One person's perception of military documentation

by DON MATHER

Sanders Associates
Hudson, New Hampshire

ABSTRACT

Culture shock is perhaps the best way to describe what one experiences in moving from the world of commercial documentation into the world of military documentation. This paper uses software documentation to describe the world of military documentation. After presenting some similarities and differences between the two worlds, it describes the military's software development process in a way that highlights documentation. In so doing, it also describes the military's software documents and points out the relationships between them.
INTRODUCTION

Culture shock is perhaps the best way to describe what one experiences in moving from the world of commercial documentation into the world of military documentation. On entering the world of defense work, one quickly (1) encounters a flurry of new acronyms—CDRL, DID, PPS, B5, PDR, CM, PDS, DBDD, PDD, C5, CDR, IDS, and so on; (2) hears reference to CDRL items, military standards, data item descriptions, data items, binding requirements, configuration management, and so forth; (3) learns that plans, specifications and even end-user documents must be written “in accordance with” standardized annotated outlines; (4) learns that some information in some documents is classified and must be marked and handled according to set procedures; and (5) learns that the money funding documentation projects comes from a contract with a military customer who has a good deal of influence over documentation.

The main purpose of this paper is to describe one person’s perception of the world of military documentation. A secondary purpose of this paper is to show that there are some similarities between writing documents in the two worlds of military and commercial documentation. Since the author’s experience has been mostly with software engineering and writing, this paper will use software documentation to describe the world of military documentation. The intended audience of this paper is mainly technical writers, editors, and managers of editors and writers who have had little or no experience in the world of military documentation (especially military software documentation).

This paper presents the following topics: (1) The contract, CDRL, military standards, DIDs, and binding requirements, (2) a summary of similarities and differences between the worlds of military and commercial documentation, (3) the military’s software development process and its documents.

THE CONTRACT, CDRL, MILITARY STANDARDS, DIDs AND BINDING REQUIREMENTS

When the Department of Defense selects a company (or team of companies) to perform some service for it, it awards that company a contract. That company is referred to as a “contractor.” The contractor refers to that part of the military, which awarded it a contract, as the “customer.”

The contract contains a list of documents to be written and delivered to the government. The list is called a “Contract Data Requirements List,” or CDRL. Normally, CDRLs are written on standard government forms called DD-1423s (See Figure 1). Any item on the CDRL list is referred to as a “CDRL item,” “data item,” or simply a “deliverable.”

A military standard is simply a document that specifies how something shall be done. A standard is stronger than a guideline—a standard must be complied with. There are many military standards and the subject of a standard varies from standard to standard.

The most widely used military standards for software development are MIL-STD-1679(NAVY), MIL-STD-483(USAF), and MIL-STD-490(USAF). MIL-STD-1679(NAVY) covers nearly all aspects of software development; it does not cover the style and format of software documents much beyond stating that the word “shall” is reserved for identifying binding requirements. MIL-STD-483(USAF) is used to control software development. MIL-STD-490(USAF) was intended to be a universal standard. It can be used to cover the development of software, hardware, a building, a desk, a train car, etc. MIL-STD-490 does have a section pertaining to the style and format of documents. Military standards also specify the data item descriptions that are intended to be used with them. This paper uses the names of documents given in MIL-STD-1679 to discuss software documents. Both the Air Force and the Navy have guidebooks that provide a good deal of information about how those services manage their software acquisition.

A data item description, or DID for short, is an annotated outline of one kind of document, e.g., a QA plan or a design specification. A CDRL list will specify that a certain document must be written in accordance with a particular military standard and data item description. With a DID in hand, all the contributors to a document know the title of the document, the outline of the document, have an idea of what kind of information goes in each section, and who needs the document and why.

A binding requirement is a requirement that a contractor must meet. It is a legally binding requirement. The word shall identifies a binding requirement. For example, if a sentence is worded “The operator interface module enables the operator to set the time of day,” then the software does not necessarily have to provide the operator with that capability. On the other hand, if the sentence is worded “The operator interface module shall enable the operator to set the time of day,” then the contractor can be held accountable in a court of law for supplying to the government an operator interface module which fulfills that requirement. This style device enables the contributors to a document to distinguish between explanatory information and what they believe they are required to do in order to satisfy the contract. This mechanism also enables the customer to perceive that distinction.
<table>
<thead>
<tr>
<th>SEQUENCE NUMBER</th>
<th>TITLE OR DESCRIPTION OF DATA</th>
<th>CATEGORY</th>
<th>CONTRACTOR</th>
<th>SYSTEM/ITEM</th>
<th>AUTHORITY (Data Item Number)</th>
<th>CONTRACT REFERENCE</th>
<th>TECHNICAL OFFICE</th>
<th>FREQUENCY</th>
<th>DATE OF 1ST SUBMISSION</th>
<th>DATE OF SUBSEQUENT SUBMISSIONS</th>
<th>DISTRIBUTION AND ADDRESSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Program Performance Specification (PPS)</td>
<td>Super System Program</td>
<td>Super Contractor</td>
<td>Super System</td>
<td>A001</td>
<td>DI-E-2138, MIL-STD-1679</td>
<td>SOW Para 3.1</td>
<td>Customer</td>
<td>1</td>
<td>1</td>
<td>5 MAC</td>
</tr>
</tbody>
</table>

