Software Design Documentation Approach for a DOD-STD-2167A Ada Project

Michael Springman
TRW Defense Systems Group
Redondo Beach, California

ABSTRACT

DOD-STD-2167A and its predecessor DOD-STD-2167 impose significant documentation requirements on software development projects. The 2167 documentation set, particularly for documenting the software design through the life cycle, contained a significant amount of redundancy. Also, for Ada development projects, 2167 did not adequately recognize the benefits achievable from using Ada as a uniform representation of the design and code products throughout the software life cycle. DOD-STD-2167A is an improvement over 2167, but a contractor and the customer must still be conscious of the possibility of generating documents with limited utility to document producers and reviewers. This paper describes a software design documentation approach being used on the Command Center Processing and Display System Replacement (CCPDS-R) project that uses heavily tailored 2167 Data Item Descriptions (because 2167A was still in the formulation stage when CCPDS-R began) to: (1) provide reviewers with appropriate design information during the software development process; (2) provide the system user with the documentation needed to maintain the delivered software; (3) eliminate redundancy; and (4) streamline the generation of the deliverable documents through reliance on information already contained in the Software Development Files (SDFs). The resulting design document set satisfies DOD-STD-2167A requirements.

PROJECT BACKGROUND

The CCPDS-R project will provide display information used during emergency conferences by the National Command Authorities; Chairman, Joint Chiefs of Staff; Commander in Chief North American Aerospace Command; Commander in Chief United States Space Command; Commander in Chief Strategic Air Command; and other nuclear capable Commanders in Chief. It is the missile warning element of the new Integrated Attack Warning/Attack Assessment System Architecture developed by North American Aerospace Defense Command/Air Force Space Command.

The CCPDS-R project is being procured by Headquarters Electronic Systems Division (ESD) at Hanscom AFB and was awarded to TRW Defense Systems Group in June 1987. The project consists of three separate subsystems of which the first, identified as the Common Subsystem, is 24 months into development. The Common Subsystem consists of approximately 350,000 source lines of Ada with a development schedule of 40 months. When software development for all three subsystems is complete in 1992, over 600,000 Ada source lines plus developed tools and commercial off-the-shelf (COTS) software will have been delivered to the Air Force. CCPDS-R is characterized as a highly reliable, real-time distributed system with a sophisticated user interface and stringent performance requirements. All CCPDS-R software is being developed using DEC's VAX Ada compiler on DEC VAX/VMS machines, augmented with Rational's R1000 Ada environment. The software will execute on a network of DEC mainframes and workstations.

CCPDS-R was planned and bid prior to the establishment of DOD-STD-2167A [2167A], so the software is being developed using a heavily tailored DOD-STD-2167. The 2167 tailoring was done in parallel with the formulation of DOD-STD-2167A, which has resulted in a CCPDS-R methodology and documentation set that is consistent with DOD-STD-2167A.

CCPDS-R exhibits the characteristics of a typical large 2167/2167A Ada development project, including:

1. Large number of software requirements (approximately 2,000)
2. Multiple CSCIs (6 for the Common Subsystem; 15 total)
3. Large number of 2167A components (approximately 7,000 CSCs/CSUs) and architecture objects (30 VAX/VMS processes, 110 Ada tasks)
4. Informal test of individual components to test all nominal, off-nominal and boundary conditions
5. Informal integration of tested components into working capability strings

6. Formal requirements verification per Government-approved test plans and test procedures

DESIGN DOCUMENTATION OBJECTIVES

Design documentation is good practice for any software development project, and is required by all Government contracts. The volume and level of detail vary by customer and contract, but the universal purposes of the documentation are to: (1) provide a review mechanism for Government and contractor personnel during development and (2) provide a maintenance resource for the eventual software maintainers. The requirements levied on a contractor should balance these objectives to ensure the reviewers get what they need during the development and the maintainers get what they need to maintain the as-built product. The specific objectives of a design document set should be:

Efficiency. Formal deliverable documents should be natural byproducts of the design/development process by taking advantage of items normally produced by the developers (e.g., Ada as a design language (ADL), commented Ada source code, SDF sections). Milestones for deliverable documentation serve as forcing functions for the developers to produce and update required documents. This is useful to a contractor because of the general tendency of software developers to place documentation lower on their priority queue.

