ESD-TR-89-315

M89-55

A MITRE View of the CCPDS-R Software Development Process

By

Gerard R. LaCroix

February 1990

Prepared for
Program Manager, CCPDS-R Program
Space and Missile Warning Systems Program Office
Electronic Systems Division
Air Force Systems Command
United States Air Force
Hanscom Air Force Base, Massachusetts

Project No. 6030
Prepared by
The MITRE Corporation
Bedford, Massachusetts
Contract No. F19628-89-C-0001

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

RICHARD K. JENNINGS, MAJ, USAF
Chief, Software Engineering
CCPDS-R Program

FOR THE COMMANDER

Paul Heartquist

PAUL E. HEARTQUIST, LT COL, USAF
Program Manager, CCPDS-R Program
Space and Missile Warning Systems
Program Office

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release;
distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
M89-55
ESD-TR-89-315

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
The MITRE Corporation

6b. OFFICE SYMBOL *(If applicable)*

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State, and ZIP Code)*
Burlington Road
Bedford, MA 01730

7b. ADDRESS *(City, State, and ZIP Code)*

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
Program Manager, (continued)

8b. OFFICE SYMBOL *(If applicable)*
ESD/SRW

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
F19628-89-C-0001

8c. ADDRESS *(City, State, and ZIP Code)*
Electronic Systems Division, AFSC
Hanscom AFB, MA 01731-5000

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
	6030		

11. TITLE *(Include Security Classification)*
A MITRE View of the CCPDS-R Software Development Process

12. PERSONAL AUTHOR(S)
LaCroix, Gerard R.

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM ________ TO ________

14. DATE OF REPORT *(Year, Month, Day)*
1990 February

15. PAGE COUNT
46

16. SUPPLEMENTARY NOTATION

aac,

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Ada programming, Software Development, (SWD)
Ada Software Development,
~~CCPDS-R~~

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This briefing, prepared at the request of the MITRE Software Center, presents the MITRE Project Leader's view of the software development approach being followed by TRW for development of the CCPDS-R System. The briefing in intended as a follow-up to the view presented by TRW ~~in April~~ 1989 at a seminar, hosted by ESD/MITRE, entitled "Managing to Realize the Potential of Ada." Keywords.

Command Center Processing and Display System

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
□ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT □ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
Judith Schultz

22b. TELEPHONE *(Include Area Code)*
(617) 271-8087

22c. OFFICE SYMBOL
Mail Stop D135

83 APR edition may be used until exhausted.
All other editions are obsolete.

8a. CCPDS-R Program, Space and Missile Warning Systems Program Office

ACKNOWLEDGMENTS

This document has been prepared by The MITRE Corporation under Project No. 6030, Contract No. F19628-89-C-0001. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, United States Air Force, Hanscom Air Force Base, Massachusetts 01731-5000. The author wishes to thank the many contributions of Bob Howe to the preparation of this briefing and its accompanying text.

A MITRE View of the CCPDS-R* Software Development Process

Gerard R. LaCroix
October 1989

* CCPDS-R -- Command Center Processing
and Display System - Replacement

MITRE

Briefing Outline

- Background
- Review of CCPDS-R Architecture
- Characteristics and Assessment of Software Development Approach
- Areas for Improvement
- Conclusions

MITRE

The briefing is organized into five parts. To provide context, several vugraphs will be used to describe the background of the Command Center Processing and Display System - Replacement (CCPDS-R) Program and to review the CCPDS-R hardware and software architecture. The bulk of the briefing will characterize and assess the CCPDS-R software development approach. This assessment will be followed by a listing of areas for improvement, and conclusions.

Briefing Purpose

- To describe the CCPDS-R software development approach

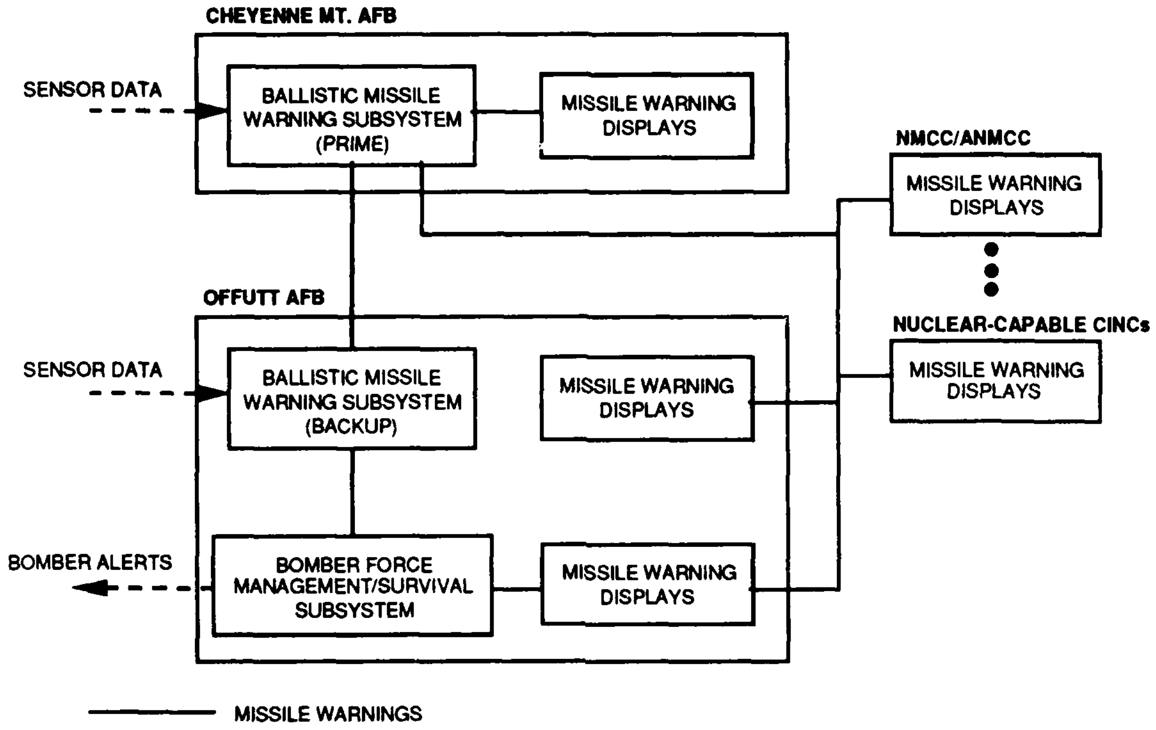
and

- Assess benefits and limitations based on experience to date

MITRE

The briefing has two main purposes. The first purpose is to describe the approach, and the second is to assess the benefits and limitations based on experience to date.

CCPDS-R System Overview



MITRE

The CCPDS-R consists of three subsystems. The Ballistic Missile Warning Subsystem will be located at Cheyenne Mountain and at Offutt AFB in the Offutt Processing and Correlation Center. The facility at Offutt AFB is intended as a backup to the prime missile warning facility at Cheyenne Mountain. The second CCPDS-R subsystem consists of remote missile warning displays to be located throughout the world, principally at the National Military Command Center (NMCC) and its alternate, and at the command centers of the various nuclear-capable theater commanders. The third CCPDS-R subsystem is the Strategic Air Command (SAC) Bomber Force Management/Survival Subsystem to be located at SAC Headquarters. Its mission is to manage and reconstitute the SAC bomber fleet in response to a ballistic missile attack on either the United States or other areas of interest.

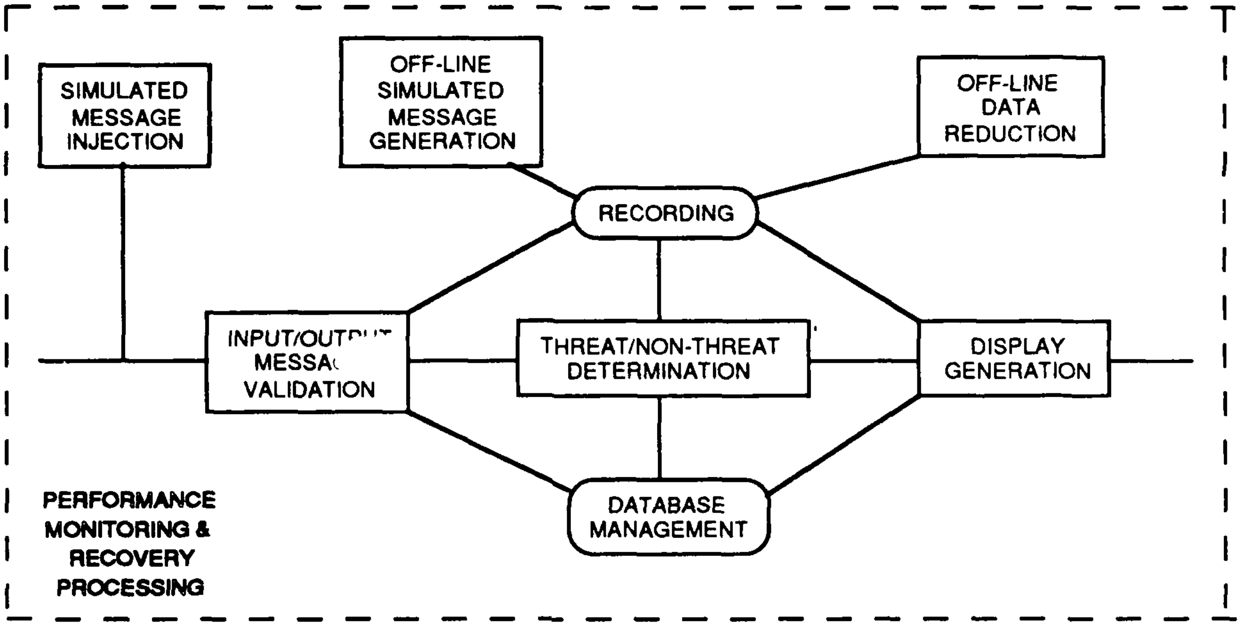
Operational Objectives

- Common displays at all user locations
- Direct sensor data to users
- Throughput and response time for twice predicted load
- Functionally improved and more accurate algorithms
- Color graphic and tabular displays
- State-of-the-art processing and display for maintainability
- Fail-safe missile warning backup for survivability
- Ada implementation for maintainability and reusability

MITRE

The operational objectives of the CCPDS-R are listed here for reference. Except for the last item dealing with Ada, these will not be addressed further in this briefing. With regard to the Ada computer programming language, it should be mentioned that all TRW-developed code for the CCPDS-R is being written in Ada. Ada is being used to enhance the maintainability and reusability of the CCPDS-R software. More will be said on these topics later in this presentation.

