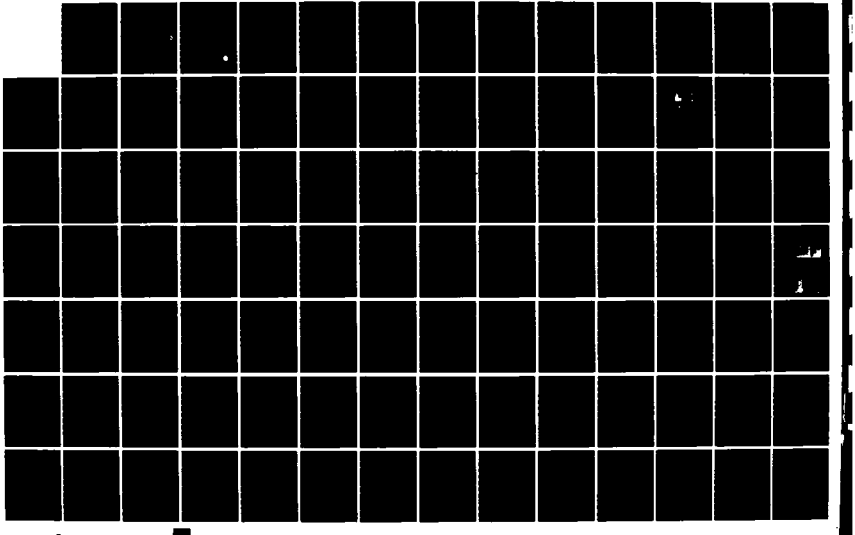
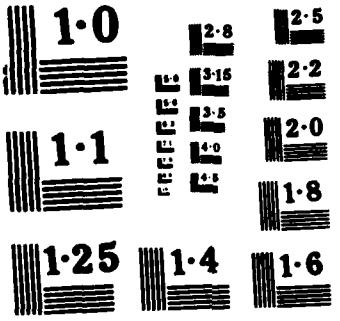
AD-A163 236 PROCEEDINGS OF THE INTERNATIONAL TEST AND EVALUATION ASSOCIATION 1984 SYM..(U) GEORGIA INST OF TECH ATLANTA SCHOOL OF INFORMATION AND COMPUT.. APR 85 1/2

UNCLASSIFIED GIT-ICS-85/05 F33657-82-G-2083 F/G 9/2 NL





AD-A163 236

DTIC ACCESSION NUMBER

PHOTOGRAPH THIS SHEET

4/0044

LEVEL

1

INVENTORY

SUPPLEMENT TO THE PROCEEDINGS OF THE INTERNATIONAL TEST AND EVALUATION ASSOCIATION 1984 SYMPOSIUM (STEP SESSION)

DOCUMENT IDENTIFICATION

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR	
NTIS GRA&I	☑
DTIC TAB	☐
UNANNOUNCED	☐
JUSTIFICATION	
BY	
DISTRIBUTION /	
AVAILABILITY CODES	
DIST	AVAIL AND/OR SPECIAL
A-1	

DISTRIBUTION STAMP

DTIC
ELECTE
JAN 21 1986
S D
D

DATE ACCESSIONED

DATE RETURNED

86 1 14 028

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NO.

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDAC

DTIC FORM 70A DEC 83

DOCUMENT PROCESSING SHEET

PREVIOUS EDITION MAY BE USED UNTIL STOCK IS EXHAUSTED.

TECHNICAL REPORT NO. GIT-ICS 85/05

SUPPLEMENT TO THE PROCEEDINGS OF THE INTERNATIONAL TEST AND EVALUATION ASSOCIATION 1984 SYMPOSIUM (STEP SESSION)

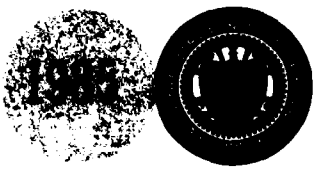
ed for

Air Development Center
nster, PA 18974

ct No. F33657-82-G-2083

1985

GEORGIA INSTITUTE OF TECHNOLOGY
A UNIT OF THE UNIVERSITY SYSTEM OF GEORGIA
SCHOOL OF INFORMATION AND COMPUTER SCIENCE
ATLANTA, GEORGIA 30332



unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER GIT-ICS 85/05	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Supplement to the Proceedings of the INTERNATIONAL TEST AND EVALUATION ASSOCIATION 1984 Symposium (STEP Session)		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s) F33657-82-G-2083
9. PERFORMING ORGANIZATION NAME AND ADDRESS Georgia Institute of Technology School of Information and Computer Science Atlanta, Georgia 30332		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS NADC Warminster, PA 18974		12. REPORT DATE April 1985 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) 15a. DECLASSIFICATION DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) unlimited		
18 SUPPLEMENTARY NOTES		
19 KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20 ABSTRACT (Continue on reverse side if necessary and identify by block number)		

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

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

This supplement contains the presentations and panel session of the Software Test and Evaluation Project (STEP) Session of the International Test and Evaluation Association Symposium on November 8, 1984.

