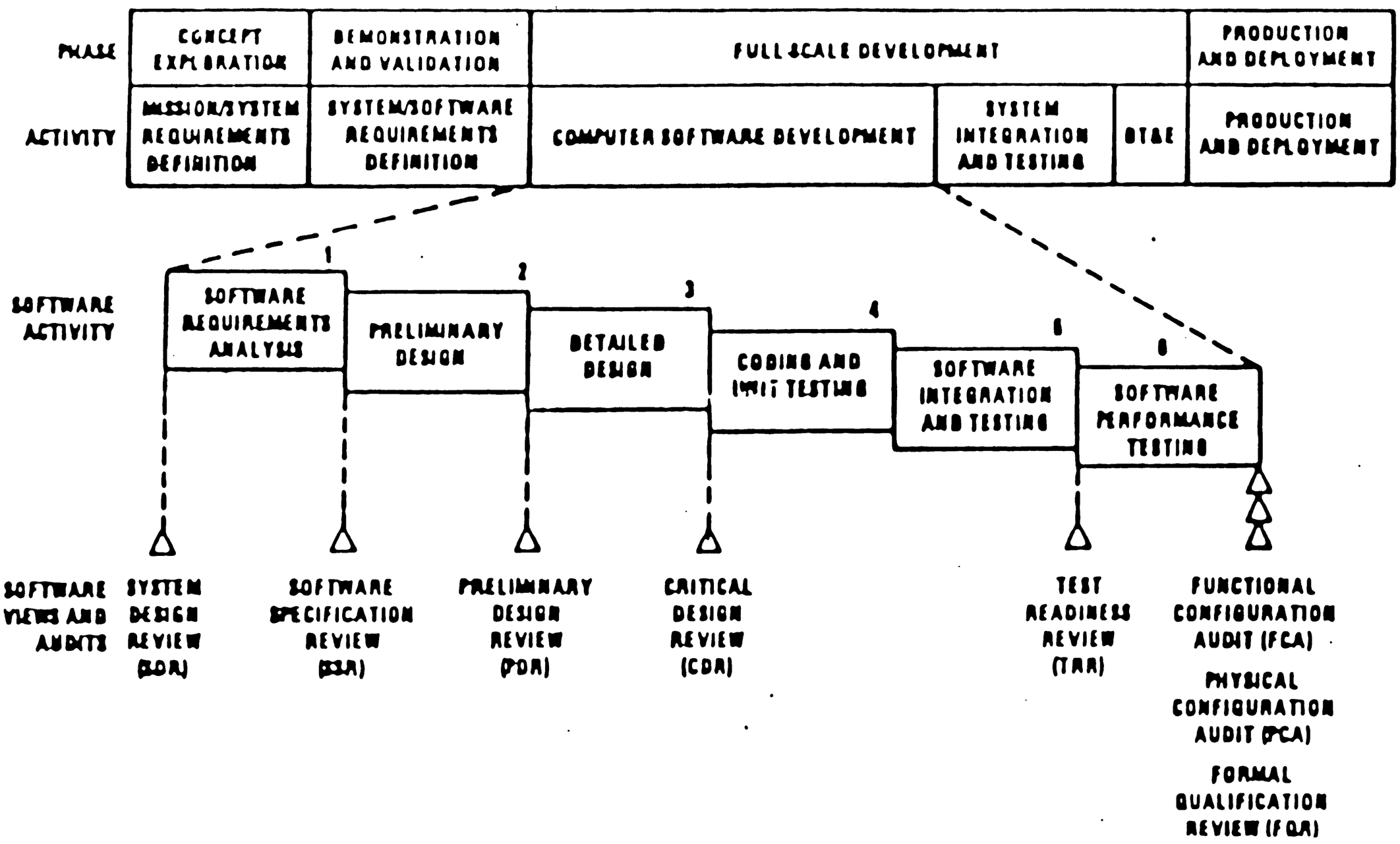

Figure 3-1: Software Life Cycle

characteristics are refined through studies, system engineering, development of preliminary equipment and prototype computer software, and test and evaluation. The objectives are to validate the choice of alternatives and to provide the basis for determining whether to proceed into the next phase.

The Full Scale Development phase is the period when the system, equipment, computer software, facilities, personnel subsystems, training, and the principal equipment and software items necessary for support are designed, fabricated, tested, and evaluated. It includes one or more major iterations of the software development cycle. The intended outputs are a system which closely approximates the production item, the documentation necessary to enter the system's Production and Deployment phase, and the test results that demonstrate that the system to be produced will meet the stated requirements. During this phase the requirements for additional software items embedded in or associated with the equipment items may be identified. These requirements may encompass firmware, test equipment, environment simulation, mission support, development support, and many other kinds of software.

The Production and Deployment phase is the combination of two overlapping periods. The production period is from the production approval until the last system item is delivered and accepted. The deployment period commences with delivery of the first operational system and terminates when the last systems are removed from operations.

3.2 Application

Software development is usually an iterative process, in which an iteration of the software development cycle may occur one or more times during each of the system life cycle phases. Successive iterations of software development usually build upon products of previous iterations. For example, design may reveal problems which lead to the revision of requirements. Testing may reveal errors in design, which in turn may lead to redesign or requirements revision.

Documentation preparation is a continuous effort covering the software life cycle. It evolves from preliminary drafts during project initiation through various reviews and changes in development. It continues through all iterations of the software life cycle with iterations caused by user feedback, changed user requirements, and changed system requirements.

The software life cycle is broken into the six phases:

- Software Requirements Analysis
- Preliminary Design
- Detailed Design
- Coding and Unit Testing
- Integration and Testing
- System Testing

3.3 Software Requirements Analysis

The purpose of the software requirements analysis is:

- The requirements are clearly understood by the developer
- The requirements are mutually agreeable to the developer
- The requirements precisely state the constraints on the desired software system
- The requirements precisely state all functions of the desired software
- The requirements provides default and error conditions whenever necessary
- The requirements provides testable criteria for the acceptance of the system
- The requirements indicates desired system qualities, their relative importance, and how they will be measured

The inherent ambiguity of natural language and the complexity of prose description makes it difficult to verify if requirements are complete and non-conflicting. The length and complexity of prose specifications also make them difficult to understand.

The inadequacies of unstructured paragraphs of natural language for requirements specifications include:

- They rely on the shared linguistic experience of those responsible for reading and writing the specification.
- They are unable to express the description of activities performed by the system and the interacting entities within the system in a clear and concise way.
- They are over-flexible as they allow related requirements to be expressed in completely different ways. The task of identifying and partitioning related requirements is more prone to error.
- They do not partition requirements effectively. As a result, the effect of changes can only be determined by examining every requirement

rather than a group of related requirements.

Unstructured natural language does not always express requirements clearly and unambiguously. However, no formal specification language has been widely accepted.

Requirement analysis is performed by having the developer write requirements documents which demonstrate the understanding of the contractor's requirements document.