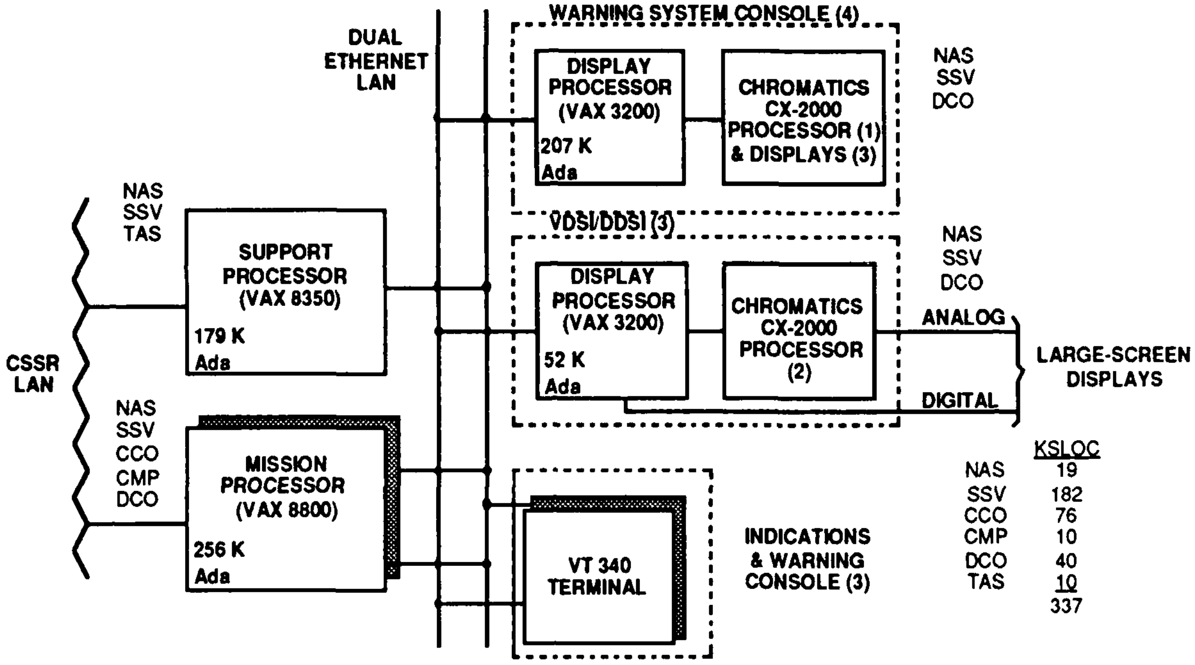
Missile Warning Processing Functions


SIMULATED MESSAGE INJECTION
OFF-LINE SIMULATED MESSAGE GENERATION
OFF-LINE DATA REDUCTION
RECORDING
INPUT/OUTPUT MESSAGE VALIDATION
THREAT/NON-THREAT DETERMINATION
DISPLAY GENERATION
DATABASE MANAGEMENT
PERFORMANCE MONITORING & RECOVERY PROCESSING

MITRE

The Missile Warning Subsystem is the first of the three CCPDS-R subsystems to be implemented. The heart of the Missile Warning Subsystem functionality consists of a processing thread involving input/output message validation, threat/non-threat determination, and display generation. This thread represents the external interfaces, the missile warning algorithms, and all operational displays. It is the thread that has the strictest time-critical requirements imposed upon it, i.e., three seconds from receipt of an external message to display, and one second response time to operator requests for displayed information. The other functions indicated on the vugraph include performance monitoring and recovery processing, and several off-line functions.

Missile Warning Subsystem Architecture

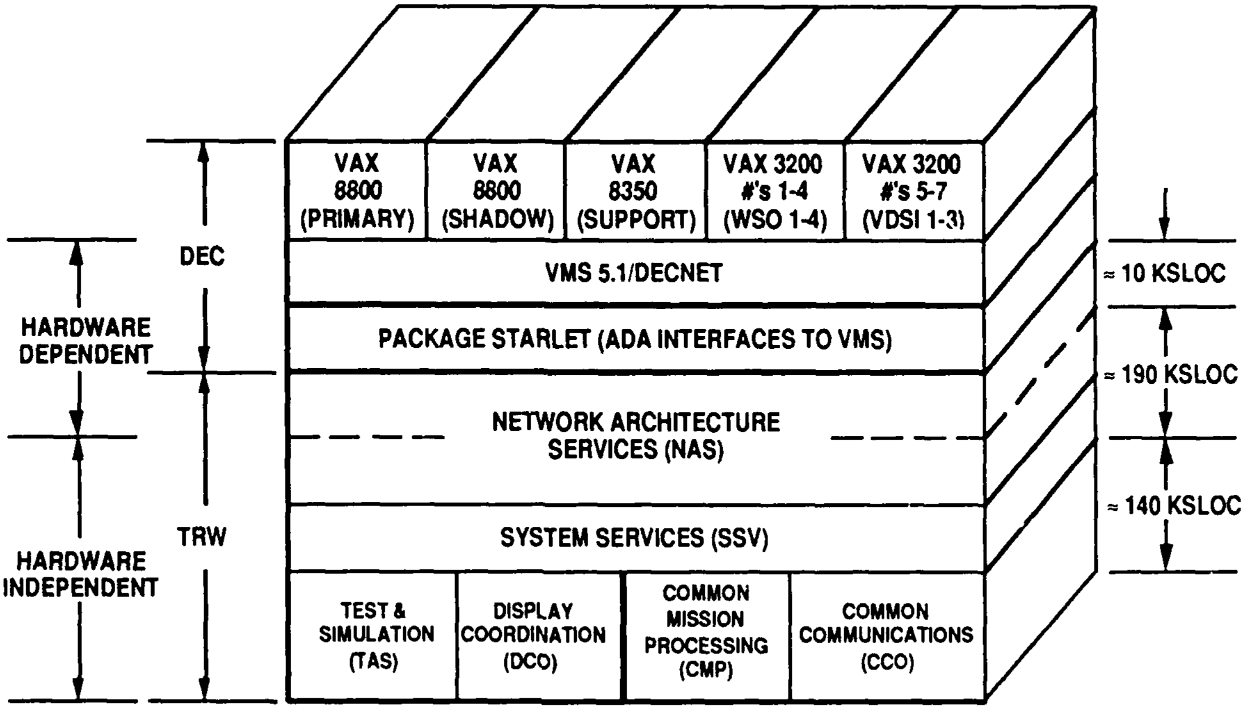

DUAL ETHERNET LAN
WARNING SYSTEM CONSOLE (4)
DISPLAY PROCESSOR (VAX 3200)
207 K Ada
CHROMATICS CX-2000 PROCESSOR (1) & DISPLAYS (3)
NAS
SSV
DCO
VDSI/DDSI (3)
DISPLAY PROCESSOR (VAX 3200)
52 K Ada
CHROMATICS CX-2000 PROCESSOR (2)
NAS
SSV
DCO
ANALOG
DIGITAL
LARGE-SCREEN DISPLAYS
NAS
SSV
TAS
SUPPORT PROCESSOR (VAX 8350)
179 K Ada
CSSR LAN
NAS
SSV
CCO
CMP
DCO
MISSION PROCESSOR (VAX 8800)
256 K Ada
VT 340 TERMINAL
INDICATIONS & WARNING CONSOLE (3)

	KSLOC
NAS	19
SSV	182
CCO	76
CMP	10
DCO	40
TAS	10
	337

MITRE

The Missile Warning Subsystem hardware suite comes largely from the Digital Equipment Corporation (DEC). The VAX 8800 Mission Processor is used for communications and algorithmic processing, the VAX 3200 is used for display processing, and the VAX 8350 Support Processor is used for off-line functions and simulated message injection. The software consists of six Computer Software Configuration Items (CSCIs) for the TRW-developed software, and Commercial Off-the-Shelf (COTS) software provided with the DEC hardware family. This software includes VAX VMS and Rdb. The size of the TRW software for the Missile Warning Subsystem is 337K source lines of code (SLOC), all written in Ada.

Missile Warning Subsystem Hardware/Software Hierarchy



VAX 8800 (PRIMARY)

VAX 8800 (SHADOW)

VAX 8350 (SUPPORT)

VAX 3200 #'s 1-4 (WSO 1-4)

VAX 3200 #'s 5-7 (VDSI 1-3)

VMS 5.1/DECNET

PACKAGE STARLET (ADA INTERFACES TO VMS)

NETWORK ARCHITECTURE SERVICES (NAS)

SYSTEM SERVICES (SSV)

TEST & SIMULATION (TAS)

DISPLAY COORDINATION (DCO)

COMMON MISSION PROCESSING (CMP)

COMMON COMMUNICATIONS (CCO)

DEC

TRW

HARDWARE DEPENDENT

HARDWARE INDEPENDENT

≈ 10 KSLOC

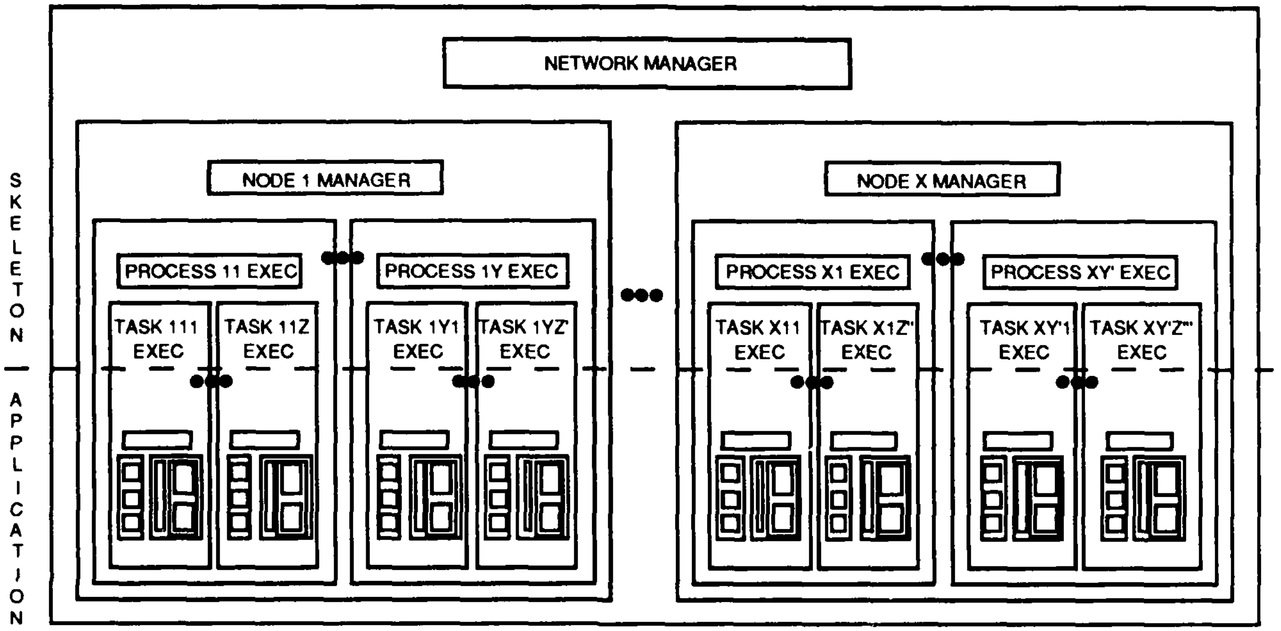
≈ 190 KSLOC

≈ 140 KSLOC

MITRE

Hierarchically, the TRW-developed software has been constructed to isolate DEC hardware and operating system (VMS/Starlet) dependencies to essentially one CSCI, i.e., Network Architecture Services (NAS), and within that CSCI to about 10K SLOC. TRW has also isolated the more difficult features of Ada such as tasking and rendezvous to the NAS CSCI which is about 20K SLOC. As a result, TRW has been able to develop the applications software using relatively inexperienced Ada designers and coders, while limiting the need for expert Ada professionals to a core group of about 10 individuals who are concerned with the design and coding of NAS. Due to its complexity, NAS will not be easy to maintain, but should not require frequent extensive modification as will the applications software which can be severely impacted by changes in operational requirements.