The motivation for this session of the ITEA symposium is to showcase some good examples of software testing in the DoD and to update the test community on some current activities of the DoD to improve the testing of software. The session is organized around three technical areas; Software T&E Policy and Guidance; Software Test Management; and Software Test Tools. The Software T&E Policy and Guidance section is concerned with presenting policy level changes to DoDD 5000.3 that will improve the visibility and accountability of software in major systems acquisitions. Included in this section is a presentation of a guidebook that will be issued to aid in implementing DoD policy. The Software Test Management section presents two perspectives of the management of software testing during development test and operational test. The Software Test Tools section is used to describe the effectiveness of using software test tools on two DoD programs. In addition, it describes the to be issued DoD-StD-SDS and SOS, the new Joint Logistics Commanders Standards for software development and software quality assurance.

A panel session to discuss in an open forum the current experiences of the panel and the audience related to testing software completes this session.

W. Michael McCracken
Session Chairman
Software Test and Evaluation Project
Georgia Institute of Technology

TABLE OF CONTENTS

Pending Revisions to DoDD 5000.3

Donald R. Greenlee

Pending revisions to DoDD 5000.3 are presented to inform the test community of these changes. The changes establish improved guidelines for the testing of software that is a part of the mission critical computer resources of major weapons systems. The material presented is for information only, as the changes have not been coordinated in the Services.

The changes to DoDD 5000.3 that are related to software testing include the following:

Test and Evaluation of Computer Software: additional language has been added to the draft Directive that clarifies the intent of this section. In summary, this language is intended to require that software components that implement critical functions of a system be identified and that those components be tested throughout the development/integration portion of the software lifecycle. This section is planned to require that the results of those tests be objective, repeatable, available to subsequent test groups and interpretable in terms of the overall system objectives. The level of testing of these components should be sufficient to achieve a balanced risk with the hardware on which they are implemented.

In Part I, Section 3 of the TEMP, there will be a description of the critical software issues that must be addressed by testing. The description will contain an outline of the software program and the T&E management responsibilities of the participating T&E organizations. It will show the integrated time sequencing of critical software T&E events. It will also state the corresponding test-verifiable objectives and the corresponding goals, specifications and thresholds for each critical software issue. It will contain a summary of the software T&E already conducted as well as the planned software T&E. Finally, it will contain the identification of critical software components and subsystems required for testing in each phase and the test tools required, including how they support the software test objectives.

Mr. Greenlee's presentation did not use viewgraphs nor was there any material for handouts.

Donald R. Greenlee

Mr. Greenlee is currently acting as Deputy Director, Defense Test and Evaluation for Strategic, Naval and Communications, Command, Control and Intelligence Systems. Prior to joining the Office of the Secretary of Defense, he was Associate Head of the Data Processing Systems Department in the MITRE Corporation's National Command and Control Systems Division. He held earlier positions with the Johns Hopkins University Applied Physics Laboratory and the Apollo Support and Defense Systems Departments of the General Electric Company.

Mr. Greenlee attended the Universities of Florida and Maryland, Syracuse and Ohio State Universities and MIT; he holds a B.S. in Mathematics and Master's degrees in Mathematics and in Engineering. He is a member of the Tau Beta Pi and Alpha Pi Mu national engineering honor societies, a U.S. Soccer Federation-licensed coach and referee, and a master ultramarathoner, for which he trains in the corridors of the Pentagon.

Software Test and Evaluation Guidebook

Richard A. DeMillo

The focus of this presentation is a guidebook that is being developed by the Software Test and Evaluation Project (STEP) for the Director, Defense Test and Evaluation. The Software Test and Evaluation Project was initiated in 1981 by the Director Defense Test and Evaluation. The primary objective of STEP is to develop new DoD guidance and policy for the test and evaluation of computer software for mission-critical applications.

The Software Test and Evaluation Guidebook is a two volume reference set that provides checklists and guidance to DoD components in the area of software test and evaluation for major Defense system acquisition. Volume I is devoted to the evaluation of the treatment of software in Test and Evaluation Master Plans. Volume II (not presented) addresses the structuring, planning, conduct, and evaluation of software tests throughout the acquisition process.