3.4 Preliminary Design

The purpose of preliminary design is to develop a design approach which includes:

- Mathematical models
- Functional flows
- Data flows

During this phase various design approaches are considered, analysis and trade-off studies are performed, and design approaches selected. Preliminary design allocates software requirements to the top level computer software components (subsystems), describes the processing that takes place within each top level computer software component, and establishes the interface relationship between top level computer software components. Design of critical lower elements of the system may also be performed. The result of this phase is a documented and approved top-level design of the software. The top-level design is reviewed against the requirements prior to initiating the detailed design phase.

3.5 Detailed Design

The purpose of the detailed design is to refine the design approach so the each top level computer software component is decomposed into a complete structure of lower level computer software components (modules) and units. The detailed design approach is provided in detailed design documents and reviewed against the requirements and top-level design prior to initiating the coding phase.

3.6 Coding and Unit Testing

The purpose of coding and unit testing is to code and test each unit of code described in the detailed design documentation. Each unit of code is reviewed for compliance with the corresponding detailed design description and applicable coding standard prior to establishing internal control of the unit and releasing it for integration.

3.7 Integration and Testing

The purpose of integration and testing is to integrate and test groups of coded units. Integrations tests should be performed based on documented integration test plans, test descriptions, and test procedures. Integration test results, and system test plans, descriptions, and procedures for testing the fully implemented software are reviewed prior to the next phase of testing.

3.8 System Testing

The purpose of system testing is to test the fully implemented software system. Test results are reviewed to determine whether the software satisfies its specified requirements.

3.9 Reliability through Documentation

A major concern of the software system throughout the software life cycle is the reliability of the software. Reliability is composed of:

- Correctness of system design
- Correctness of mapping of system design to implementation
- Reliability of components making up the system
- Meeting of all specifications
- Traceability of specifications to design
- Not producing incorrect output regardless of the input
- Not allowing itself to be corrupted
- Taking meaningful and useful actions in unexpected situations
- Only completely failing when further progress is completely impossible

Unless the system documentation is accurate, reliability will not be visible.

3.10 Department of Defense Documentation

The Department of Defense currently has definitive software documentation standards. These standards reflect the user/buyer experience gained from being the largest purchaser of software with an annual ten billion dollar budget. Because of the large investment in these software products, research was done to optimize the software effort to increase reliability, understandability, and

maintainablility. Documentation was a key component during the software life cycle. Until coding begins, documentation is the specification and the design.

The first attempt, MIL-STD-490, was a standard on how to write documentation. This was not enough since it did not define all the documents needed. More software documents were established in DOD-STD-1679. As more projects were completed using this standard, further documentation standards were established in DOD-STD-2167.

These Department of Defense documentations standards provide the basis for the documentation standards of the IEEE. They also provide excellent resource material for internal standards of a software project. Some of the factors for determining the amount and detail of documentation are:

- Functional complexity
- Size
- Criticality
- Interface complexity
- Database complexity
- Integration complexity
- Complexity of security requirements
- Certification requirements
- Probability of change
- Intended end-use
- Support concept
- Development location(s)
- Schedule

Chapter 4
MIL-STD-490

MIL-STD-490 establishes the format and contents of specifications for development, procurement, production, assembly, installation, testing, or support of items, processes, and materials. Uniform practices are established for specification preparation, to ensure the inclusion of essential requirements, and to aid in the use and analysis of specification content. This common style, format, and general instructions allows the reader to know in advance what items of information to expect to quickly gain familiarity with the document.

The types of specifications available are:

- Type A - System Specification
- Type B - Development Specifications
- Type C - Product Specifications
- Type D - Process Specification
- Type E - Material Specification

Only Type A, Type B, and Type C are applicable to software systems. When a system (hardware and software) is to be implemented, a Type A specification is used. When the hardware is determined, the Type C specification is used. Both the Type A and Type C specifications describe "what to do" by the user or contracting agency. The Type B specification describes "how to do" by the developing agency.

Specifications will contain six numbered sections, and appendixes as required. These sections are titled and numbered as follows:

1. Scope

2. Applicable Documents

3. Requirements

4. Quality Assurance Provisions

5. Preparation for Delivery

6. Notes

10. Appendix

Subject matter is kept within the scope of the sections so that the same kind of requirements or information will always appear in the same section of every specification. Except for Appendixes, if a section contains no pertinent information, the section cites that this section is not applicable to this specification.

Other items addressed were: language style, capitalization and spelling, abbreviations, symbols, propriety names, commonly used words and phrases, use of "shall", "will" and "may", paragraph numbering, paragraph identification, underlining, cross references, location and preparation of figures, location and preparation of tables, foldouts, contractual and administrative requirements, definitions in specifications, references to other documents, identification of specification, and changes and revision guidelines.

Section 1 or Scope consists of a clear, concise abstract of the coverage of the specification. It may include, where necessary, information as to the use of the item other than specific detailed applications covered under "Intended Use" in Section 6 of the specification.

Section 2 or Applicable Documents consist of all and only those documents referenced in Sections 3, 4, 5 and Appendixes. These documents are further subdivided into Government Documents and Non-Government Documents and

are listed in the following order:

- Specifications
- Standards
- Drawings
- Other Publications

Section 3 or Requirements states the essential requirements and descriptions that apply to performance, design, and reliability of the item, material or process covered by the specification. This section is intended to indicate the minimum requirements that must be met to be acceptable as necessary and practicably attainable.

Section 4 or Quality Assurance Provisions includes all the examinations and tests to be performed in order to ascertain that the product or process to be developed or offered for acceptance conforms to the requirements in Sections 3 and 5 of the specification. The order of presentation of Section 4 material, insofar as practicable, follows the order of requirements as presented in Section 3 of the specification, or alternately, in the most logical order of conducting the tests listed.

Section 5 or Preparation for Delivery is the applicable requirements for preservation, packaging, and packing an item and marking of packages and containers.

Section 6 or Notes contains information of a general or explanatory nature, and no requirements appear therein. The information contained is designed to assist in determining the applicability of the of the specification. The section should include the following, as applicable, in the order listed:

- Intended Use

- Ordering Information
- Preproduction Sample
- Standard Sample
- Definitions
- Qualification Provisions
- Cross Reference of Classifications
- Miscellaneous Notes

Section 10 or Appendix is a section of provisions added at the end of the specification. An appendix may be used to append large (multi-page) data tables, plans pertinent to the submittal of the item, management plans pertinent to the subject of the specification, classified information or other information or requirement related to the subject that would normally be invoked by the specification but would, by its bulk or content, tend to degrade the usefulness of the specification. In all cases where an appendix is used, reference to the appendix is included in the body of the specification.

4.1 Type A - System Specification

The System Specification does the following:

- States the technical and mission requirements for a system as an entity
- Allocates requirements to functional areas
- Defines the interfaces between or among the functional areas

Normally, the initial version of a system specification is based on parameters developed during the concept exploration phase. This specification (initial version) is used to establish the general nature of the system that is to

be further defined during the system life cycle. The system specification is maintained current forming the future performance base for the development and production of prime items and subsystems, the performance of such items being allocated from system performance requirements.