Understandability/Uniformity. During the development process, documents should satisfy the needs of a reasonably informed reviewer, i.e., someone who is familiar with the system and software requirements and who can read Ada. The documentation should provide a comprehensive overview of the architecture, including graphic representations, to help reviewers understand the top level design and determine potential design/performance issues. It should also provide a mapping of static design elements (i.e., CSCs, CSUs, Units) to the dynamic architecture elements (e.g., Ada tasks, Ada main programs) so that operational capability strings can be followed in the source code.

Maintainability. For software maintenance following turnover of the completed system to the customer, the documents should provide complete as-built descriptions of each component (text and commented source code) and provide maintenance guidance for those software areas that are known to be potentially changeable or adaptable.

DEFINITION of DOD-STD-2167/2167A TERMS

The DOD-STD-2167/2167A definitions of CSCIs, TLCSCs/LLCSCs/Units (2167) and CSCs/CSUs (2167A) can result in categorizations of the software products for management and documentation purposes instead of pertaining to the architectural objects (e.g., Ada tasks). For example, the CCPDS-R architecture is described in terms of DEC VAX *nodes*, VAX/VMS *processes* (or *Ada main programs*), Ada *tasks*, and intertask communications *circuits* and *sockets*. (The architecture components are described in detail in Royce [1989-1], which discusses the Network Architecture Services (NAS) software developed initially for CCPDS-R.) The 2167/2167A partitioning of components affects the documentation structure. CCPDS-R's definitions of the 2167 components are:

Computer Software Configuration Item (CSCI): A collection of TLCSCs, LLCSCs and Units that can be allocated to a single functional organization (i.e., skill center) to implement. For example, CCPDS-R has display, communications, system services, test and simulation, and algorithm CSCIs.

Top Level Computer Software Component (TLCSC): A component which maps directly to Ada library units or collections of functionally cohesive Ada library units. A TLCSC may contain nested LLCSCs and Units, and must be separately testable (termed "standalone test", or SAT). For documentation purposes, a logically related collection of TLCSCs within a CSCI is termed a "TLCSC Group".

Lower Level Computer Software Component (LLCSC): A program unit declared within a program unit (which could be either a TLCSC or a higher level LLCSC) that is sufficiently complex to require standalone testing prior to its inclusion in the standalone testing of its parent.

Subordinate Unit: A component of an LLCSC or TLCSC whose standalone test is wholly provided by the standalone test of its parent program unit. A Unit may also be defined as a library unit as long as its services are not shared across TLCSC boundaries.

The 2167A components map to CCPDS-R's 2167 definitions as follows:

Computer Software Component (CSC): Equivalent to TLCSC Groups. Per the standard,

CSCs may be further decomposed into other CSCs and CSUs.

Computer Software Unit (CSU): An element of a CSC that can be standalone tested. CSUs equate to individual TLCSCs and LLCSCs.

Subordinate Unit: These are the lower level "units" that comprise the standalone testable CSUs. These generally must be tested in the context of their parent CSUs.

DESIGN DOCUMENTATION SET

The 2167 and 2167A document sets are very similar in overall information content, although the individual document definitions differ (Figure 1). The required 2167 documents are:

Software Top Level Design Document (STLDD). Per CSCI, the STLDD is intended to provide a top level description of the architecture and component/data flow, a summary of CSCI requirements traceability to the CSCI's TLCSCs, and a detailed description of each TLCSC, including input/output interfaces, processing, control flow, limitations and interrupt handling.

Software Detailed Design Document (SDDD). Per CSCI, the SDDD is intended to provide a detailed design description of each CSCI component in a hierarchical manner, including each TLCSC, its constituent LLCSCs, and subordinate program units.

Software Product Specification (SPS). Per CSCI, the SPS is required to include the final STLDD, final SDDD, and the final code listings.

Software Development Files (SDFs). SDFs are required to be generated and maintained during the development of the software, although they are generally not formally deliverable items (e.g., there is no 2167 DID specifying the format of an SDF). The SDFs are intended to be the developer's repository for all design and test data generated for a software component.

The 2167A design document set consists of:

Software Design Document (SDD). Per CSCI, the SDD includes the top level architecture description, CSC/CSU descriptions and CSCI data descriptions. It combines the 2167 STLDD and SDDD.

Software Product Specification (SPS). Same as for 2167, except the final SDD replaces the final STLDD and SDDD.

Software Development Files (SDFs). Same as for 2167.