Richard A. DeMillo

Richard A. DeMillo has an extensive background in computer science research and computing practice. Dr. DeMillo holds the Ph.D. degree in Information and Computer Science from the Georgia Institute of Technology (1972). He is the author or co-author of over 50 reports, books and articles reporting basic research results in theoretical computer science, computer security, and software engineering.

Prior to receiving his Ph.D., Dr. DeMillo held programming positions and research assistantships. From 1972 to 1976 he was an Assistant Professor of Electrical Engineering and Computer Science at the University of Wisconsin-Milwaukee. In 1976, he returned to Georgia Tech and was promoted to the rank of Professor in 1981.

In the area of software engineering, Dr. DeMillo has concentrated on software reliability and the emerging area of software metrics. He is a co-developer of the program mutation approach to software testing and is the author of the book Program Mutation: an approach to software testing. Dr. DeMillo is currently Principal Investigator for the DoD Software Test and Evaluation Project (STEP). The goal of STEP is to revise the policy and procedures in the Department of Defense for the testing of software in major DoD systems. In this capacity, Dr. DeMillo reports to the Office of the Secretary of Defense.

Dr. DeMillo is on the editorial board of several professional journals and has served as chairman of a number of national meetings. He has been a Distinguished Visitor of the IEEE Computer Society and is a frequent speaker at professional and technical gatherings.

THE SOFTWARE TEST AND EVALUATION PROJECT

PROGRESS AND PLANS
1984–1985

DIRECTOR, DEFENSE TEST AND EVALUATION
OFFICE OF THE SECRETARY OF DEFENSE

STEP INITIATED BY DDT&E IN 1981

GOAL IS TO IMPROVE THE PRACTICE OF SOFTWARE TEST AND EVALUATION

* DEVELOP POLICY AND STANDARDS
* INSERT EXISTING TECHNOLOGY
* COORDINATE WITH RELATED EFFORTS

RATIONALE FOR STEP

SOFTWARE T&E IS A CRITICAL TECHNOLOGY

* MISSION CRITICAL SOFTWARE
* HARDWARE AND SOFTWARE IMBALANCES
* COSTS: TESTS VS. ERRORS
* INADEQUACIES OF CURRENT GUIDANCE

STUDY RESULTS: 20 RECOMMENDATIONS

MODIFY EXISTING POLICY AND PROVIDE NEAR AND LONG TERM IMPLEMENTATION SUPPORT

* Emphasis on test planning: chain beginning at system level
* Concentrate resources on critical software components
* Insert available technology into practice

SOFTWARE IN THE MODIFIED TEST AND EVALUATION MASTER PLAN

-- Criteria for evaluation of TEMP

-- Criteria for additional guidance to Project Offices

-- Examples of software characteristics and test issues

-- Examples of software test articles and special support requirements

GEORGIA INSTITUTE OF TECHNOLOGY
THE SOFTWARE TEST AND EVALUATION PROJECT

POLICY IMPLEMENTATION THROUGH TEST AND EVALUATION MASTER PLAN

MAJOR PLANNING DOCUMENT -- SUBMITTED AND EVALUATED BEFORE MILESTONES

MISSION/FUNCTION MATRIX	*Relate functions to be demonstrated to mission to be performed*
OPERATIONAL CHARACTERISTICS TECHNICAL CHARACTERISTICS	*Primary indicators of conformance to written specifications or operational requirements*
OPERATIONAL ISSUES TECHNICAL ISSUES	*Aspects of system capability questioned before estimating overall worth*
T&E OUTLINES	*To date, planned, objective, scope*
SPECIAL RESOURCE SUMMARY	*Test articles, test tools*

PART I – DESCRIPTION

SECTION	QUESTIONS
1. Mission	
2. System	a. Does the system contain Mission–critical Computer Resources? b. Does software implement critical functions? c. Is the system software intensive?

PART I – DESCRIPTION (CONTINUED)

SECTION	QUESTIONS
2. System (continued) a. Key Functions	a. Does Mission/Function Matrix identify primary functional capabilities to be implemented by software? b. Are the functions: ––New? ––Modifications of existing capabilities? ––Automation of existing capabilities? ––Mature?

PART I – DESCRIPTION (CONTINUED)

SECTION	QUESTIONS
2. System (continued)	
a. Key Functions	
b. Interfaces	a. Is software important to the interfaces? b. Do the interfaces have software implications?