Software Architecture Skeleton & Applications Software Relationship



MITRE

To facilitate the development of the applications software, TRW devised a top-down architecture skeleton consisting of generically instantiated software for Node Managers, Process Executives, and Task Executives. The skeleton, which was developed early in the CCPDS-R Program, implements a message-based design in which processors (or nodes) communicate with other processors by means of messages. A similar scheme exists for VMS processes resident in a processor to communicate with other VMS processes. Ada tasks resident within a process also communicate with other Ada tasks by means of messages. The forms of communication are referred to as Inter-Task Communications (ITC) Level 1 (Task-to-Task), Level 2 (Process-to-Process), and Level 3 (Node-to-Node). The overhead associated with such an architecture, specially for node-to-node communications, is higher than would be obtained using VMS alone. MITRE estimates the overhead at about 30-35 percent CPU utilization vs. 10-15 percent for VMS. However, the overhead penalty is offset by the ease with which applications software can be added by the developers who need not know, nor be concerned with, how Ada tasks communicate with other tasks; rather, they need only be involved with the design of Ada task bodies.

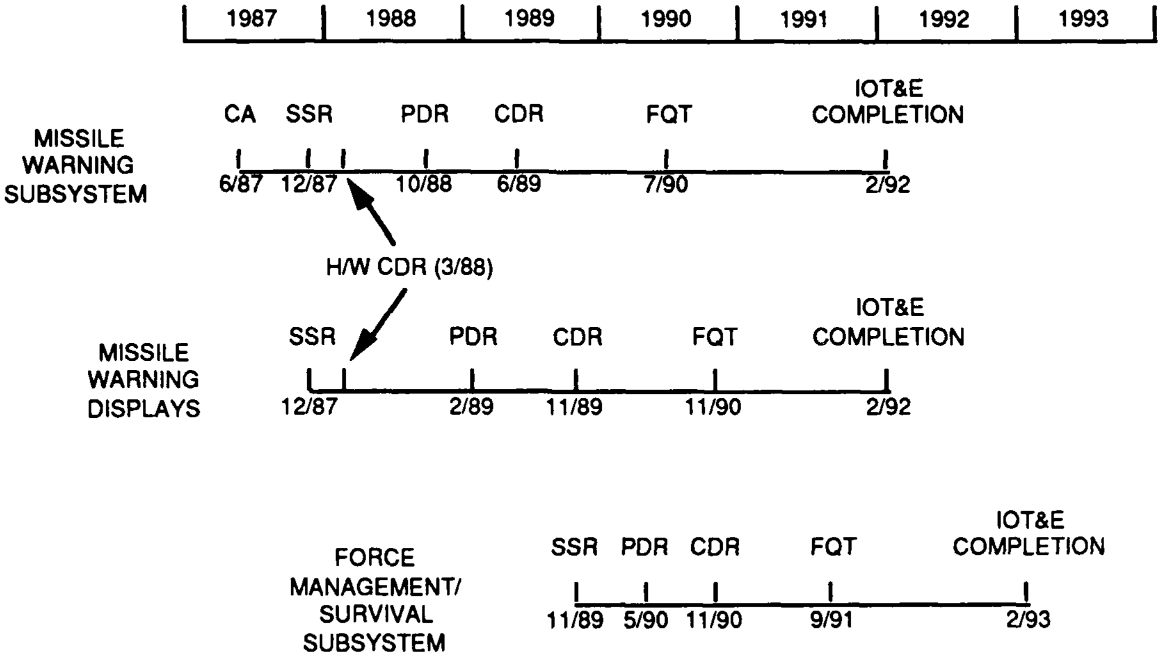
Software Architecture Skeleton

- Consists of generic software instantiated at compile time
- Identifies nodes, processes, tasks, and task connections (circuits and sockets)
- Composed of network manager, node managers, process executives, task executives, and a database controlling inter-task communications
- Encapsulates difficult features of Ada, such as tasking
- Permits early integration with stubs, as appropriate, and step-wise integration with actual applications software thereafter
- Reused repeatedly as applications are added
- Applications software is
  - Independent of DEC hardware architecture
  - Consists of logically interconnected software modules

MITRE

This vugraph summarizes the salient features of the software architecture skeleton. Instantiation refers to a process whereby software operations are converted from generic form through the specification of data types at compile time. The skeleton permits software integration to take place top-down with stubs used for missing applications. The skeleton also provides a source of early software reliability data because it is used repeatedly by each of the developers throughout the applications software development cycle. The skeleton is the mechanism by which the applications software is isolated from hardware and operating system dependencies. (Note: Small portions of the CCO CSCI contain hardware-dependent code for the Excelan and Simpact interfaces. Likewise, the DCO CSCI contains some hardware-dependent code for the Chromatics display generator, and the large-screen digital and video interface equipments.)

Major CCPDS-R Milestones

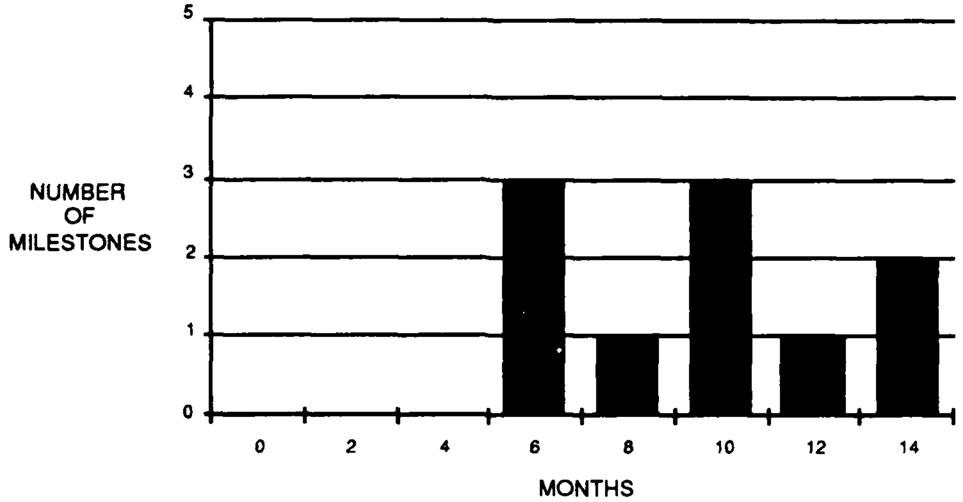


MITRE

This next portion of the briefing attempts to characterize and assess the overall software development approach.

The first characteristic of importance is the development schedule. CCPDS-R is favored with a realistic schedule in which sufficient time has been allocated between the major review milestones for each of the three CCPDS-R subsystems to accomplish the work required. Although each individual schedule is reasonable, a moderate degree of schedule overlap (concurrency) exists among the three developments. This may cause key resources (facilities and personnel) to be stretched too thin. The first signs may appear in late 1989 when major activities on all three subsystems will be proceeding in parallel for the first time.

Months between Major Milestones*



9.4 MONTHS AVE. BETWEEN MILESTONES

17.7 MONTHS AVE. FROM SSR TO CDR

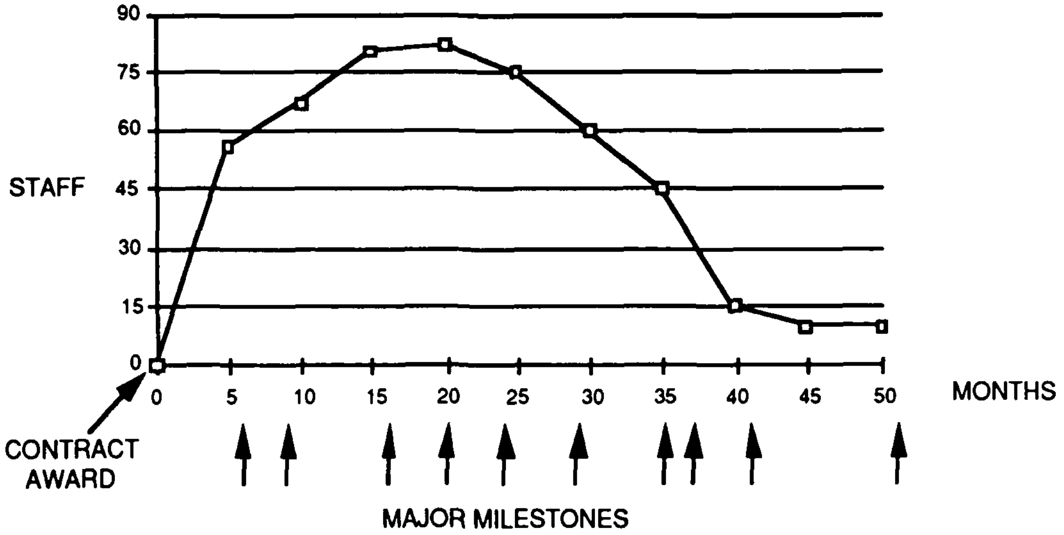
29.3 MONTHS AVE. FROM SSR TO FQT

* CONTRACT AWARD TO SSR
SSR TO PDR
PDR TO CDR
CDR TO FQT

MITRE

To further highlight the reasonableness of the separate development schedules, the numbers of months between major software milestones were plotted. This chart shows that in all instances at least six months have been scheduled between the major milestones indicated in the footnote. The average is 9.4 months. The average for the design phases (i.e., Software Specification Review (SSR) to Critical Design Review (CDR)) of each of the three subsystems is 17.7 months, and the average duration of the software development phases (design through test) is 29.3 months.