The requirements contains the following:

- The performance and design requirements of the system
- The performance requirements related to manning, operating, maintaining, and logistically supporting the system, to the extent these requirements define or limit design of the system equipment
- The design constraints and standards necessary to assure compatibility of system hardware
- The definition of the principal interfaces between the system being specified and other systems with it must be compatible
- The functional areas of the system, and the principal interfaces between and within each functional area
- The allocation of performance to, and the specific design constraints peculiar to, each functional area
- The identification and use of major Government-furnished property to be designed into and delivered with system equipment, or to be used with other system equipment as an entity and an integral part of system capability

Unless purely descriptive by nature, requirements included are stated in quantitative physical terms which can be verified. Since a system may be either hardware, or a combination of hardware and software, the requirements fall into three categories:

- Hardware
- Software
- Combination of hardware and software

A requirement that appears to be hardware may actually be a combination of hardware and software. The type of requirement it becomes is dependent on system design. Some examples are maintainability and availability.

An example of the maintainability requirement is if the system has a hardware failure, that failure is pinpointed to the broken circuit board. The system design may use visible latches to indicate a failure. These latches may be set solely by hardware circuitry or by a software command to the hardware circuitry.

An example of the availability requirement is that the system will continue its processing if the power interruption is less than 200 milliseconds. If system design has all the random access memory (RAM) power-protected, this is solely a hardware requirement. If only part of the RAM is power-protected, then this requirement becomes both a software and hardware requirement.

4.2 Type B - Development Specification

The Development Specification states the requirements for the design or engineering development of a software system during the development period. The requirements are described in operational, functional, and mathematical language necessary to design and verify in terms of performance criteria. The specification provides the logical, detailed descriptions of performance requirements of a software system and the tests required to assure development of the software system satisfactory for the intended use.

The requirements consists of

- Program Definition of Major Functions
- Detailed Functional Requirements

- Adaptation Information

Program definition provides details imposed by interfacing equipment, timing and sequencing requirements, and the interactions of the major functions of the system. An example of this is a message from the data bus is available for only 100 milliseconds.

Detailed functional requirements contain the detail text and mathematical description of each required function. This information is broken into input, processing, output, and special processing. An example for a top level design of a data entry system is:

Input - Receive Keystroke from Keyboard Unit

Processing - Determine Meaning of Keystroke

Output - Display Keystroke Meaning on Display Unit

Special Considerations - Perform in 50 milliseconds.

Adaptation Data contains a description of data requirements with respect to system environment, system parameters, and system capabilities. Examples of items affecting adaptation data are hardware changes and constants changes for operational needs.

4.3 Type C - Product Specification

The product specification is applicable to any item below the system level. For a software system, this is the production of the computer programs and the implementing media, i.e. magnetic tape, disc. etc. It does not cover the details requirements for material or the manufacture of the implementing medium. A translation of the performance requirements into programming terminology and quality assurance procedures is provided.

The requirements contain a comprehensive description of the structure and

function of the software system as a whole. It consists of the following:

- Function Allocation Description
- Functional Description
- Storage Allocation
- Computer Program Functional Flow

Function Allocation Description identifies and describes the allocation of functions and tasks to be performed by the individual subprograms.

Functional Description describes a general summary of inputs, outputs, and functions to be performed for each subprogram and common subroutine.

Storage allocation describes the approximations of the allocation of memory storage to subprograms, the executive routine, subroutines, and the data base. The timing, sequencing requirements, and equipment constraints used in determining the allocation is described.

Computer Program Functional Flow shows the general system flow of both data and control. If the system is to operate in more than one mode, each mode is clearly distinguished. Program interrupts, control logic involved in referencing each subprogram, and special control features that affect the design of the control logic but are not part of the normal operational are discussed.

Chapter 5
DOD-STD-1679

MIL-STD-490 addressed the issues of the requirements and design documents. Other formal documentation types were needed. DOD-STD-1679 established uniform requirements for software development. Documentation types were developed based on the phases of the software life cycle and the needs of particular groups of users.

5.1 Documentation Types

The main goal of these documents types is to allow project control for development, maintenance and reliability considerations.

5.1.1 Management Documentation

At the beginning of the software life cycle, management determines how the software project will be planned. Clear lines of authority and responsibility are established. Coordination of the software project is through a schedule of events and milestones. These plans are documented in the Software Development Plan, The Software Quality Assurance Plan, and the Software Configuration Plan. Some variations of these plans would be to include the Software Quality Assurance Plan as part of the Software Development Plan, to include the Software Configuration Management Plan as part of the Software Development Plan, or include the Software Management Plan as part of the System Configuration Management Plan.

5.1.2 Requirements Documentation

To determine whether the requirements were understood by the developer, the Program Performance Specification and the Interface Design Specification documents were established.

The Program Performance Specification determines the detailed performance requirements of the software from documentation provided by the user which is considered the user's baseline. This information may be augmented by studies, analyses, visits to the user, and surveys as necessary. However, if the augmented information is in conflict with the user's baseline, the documented user's baseline requirements take precedence. The rigid format of the document requires a thorough understanding of the requirements. Incomplete or conflicting requirements becomes apparent as the document is produced enabling the user to evaluate and implement changes in the requirements.

The Interface Design Specification contains the data of the interfacing of the software system with other computer programs or systems. The level of detail required to adequately produce the document guarantees that enough information is available for the eventual design of the software system.

Both the Program Performance Specification and the Interface Design Specification are reviewed by the users in a formal design review. Each requirement specified in the user's baseline must be accounted for in the Program Performance Specification. Until these documents are formally accepted by the user, proceeding with the design documentation involves a risk to the developer of unnecessary or invalid work. The formal acceptance of these requirements establishes the developer's baseline that the design documentation is based on. These requirements documents are complete that reference to the

user's documents are no longer necessary and constitute the developer's baseline.

5.1.3 Design Documentation

DOD-STD-1679 has a single design phase of the software life cycle rather than a separate preliminary design phase and detailed design phase. The documents produced were the Program Design Specification, the Data Base Design Specification, and the Program Description Document.

The Program Design Specification is the top level design of the software system. The Program Design Specification demonstrates that the requirements can be broken into functions or subprograms. The naming and programming conventions are established.

The Data Base Design Document contains the detailed data descriptions for the data which is used for the communication between the various functions or subprograms as established in the Program Design Specification.

The Program Description Document is broken into separate volumes for each function or subprogram. The Program Description Document further shows the breakdown of the function or subprogram into modules and units that satisfies the performance requirements. The Program Description Document contains the detailed data descriptions for the data which is globally used by more than one module or unit.

The Program Design Specification, the Data Base Design Specification, and the Program Description Document are reviewed by the user in a formal design review. Each requirement specified in the developer's baseline must be accounted for in the Program Design Specification. After the validation of the design, the developer gives these documents to the computer programmers to create the source code.

5.1.4 Operations Documentation

Operations documentation contains the procedures for the loading, initialing, and operating the software system. The documents produced are the Operator's Manual and the System Operator's Manual.

The Operator's Manual is concerned with the non-functional operations. The Operator's Manual defines minimal processor and peripheral equipment requirements, equipment set-up for system operation, software set-up, special parameter entering requirements, standby/operate procedures, monitoring procedures, and recovery procedures.

The System Operator's Manual is concerned with the functional operations. The System Operator's Manual defines the individual and station functions, the coordinated stations procedures, all user aspects, modes and procedures to perform the system operation, the function of every control button, switch, readout, and display affected by or affecting the system, and all constraints imposed on the operator.