DOD-STD-2167 and 2167A include detailed DIDs for the design documents that are intended to apply to any DOD software development application. There are certain aspects of these DIDs that do not lend themselves well to an Ada development process, particularly the incremental development and review process being employed on CCPDS-R. The DOD-STD-2167 series of design DIDs can result in duplicative, voluminous documents which are of dubious value to the document producers and reviewers, particularly where Ada itself is used to describe the design. Each standard encourages tailoring of the standard and its DIDs for each particular application. This paper describes: (1) revisions made to the 2167 software design document DIDs consistent with the CCPDS-R Ada Process Model (Ref.Royce [1989-2]); (2) the planned evolution of the design documents as the software is developed; and (3) a 2167A documentation approach based on CCPDS-R experience.

Ada PROCESS MODEL OVERVIEW

The CCPDS-R software development approach is the initial application of TRW's "Ada Process Model", which is based on early definition, demonstration, implementation and test of incremental capabilities termed *builds*. DOD-STD-2167 has been tailored for CCPDS-R to accommodate this process model, including the incremental generation and review of the design and documentation products. A subsystem build consists of a collection of CSCs from one or more CSCIs which are integrated to form an incremental set of subsystem capabilities. Each CSCI is developed incrementally, with each CSCI build having its own top level design, detailed design, code and test cycle.

The builds are defined so that the foundation architecture components that are relatively independent of the required System Specification capabilities are developed, integrated and tested as early as possible, while the generally more volatile, application-specific components are allocated to later builds. The Ada Process Model requires that software capabilities be demonstrated at informal design walkthrough milestones and at formal review milestones to provide tangible evidence of design progress. Such reviews involving capability demonstrations provide a much sounder basis than traditional paper/viewgraph reviews for the customer and the contractor to assess readiness to proceed with subsequent development activities.