Software Staffing*



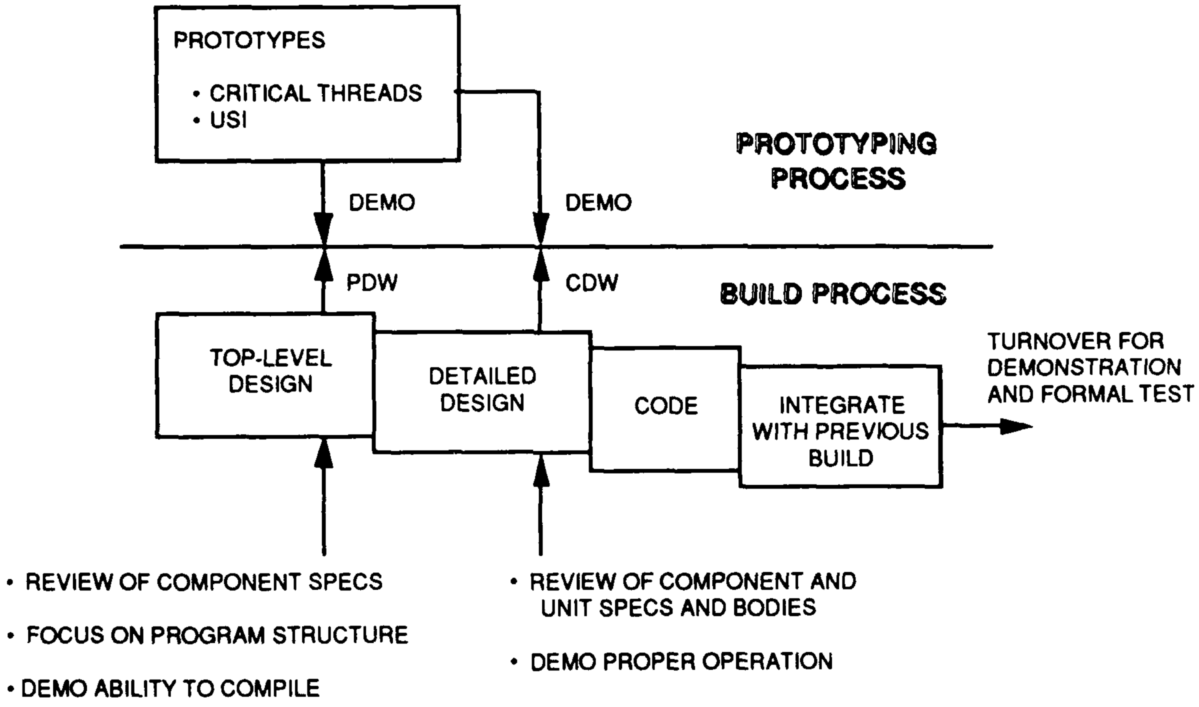
TOTAL STAFF-MONTHS = 2531
AVE. MONTHLY STAFF = 49.6

* CONTRACT AWARD TO FQT OF FORCE MANAGEMENT/ SURVIVAL SUBSYSTEM

MITRE

The second characteristic which the CCPDS-R Program has in its favor is software staffing. The staff size rises quickly and stays in the 60 - 80 range for the bulk of the schedule, from contract award to completion of Formal Qualification Testing (FQT) of the third subsystem. This staff complement is of manageable proportions. It includes a cadre of about 10 Ada experts who are responsible for the more difficult software (i.e., NAS) and who have been with the NAS development from its inception as a TRW Independent Research and Development (IR&D) project in 1983 through the CCPDS-R Concept Development Phase (1985 -1986), and since contract award for the current Full-Scale Development Phase (June 1987). This cadre also provides Ada training for the other software professionals working on the CCPDS-R Program.

Prototyping and Build Processes



PROTOTYPES
• CRITICAL THREADS
• USI
DEMO
DEMO
PROTOTYPING PROCESS
PDW
CDW
BUILD PROCESS
TOP-LEVEL DESIGN
DETAILED DESIGN
CODE
INTEGRATE WITH PREVIOUS BUILD
TURNOVER FOR DEMONSTRATION AND FORMAL TEST
• REVIEW OF COMPONENT SPECS
• FOCUS ON PROGRAM STRUCTURE
• DEMO ABILITY TO COMPILE
• REVIEW OF COMPONENT AND UNIT SPECS AND BODIES
• DEMO PROPER OPERATION

MITRE

The third characteristic and the one that tends to distinguish the CCPDS-R Program from others is the software prototyping and build process culminating in functional capability demonstrations. The prototyping of critical performance components and the prototyping of the User-System Interface (USI) are enabling TRW to better define the development requirements for the software. The builds, of which there are about ten, are in essence mini-developments, each with top-level and detailed design phases, a coding phase, and an integration/test phase. Each build has a Preliminary Design Walkthrough (PDW) and a Critical Design Walkthrough (CDW) in which a specialized technical audience of about 40 to 50 people, including the Government, participates. Each build can then be formally demonstrated to the Government in accordance with an approved Demonstration Plan which includes pass/fail criteria. The overall process focuses the work on near-term products, visible to all, and forces early accountability from the developers. The functional contents of the demonstrations are specified by the Government in the Statement of Work (SOW) *prior to contract award to insure* that the difficult parts of the job are accomplished first and to insure that the builds are operationally useful.

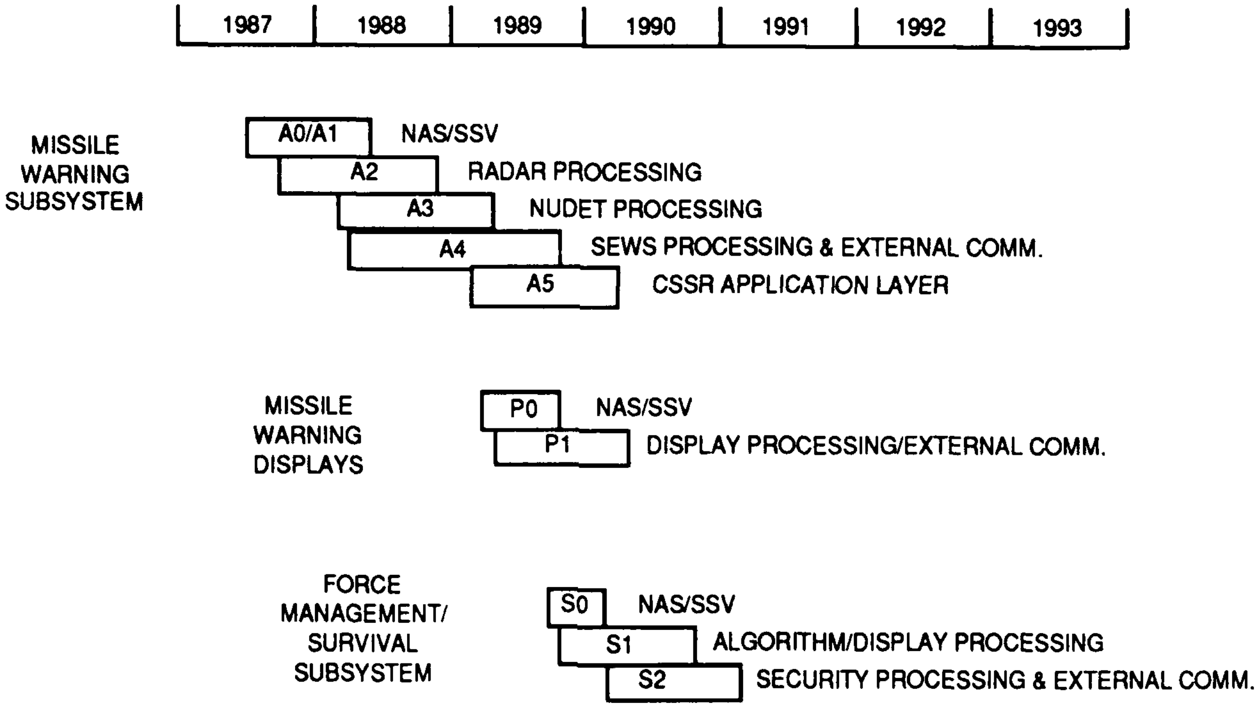
Prototyping of Components

- Define required inputs and outputs
- Define top-level architecture (Process and Task Specifications) to achieve partial component functionality
- Define task-to-task interfaces
- Instantiate Software Architecture Skeleton (SAS)
- Demonstrate ability to compile critical components used in Task Bodies at PDW
- Further define Task Bodies to achieve full component functionality
- Demonstrate proper operation at CDW

MITRE

The software prototyping is achieved in two phases. Prior to PDW, required inputs and outputs are defined, along with process and task specifications, and task-to-task interfaces. The software architecture skeleton is instantiated and the critical software components are compiled. After PDW, the task bodies are further defined. When CDW is conducted, proper operation of the prototyped component is demonstrated.

Software Builds



1987 1988 1989 1990 1991 1992 1993

MISSILE WARNING SUBSYSTEM

A0/A1 NAS/SSV

A2 RADAR PROCESSING

A3 NUDET PROCESSING

A4 SEWS PROCESSING & EXTERNAL COMM.

A5 CSSR APPLICATION LAYER

MISSILE WARNING DISPLAYS

P0 NAS/SSV

P1 DISPLAY PROCESSING/EXTERNAL COMM.

FORCE MANAGEMENT/ SURVIVAL SUBSYSTEM

S0 NAS/SSV

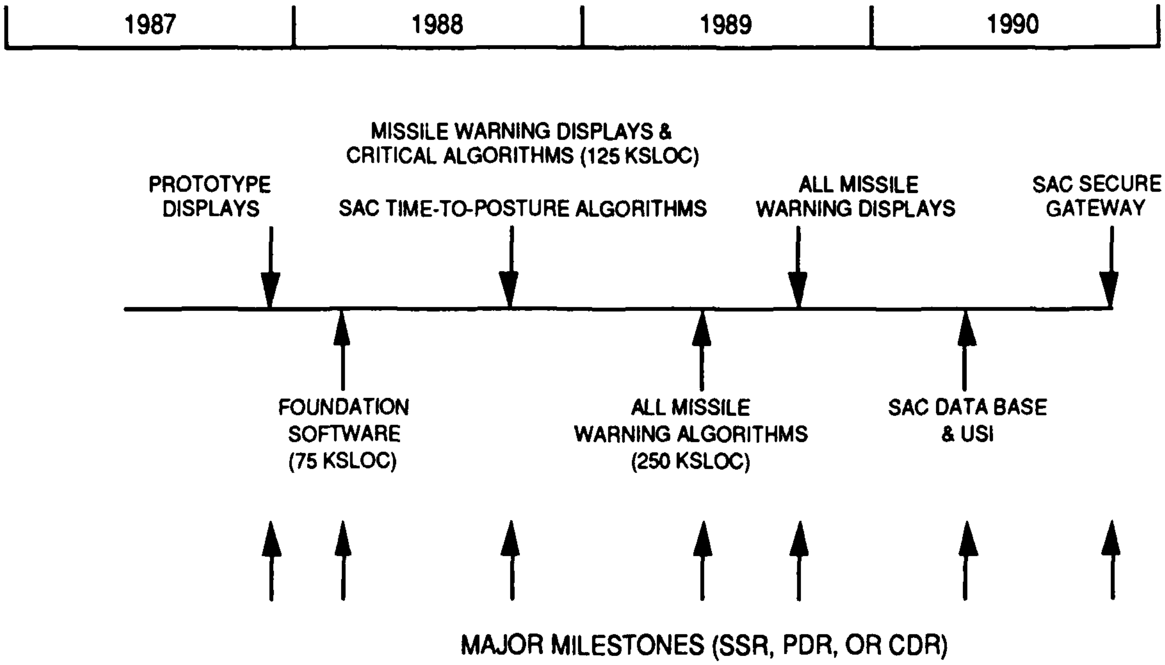
S1 ALGORITHM/DISPLAY PROCESSING

S2 SECURITY PROCESSING & EXTERNAL COMM.