PART I – DESCRIPTION (CONTINUED)

SECTION	QUESTIONS
2. System (continued)	
a. Key Functions	
b. Interfaces	
c. Unique Characteristics	a. Does the system use software technology that: ––Affects risk? ––Has lifecycle impact? ––Distinguishes it from other systems?

PART I – DESCRIPTION (CONTINUED)

SECTION	QUESTIONS
2. System	
3. Required Operational Characteristics	
4. Required Technical Characteristics	
5. Required Software Characteristics	a. Are there software characteristics that: -- Are unique to software? -- May be overlooked?

REQUIRED SOFTWARE CHARACTERISTICS

"Software parameters that are primary indicators of conformance to written requirements/specifications and operational suitability/effectiveness."

EXAMPLES

TYPE	PARAMETER	UNIQUE?
Operational	performance	No
Operational	maintainability	Unique aspects
Technical	precision/accuracy	Minimal
Technical	robustness	Yes

PART I – DESCRIPTION (CONTINUED)

SECTION	QUESTIONS
2. System	
3. Required Operational Characteristics	
4. Required Technical Characteristics	
5. Required Software Characteristics	
6. Critical T&E Issues	a. Do the required software characteristics raise unique or easily missed T&E issues?
a. Technical Issues	
b. Operational Issues	
c. Software Issues	

REQUIRED SOFTWARE CHARACTERISTICS

"Those aspects of software capability...that must be questioned before a system's overall worth can be estimated..."

EXAMPLES

TYPE	CHARACTERISTIC	ISSUE?
Operational	performance	degraded mode operation
Operational	maintainability	Adequacy of support env.
Technical	precision/accuracy	algorithm success
Technical	robustness	design and architecture

PART II – PROGRAM SUMMARY

SECTION	QUESTIONS
1. Management	
2. Integrated Schedule	a. Are key software subsystem demonstration included on schedule?

T&E OUTLINES

PART III – DT&E Outline

PART IV – OT&E Outline

PART V – Software T&E Outline

PART V – SOFTWARE T&E OUTLINE

SECTION	QUESTIONS
1. Software T&E to Date	a. Have software characteristics been demonstrated using systematic test methods? b. Have the planned levels of testing been achieved? Is the documentation cited? c. Have testing deficiencies been interpreted in terms of overall system evaluation criteria?

PART V – SOFTWARE T&E OUTLINE (CONT)

SECTION	QUESTIONS
1. Software T&E to Date	
2. Future Software T&E	a. Is the test environment (i.e., development, operational, logistics support) appropriate for the characteristics to be demonstrated?

PART V – SOFTWARE T&E OUTLINE (CONT)

SECTION	QUESTIONS
1. Software T&E to Date	
2. Future Software T&E (continued)	
a. Software T&E Objectives	a. What software characteristics are not adequately addressed in the DT&E and OT&E Outlines?
b. Software T&E Events/ Scope of Testing/ Basic Scenarios	

PART V – SOFTWARE T&E OUTLINE (CONT)

SECTION	QUESTIONS
1. Software T&E to Date	
2. Future Software T&E (continued)	
c. Critical Software T&E Items	a. Are subsystems needed for adequate software T&E prior to next decision point identified?

PART VII – SPECIAL RESOURCE SUMMARY

SECTION	QUESTIONS
1. Test Articles	a. Are critical software components and key subsystems identified?

GEORGIA INSTITUTE OF TECHNOLOGY
THE SOFTWARE TEST AND EVALUATION PROJECT

PART VII – SPECIAL RESOURCE SUMMARY

SECTION	QUESTIONS
1. Test Articles	
2. Special Support Requirements	a. Is there an explanation of how test tools support software test objectives?
	b. Are adequate steps being taken to acquire each tool? Do any of the tools increase risk?