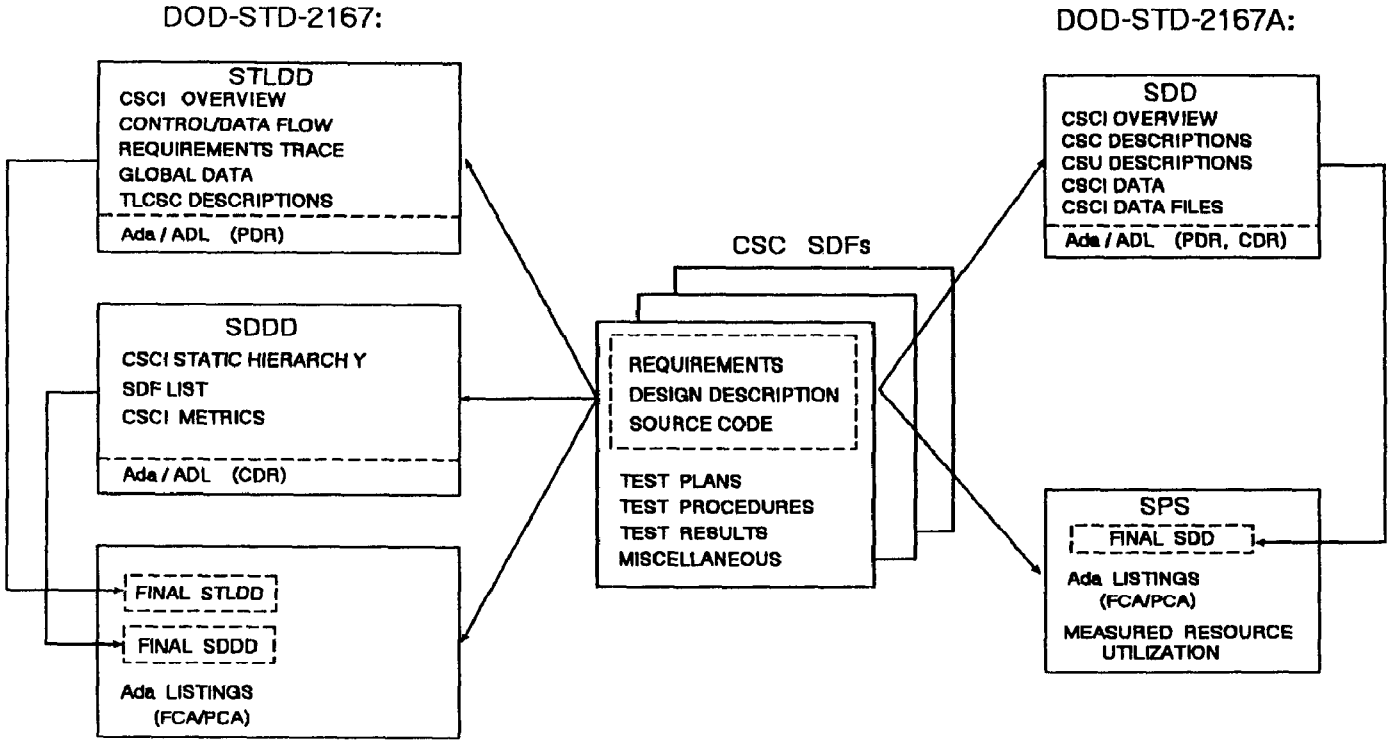

Figure 1: 2167/2167A Design Document Relationships

The Software Architecture Skeleton (SAS) is defined and baselined early, and consists of the top level executive structure for all processes and tasks and their interconnections (i.e., circuits and sockets). The process and task executives are all instantiated generics, with the Ada source code produced by a tool which has all the architecture objects described in a database. The SAS concept enables rapid construction of a complete functioning network, which facilitates early discovery of design, interface and integration problems.

The primary advantage of Ada in supporting incremental development as defined above is its support for partial implementations. Separation of specifications and bodies, packages, sophisticated data typing and Ada's expressiveness and readability provide powerful features which can be exploited to provide an integrated, uniform development approach. The uniformity gained through the use of Ada throughout the software development cycle as a representation format is also useful for providing consistent and insightful development progress metrics for continuous assessment of project status from multiple perspectives.

The software design/implementation phases of the Ada Process Model are described in detail in [Royce 1989-2] and summarized below:

- **Top level architecture design** of the foundation software components, resulting in definition of the System Global Interface (SGI) packages and the Software Architecture Skeleton (SAS). Also produced is the allocation of software for each CSCI to specific incremental builds to maximize early availability of functionality and minimize downstream breakage. Preliminary Design Walkthroughs (PDWs) are conducted during this phase for the contractor and the Government to periodically review the evolving top level design.
- **Top level design** for each applications build, which refines the overall top level architecture design and iterates the SAS/SGI architecture as the

design progresses. An applications oriented PDW culminates this phase.

- **Detailed design** for each build, culminating in a Critical Design Walkthrough (CDW).
- **Implementation and informal standalone test** of all build components.
- **Turnover** of completed build components to the I&T organization for formal baselining and test activities. The turnover process involves a significant amount of integration by the developers and testers as the software is built into a functioning configuration.

CCPDS-R DOCUMENTATION APPROACH

The CCPDS-R contract requires that the formal 2167 design documents be delivered initially to the Government as follows:

- STLDD: 30 days before PDR
- SDDD: 60 days before CDR
- SPS: 30 days before FQT

In the CCPDS-R software development process, the PDR is conducted after the final build Preliminary Design Walkthrough (PDW) is conducted. Similarly, the CDR is conducted after the final build Critical Design Walkthrough (CDW) is conducted. The PDR and CDR summarize the results of the incremental design walkthroughs and the status of action items resulting from the walkthroughs.

The intent of TRW's incremental build development approach is to evolve the design and the accompanying documentation from build to build, culminating in a complete representation of the top level design at PDR and of the detailed design at CDR. (See Figure 2.)

The PDW/CDW activities are informal milestones at which each CSCI developer's products are reviewed by his/her peers and other interested reviewers, including Government personnel. The design documentation is generated and maintained in the SDFs by the software developers as the design incrementally evolves. Reviewers rely on the walkthrough process and briefing materials (heavily dependent on Ada/ADL source code examples) to provide insight into the design approach and to provide a forum for constructive criticism. A major goal of the design documentation approach is to include as much of the required design description information as possible in the Ada/ADL. Ada is a descriptive language, particularly in the definition of global and local interfaces and data structures. The design and coding standards documented in the CCPDS-R Software Standards and Procedures Manual (SSPM) support this goal. This approach enables the design products to stabilize before publishing documentation over and above that needed by the design team to prove design feasibility. This relieves the designers from expending effort preparing voluminous documentation that would quickly obsolesce.

It became apparent during the design process that it is not practical to attempt to maintain a baselined top level version or detailed version of the Ada/ADL through the software product development cycle. It is much more effective and productive to maintain an updated version of the STLDD and current versions of the SDFs, which include the latest Ada/ADL. The Ada/ADL evolves in the SDFs to the actual product representation (Table 1), consisting of commented, readable Ada code that is generally considered the most accurate documentation of the software by software maintainers. TRW therefore uses a tailored STLDD and the SDFs as the key documents from which the deliverable design documents are produced. This corresponds directly to 2167A's SDD and SDFs.