MITRE

Altogether, there are ten software builds currently defined for the three CCPDS-R subsystems. The first build for each subsystem contains the foundation software (i.e., NAS and Software Architecture Skeleton) on top of which applications software may be overlaid in the later builds. This process of iteration is permitting TRW and the Government to identify procedural problems (and differences in expectation) and to resolve them early. The experience gained in the first few builds have been and are being applied to the remaining builds resulting in a more efficient design, better documentation, and a cost-effective test process.

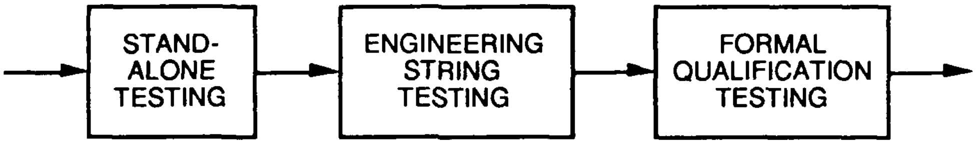
Capability Demonstrations


1987
1988
1989
1990
MISSILE WARNING DISPLAYS & CRITICAL ALGORITHMS (125 KSLOC)
PROTOTYPE DISPLAYS
SAC TIME-TO-POSTURE ALGORITHMS
ALL MISSILE WARNING DISPLAYS
SAC SECURE GATEWAY
FOUNDATION SOFTWARE (75 KSLOC)
ALL MISSILE WARNING ALGORITHMS (250 KSLOC)
SAC DATA BASE & USI
MAJOR MILESTONES (SSR, PDR, OR CDR)

MITRE

Seven capability demonstrations are specified in the CCPDS-R SOW. The demonstrations conducted to date have been invaluable in providing concrete visibility into true development progress. Problems identified by the demonstrations have been and are being used to formulate corrective actions. The demonstrations are tied to formal reviews, such as SSR, Preliminary Design Review (PDR), or CDR, and to the award fee structure. TRW distributes half of the award fee as bonuses to project personnel based on merit.

Overall Software Test Approach



STAND-ALONE TESTING

ENGINEERING STRING TESTING

FORMAL QUALIFICATION TESTING

- DETAILED SRS REQUIREMENTS
- PERFORMED ON BUILD COMPONENTS

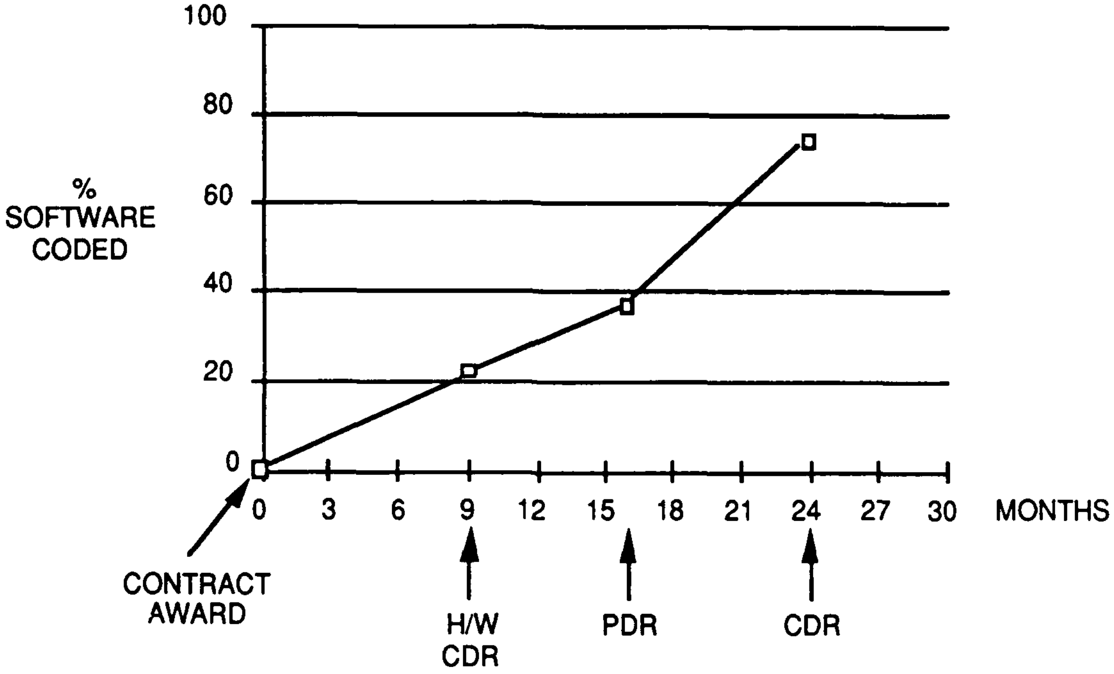
- END-TO-END REQUIREMENTS
- PERFORMED ON BUILDS

- END-TO-END REQUIREMENTS
- PERFORMED ON TOTAL SOFTWARE
- REGRESSION TESTING IN AREAS CHANGED

MITRE

Two types of formal tests are being applied to each of the CCPDS-R software builds. Stand-alone tests are performed on build components to verify satisfaction of detailed Software Requirement Specification (SRS) requirements, such as algorithms (equations, logic, etc.) and message validation (e.g., field contents). About 25 percent of the SRS requirements are expected to be verified by stand-alone tests. In addition, engineering string tests are performed to verify satisfaction of SRS end-to-end functional requirements, such as ability to receive a missile launch event message, process it, and generate a threat fan for display. Engineering string tests implicitly verify many detailed SRS requirements without the need for detailed stand-alone testing. Approximately 40-45 percent of the SRS requirements are expected to be verified by engineering string tests. At the conclusion of the last build, FQT will be performed on the total subsystem software against SRS requirements, both detailed and end-to-end. Actual FQT testing may be limited to new functionality, implemented since testing of the last build, and to regression testing of changed areas. Other SRS requirements may be verified through analysis of the prior stand-alone and engineering string test results obtained from testing each build. Only about 30-35 percent of the SRS requirements are expected to be specifically verified at FQT.

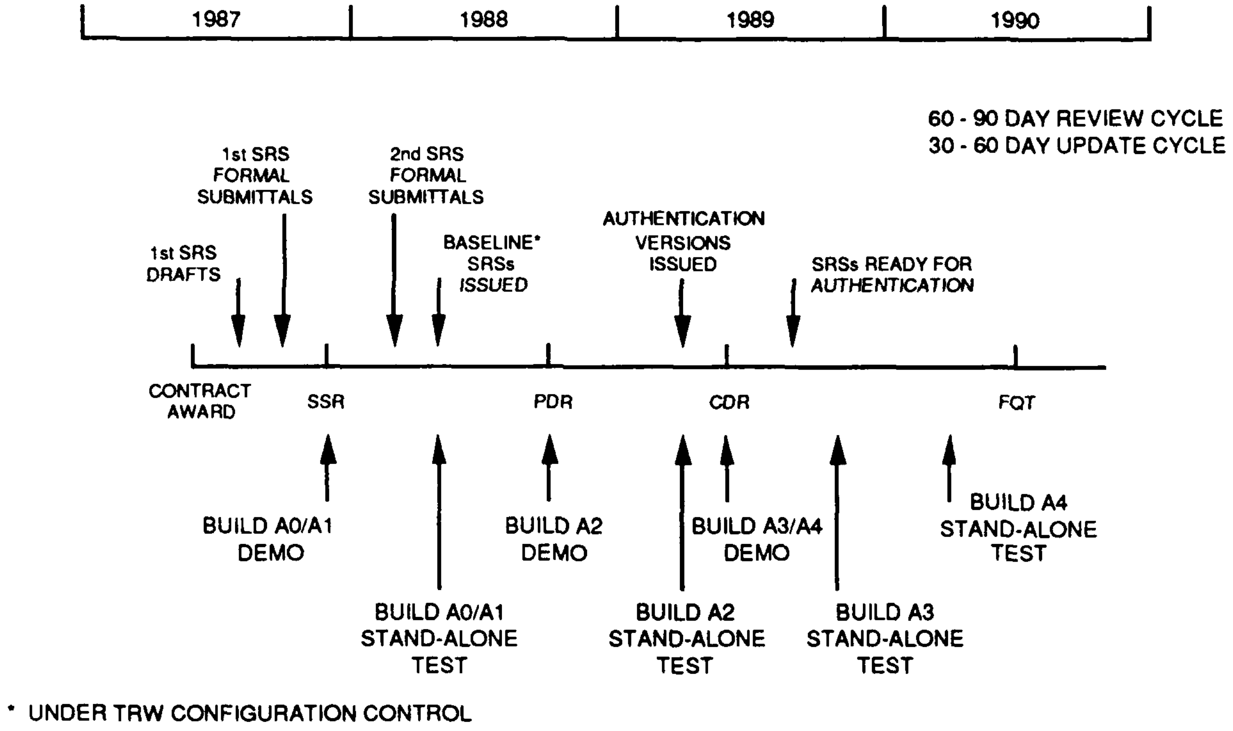
Missile Warning Software Coded vs. Time



MITRE

This chart plots the percentage amount of the Missile Warning Subsystem software coded to date as a result of the build process. At CDR, which took place in June 1989, approximately 75 percent of the software was demonstrated. This experience runs counter to the usual case where CDR is a review of detailed design and the outcome of CDR is approval to proceed to code. On the CCPDS-R Program, the Government specified that a build approach was to be taken throughout the development cycle. As a result, reviews like PDR and CDR have become the culmination of a series of PDWs and CDWs, respectively. Formal reviews, like PDR and CDR, continue to provide an opportunity for all parties to meet periodically to review the progress made to date, and to participate in the formal demonstrations conducted in conjunction with the reviews.