**Remarks:**
A preliminary copy will be submitted for approval. The government will respond with comments 30 days after receipt. Final will be delivered 60 days after receipt of comment. Submission dates based on Jan 1 start.

**Total:** 6

---

**Prepared By:**

Customer Representative 1

Date

Approved By:

Customer Representative 2

Date

Page 1 of 1

---

From the collection of the Computer History Museum (www.computerhistory.org)
Perhaps one of the most obvious differences between the two worlds is the presence of the customer in the development of military documentation. In the military world, the customer (1) defines what documents will be written and when they are due, (2) requires that the documents comply with standard outlines, (3) sets some style and format conventions, (4) imposes policies and procedures for marking and handling classified information when documents contain classified information, (5) has the right to approve or reject documents, (6) may include in the contract the right to award the contract money in parts paying a portion every time a document is approved. Thus, in the world of military documentation, one's freedom to develop a document as one sees fit is much more restricted than in the world of commercial documentation. On the other hand, those commercial companies which offer their services to other companies may notice a resemblance between their situation and a defense contractor's.

Technical writers and editors in the two worlds probably work on different kinds of documents most of the time. In the world of military software documentation, writers and editors work mostly on development documents rather than end-user documents. Just the opposite seems to be true in the commercial world.

Companies that are practicing a methodical software development process which emphasizes documentation may perceive a similarity between their process and the military's. The names of the documents and the emphasis given topics may be different. Generally, however, those companies will probably write the same kinds of documents, produce them in the same order and in the same software development phases, and cover the same topics.

The planning of documentation projects is probably similar. In both worlds, planning must answer the questions: (1) Who needs documentation and why? (2) What documents will be written and what are the objectives of each one? (3) How will those objectives be met? (4) How will it be determined which purposes were achieved and the degree to which the others were met? (5) What are the required sources, schedule, and costs?

In the world of military documentation, the CDRL list answers the question “What documents will be written?” and specifies when they will be due and how many copies will be delivered. Furthermore, the data item descriptions (1) identify who needs documentation and indicates why, (2) state the objectives of each document, (3) provide a partial answer to the question “How will those objectives be met?” in providing an annotated outline. The applicable military standard will shed more light on the question “How will those objectives be met?” to the extent that it specifies style and format.

Perhaps the most obvious and greatest similarity between the world of military documentation and the world of commercial documentation is the existence of a need for documentation. The fundamental goal of every document, whether it is a military or commercial document, is to communicate with someone with this need.

The development process is the heart of software management. A “good” development process plus “good” scheduling and cost control result in a high percentage of successful software projects. A “good” development process is nearly always methodical, i.e., development occurs as a sequence of refinements, each of which is produced in a methodical way.

This paper briefly describes the military’s software development process in a way that brings out the role of documentation.

Software documentation expresses plans and software specifications. The government has developed standard sets of software documents and each document has a standard annotated outline. There are a variety of reasons for moving toward a standardized approach to documentation. This approach, for instance, is one way to deal with the complexity of working with multiple contractors and to aid end users—everyone interested in the same information can find it in the same place regardless of the contractor.

Software documentation is useful to contractors, the government, and the end-users alike. Software development documentation increases the likelihood of an orderly development process, establishes well-defined baselines, provides a vehicle for change control, provides for personnel changes during the entire life of development and maintenance, and facilitates maintenance. The end-user documentation provides the ultimate users of the developed system with the information they need to perform their jobs well. Ideally, the attributes of software documentation are completeness, accuracy, appropriateness, and clarity. These attributes result in specifications that are internally consistent, explicit, designable and/or testable, traceable between documents, and assignable to programming personnel.
Structured programming is a discipline for producing code that can significantly improve software reliability and maintainability. The main attributes of structured programming are that the code is modular, top-down, sequential, indented to bring out the structure of the logic, has one entry and one exit, and uses a restricted set of control and data structures. These attributes lead to code that is simpler, clearer, and easier to test than unstructured code.

Software is developed in phases. There are many ways to identify those phases. For the purposes of this paper, those phases are designated as

1. Initial Planning
2. Requirements Analysis
3. Preliminary Design
4. Detailed Design
5. Code, Debug, and Unit Test
6. Contractor Testing
7. Acceptance Testing

Documentation is developed or used in each of these phases. The remainder of this paper describes each of these phases and the role of documentation in each phase. It also points out the relationships between the software documents.