DESIGN DOCUMENT EVOLUTION

The SDF is the central source for design information for individual software components. The SDF outline is shown in Table 3. This outline includes all items required by the DOD-STD-2167 STLDD and SDDD DIDs and 2167A SDD DID for software component design description information. Each SDF evolves through the design walkthrough and turnover milestones as shown in Table 2. There is one SDF per TLCSC and LLCSC, which for 2167A equates to one SDF per CSC per build (assuming a CSC may consist of multiple sub-level CSCs allocated to different builds).

STLDD Section 3.6 (top level design descriptions for all TLCSCs) is generated using Section 3 of each TLCSC's SDF, while the STLDD Ada/ADL appendix includes the PDR snapshot of the Ada/ADL source code that is generated from Section 4 of each TLCSC's SDF. The SDDD simply references the CSCI's SDFs and includes the CDR snapshot of the Ada/ADL source code. The SDFs are made available to the Government for review in support of the CDR. Using this approach to the STLDD and SDDD, generation and maintenance of the design description and Ada/ADL information is centrally performed in the SDFs, which enables single-point control, currency of the information, and efficient document generation.

The STLDD, SDDD and SPS evolve through the walkthrough, formal review and turnover milestones as

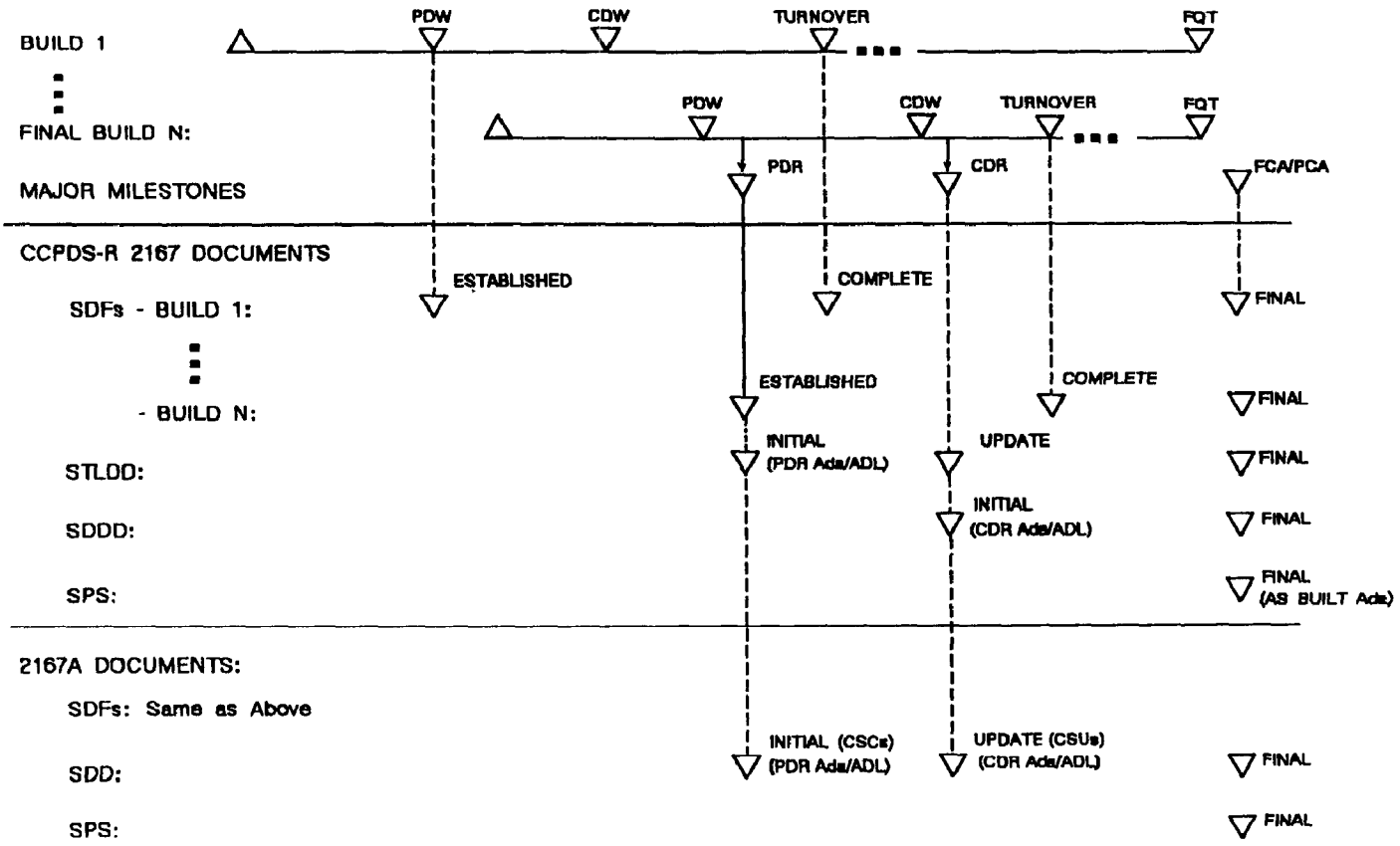

Figure 2: Design Document Generation Schedule

shown in Table 4. The STLDD reflects the complete and evolving top level design at each milestone and includes listings of the Ada/ADL to support the walkthroughs and PDR. The SDDD reflects the complete detailed design at CDR, including listings of the Ada/ADL detailed design representation. The SPS consists of the final updated STLDD and SDDD (which refers to the CSCI's SDFs) and the complete Ada product listing. The SDFs are part of the Software Development Library that most contracts require to be delivered at the end of the contract, and thus are a delivered component of the final design documentation set.

Duplication among the documents is minimized by making the STLDD the textual and graphical representation of the complete top level design for the entire CSCI, while the SDDD addresses the designs of the individual components (TLCSCs, LLCSCs, Subordinate Units) by reference to the SDFs. The top level design version of the Ada/ADL is captured in the PDR version of the STLDD, while the detailed design version is captured in the CDR version of the SDDD. No attempt is made to maintain these versions as the design and implementation progress. The current Ada/ADL design representation is captured instead in the SDFs.