GEORGIA INSTITUTE OF TECHNOLOGY
THE SOFTWARE TEST AND EVALUATION PROJECT

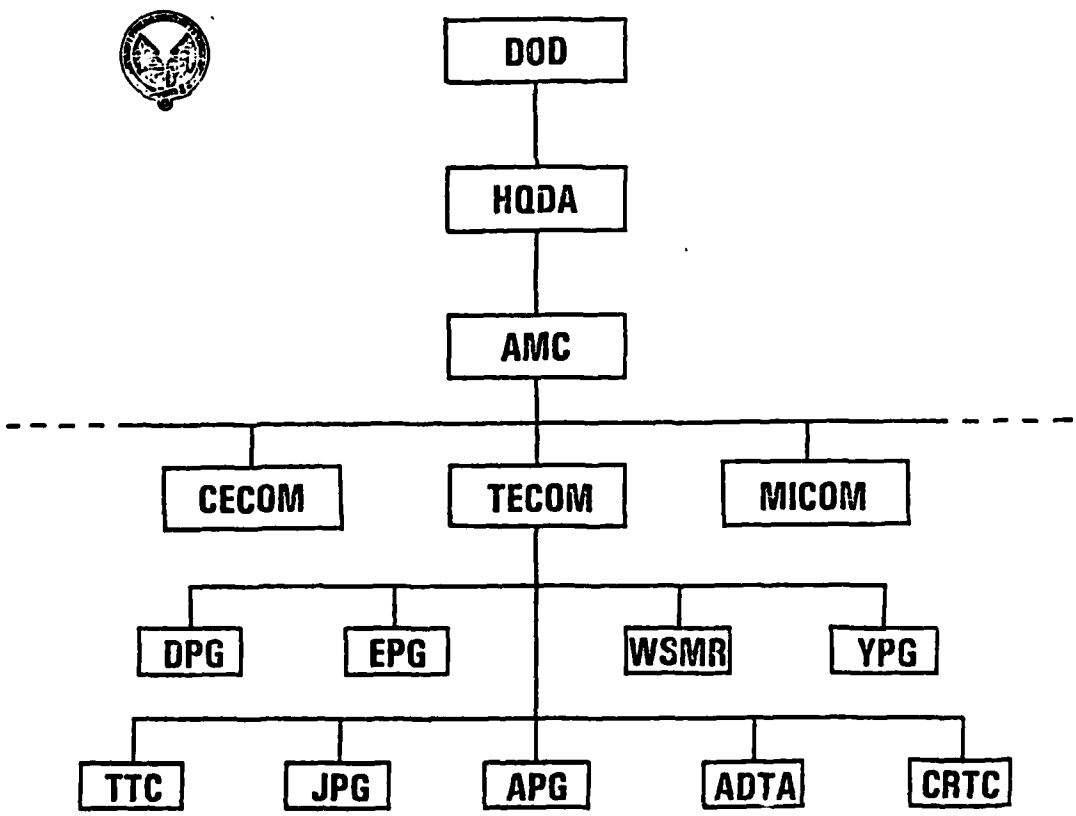
TECOM Policy, Procedures and Responsibilities for T&E of Software Embedded in Battlefield Automated Systems