5.1.5 Documentation of Changes

For changes in the user's baseline to be reflected in the requirements documentation, the user must issue Software Change Proposals and Software Enhancement Proposals. These new or modified requirements become part of the new user's baseline. The importance of formalized requirements changes become apparent in management schedule and budget considerations caused by the impact upon the software life cycle. What may appear as a deletions of requirements, may still have a negative effect on budget when documentation considerations are evaluated.

For changes not related to the requirements, the Software Trouble Report

is used. These changes occur from problems in the documentation and operation of the software system.

The earlier a change is implemented in the software life cycle, the less costly it becomes. For example, if the change is made during the requirements phase of the software life cycle, then only the requirements documentation is affected. If the change is made after the software is coded, the requirements document, the design documentation, the test documentation, and the code may be affected. Configuration control procedures are followed for the implementation of changes to the software documentation and software system.

5.1.6 Support Documentation

Support documentation contains the information necessary to create the software system using the source code. The document produced is the Program Package Document. The Program Package Document contains all the program materials used to produce, maintain, and update the software system.

This documentation type needs more information concerning the hardware and software procedures for compiling, debugging and installing. These deficiencies are to be addressed in DOD-STD-2167.

5.1.7 Test Documentation

The quality assurance portion of the MIL-STD-490 documents gave rise to a series of test documents. Since reliability is an important consideration of the software system, testing becomes a significant part of the software development. The documents produced are the Computer Test Plan, the Computer Test Specifications, the Computer Test Procedures and the Computer Test Report.

The Computer Test Plan defines the total scope of testing. Each

requirement in the Program Performance Specification must be accounted for in the Computer Test Plan. The testing validates these requirements and is independent of the design.

The Computer Test Specifications is prepared for each test specified in the Computer Test Plan. The Computer Test Specifications are delineated as the System Test Specification and the Function Test Specification. The requirements that can not be tested by the operation of the software system are specified in the Function Test Specification. All other requirements are specified in the System Test Specification. For example, if the system must maintain a 50% memory reserve, this requirement cannot be tested by operating the software system. Therefore this requirement is tested in the Function Test Specification.

The Computer Test Procedures present detailed instructions for test setup, execution, and evaluation for each test specified in the Computer Test Procedures. The Computer Test Report documents the result of the tests generated by the Computer Test Procedures. Any discrepancies in the expected result and the actual result are described.

5.2 Software Development Plan

The Software Development Plan describes the comprehensive plan for the management of the development of effort for the software system. The Software Development Plan provides the means for the development effort to:

- Coordinate schedules
- Control resources
- Initiate actions
- Monitor Progress

The Software Development Plan contains:

- Project development organization and their titles, duties, and relationship to other organizational entities

- Program design approach of methods and techniques to ensure the software system design satisfies all technical, operational, and performance requirements.

- Implementation approach to conform with DOD-STD-1679 development techniques as well as special coding techniques and production methods

- Resource utilization control of memory usage, mass storage allocation and usage, system response times, central processor usage, and I/O utilization

- Certification test philosophy and plans for each phase of the software life cycle such as unit and module tests, program debug, integration tests, and acceptance testing

- Program support center facilities capabilities, projected usage, manning, special tools, and arrangements for post-delivery maintenance

- Quality assurance with the information required of the Software Quality Assurance Plan or, if separate, a reference to the Software Quality Assurance Plan

- Programming standards and conventions that apply to the design and production of the software system

- Configuration management with the information required of the Software Configuration Management Plan, or, if separate, a reference to the Software Configuration Management Plan or System Configuration Management Plan.

- Government furnished equipment and services to be used for the production or test of the software system

- Software integration approach, plan, and organization to achieve software integration in all system elements in both software and hardware

- Risk areas in cost, schedule, and technological risks

- Schedules and milestones in the software development schedule,

including status reports, reviews, and audits

- Resource allocation of personnel, material, and financial resources

Graphic representations of project organization, schedule and milestones, and resource allocation could present a clearer representation of these areas. A minimal amount of narrative would then be needed for full understanding.

5.3 Software Quality Assurance Plan

The Software Quality Assurance Plan describes the organization and procedures to assure the software written complies with the requirements. The Software Quality Assurance Plan is oriented toward the design and production of software that is effective and reliable, and planned and developed in consonance with other adminstrative and technical programs.

The Software Quality Assurance Plan contains:

- The software quality assurance organization of authority and responsibility
- The software quality assurance procedures with rules, techniques, and methodologies for:
    - Software development management
    - Software configuration management
    - Software specification, design, and production
    - Software testing
    - Corrective actions on deficiencies
- Plan implementation of specific tasks, responsibilities, and resources
- Reporting and control system to:
    - Monitor overall development status

- Base decisions on quality control data
- Disclose inadequacies, discrepancies, and deficiencies as well as proposed improvement
- Allow rapid and effective corrective actions

5.4 Software Configuration Management Plan

The Software Configuration Management Plan describes how to assure proper configuration identification, configuration control, and configuration status accounting.

The Software Configuration Management Plan contains:

- The configuration management organization with the responsibilities of the members and their relationship to the overall organization and the policies and directives relating to configuration management
- The configuration identification policies and procedures for identifying the documentation of the functional and physical characteristics of configuration items
- The configuration control for changes to the baseline
- The software configuration authentication of the software to the documentation
- The configuration status accounting procedures for collecting, recording, processing, and maintaining data
- The interface management for coordinating efforts to ensure compatibility with other systems
- The configuration audits plans, procedures, schedule, quality assurance measures, and format
- The control over subcontractors and vendors
- The configuration management milestones for the software life cycle phases

5.5 Program Performance Specification

The Program Performance Specification describes in detail all the operational and functional requirements necessary to design, test, and maintain the software system. It provides the logical, detail description of the performance requirements of the system software.

The Program Performance Specification contains:

- System description - All the components in the system which affect the software or the software performance requirements are described. How the software interfaces to perform the required system functions with the other components is determined.
    - Peripheral equipment identification - all the equipment which the software will interface by then physical characteristics and type of interface
    - Interface identification - all other computer programs or systems with which the software will interface
- Functional description - The major functions and the functional relationships of the software with the interfaces are analyzed. The performance of each function supported by the software, it purpose, and functional design is described.
    - Equipment description - The requirements imposed on the software by each interfacing equipment, the purpose of the equipment, and the use of the options and controls
    - Diagrams - Diagrams of equipment and software relationships with internal and external data flow
    - Intersystem interface - The requirements imposed on the software by each interface with other systems and equipment, the purpose of the interface, the data to be exchanged, and the data quantity, frequency, rate, format, content, scaling requirements, and conventions of the data
- Detail functional requirements - Each function is described in detailed text with logical and mathematical descriptions
    - Inputs - Both internal and external with their source, format,

method of reception, quantity, timing, range, and scaling

- Processing - Textual and, as appropriate, mathematical descriptions of the processing requirements of each function, including functional parameters and geometric diagrams

- Outputs - Both internal and external with their method of transmission and timing, meaning, format, quantity, destinations, range, and scaling

- Special Requirements - Requirements imposed by higher-level constraints or by exigencies of the function