The SDFs are formally established following the appropriate PDW. "Establishing" an SDF consists of producing a hard-copy version with appropriate cover sheet and constituent sections that can be audited by the software QA organization. Prior to this time, individual developers work in their on-line SDF environments to generate the design description information required for inclusion in the STLDD. The reason to delay formal establishment of the SDF until after PDW is to be able to efficiently accommodate architecture design decisions that result from PDW (e.g., combining/redefining/eliminating TLCSCs/LLCSCs).

The SDFs are maintained electronically (i.e., on-line) and are generated using an SDF Build tool developed for CCPDS-R. This tool generates requirements and test traceability tables, software metrics, code audi-

Table 1: Ada/ADL Evolution

Components				
2167	2167A	PDW	CDW	Turnover
TLCSCs	CSCs/CSUs	Ada Specs ADL Bodies	Ada Specs Ada Bodies	Ada
LLCSCs	N/A	ADL Specs	Ada Specs ADL Bodies	Ada
Subordinate Units	Subordinate Units		ADL Specs ADL Bodies	Ada Ada

Table 2: Software Development File (SDF) Evolution

SDF SECTION	PDW		CDW		TURNOVER & FCA/PCA
	TLCSC	LLCSC	TLCSC	LLCSC	TLCSC/LLCSC
1. Cover Sheets	Complete	Complete	Complete	Complete	Complete
2. Requirements	Complete	Prelim	Complete	Complete	Complete
3. CSC Design Descriptions	Complete	Prelim	Complete	Complete	Complete
4. CSC Program Unit	Ada/ADL	ADL	Ada	Ada/ADL	Ada
5. Subordinate Program Units	ADL		Ada/ADL	ADL	Ada
6. SAT Plan	Prelim		Complete	Prelim	Complete
7. SAT Procedures			Prelim		Complete
8. SAT Results					Complete
9. SPR Log			Updated		Updated
10. Metrics/Code Auditor Results	Prelim	Prelim	Updated	Updated	Updated
11. Notes (Waivers, etc.)	As Applicable	As Applicable	As Applicable	As Applicable	As Applicable

tor results, and other required items automatically, assuming the software adheres to the standards specified in the CCPDS-R SSPM. Another utility tool is used to extract appropriate information from the CSCI's SDFs to generate design description subsections for section 3.6 of the STLDD. These tools minimize the tedious labor associated with generating formal design documentation.