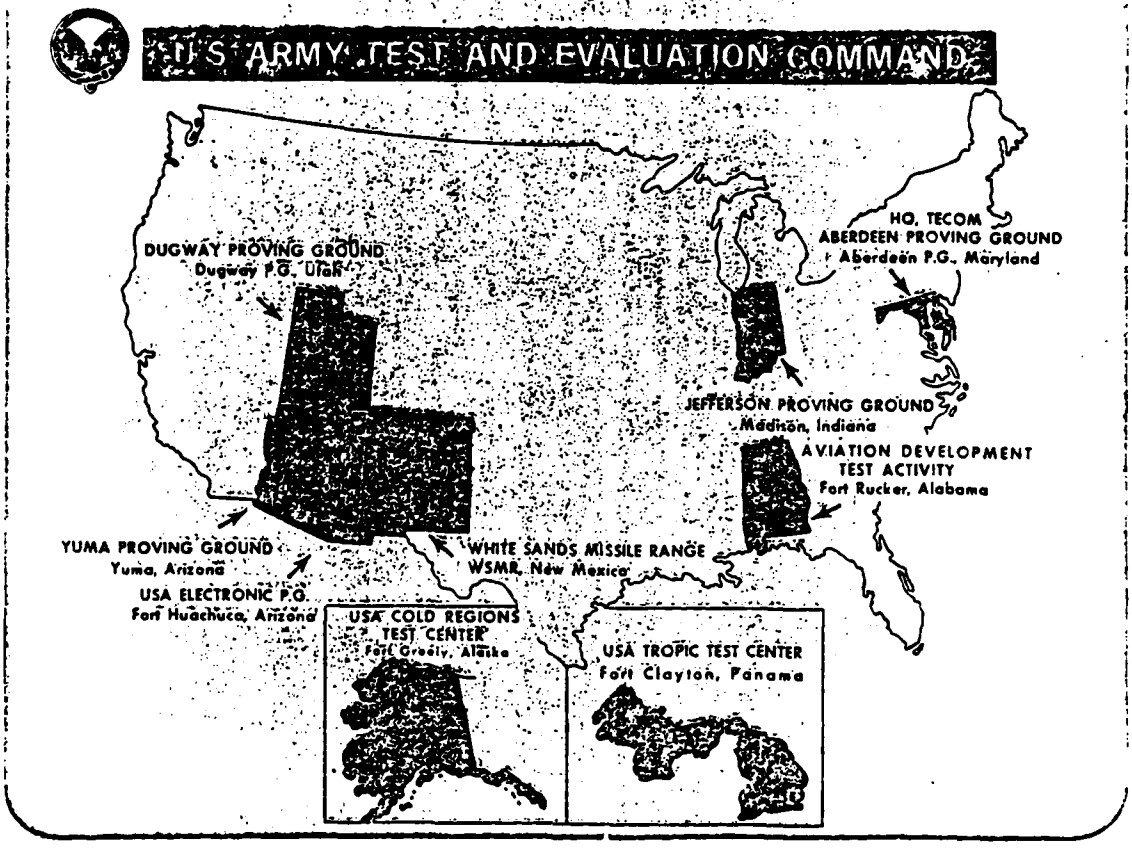
Francis Bartosik

The U.S. Army Test and Evaluation Command is responsible for test and evaluation of assigned Army and other service material systems. TECOM is headquartered at Aberdeen Proving Ground, MD. TECOM contains nine subordinate installations/field operating agencies. TECOM relies on repeatable, highly instrumented, system-level testing to demonstrate system capabilities. To aid in the testing of software portions of systems, TECOM has developed a policy for that activity. This policy is used to reduce the art of testing to more of a standardized, scientific process. The intentions behind implementing this policy are to: (1) establish consistency among HQ staff elements and field test agencies in the approach to software testing; (2) to formalize and discipline the software test process and the scheduling of the events comprising that process to better support the total TECOM mission of test and evaluation; (3) to focus attention on the mission-critical embedded computer resources in the battlefield automated systems being developed.

TECOM SOFTWARE TESTING POLICY

- BACKGROUND
- POLICY
  - WHY
  - WHAT
  - HOW
- EXPECTATIONS
- PERSONAL OBSERVATIONS





TECOM SOFTWARE TESTING POLICY

- TECOM TESTING
  - CONTROLLED
  - REPEATABLE
  - INSTRUMENTED

TECOM SOFTWARE TESTING POLICY

WHY IS IT NECESSARY?

- INCONSISTENCY IN FIELD AGENCIES' SOFTWARE TESTING APPROACH
- INVOLVEMENT BY TESTER TOO LATE FOR ALL BUT "BLACK BOX" TESTING
- ADEQUATE TESTING PRECLUDED BY (AMONG OTHERS) INSUFFICIENT KNOWLEDGE OF SYSTEM TO BE TESTED

TECOM SOFTWARE TESTING POLICY

WHAT IS THE INTENTION?

- ESTABLISH CONSISTENT APPROACH TO SOFTWARE TESTING THROUGHOUT TECOM
- FORMALIZE SOFTWARE T&E PROCESS IN SUPPORT OF TECOM MISSION
- EMPHASIZE MISSION-CRITICAL EMBEDDED COMPUTER RESOURCES

TECOM SOFTWARE TESTING POLICY

- **BASIS**
  - PRACTICAL EXPERIENCE (SCHOOL OF HARD KNOCKS)
  - METHODOLOGY STUDIES 1974-76
  - ARTADS SOFTWARE ACQUISITION STUDY 1975
  - DARCOM STUDY 1976
  - TECOM TECHNICAL REPORT SY–2-77 1977
  - TEST OPERATING PROCEDURE 1-1-056 1977
  - JLC
    - ELECTRONICS RELIABILITY SOFTWARE WORKING GROUP 1975
    - AD HOC GROUP ON T&E PLANNING 1978
    - COORDINATING GROUP ON COMPUTER RESOURCE MGMT 1978
    - MONTEREY I/II 1979/81
    - ORLANDO I 1983

TECOM SOFTWARE TESTING POLICY

- **BASIS (Cont'd)**
  - CONFERENCES AND SYMPOSIA
    - SOFTWARE QUALITY MGMT CONFERENCE 1971
    - SECOND ARMY SOFTWARE SYMPOSIUM 1978
    - DARCOM TACTICAL COMPUTER CONFERENCE 1978
    - DOD/NSIA SYMPOSIUM ON SOFTWARE T&E 1983
    - MULTI-SERVICE DT&E COMMANDERS 1980-84
  - OTHER ACTIVITIES
    - BAISEMP 1978-81
    - ACCSSEIMP 1982-84
    - DEFENSE MANAGEMENT JOURNAL MARCH 1979
    - STEP 1983–