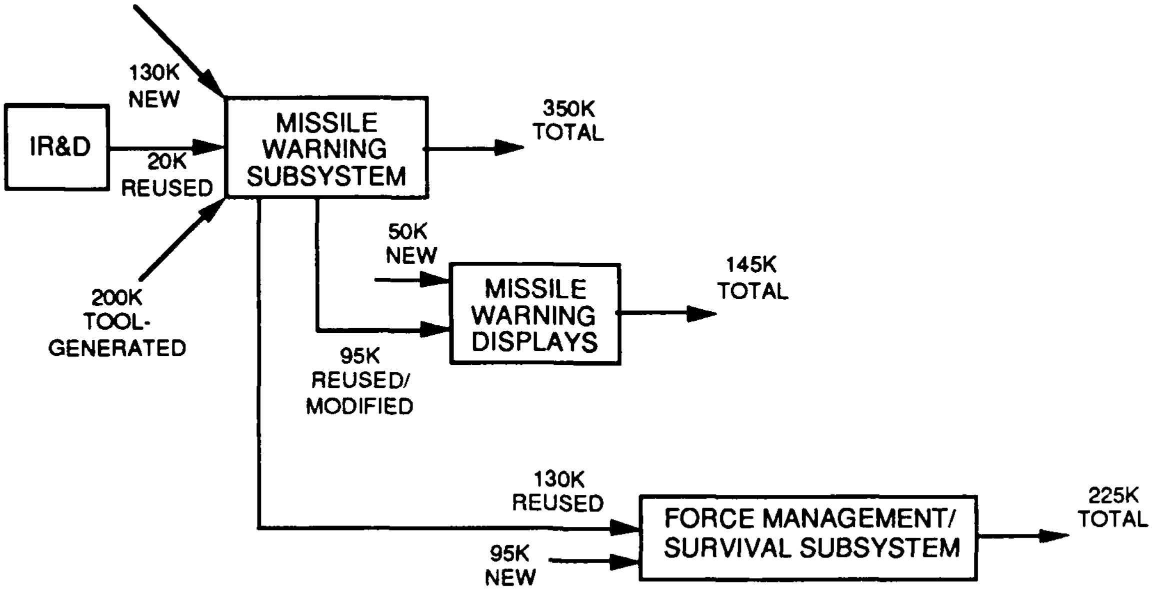
SRS Review Process vs. Build Demos and Stand-Alone Tests


1987
1988
1989
1990
60 - 90 DAY REVIEW CYCLE
30 - 60 DAY UPDATE CYCLE
1st SRS FORMAL SUBMITTALS
2nd SRS FORMAL SUBMITTALS
AUTHENTICATION VERSIONS ISSUED
1st SRS DRAFTS
BASELINE* SRSs ISSUED
SRSs READY FOR AUTHENTICATION
CONTRACT AWARD
SSR
PDR
CDR
FQT
BUILD A0/A1 DEMO
BUILD A2 DEMO
BUILD A3/A4 DEMO
BUILD A4 STAND-ALONE TEST
BUILD A0/A1 STAND-ALONE TEST
BUILD A2 STAND-ALONE TEST
BUILD A3 STAND-ALONE TEST
* UNDER TRW CONFIGURATION CONTROL

MITRE

The iterative nature of the CCPDS-R software development extends also to the documentation. The review and approval process for the Missile Warning Subsystem SRSs took about two years from receipt of early drafts to readiness for authentication. This process included four formal submittals of the SRSs prior to CDR, each with a review and comment resolution cycle. The SRSs were first placed under TRW configuration control in May 1988 as a prerequisite for conduct of the first formal stand-alone tests. They remain under TRW configuration control and are not expected to be authenticated (i.e., put under the Government's configuration control) until shortly before FQT.

Software Reuse (Case 1)

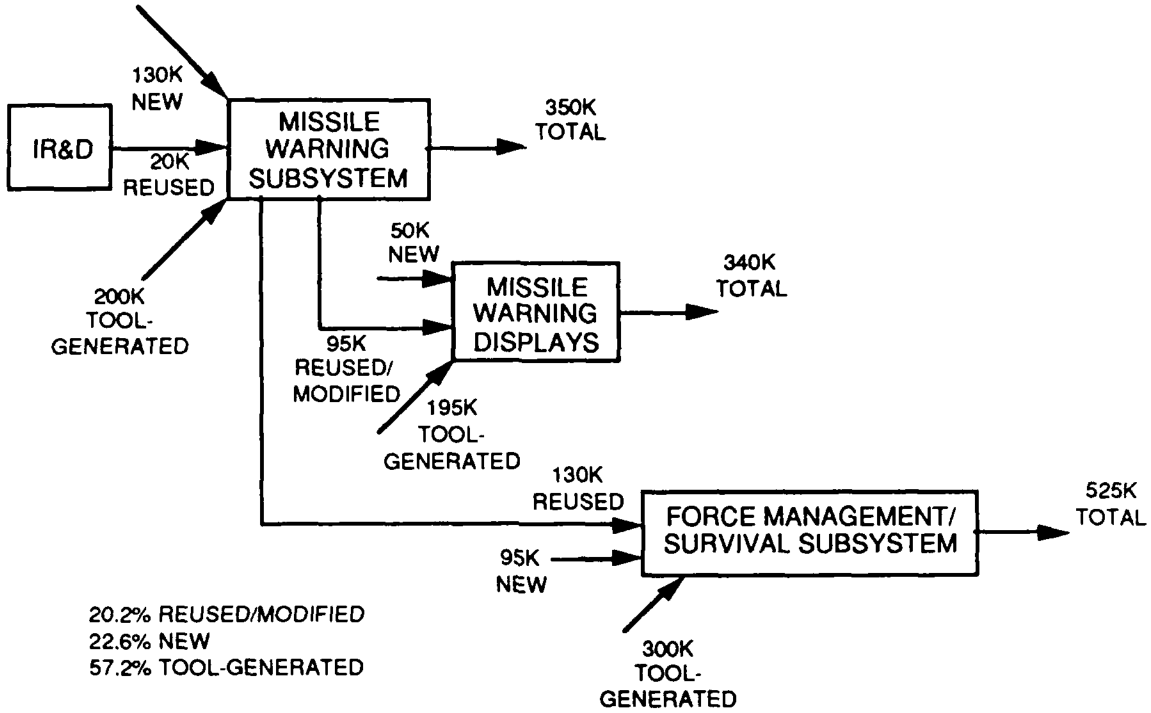


34.0% REUSED/MODIFIED
38.2% NEW
27.8% TOOL-GENERATED

MITRE

This chart depicts TRW's current projection of software reuse in which a total of 210K SLOC are reused from the development of one CCPDS-R subsystem to the next. Due to similarities between portions of the Missile Warning Subsystem and Missile Warning Displays Subsystem, an additional 35K SLOC can be modified. This results in about 34 percent of the total 720K SLOC being reused or modified. Approximately 28 percent of the CCPDS-R software is expected to be generated by tools.

Software Reuse (Case 2)

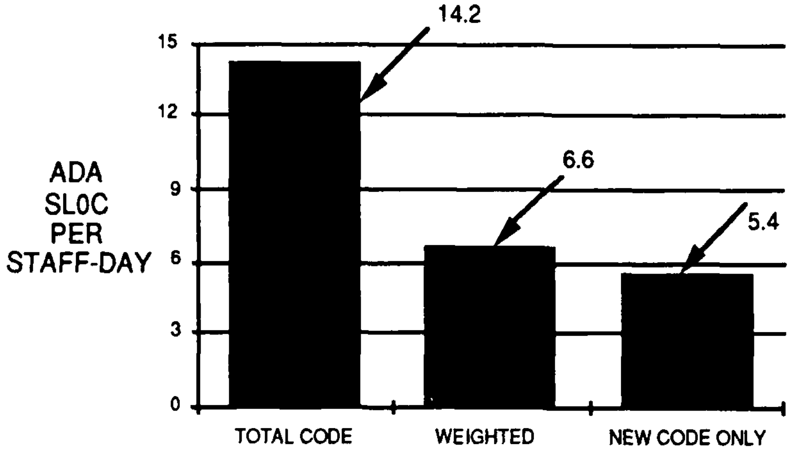


20.2% REUSED/MODIFIED
22.6% NEW
57.2% TOOL-GENERATED

MITRE

A second scenario is depicted on this chart which assumes the existence of additional tool-generated code in the same proportion as for the Missile Warning Subsystem (Case 1). Under this case, the same amount of code (245K SLOC) is reused or modified. However, the percentage is reduced to 20.2 percent due to the increased total code (1215K SLOC). In either absolute terms or as a percentage of the total software, the amount of projected reuse is not insignificant.

Software Productivity



NEW	275K
MODIFIED	35K
REUSED	210K
TOOL-GENERATED	200K
TOTAL	720K

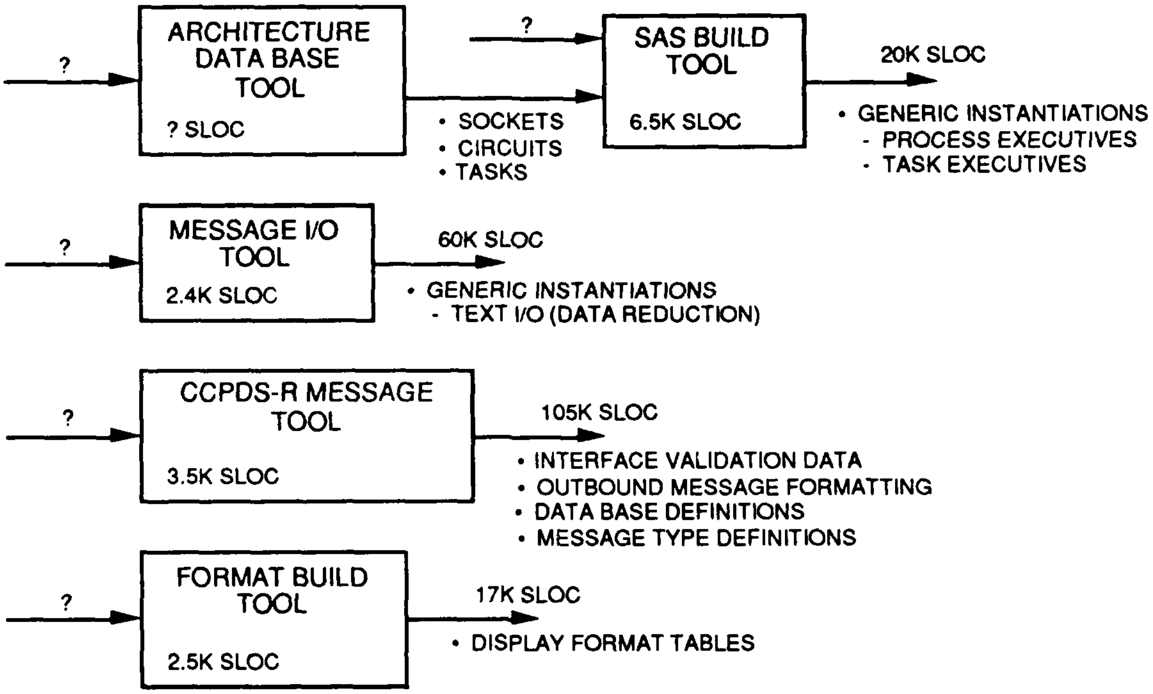
TOTAL CODE = NEW + MODIFIED + REUSED + TOOL-GENERATED
WEIGHTED = NEW + .5 (MODIFIED) + .1 (REUSED + TOOL-GENERATED)
NEW CODE ONLY = NEW

SLOC = SPECIFICATION CARRIAGE RETURNS + BODY SEMI-COLONS

MITRE

The amount of reused/modified code coupled with the large amount of tool-generated code is enhancing TRW's software productivity, measured from Contract Award to FQT of the third subsystem. Current estimates range from a low of 5.4 SLOC per staff-day, if labor credit is given to "new code only," to a high of 14.2 SLOC per staff-day, if all code is equally weighted with respect to the amount of labor needed to produce it. The true productivity is somewhere in between. Under one scheme where tool-generated code is weighted at 10 percent of the labor required for new code, the productivity figure is 6.6 SLOC per staff-day. Under another in which tool-generated code is weighted at 60 percent (per most recent TRW data), the productivity figure is 8.6 SLOC per staff-day. By any measure, the productivity is very good due largely to the staffing, schedule, prototyping, and build process considerations cited previously, in addition to the enhanced interface discipline imposed through the use of Ada, the flexibility of TRW's message-based design, and the development of the most dificult foundation software first.