DESIGN DOCUMENT SET RECOMMENDATIONS

The CCPDS-R tailored 2167 design document set is basically a 2167A approach (Table 5). Because the SDDD is primarily a pointer to the SDFs for detailed design description information, the CCPDS-R document set consists of the STLDD, SPS, and SDFs, which correlates directly to the 2167A SDD, SPS and SDFs. CCPDS-R also defined a pair of higher level documents that addressed topics that spanned across all CSCIs. These were:

System Description Document, which describes the overall hardware and software architecture and summarizes each CSCI's role in the architecture.

Technical Performance Measurement (TPM) Report, which reports the progress towards meeting the system's quantitative performance requirements, including processing and memory resource utilization.

The contents of these documents can be included in 2167A's "System/Segment Design Document".

Some recommendations follow that are based on CCPDS-R experience, which includes customer feedback concerning what is required for managing and monitoring the software development process and for maintaining the final software product. The result is a design document set that satisfies contractor, customer and contractual needs. These recommendations assume DOD-STD-2167A is required for the contract.

- Include as much of the required design description information as possible in the Ada/ADL. Use the Ada/ADL as the basis for detailed review of the design.

- Use the SDFs as the primary source for the deliverable SDD and SPS design description information.

Table 3: Software Development File (SDF) Outline

1. Cover Sheets
2. Requirements
   - 2.1 Requirements Allocation
   - 2.2 CSC Problem Statement
3. CSC Design Description
   - 3.1 External Interfaces
   - 3.2 Critical Object Design
   - 3.3 Interrupts
   - 3.4 Timing and Sequencing
   - 3.5 Limitations and Constraints
   - 3.6 Components Description (including subordinate CSCs, CSUs, Units)
4. CSC Ada/ADL
5. Subordinate CSCs, CSUs and Units Ada/ADL
6. Plan for Standalone Test
   - 6.1 Objectives
   - 6.2 Test Classes
   - 6.3 General Acceptance Criteria
   - 6.4 Requirements Traceability Matrix
7. CSC Standalone Test Procedures
8. CSC Standalone Test Results
9. Software Problem Report (SPR) Log
10. Metrics and Code Auditor Results
11. Notes

Table 4: STLDD, SDDD, SPS Evolution

DOCUMENT	SECTION		PDR	CDR	FCA/PCA
STLDD	3.1	CSCI Architecture	Complete	Updated	Updated
	3.2	Functional Allocation	Complete	Updated	Updated
	3.3	Memory/Processing	Complete	Updated	Updated
	3.4	Component Control & Data Flow	Complete	Updated	Updated
	3.5	Global Data	Complete	Updated	Updated
	3.6	Top Level Design Description	Complete	Updated	Updated
	3.7	Adaptation Data	Complete	Updated	Updated
	App 10	Ada/ADL Listings	Mag Tape		
SDDD	3.1	CSCI Static Hierarchy		Complete	Updated
	3.2	CSCI SDF Index		Complete	Updated
	3.3	CSCI Metrics		Complete	Updated
	App 10	Ada/ADL Listings		Mag Tape	
SPS	3.1	Physical Media Descriptions			Complete
	App 10	STLDD			Complete
	App 20	SDDD			Complete
	App 30	Ada Listings			Mag Tape

- The PDR version of the SDD should contain: (1) a complete description of the top level CSCI architecture, (2) CSC design descriptions, and (3) the PDR version of the Ada/ADL, which represents the top level design developmental configuration. The top level CSCI architecture section should include graphics as needed to describe the process/task configuration and control/data flow.
- The CDR version of the SDD should contain (1) updates to the sections provided at PDR, (2) CSU design descriptions, and (3) the CDR version of the Ada/ADL, which represents the detailed design developmental configuration.
- Design description information for lower level subordinate units should be included in the SDFs. The detailed design should be reviewed by contractor and customer personnel via interactive design walkthroughs using the Ada/ADL supplemented by top level design graphics.
- Generate the SPS using the SDD and SDFs as the basis. The SPS will be the final updated SDD (no Ada/ADL appendix) and the final source code listings. The SDFs should be considered a component of the delivered product specification.
- Include system-level software architecture and performance information in a higher level document such as the System/Segment Design Document.

The outline for the SDD is listed in Table 6. The organization and contents of the SDD are supported by the SDF outline described earlier in this paper.