- Adaptation - Parameters which reflect the system environment, limits, and capacities and which can be defined symbolically to subsequently be modified without altering the software logic

- Testing requirements of the functions for all levels

5.6 Interface Design Specification

The Interface Design Specification establishes the requirements for any system which utilizes direct digital processor interfaces. The Interface Design Specification provides a detailed logical description of:

- All data units

- All messages

- Use of all control signals for defining interdigital processor communication conventions

The Interface Design Specifications contains for the software system to other systems interface signals and the other systems to the software system interface signals:

- The summary cross reference with an alphabetical list of all signals with page number cross references for the signal in the:

  - Signal definition list

  - Narrative signal flow table

- Data unit description
- Message description

- Signal Definition List which, for each signal, provides a detailed description of the:
  - Initiation
  - Use
  - Effects
- Narrative Signal Flow Table in logical groupings for all signal in a specified interface, arranged and numbered in a sequence of normal expected occurrences of events
- Interdigital Processor Communications consisting of:
  - Communication control signals
  - Enable and disable procedures
  - Unique input/output requirements
  - Control word formats
  - Communication responsibilities
  - Data transfer technique and sequence
  - Error conditions
  - Data transfer requirements
  - Data transfer rate
  - Whether Periodic/aperiodic
  - Minimal interval between transfers
  - Interface testing techniques
- Data Unit Descriptions consisting of:
  - Positions of the fields

- Use of field
- Name of field
- Beginning and ending bit positions
- Bit positions not used and available
- Scaling and units
- Scaling convention
- On or set conditions

- Message Descriptions consisting of:
  - Positions of the fields
  - Data unit source and use
  - Word positions of the words of the message
  - Name of field
  - Beginning and ending bit positions
  - Bit positions not used and available
  - Scaling and units
  - Scaling convention
  - On or set conditions

5.7 Program Design Specification

The Program Design Specification document is the design description of the software system. It is based upon the performance requirements defined in the Program Performance Specification. The Interface Design Specification is also accommodated. It specifies the programming approach for implementing the computer program and defines the program architecture for further program

decomposition.

The Program Design Specification contains:

- Functional allocation to be performed by the subprogram or modules with a table showing which subprogram or module satisfies which requirement from the Program Performance Specification

- Functional description for each subprogram or module containing:

    - Inputs with specific data required and its sources

    - Processing

    - Outputs with intended destination

    - Other functions performed by the subprogram

    - Common subroutines

    - The interface between the executive control routines and the subprogram with scheduling requirements and conditions

- Subprogram storage and processing allocation of memory storage and processing time

- Program functional flow of both system data and execution control with diagrams and containing:

    - Program interrupt control with the source, purpose, type, predicted rate, of occurrence, and required control response

    - Subprogram reference control with the control logic, assignment of priorities, and permissible cycle times

    - Special control features which affect the design of the control logic

- Programming guidelines containing:

    - The programming language and its supporting system

    - Manuals for the programming language and supporting system

    - Mnemonic labeling conventions

- Program version identification

- Quality Assurance

5.8 Data Base Design Document

The Data Base Design Documents provides a complete detailed description of all common data items necessary to carry out the functions of the software system. Common data is that data required by two or more subprograms. Common data includes

- Tables
- Variables
- Constants
- Flags
- Indexes

The detailed information of each table contains:

- Table name
- Purpose and type
- Size and indexing procedure
- Subitems
  - Field name
  - Purpose and type
  - Size
  - Binary point
  - Range of values and initial conditions
  - Static or dynamic

- Structure and bit layout

The detailed information of each variable contains:

- Variable name
- Purpose and type
- Size
- Binary point
- Range of values
- Static or dynamic
- Structure and bit layout

The detailed information of each constant contains:

- Constant name
- Purpose
- Initial condition
- Structure and bit layout

The detailed information of each flag contains:

- Flag name
- Purpose
- Initial condition
- Structure and bit layout

The detailed information of each index contains:

- Index name
- Purpose

The Data Base Design Document also contains a matrix of all the data

base items with the referencing subprograms. The matrix indicates whether the data item was set, used, or both.

The Data Base Design Document is based on the Program Performance Specification and is developed in consonance with the Program Design Specification, and concurrently with the Program Description Document.

5.9 Program Description Document

The Program Description Document represents the further detailing of the software system into individual operations to be performed by the software system. The Program Description Document provides a complete technical description of all software system subprogram functions, structures, operation environments, operating constraints, data base organization, source and object code listings, and diagrammatic/narrative flows. Each subprogram or function is described in its own volume with referenced appendixes as software printout listings. The Program Description Document is oriented to programming logic and programmer's language. The aim is to design and completely define the basic subprogram logic and program procedures for each application subprogram and each system control subroutine. The Program Description Document is generated from the Program Design Specification.

The Program Description Document contains:

- A detailed subprogram description with its processing capability using the same mnemonics which will appear in the source code
- The subprogram data base in the same detail as found in the Data Design Document for each:
    - Table
    - Variable

- Constant
- Flag
- Index

- A common data base reference to local and common data base items and the location of each reference
- Input/output formats processed by the subprogram
- Required system library subroutines and volume the subroutine is described
- Conditions for initiation of the subprogram
- Subprogram limitations including:
  - Timing requirements
  - Limitations of algorithm and formulas used
  - Design limits of input and output data
  - Associated error condition sensing provided
  - The error and reasonableness checks that are programmed into the various routines
- Interface description with other subprograms and system or executive which it interfaces

5.10 Operator's Manual

The Operator's Manual presents procedures for prestandby/operate, monitoring, and recovery of the software system. It is limited to instructions for preparing and maintaining the software system in the required state of capability.

The Operator's Manual contains:

- Operational Environment to allow operation of the software system which include:

- Equipment requirements
- Program materials
- Supporting Documentation

- Prestandby Procedures for preparation and setup prior to the software system operation which include:
  - Equipment Setup
  - Program Setup
  - Adaptation Data Setup
- Standby/Operate Procedures which start the system after prestandby procedures are complete
- Monitoring Procedures for trouble and malfunction indications
- Recovery Procedures for restating the software system after an abort or interruption

5.11 System Operator's Manual

The System Operator's Manual is intended to be the sole reference that is required for the individual operator.

The System Operator's Manual contains:

- Instructions from the system control panel for:
  - Program loading
  - Initiation
  - Modification
  - Operation
  - Termination
- Communication links descriptions and the requirements for link operation

- Instructions for operating the system keyset devices with figures for clarity
- Instructions for the basic operation of the consoles with figures for clarity
- The normal sequence of functions the operator must perform
- The random conditions requiring an operator's response
- The corrective response in response to illegal actions that the operator must perform

The operational procedure is written with the following guidelines:

- The procedures agree exactly with the performance requirements
- If the procedure includes several modes of operation, each mode is clearly defined and presented individually
- The material is presented in a step-by-step manner
- Procedural steps are expressed precisely in the imperative mode, using system nomenclature
- Each step is listed as an individual item
- Each step is listed in the order in which it is to be performed
- Each step is precise and unambiguous
- Supporting illustrations is comprised of view of the equipment with each control and indicator identified
- Data entry number and name is to be contained within each appropriate mode
- The difference in mode operations while using any reduced capability modes or program is described

5.12 Software Change/Software Enhancement Proposal

The Software Change Proposal and the Software Enhancement Proposal are used to request changes to established baselines. A Software Change Proposal is used in cases where the change is an addition, deletion, or modification of a capability which would be evident to the user/operator of the system and/or affect the compatibility of the software with previous unchanged versions. A Software Enhancement Proposal is used in cases where the change will alter the component operation but has no externally visible effect on the operation of the system nor alter the compatibility of the software with previous versions. The Software Change Proposal/Software Enhancement Proposal is the vehicle for analyzing, approving, and acting on the propsed change and must be complete in detail and supporting documentation to fulfill this function.