TECOM SOFTWARE TESTING POLICY

- **PROVISIONS**
  - EMBEDDED COMPUTER RESOURCES ADDRESSED IN CONTEXT OF SYSTEM T&E
  - SOFTWARE ASSESSED AGAINST SPECIFICATIONS AND STANDARDS
  - IMPACT OF SOFTWARE PERFORMANCE ON SYSTEM PERFORMANCE ASSESSED
  - SOFTWARE QUALITY ASSESSED IN TERMS OF USABILITY, CORRECTNESS, INTEROPERABILITY, MAINTAINABILITY
  - CAPABILITY TO TEST AND EVALUATE BATTLEFIELD AUTOMATED SYSTEMS UPDATED THROUGH FEEDBACK TO COMMAND METHODOLOGY AND INSTRUMENTATION PROGRAMS

TECOM SOFTWARE TESTING POLICY

- **WHAT ARE THE RESPONSIBILITIES?**
  - TECOM HQ
    - AS EVALUATOR
    - AS TEST MANAGER
  - FIELD TEST AGENCIES

TECOM SOFTWARE TESTING POLICY

- **RESPONSIBILITIES**
  - TECOM HQ, AS EVALUATOR, WILL:
    - DEVELOP SOFTWARE PLANNING DIRECTIVE IN TIME FOR FIELD AGENCY EARLY INVOLVEMENT
    - DEVELOP INDEPENDENT EVALUATION PLANS (IEP)
    - DEVELOP TEST DESIGN PLANS (TDP)
    - ASSURE ADEQUACY OF DETAILED TEST PLANS (DTP)
    - DEVELOP INDEPENDENT EVALUATION REPORTS (IER)

TECOM SOFTWARE TESTING POLICY

- **RESPONSIBILITIES (Cont'd)**
  - TECOM HQ, AS TEST MANAGER, WILL:
    - DEVELOP SOFTWARE PLANNING DIRECTIVE
    - ASSURE AVAILABILITY OF TEST TOOLS/INSTRUMENTATION
    - ASSURE ADEQUACY OF DTP
    - APPROVE TEST REPORT
    - OBTAIN AND FORWARD DATA TO INDEPENDENT EVALUATOR

TECOM SOFTWARE TESTING POLICY

RESPONSIBILITIES (Cont'd)

- FIELD TEST AGENCIES WILL:
  - PARTICIPATE IN SYSTEM DEVELOPMENT PROCESS
  - ASSURE TEST PLANNING ADDRESSES SOFTWARE
  - PREPARE DTP THAT IS RESPONSIVE TO:
    - SOFTWARE PLANNING DIRECTIVE
    - TEST PLANNING/EXECUTION DIRECTIVES
    - TEST DESIGN PLAN
    - INDEPENDENT EVALUATION PLAN
    - SYSTEM REQUIREMENTS (LOA, LR, ROC, A–, B–5, C–5 SPECS)
    - EXECUTIVE APPROVED DTP
    - PREPARE TEST REPORT
  - CONDUCT METHODOLOGY STUDIES TO IMPROVE CAPACILITY

TECOM SOFTWARE TESTING POLICY

SOFTWARE TESTING OBJECTIVES

1. PARTICIPATE IN SPEC DEVELOPMENT
2. ASSURE ALGORITHMIC CORRECTNESS
3. DETERMINE DOCUMENTATION ADEQUACY
4. MONITOR IV&V ACTIVITIES
5. VALIDATE SIMULATIONS
6. SYSTEMATICALLY DETECT AND ANALYZE SOFTWARE FAILURES
7. DETERMINE RESOURCE CONSTRAINTS AND EXCESSES
8. ASSESS SOFTWARE ASPECTS OF SYSTEM SPEC COMPLIANCE
9. DETERMINE RETEST REQUIREMENTS
10. MEASURE OPERATING SYSTEM FUNCTION
11. RELATE SYSTEM AND COMPUTER TIME LINES
12. ENSURE A CONSISTENT SOFTWARE INCIDENT REPORTING SYSTEM
13. SUPPORT SOFTWARE PORTION OF SYSTEM EVALUATION

TECOM SOFTWARE TESTING POLICY

- WHAT ARE THE MECHANICS?
  - SOFTWARE PLANNING DIRECTIVE
  - EARLY PARTICIPATION IN SYSTEM DEVELOPMENT
  - INCREASED RELIANCE ON SINGLE INTEGRATED DEVELOPMENT TEST CONCEPT (SIDTC)
    - MATERIEL DEVELOPER
    - DEVELOPING CONTRACTOR(S)
    - IV&V AGENT
    - SYSTEM DEVELOPMENT TESTER
    - COMBAT DEVELOPER
    - OPERATIONAL TESTER
    - DT AND OT EVALUATORS
    - LIFE CYCLE SOFTWARE SUPPORT CENTER

TECOM SOFTWARE TESTING POLICY

WHAT ARE OUR EXPECTATIONS?