Tool-Generated Ada Source Code



?

ARCHITECTURE DATA BASE TOOL

? SLOC

?

- SOCKETS
- CIRCUITS
- TASKS

SAS BUILD TOOL

6.5K SLOC

20K SLOC

- GENERIC INSTANTIATIONS
  - PROCESS EXECUTIVES
  - TASK EXECUTIVES

?

MESSAGE I/O TOOL

2.4K SLOC

60K SLOC

- GENERIC INSTANTIATIONS
  - TEXT I/O (DATA REDUCTION)

?

CCPDS-R MESSAGE TOOL

3.5K SLOC

105K SLOC

- INTERFACE VALIDATION DATA
- OUTBOUND MESSAGE FORMATTING
- DATA BASE DEFINITIONS
- MESSAGE TYPE DEFINITIONS

?

FORMAT BUILD TOOL

2.5K SLOC

17K SLOC

- DISPLAY FORMAT TABLES

MITRE

Approximately 15K SLOC of tool software is responsible for generating over 200K SLOC of compilable Ada source code -- a 13:1 expansion ratio. The existence of this tool-generated software has been and continues to be a problem area for the Government on the CCPDS-R Program. First of all, this software was not anticipated at the time the CCPDS-R Program was formulated. As a result, the Government has little visibility into this area. Issues such as tool design documentation and test of the tools are not addressed contractually. Yet, the code generated by the tools represents anywhere from 28-57 percent of the total CCPDS-R software. Secondly, the tools which TRW has found necessary for development, will probably be needed by the Government for software maintenance. Provisions for user's manuals addressing tool parameters (legal values, capacities), rules for use, and interdependencies between files remain undefined. The Government is working diligently with TRW to resolve these two areas of concern.

Flexible vs. Tailored Designs for Missile Warning Subsystem

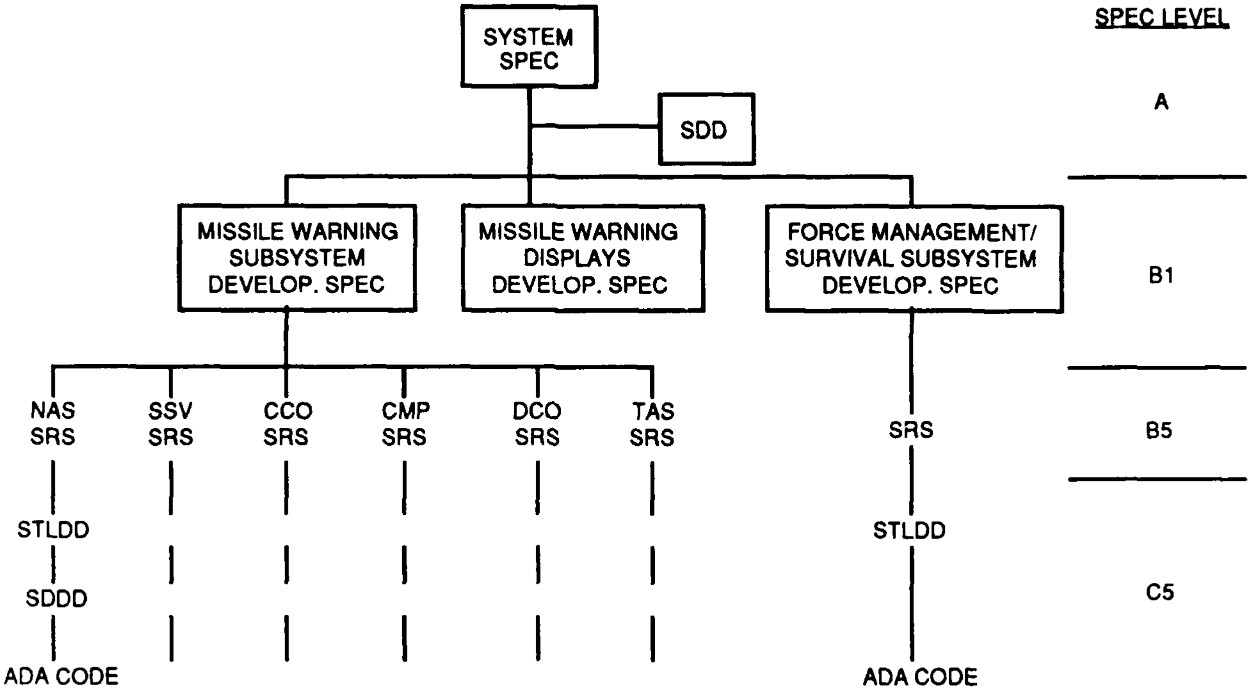
FUNCTION	TRW DESIGN (KSLOC)	TAILORED DESIGN (KSLOC)
EXECUTIVE CONTROL	74	5
COMMUNICATIONS	76	20
ALGORITHMS	10	10
DISPLAYS	40	30
TEST & SIMULATION	10	10
SYSTEM SERVICES	127	70
	337	145

CPU UTILIZATION (%)	MIPS	MIPS
APPLICATION	4.2 (35%)	3.6 (30%)
OVERHEAD	4.2 (35%)	1.2 (10%)
	8.4 (70%)	4.8 (40%)

MITRE

This vugraph compares the software size and CPU utilization associated with the TRW Missile Warning Subsystem design against estimates for a tailored, relatively inflexible, design for that same subsystem. As expected, the flexible TRW design is bigger in total size due to its general-purpose nature. Similarly, the CPU utilization for the TRW design exceeds that of the tailored design. Such is the price for flexibility. In the past, these overhead factors could have been severely limiting in terms of achieved functionality and quantitative performance. However, with the continuing rapid advances in processor memory capacity and CPU speed, TRW is demonstrating that the extra code to provide flexibility can be absorbed without serious effects.

Documentation Tailoring


SPEC LEVEL
SYSTEM SPEC
SDD
A
MISSILE WARNING SUBSYSTEM DEVELOP. SPEC
MISSILE WARNING DISPLAYS DEVELOP. SPEC
FORCE MANAGEMENT/ SURVIVAL SUBSYSTEM DEVELOP. SPEC
B1
NAS SRS
SSV SRS
CCO SRS
CMP SRS
DCO SRS
TAS SRS
SRS
B5
STLDD
STLDD
SDDD
C5
ADA CODE
ADA CODE

MITRE

Despite the best efforts of all concerned during contract definition, the CCPDS-R Program is inundated with paper. Overall, the contract calls for a total of 1,029 document deliveries across the three subsystems. This chart shows some of that paper, much of it driven by the numbers of CSCIs. There are six CSCIs for the Missile Warning Subsystem, five CSCIs for the Missile Warning Displays Subsystem (not shown), and seven CSCIs for the Force Management/Survival Subsystem (also not shown). As the Government's and TRW's experiences with the Missile Warning Subsystem mature, we are finding that fewer CSCIs will suffice to provide the proper degree of visibility and control. Accordingly, for the Force Management/Survival Subsystem, the Government and TRW are now planning on a single CSCI which will greatly alleviate the number of software requirements and design deliverables. In addition, we are planning to eliminate the Software Detailed Design Document (SDDD) as a deliverable. This is consistent with DOD-STD-2167A. Neither the Government nor TRW has found the SDDD useful for either software development or software maintenance because of the high readability of Ada code.

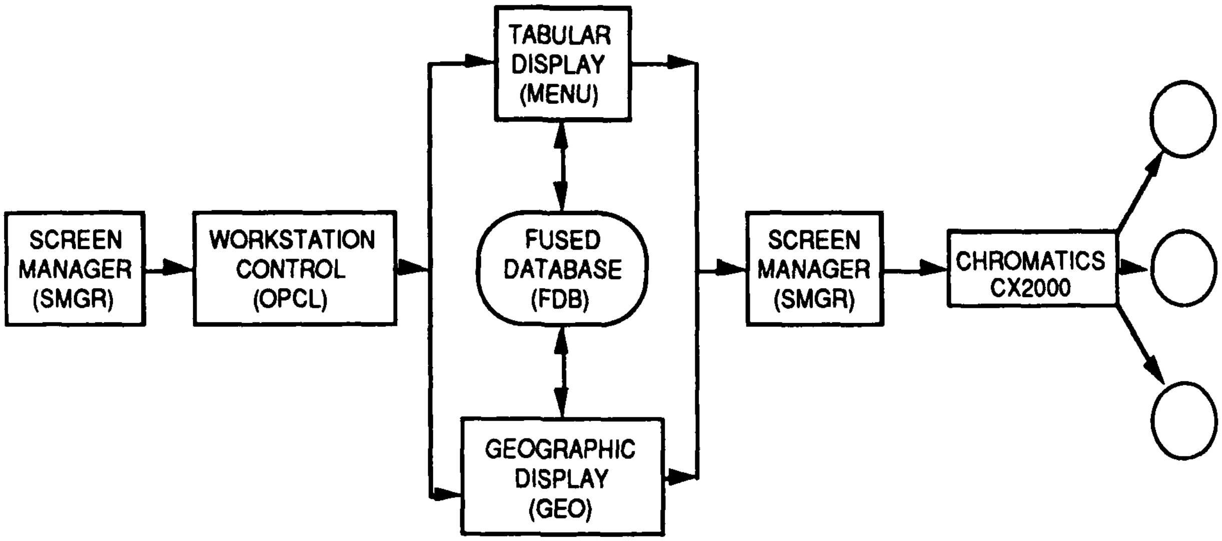
Software Progress Metrics

- Software staffing
- Software size
- Development progress
  - Design/code/unit test/document
- Test progress
  - Integration/component/string tests
  - Test procedures
  - Requirements verified
- Software problem reports
  - SPRs/SDPRs
- Action items from reviews
  - Monthly/PDW/CDW/demos
- Cost/schedule variances

MITRE

The CCPDS-R Program employs extensive metrics to manage the software development. This chart lists the major metrics reported monthly at each Program Management Review (PMR) and recorded in a monthly software metrics report. Several of these metrics, such as development progress, are produced automatically by tools which examine the Ada source code and determine the state of its completion. Notably missing from the list of metrics is one that deals with software timing (CPU utilization). In retrospect, such a metric based on measurements (as opposed to modelling only) should have been included as a deliverable. The Government is currently working with TRW to remedy this situation.