The Software Change Proposal/Software Enhancement Proposal contains:

- System/Project name
- Date Prepared
- Identification number for the proposal
- Title of the proposal
- Originator of the proposal
- Component affected
- Description of problem or need
- Alternative/Impacts if not approved
- Baselines affected
- Documentation affected
- Other systems affected

- Effect on user
- Net effect on system resources
- Developmental requirements
- Most effective point in the development recommended
- Accomplishment of the proposal
- Supersedes or replaces other proposals
- Cost, schedule or interface impact
- Configuration control information

5.13 Software Trouble Report

The Software Trouble Report shows all essential data on each software problem detected. Software problems are classified by category as follows:

- Software Trouble - The software does not operate according to supporting documentation and the documentation is correct.

- Documentation Trouble - The software does not operate according to supporting documentation but the software operation is correct.

- Design Trouble - The software operates according to the supporting documentation but a design deficiency exists. The design deficiency may not always result in a directly observable operational symptom, but possesses the potential of causing trouble.

The Software Trouble Report is the basic input to the Software Quality Assurance program during the test and acceptance phase of the development effort.

The Software Trouble Report contains:

- Date Prepared
- Category of Software Trouble
- Priority of the Severity of the Software Trouble

- Number for control purposes
- Title describing the problem
- Official designation of the problem
- Document affected
- Unit/Test which trouble was detected
- Program identification number
- Reference document
- Function affected
- Responsible module
- Test step being executed when the trouble was discovered
- Originator, title, and phone
- Run time elapsed until trouble occurred
- Simulation used for operational conditions
- Linking for intersystem communications
- Configuration in Memory
- Problem duplicability
- Data dump
- System status/environment
- Trouble description
- Stop data available
- Testing performed
- Software Quality Assurance sign-off
- Current status of the trouble

5.14 Computer Test Plan

The Computer Test Plan defines the total scope of the testing to be performed. It identifies the particular level of testing and describes its contributing role for ensuring the reliability and certified acceptance of the software system. The Computer Test Plan is used to review and ensure that the software system is effectively meeting the technical requirements and the system integration is ensured.

The Computer Test Plan contains:

- The test requirements for each level of testing addressing each:
  - Input
  - Output
  - Operator actions
  - Any other requirements deemed necessary for evaluation
- The test management requirements
- Personnel requirements
- Hardware requirements
- Supporting software requirements
- Schedule
- Quality assurance

5.15 Computer Test Specifications

The Computer Test Specification is prepared for each test specified in the corresponding Computer Test Plan, normally one for each subprogram or specified function and one for the pertinent test. A Computer Test Specification is prepared before, and is the basis of the development of the Computer Test Procedures.

The Computer Test Specifications are broken into two parts, the System Test Specification and the Function Test Specification. The System Test Specification test the requirements of the Program Performance Specification by the normal operation of the software system. The Function Test Specification test the requirements of the Program Performance Specification that cannot be verified by normal operation of the system.

Both parts of the Computer Test Specifications contain:

- Test management requirements
- Personnel requirements
- Hardware requirements
- Support software requirements
- Schedule
- Quality assurance

These topics are only addressed if there was a change in the Computer Test Plan.

The System Test Specification contains:

- Test inputs of upper and lower limits with methods of generation
- Required accuracies

- Expected output values and, if the output is a range, the upper and lower limits of the range
- Data collection methods type of recording, frequency, and duration
- Interface of hardware/man-machine, input/output system interface, the destination, and result intended
- Method of data exchange for the data flow
- Timing requirements for the input/output to the various subsystems
- Degradation of maximum time limit for continuous operation
- Casualty recovery techniques
- Display requirements that are obtained
- Communications requirements both internal and external

The Function Test Specification contains:

- Pretest inputs that are used to replace the dynamic values necessary for the operation of the function but not the evaluation of the function
- Test inputs of upper and lower limits with methods of generation
- Required accuracies
- Expected output values and if the output if a range, the upper and lower limits of the range
- Data collection methods type of recording, frequency, and duration

The Computer Test Specifications can be reviewed to ensure that the overall objectives are fulfilled and the primary features of the software system are evaluated, e.g., that the system or functional characteristic or particular modes of operation are tested.

5.16 Computer Test Procedures

The Computer Test Procedures provide detailed instructions for the execution and for the evaluation of the results for each level of testing specified. The Computer Test Procedures provide the quantitative results which are later extracted from the tests themselves. The Computer Test Procedures are developed from the System Computer Test Specifications, the Function Computer Test Specification, and relevant design documents.

These Computer Test Procedures contain:

- Materials needed to run test
- Personnel necessary and required experience
- Setup of the computer hardware
- Power-on of the hardware
- Loading of the software
- Step by step operating instructions
- Recording instructions
- Deviations from the Computer Test Specification

The Computer Test Specification also contains:

- Test management requirements
- Personnel requirements
- Hardware requirements
- Support software requirements
- Schedule
- Quality assurance

These topics are only addressed if there was a change in the Computer

Test Plan.

5.17 Computer Program Test Report

The Computer Test Report is the vehicle by which the results of the validation of the software system are documented. The Computer Test Report is used to describe, define, and evaluate discrepancies between the intended system requirements and design and the program capability as produced by the code.

The Computer Test Report contains:

- The test criteria which are:
    - Range of data and parameter values
    - Accuracy requirements
    - Program/subprogram/module capabilities
    - Data rates from minimum to maximum
    - Duration of test in time or number of events
    - Definition of error
    - Definition of failures
- Test results based on the data collected with any discrepancies with its impact and validity noted
- Evaluation criteria with range of data and parameter tested and identification of any functional deficiencies, limitations, or constraints detected during the test
- Test evaluation that the functional capability was demonstrated
- Recommendations of improvements in design or operation as determined during the test period

5.18 Software Program Package Document

The Software Program Package Document consists of the code and any data which are necessary to run the software properly. This data may include, but not limited to:

- Adaptation Data
- Data File Contents
- Set-up Data
- Program Parameter Values

The Software Program Package Document contains:

- Software source code suitable for assembly or compilation
- Complete object form suitable for loading and execution
- Source program listing
- Error-free source/object listing
- Cross reference listing of each mnemonic and statement that references that mnemonic
- Miscellaneous listings used in the program reproduction

The program material items are used to produce, maintain, and update the software system.