SUMMARY

TRW and the Government recognized early in the CCPDS-R contract that the documentation requirements of an Ada DOD-STD-2167/2167A project could overwhelm everyone, including document producers, reviewers, and data managers. The approach described in this paper generates information that is useful for both review and maintenance purposes. The deliverable documents are generated efficiently using the Software Development Files as a primary information source. This ensures a current and consistent documentation set throughout the software life cycle. Although there has been no formal customer maintenance experience to date for the system, TRW's experience in maintaining the incremental build deliveries has been positive, relying primarily on the source code and SDFs. The design documentation set described will satisfy the needs of a competent, Ada-trained customer maintenance team.

ACKNOWLEDGEMENTS

The CCPDS-R project invested over 3 years of pre-contract preparation, proposal generation, and concept definition before any of the concepts described in this paper could be put into practice. Since award of the contract, much thinking of the originally proposed methods has occurred, resulting in the project's current approach and success to date. Acknowledgements are due to Steve Patay, Bruce Kohl and Walker Royce, who were instrumental in helping define and sell the original innovative development approach, and to Don Andres, Joan Bebb, Tom Herman, Chase Dane and Patty Shishido for helping to make the original ideas work in practice. Also, TRW's Government counterparts at USAF/ESD are acknowledged for supporting this approach and for being open to suggestions for improvements.

BIOGRAPHY

Michael Springman is the Assistant Project Manager for Software Development on the CCPDS-R project. He is responsible for the development of over 500,000 Ada source lines to the Air Force for this real-time system. He received a BS in Mathematics and Physics from Southwest State University (Minnesota) in 1973 and an MS in Applied Mathematics and Computer Science from the University of Colorado (Boulder) in 1975. Mr. Springman has been at TRW for 13 years, where he has been involved on a variety of real-time C^3 and avionics software projects. He has been responsible for all aspects of the software development life cycle, including system and software requirements definition, software design and development, and requirements verification. He has been dedicated to CCPDS-R for the last three years, which included the conceptual definition, proposal, and contract startup activities.

REFERENCES

[Royce 1989-1] Royce, W. E., "Reliable, Reusable Ada Components for Constructing Large, Distributed Multi-Task Networks: Network Architecture Services (NAS)", *TRI-Ada Proceedings*, Pittsburgh, October 1989.

[Royce 1989-2] Royce, W. E., "TRW's Ada Process Model For Incremental Development of Large Software Systems", *TRI-Ada Proceedings*, Pittsburgh, October 1989.

[2167A] "DOD-STD-2167A: Military Standard, Defense System Software Development", 29 February 1988.

Table 5: 2167/2167A Documentation Mapping

Design Topic	Standard 2167 Documentation	CCPDS-R Tailored 2167 Documentation	Standard 2167A Documentation
SRS Requirements Allocation	STLDD 3.2	STLDD 3.2	SDD 3.1
CSCI Architecture Overview	STLDD 3.1, 3.4	STLDD 3.1, 3.4	SDD 3.1
System States and Modes	STLDD 3.1, 3.4	STLDD 3.1, 3.4	SDD 3.1
Memory/Processing Allocation	STLDD 3.3	System Desc Doc	SDD 3.1
CSCI Data	STLDD 3.5	STLDD 3.5	SDD 5
CSCI Data Files	STLDD 3.5	STLDD 3.5	SDD 6
TLCSC (CSC) Descriptions	STLDD 3.6	STLDD 3.6	SDD 3.2
LLCSC (CSU) Descriptions	SDDD	SDFs (Ref. in SDDD)	SDD 4
Unit Descriptions	SDDD	SDFs (Ref. in SDDD)	SDD 4
Source Listings	SPS	SPS	SPS
Measured Resource Utilization	N/A	TPM Report	SPS

Table 6: SDD Outline

1. SCOPE

1.1 Identification

1.2 Purpose

1.3 Introduction

2. REFERENCED DOCUMENTS

3. TOP LEVEL DESIGN DESCRIPTION

3.1 CSCI Overview

3.1.1 CSCI Architecture

3.1.2 System States and Modes

3.1.3 Memory and Processing Time Allocation

3.2 CSCI Design Description

3.2.X CSC X (from CSC X SDF "Design Description" Section)

3.2.X.Y Sub-level CSC Y (from CSC Y SDF "Design Description" section)

4. DETAILED DESIGN

4.X CSC X

4.X.Y CSU Y (from CSU Y SDF "Design Description" section; can reference SDF if SDF is delivered)

5. CSCI DATA

6. CSCI DATA FILES

APPENDIX

PDR: PDR version of Ada/ADL

CDR: CDR version of Ada/ADL

FCA/PCA: Empty (Ada source code is separate appendix of SPS)