Initial Planning Software Development Phase

The documents produced during the Initial Planning Phase convey the contractor’s plans for fulfilling the contract. Four types of software planning documents can be written during this phase:

1. Software development plan (SDP)
2. Software quality assurance plan (SQA or, simply, QA plan)
3. Software configuration management plan (SCM or, simply, CM plan)
4. Software standards and conventions

The software development plan is software management’s plan for developing the program performance specification and producing software, which satisfies the requirements specified in the program performance specification, within budget and on time.

The software quality assurance plan is the quality assurance group’s plan for verifying that all the requirements stated in the contract are met. Important parts of a QA plan are the plans for verifying that the software group and the configuration management group (whose function is explained below) are complying with the SDP and CM plan, respectively.

A software configuration management plan is the configuration management group’s plan for managing changes in the software’s configuration during software development. (The word configuration may require some explanation. Suppose a contractor is developing a not-so plain, everyday, homely desk for the government. By the configuration of the desk is meant all the information needed to completely describe the desk. For instance, if the current configuration of the desk calls for a 24-inch drawer and someone wants to make it a 30-inch drawer, then that is a change in the desk’s configuration.)

Software standards and conventions can be covered in either a section of the software development plan or in a separate document. They specify programming standards and how some aspects of software development will be conducted.

Requirements Analysis Software Development Phase

The documentation produced during the Requirements Analysis Phase conveys the contractor’s understanding of the functional performance requirements to the customer. Two kinds of documents can be written during this phase:

1. Program performance specification (PPS)
2. Interface design specification (IDS)

The program performance specification is a functional specification. This kind of document describes what functions the software will perform, not how the software will perform them. If a function should be tested at the system level, then it belongs in the PPS and, otherwise, it does not. A PPS addresses

1. System-level functions that have been delegated to software and some implied functions
2. Interfaces external to the product being developed and between the major software functions
3. Hardware environment in which the software will perform
4. Kinds of tests required to verify that the software does indeed comply with the requirements described in the PPS

The PPS is a necessary preliminary to setting up test requirements and beginning the software design. Some software projects are sufficiently large or complicated to warrant developing more than one PPS on a project. The CDRL list specifies what PPSs must be developed and then delivered to the government.

A program performance specification is referred to by several names. MIL-STD-1679(NAVY) refers to it as a program performance specification. MIL-STD-490 calls it a B5 Specification. MIL-STD-483 calls it a Part I Specification. This kind of document can also be called a data processing system requirements specification (DPSR).

The program performance specification and the software development plan are the two most important software engineering documents. A software project can be defined as a project to produce software, which has agreed-upon functions, within budget, on time, and in a manner that has an amount of risk that is acceptable to the software development manager. The PPS is software management’s written vehicle for gaining and communicating agreement as to what functions the software is supposed to perform. The SDP is software management’s written plan for producing the software.
within budget and on time. Furthermore, any significant error in either of these two documents can lead to a situation that is singularly challenging (and expensive) to correct.

The interface design specification describes the software interfaces and the data flowing between two digital processors. By “software interfaces” is meant those interfaces which send data to the software under development, which the software hands off data to, or which the software controls. The interface design specification was mainly intended to cover the interaction of the software being developed with software in another system.

The interface specification has another use when more than one company is developing the software. Specifically, it can be used to specify the interfaces between the software being developed by two of the companies. This is one way the two companies can know what to expect in the way of input from the other company and what they are expected to hand off to the other company. The interface design specification then becomes one basis for managing the interface between the two companies.

The chief importance of an interface specification to a software development manager is in its potential for shortening the Contractor’s Testing Phase. This potential can be realized when the software engineers know precisely what requirements they are to implement and their manager exercises rigid control over the interfaces between the software developed by different programmers and programming teams. An interface document increases the likelihood of software developed by different programmers or teams of programmers interacting correctly. This single improvement can dramatically reduce integration time.

Preliminary Design Software Development Phase

The documents produced during the preliminary design phase describe the top-level design and planning of the contractor’s approach to fulfilling and verifying the requirements specified in the program performance specification. Three kinds of documents can be written during this phase:

1. Program design specification (PDS)
2. Data base design document (DBDD)
3. Test plan

Once again, the CDRL list will define which of these documents must be written and delivered to the government as a contract requirement, but the main purpose of a PDS is to describe the design approach. It describes the architecture and organization of program modules. It provides the programmers with a logical description of the internal design of the software. A PDS is not a detailed design document, but rather, it communicates the design idea.