5.19 Documentation in the Source Code

Thes source code is documented different ways. Each program, subprogram, module, and unit has at the beginning of the executable code an abstract. This textual description contains:

- Inputs with the allowed and expected range of values for all inputs
- Outputs with the allowed and expected range of values for all

outputs

- Function or task
- List of other components called
- The original and updating programmer names
- Dates and reasons for all changes
- A description of any transportability constraints

In addition to a general explanations, to assist understanding, precise references to the appropriate statements labels and data-names are included in each module and unit.

Comments are used throughout the software to facilitate software comprehension. Each source statement is to be self-defined or defined by a comment phrase to a level of understandable by a person knowledgeable in software but not associated with the original development effort. Logical groups of comments may be included in a single comment line.

Other methods to improve readability and clarity include:

- Software structural indentation
- Paragraphing
- Blocking by blank lines

Chapter 6
DOD-STD-2167

DOD-STD-1679 documents are all included in the DOD-STD-2167. More documents were added, either by splitting DOD-STD-1679 documents into separate documents, or adding documentations requirements which were lacking. More detail is given concerning paragraph numbering and cover page layouts.

6.1 Documentation Types

The documentation for management documentation, requirements documentation, design documentation, operations documentation, and the documentation of changes remained essentially the same.

The major type of documentation that was changed was the support documentation. A new documentation type was added for diagnostics.

6.1.1 Support Documentation

Support documentation contains the information necessary to create the software system using the source code. The documents produced are the Version Description Document, the Software Programmer's Manual, Firmware Support Manual, and the Computer Resources Integrated Support Document.

Since software may have many different baselines, a Version Description Document is used to identify the version of the software system concerning contents and operations.

The Software Programmer's Manual contains the information necessary for programming the software on the target computer. The Firmware Support Manual contains the information necessary for installing firmware devices. The Computer Resources Integrated Support Document contains the software life

support information.

6.1.2 Diagnostic Documentation

Since a software program may be written to throughly test the software system, this program may be adapted as a diagnostic tool. Diagnostic documentation contains the information necessary to identify a software malfunction. The document produced is the Computer System Diagnostic Manual.

6.2 System/Segment Specification

The System/Segment Specification is similar to the MIL-STD-490 Type A - System Specification. The main difference is the requirements are either designated as a hardware or software requirement.

6.3 Software Development Plan

The Software Development Plan is similar to the DOD-STD-1679 Software Development Plan. A section has been added concerning software standards and procedures. This may cite a Software Standards and Procedures Manual or include the information here.

6.4 Software Configuration Management Plan

The Software Configuration Plan is similar to the DOD-STD-1679 Software Configuration Plan.

6.5 Software Standards and Procedures Manual

The Software Standards and Procedure Manual is the information that was formerly found is in the DOD-STD-1679 Program Design Specification concerning programming guidelines.

6.6 Software Quality Evaluation Plan

The Software Quality Assurance Plan is similar to the DOD-STD-1679 Software Quality Assurance Plan.

6.7 Software Requirements Specification

The Software Requirements Specifications is similar to the DOD-STD-1679 Program Performance Specification.

6.8 Interface Requirements Specifications

The Interface Requirements Specifications is similar to the DOD-STD-1679 Interface Design Document.

6.9 Software Top Level Design Document

The Software Top Level Design Document is similar to the DOD-STD-1679 Program Design Specification without the programming guidelines.

6.10 Software Detail Design Document

The Software Detail Design Document is similar to the design section of the DOD-STD-1679 Program Description Document.

6.11 Interface Design Document

The Interface Design Document is similar to the DOD-STD-1679 Data Base Design Document.

6.12 Data Base Design Document

The Data Base Design Document is similar to the data base section of the DOD-STD-1679 Program Description Document.

6.13 Software Product Specification

The Software Product Specification is similar to the DOD-STD-1679 Program Description Document.

6.14 Version Description Document

The Version Description Document is used to identify new and interim versions of the software system. This identification contains:

- Inventory of materials to be released
- Inventory of the contents of the software system
- Changes installed
- Adaptation data
- Interface compatibility
- Bibliography of reference documents
- Operational Description
- Installation Instructions
- Possible problems and known errors

6.15 Software Test Plan

The Software Test Plan is similar to the DOD-STD-1679 Computer Test Plan.

6.16 Software Test Description

The Software Test Description is similar to the DOD-STD-1679 Computer Test Specifications.

6.17 Software Test Procedures

The Software Test Procedures is similar to the DOD-STD-1679 Computer Test Procedures.

6.18 Software Test Report

The Software Test Report is similar to the DOD-STD-1679 Computer Test Reputer.

6.19 Computer Support Operator's Manual

The Computer Support Operator's Manual is similar to the DOD-STD-1679 System Operator's Manual.

6.20 Software User's Manual

The Software User's Manual is similar to the DOD-STD-1679 Operator's Manual.

6.21 Computer Support Diagnostic Manual

The Computer Support Diagnostic Manual contains the information necessary to identify a computer system malfunction and instructions to run the diagnostics. This information include:

- Identification of all support hardware, software, and procedures to perform system diagnosis
- A description of each diagnostic tool available for the system
- A description of each diagnostic test available on the diagnostic tools, including:
    - The purpose of each test
    - The procedures for executing the test
    - Additional hardware, software, or firmware for executing the test
    - All diagnostic messages

6.22 Software Programmer's Manual

The Software Programmer's Manual contains the information to facilitate programming or reprogramming software for the target computer. This information includes:

- Equipment configuration
- Operational characteristics, capabilities, and limitations
- Compilation and assembly instructions
- Programming features
- Program instructions
- I/O control features
- Examples of programming technique

- Special features
- Error detection and diagnostic features

6.23 Firmware Support Manual

The Firmware Support Manual contains the information necessary to modify or replace the read-only memory, programmable read-only memory, and other such firmware components of the system. This information includes:

- Description of the firmware components
- Installation and repair procedures
- Security implications
- Operational and environment limitations
- Hardware needed for programming firmware devices
- Software needed for programming firmware devices
- Procedures for programming firmware devices
- Vendor information

6.24 Operational Concept Document

The Operational Concept Document contains:

- The mission of the software system
- The operational environment
- The support environment
- The functions and characteristics of the software system within the overall system

6.25 Computer Resources Integrated Support Document

The Computer Resources Integrated Support Document contains the information that is required to perform life cycle support of the software system. This information includes:

- The support environment describing required
  - Support software
  - Equipment
  - Facilities
  - Personnel
- Support operations, describing:
  - General usage instructions describing
    - Initiation
    - General Operation
    - Monitoring operations of the support environment
  - Administration
  - Software modification
  - Software integration and testing
  - System and software generation
  - Software quality evaluation
  - Corrective action system
  - Configuration management
  - Simulation
  - Emulation

- Reproduction
- Operational distributions

- Training plans and provisions
- Predicted level of change to software system in the support environment

6.26 Engineering Change Proposal

The Engineering Change Proposal is similar to the DOD-STD-1679 Software Change Proposal.

6.27 Specification Change Notice

The Specification Change notice will describe changes to the baseline and accompanies an Engineering Change Proposal.

Chapter 7
Conclusion

Writing good documentation is not easy, nor is it a single stage process. Documents must be written, read, criticized, and then rewritten, and the process should continue until satisfactory documents are produced.