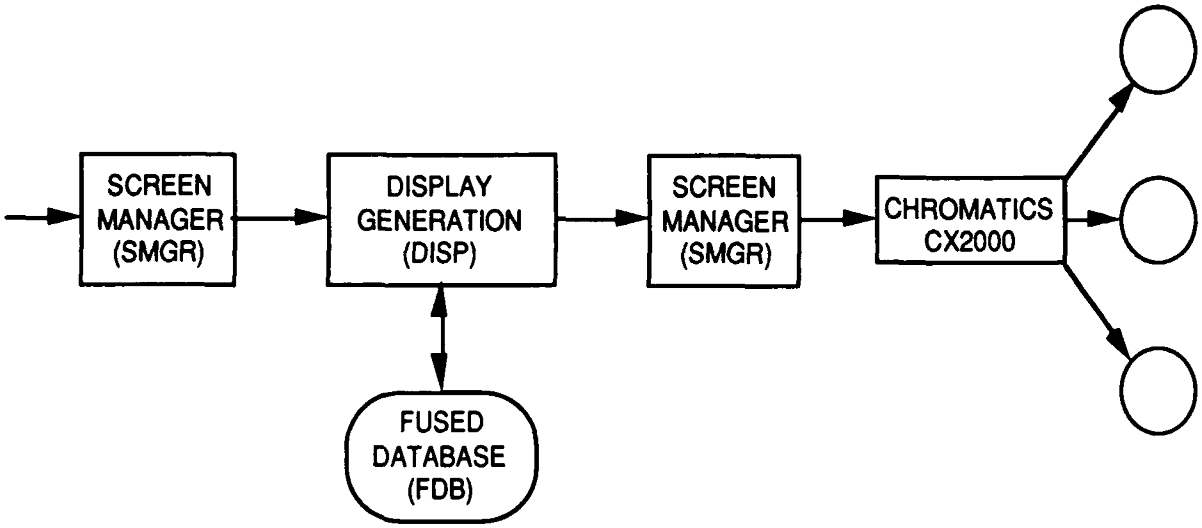
Example of Evolving Task Level Software (PDR Design)



MITRE

A feature of Ada is that it allows tasks to be defined at any level of functionality. At one extreme, all of the software could be defined as one big task. At the other, separate tasks could be defined for each and every function. In the early phases of the Missile Warning Subsystem development, such as at PDR, the latter tended to be the case. This gave both TRW and the Government visibility into and enhanced ease of understanding of the software.

Example of Evolving Task Level Software (CDR Design)



MITRE

As time has progressed, the design has evolved, and in several areas tasks are being merged. At CDR, for example, the previously separate tasks (OPCL, MENU, and GEO) each of about 10K SLOC were merged into a single new task (DISP) of about 30K SLOC. Reducing the number of interdependent tasks tends to reduce the number of overhead calls to VMS, Starlet, and ITC which, in turn, reduce processing time.

Pros and Cons of Merging Ada Tasks

- Reduces VMS and STARLET overhead
- Improves response time characteristics
- Makes software maintenance more difficult
  - Small changes entail extensive recompilation
  - Software is harder to understand

MITRE

A balance needs to be established between the need to merge tasks in order to reduce overhead and response time, and the need to design software that is both understandable early in the development and maintainable after development. The merging of tasks, which TRW is undertaking, to meet quantitative performance requirements, may have maintenance impacts in terms of compilation time and ease of understanding. The Government is continuing to assess this area actively.

TRW/MITRE Software Communications

	1987							1988												1989							
	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A
Formal Reviews			SDR				SSR			H/W CDR							PDR				PDS PDR				CDR		
Tech. Interchange Mtgs.	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
Working Groups																											
Algorithms		o	o	o	o	o	o	o		o		o	o		o							o			o		
Displays		o	o	o	o	o	o	o		o		o	o		o							o			o		
Interfaces			o		o			o			o				o						o						
Security		o			o						o			o				o				o			o		
Test Planning					o								o							o							
Design Walkthroughs																											
Preliminary			A0			A1						A2				A3/A4									A5		
Critical						A0				A1						A2			A3						A4		
Demonstrations							USI #1			CAP #1							USI #2 CAP #2 DBM #1								CAP #3		
Prog. Mgt. Reviews	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
Independent Audits							o					o					o		o						o		

MITRE

The final characteristic of the CCPDS-R Program, but not the least in terms of impact, is the extensive amount of communication which takes place between TRW and the Government. These are contractually established, and include monthly PMRs, bi-weekly Technical Interchange Meetings (TIMs), working groups in several areas, PDWs/CDWs, and formal demonstrations. As of August 1989, more than 60 such exchanges have taken place on the Program, not including daily interaction via telephone at all staff levels.

Risk Reduction Summary
(1 of 2)

- Trade studies
- Software Engineering Exercise
- Software prototyping
- Early review of draft SRSs
- Realistic schedule
- Qualified, appropriately sized staff with core group of experts
- Prototyping & evaluation of high risk items (algorithms)
- Performance modelling & testbed measurements
- Incremental software builds
- Incremental software/subsystem testing
- Early capability demonstrations

MITRE

This chart and the next summarize the various risk reduction measures taken or being taken on the CCPDS-R Program dating back to the Concept Development Phase when engineering trade studies and a Software Engineering Exercise were performed.

Risk Reduction Summary
(2 of 2)

- Exercise of delivered code in local MITRE/ESD testbeds
- Software reuse
- Tailored documentation
- Software progress metrics
- Working groups
- Technical interchange meetings
- Monthly contractor management reviews
- Periodic Red Teams/Corporate Level Assessments
- MITRE on-site representative
- Direct comment processing (APIS)

MITRE

Other risk reduction measures include exercise of the TRW build code locally, and independent audits such as Corporate Level Assessments (CLAs). Since mid-1988, the Government has had a MITRE (and a CTA) representative on-site at the TRW plant in Dominguez Hills, CA. This on-site representation has been invaluable in providing liaison between TRW and the Government, and active Government participation on TRW Program Configuration Control Board (PCCB) and Software CCB (SCCB) meetings. The ability to electronically transmit to TRW Government comments on TRW deliverables via APIS has helped to resolve differences early.

Areas for Improvement

(1 of 2)

- Improve knowledge of application
- Pin down requirements before going on contract
  - Algorithms
  - Displays
  - External interfaces
- Keep number of CSCIs to a minimum
- Tailor documentation to reduce paper load
- Keep design out of SRSs
- Eliminate software detailed design document
- Plan for collocation period if geographically dispersed
- Use graphics tools early in design process
- Plan for documentation and review of specially developed software tools
- Provide mechanism for modifying tools into properly integrated set

MITRE

Notwithstanding the progress made, this chart and the next identify areas of weakness on the CCPDS-R Program. Some are too late to remedy; however, where possible the Government and TRW are working together to improve the situation.

Areas for Improvement

(2 of 2)

- Quantify overhead CPU usage of foundation software
- Obtain early and continuing measurements of sizing and timing, and track convergence with QPRs
- Define requirements for software maintenance early
- Assess evolution towards fewer Processes and Tasks (in order to meet QPRs) against requirements for flexibility/maintainability
- Balance early verification of requirements against scope of regression testing

MITRE

This chart concludes the listing of areas for improvement.

Factors Behind Successful Software Development*

- Quality SRSs
  - Reflect detailed understanding of job to be done
  - Contain testable req'ts.
  - Guide to software designers/coders
- Quality test program
  - Thorough & extensive
  - Sufficient staffing & computer resources to do job
- Frequent Contractor/Government interaction
  - Close professional working relationship
- Contractor/Government knowledge of application
  - Key members experienced in application
  - Key members experienced in software acquisition
- Continuity of Contractor/Government key personnel
  - From Contract Award to completion of testing

* Extracted from lessons learned from the Berlin Radar Program, 1986

MITRE

At the conclusion of the Berlin Radar Program, which is generally viewed as a successful software development program, a survey was made of MITRE participants to identify success factors. The consensus is shown on this vugraph. All of the factors, except for knowledge of the application, are viewed as strong features of the CCPDS-R Program. As noted on the areas for improvement charts, elements within both the Government and TRW need to make continuing efforts to improve their understanding of missile warning, attack assessment, force management, and force reconstitution operations.

Conclusions

- CCPDS-R software development approach is working
- New Ada technology is being effectively coupled with the best of lessons learned from the past
- Programmer productivity has been greatly enhanced through software reuse, isolation of difficult Ada features, and use of tools to generate Ada source code
- Useful and demonstrable code is available early to judge true progress
- Demonstrations are invaluable in obtaining user input on requirements nuances
- Performance, schedule, and cost objectives are being met

MITRE

In conclusion, the CCPDS-R software development approach is working due, in large measure, to an effective coupling of Ada technology with the best of past lessons learned. It is doubtful whether either alone would have resulted in the degree of success enjoyed so far. Software productivity is being enhanced, demonstrations are being used as true measures of progress and proper interpretation of requirements, and the overall objectives of the program are being met.

Additional data is needed in a number of areas. These include the extent to which code can be reused, the degree to which the TRW code generation tools are essential for software maintenance, and the trade-offs between merging Ada tasks and software maintainability.

GLOSSARY

ANMCC	Alternate National Military Command Center
AFB	Air Force Base
APIS	Automated Program Information System
CA	Contract Award
CCO	Common Communications
CCPDS-R	Command Center Processing and Display System - Replacement
CDR	Critical Design Review
CDW	Critical Design Walkthrough
CINC	Commander-In-Chief
CLA	Corporate Level Assessment
CMP	Common Mission Processing
COTS	Commercial-Off-The-Shelf
CPU	Central Processing Unit
CSCI	Computer Software Configuration Item
CSSR	Communications System Segment Replacement
DCO	Display Coordination
DDSI	Digital Display System Interface
DEC	Digital Equipment Corporation
DISP	Display Generation
ESD	Electronic Systems Division
FDB	Fused Database
FQT	Formal Qualification Testing
GEO	Geographic Display
H/W	Hardware
IOT&E	Initial Operational Test and Evaluation
IR&D	Independent Research and Development
ITC	Inter-Task Communications
LAN	Local Area Network
NAS	Network Architecture Services
NMCC	National Military Command Center
NUDET	Nuclear Detonation
OPCC	Offutt Processing and Correlation Center
OPCL	Operator Control
PCCB	Program Configuration Control Board
PDR	Preliminary Design Review

PDS	Processing and Display Subsystem
PDW	Preliminary Design Walkthrough
PMR	Program Management Review
QPR	Quantitative Performance Requirement
SAC	Strategic Air Command
SAS	Software Architecture Skeleton
SCCB	Software Configuration Control Board
SDD	System Description Document
SDDD	Software Detailed Design Document
SDR	System Design Review
SEWS	Satellite Early Warning System
SLOC	Source Lines of Code
SMGR	Screen Manager
SOW	Statement of Work
SRS	Software Requirement Specification
SSR	Software Specification Review
SSV	Systems Services
STLDD	Software Top-Level Design Document
TAS	Test & Simulation
TIM	Technical Interchange Meeting
USI	User-System Interface
VDSI	Visual Display System Interface
VMS	Virtual Memory System
WSO	Warning System Operator