Program design specification is the name MIL-STD-1679(NAVY) uses to designate this kind of document. MIL-STD-490(USAF) and MIL-STD-483(USAF) do not have an exact equivalent to a PDS. Their design documents come out of the total design effort; only parts of them are developed during the preliminary design phase.

Typically, a data base design document appears on a CDRL list when there is a large data base or the data base is critical in some way. A DBDD describes all the data used by two or more software components and shows the file organization.

The test plan is a management document. It identifies the major functional areas to be tested, describes the testing methodology, and identifies the resources (people, equipment, and time) needed for testing.

These documents are often reviewed at a preliminary design review (PDR). A PDR is a formal review conducted during the preliminary design phase. The purposes of a PDR are to

1. review the top-level design,
2. evaluate progress,
3. verify the technical adequacy of the selected design and test approach, and
4. verify compatibility between the PDS and the PPS, i.e., verify that the design covers all the requirements in the PPS and covers no more than that. More than one PDR may be conducted if the PDS and DBDD are being developed in stages. (Note: The Air Force often conducts a PDR during the Requirements Analysis Phase rather than during the design phases.)

Detailed Design Software Development Phase

This is the last phase of software design. During this phase, the programming team converts the design approach expressed in the PDS and the DBDD into detailed processing steps. The results of the conversion are expressed in the program design description (PDD) document.

The PDD describes the design details of each software component to be coded. It includes functions performed, design structure, operating constraints, inputs and outputs, diagrammatic/narrative flows, and data base organization. The PDD serves as the primary document that development and maintenance personnel use for developing software, diagnosing trouble, and modifying software.

In addition to the PDD, several other documents are produced or updated during this phase:

1. The PDS is revised with comments from the PDR and possibly with improvements identified by the contractor since the PDR
2. The DBDD is revised with comments from the PDR and possibly with improvements identified by the contractor since the PDR
3. Test plans are updated with comments from the PDR and possibly with improvements identified by the contractor since PDR
4. Test specifications, which describe how the requirements will be tested, are produced by the test team using the test plans

The C5 Specification of MIL-STD-490(USAF) and the Part II Specification of MIL-STD-483(USAF) are equivalent to the combination of a PDS and PDD.

All the documents produced or updated during the detailed design phase are often reviewed at a critical design review (CDR). A CDR is a formal review at the completion of the detailed design phase and before code development begins.
The main purpose of a CDR is to review the detailed program design. There can be more than one CDR when the detailed design is evolving in stages.

It is at the end of the detailed design phase that the information needed for writing the first draft of the maintenance and operator manuals is known. The definition of menus, prompts, error messages, what conditions cause the error messages to be issued, and how to respond to error messages, initialization and recovery procedures, and so on are all defined by the end of the detailed design phase. Thus, if this information is written down by the software engineers as soon as they know it, then work on the maintenance and operator manuals can begin in the next phase. Typically, however, these documents are not started until the contractor testing phase.

**Code, Debug, and Unit Test Software Development Phase**

This software development phase is when individual programmers will code and debug their software. After a pro-
grammer has written some code, he or she will unit test it until satisfied that the software performs properly. At this point, the software is ready to enter the next phase of software development. During this phase

1. Programmers use the PDD to produce code
2. Test personnel use the test specifications to write the test procedures, which are detailed procedural descriptions of how they will perform the tests described in the test specifications
3. Technical writers can begin writing the first drafts of the maintenance and operator manuals (though usually work on these documents does not begin until the next phase)

Contractor Testing Software Development Phase

During this phase, the contractor's software development team (as opposed to an individual programmer) tests software until it is ready for acceptance testing. The programming team may do some testing of its own on the software before turning it over to the test team. The test team will perform the tests described in the test procedures document (and possibly use the operator manuals if they are available) and any other tests they deem needed. If errors are detected, then the test team writes test reports and returns the software with the test results to the programming team for correction. When the programming team is satisfied the problems have been resolved, they submit the corrected software to the test team for retesting. This continues until the test team (and QA personnel) are satisfied that the software is ready for acceptance testing.

The final versions of the maintenance manuals and end-user documentation can be produced during this or the next phase. By the end of this phase, all the documentation should be revised to reflect the as-built configuration of the software. The CDRL will specify whether this will in fact be done.

Acceptance Testing Software Development Phase

During Acceptance Testing Phase, the software is tested either by the customer or in the presence of the customer. The people performing the tests can use the test procedures and the user's manuals. When the software passes this test, the customer has accepted the software and the software development process ends.

SUMMARY

The following figures summarize the software documents. Figure 2 presents the document tree, which relates the documents to each other, and Figure 3 shows relationships between software documents and software development phases.

SUGGESTED READINGS

5. MIL-S-83490, Specifications, Types and Forms.

Note: Readings 1-5 are available from the Naval Publications and Forms Center, 500 Tabor Ave., Philadelphia, PA 19120.