7.1 General Aspects of Good Documentation

While documentation may be technically complete, certain considerations can enhance the information presented. In summary they are:

- Who - understand the user's thought process
- Simplicity - keep it simple
- Examples - use plenty of meaningful examples
- Details - sink the details
- Retrieval - provide easy access
- Perspective - keep the user point of view at all times
- Appearance - it must look nice to read well.

7.2 Writing Style Guidelines

Some writing style guidelines are:

- Active tense rather than passive tense should be used.
- Long sentences which represent a number of different facts should not be used. It is better to use a number of shorter sentences.
- Previously presented information should not be referred to only by a reference number. What the reference covered should also be stated.
- Facts should be itemized whenever possible rather than present them in a sentence.

- If a description is complex, it should be repeated by presenting two or more differently phrased descriptions of the same thing.

- A sentence should not be verbose. A concept should be stated in as few words as possible without losing the meaning.

- Terminology should be precise, and if necessary, the terms may need to be defined before they are used.

- Paragraphs should be short. As a general rule, no paragraph should be made up of more than seven sentences.

- Headings and Subheadings should be used.

- Grammatically correct constructs should be used.

- Words should be spelled correctly.

7.3 Conclusion

The key to a large software system control is the software system comprehension. Comprehension of the software system and its parts requires

- Knowledge of total system objectives

- The partition of requirements into individual capabilities or functions

- The mapping of requirements onto a system structure and its structural elements

- The algorithm used in implementing the system and its subsystem at various levels

Clearly then a large software system must be accompanied by documentation containing this information. Moreover, the documentation must be readily accessible according to the particular needs of the inquirer. And the documentation must remain correct and complete as it keeps pace with the changing system.

References

Andriole, Stephen J. *Software Validation, Verification, Testing, and Documentation.* Princeton, NJ: Petrocelli Books, 1986.

Belady, L. A. and Lehman, M. M. Characteristics of Large Systems. In Wegner, Peter (Ed.), *Research Directions in Software Technology.* Massachusetts Institute of Technology, 1980.

DOD. *Military Standard Specification Practices* (DOD-STD-490 ed.). Author, 30 October 1968.

Gilbert, Philip. *Software Design and Development.* Chicago, Ill.: Science Research Associates, Inc., 1983.

Mecham, Douglas. Writing A User Guide. In Warren, Jim (Ed.), *Conference Proceedings, The Second West Cost Computer Faire.* Palo Alto, Cal.: Computer Faire, March 1978.

Navy. *Computer Program Test Plan* (DI-T-2142 ed.). Author, 1983.

Navy. *Computer Program Test Procedures* (DI-T-2144 ed.). Author, 1983.

Navy. *Computer Program Test Report* (DI-T-2156 ed.). Author, 1983.

Navy. *Computer Program Test Specification* (DI-T-2143 ed.). Author, 1983.

Navy. *Computer Resources Integration Support Document* (DI-MCCR-80024 ed.). Author, 1985.

Navy. *Computer Software Trouble Report* (DI-E-2178 ed.). Author, 1983.

Navy. *Computer Support Diagnostic Manual* (DI-MCCR-80020 ed.). Author, 1985.

Navy. *Computer Support Operator's Manual* (DI-MCCR-80018 ed.). Author, 1985.

Navy. *Data Base Design Document* (DI-MCCR-80028 ed.). Author, 1985.

Navy. *Data Base Design Specification* (DI-S-2140 ed.). Author, 1983.

Navy. *Department of Defense Standard, Software Development* (DOD-STD-1679A ed.). Author, 22 October 1983.

Navy. *Firmware Support Manual* (DI-MCCR-80022 ed.). Author, 1985.

Navy. *Interface Design Document* (DI-MCCR-80027 ed.). Author, 1985.

Navy. *Interface Design Specification* (DI-E-2135 ed.). Author, 1983.

Navy. *Interface Requirements Specification* (DI-MCCR-80026 ed.). Author, 1985.

Navy. *Military Standard Defense System Software Development* (DOD-STD-2167 ed.). Author, 4 June 1985.

Navy. *Operational Concept Document* (DI-MCCR-80023 ed.). Author, 1985.

Navy. *Operator's Manual* (DI-M-2145 ed.). Author, 1983.

Navy. *Program Description Document* (DI-S-2139 ed.). Author, 1983.

Navy. *Program Design Specification* (DI-E-2138 ed.). Author, 1983.

Navy. *Program Package Document* (DI-S-2141 ed.). Author, 1983.

Navy. *Program Performance Specification* (DI-E-2136 ed.). Author, 1983.

Navy. *Software Configuration Management Plan* (DI-E-2175 ed.). Author, 1983.

Navy. *Software Configuration Management Plan* (DI-MCCR-80009 ed.). Author, 1985.

Navy. *Software Detail Design Document* (DI-MCCR-80031 ed.). Author, 1985.

Navy. *Software Development Plan* (DI-A-2176 ed.). Author, 1983.

Navy. *Software Development Plan* (DI-MCCR-80030 ed.). Author, 1985.

Navy. *Software Product Specification* (DI-MCCR-80029 ed.). Author, 1985.

Navy. *Software Programmer's Manual* (DI-MCCR-80021 ed.). Author, 1985.

Navy. *Software Quality Assurance Plan* (DI-R-2174 ed.). Author, 1983.

Navy. *Software Quality Evaluation Plan* (DI-MCCR-80010 ed.). Author, 1985.

Navy. *Software Requirements Specification* (DI-MCCR-80025 ed.). Author, 1985.

Navy. *Software Standards & Procedures Manual* (DI-MCCR-80011 ed.). Author, 1985.

Navy. *Software Test Description* (DI-MCCR-80015 ed.). Author, 1985.

Navy. *Software Test Plan* (DI-MCCR-80014 ed.). Author, 1985.

Navy. *Software Test Procedure* (DI-MCCR-80016 ed.). Author, 1985.

Navy. *Software Test Report* (DI-MCCR-80017 ed.). Author, 1985.

Navy. *Software Top Level Design Document* (DI-MCCR-80012 ed.). Author, 1985.

Navy. *Software User's Manual* (DI-MCCR-80019 ed.). Author, 1985.

Navy. *System Operator's Manual* (DI-M-2148 ed.). Author, 1983.

Navy. *System/Segment Specification* (DI-MCCR-80008 ed.). Author, 1985.

Navy. *Version Description Document* (DI-MCCR-80013 ed.). Author, 1985.

Parikh, Girish. *Techniques of Program and System Maintenance.* Winthrop Publishers, Inc., 1982.

Royce, Dr. Winston W. Managing the Development of Large Software Systems. In *9th International Conference on Software Engineering.* IEEE, 1987.

Sommerville, I. *Software Engineering.* Addison-Wesley Publishing Co., 1982.

Tausworthe, Robert. *Standardized Development of Computer Software.* Prentice-Hall, Inc., 1977.

Vita

Demetria Deakos was born November 18, 1950 in Hazleton, Pa. She was the youngest child of Pietro and Amalia Deakos. She attended Wilkes College from 1968 to 1972, and graduated with a Bachelor of Science in Chemistry. She attended Lehigh University from 1983 to 1987 and graduated with a Master of Science in Computer Science.