- UNIFIED COMMAND-WIDE APPROACH TO SOFTWARE TESTING
- NO SURPRISES TO PM'S
- AUDIT TRAIL FOR SOFTWARE TESTING (MONEY, MANPOWER, TOOLS)
- EARLY INVOLVEMENT FOR TESTER AND EVALUATOR
- INCREASE TESTER'S KNOWLEDGE OF SYSTEM
- FACILITATE TIMELY IDENTIFICATION OF NECESSARY TEST TOOLS
- INCREASE APPROPRIATENESS/QUALITY OF TESTS
- INCREASE QUALITY OF TEST DATA/TEST REPORTS
- INCREASE QUALITY OF EVALUATION

AFOTEC Software Test Managers Handbook

Major Frederick Foster

The Air Force Operational Test and Evaluation Center (AFOTEC) is the USAF independent test agency. It is responsible for testing new and modified weapons systems under realistic, operational conditions. The Software Evaluation Division (LG5) is tasked with evaluating the operational effectiveness and suitability of mission critical computer resources (MCCR) embedded within those systems. This presentation describes the requirement for operational test and evaluation (OT&E) of MCCR software and AFOTEC's role in managing the OT&E program.

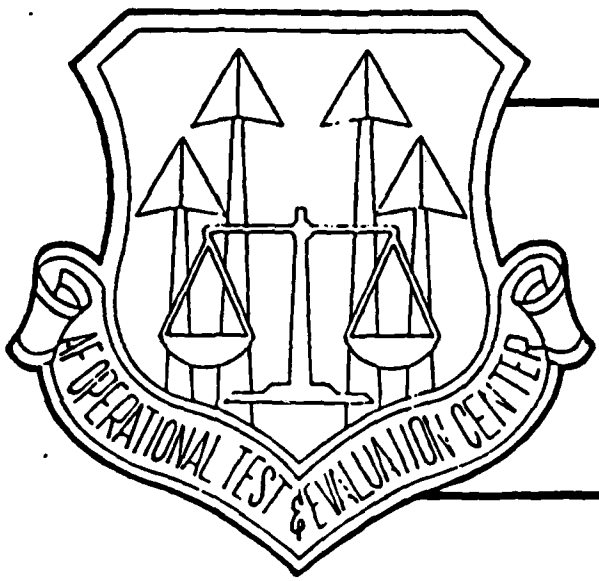
AFOTEC/LG5 was organized to meet the intent of DoD 5000.3 which requires the development of software performance objectives, operational testing of software, and participation by OT&E agencies during planning and development activities. At the Center, software test managers plan for OT&E of the MCCR software. They estimate test requirements and write the software portion of the OT&E test plan. They devise an organizational structure for software testing and insure the test team is adequately staffed. The Deputy Test Director for Software Evaluation (DSE) is a key AFOTEC position on the field test team. The DSE is supported by software evaluators from the eventual using, supporting and training organizations, to collect, evaluate, and analyze software data during test. The Software Test Manager works closely with the test team during and after test to prepare the interim and final reports.

AFOTEC/LG5 has published guidelines, AFOTECP 800-2 Volumes 1 and 2, describing the activities of the software test manager and deputy for software evaluation. Additionally, Volumes 3 through 5 of the same series support the evaluation of source listings and documentation, the operator-machine interface, and the software support resources. These guidelines are "living" documents, responsive to a strong lessons learned program, and provide a crossfeed of information between space/strategic, tactical/avionic, and C^3I software OT&E programs.

The Operational Test Center is an active participant in software test and evaluation. AFOTEC/LG5 currently uses standardized assessment tools for software maintainability, operator-machine interface, and software support resources. Future methodologies under development are aimed at evaluating computer system security, performance, and software risk analysis.

Major Frederick J. Foster

Major Foster is a branch chief with the Software Evaluation Division, Air Force Operational Test and Evaluation Center. Since 1970 he has participated in designing and testing management information and mission critical computer systems for state government and military applications. He has a Bachelor of Science in Systems Engineering and a Master of Business Administration. Major Foster is a senior pilot and a graduate of the Air Command and Staff College.



SOFTWARE TEST AND EVALUATION

SOFTWARE EVALUATION DIVISION

SOFTWARE TEST AND EVALUATION

→ • BACKGROUND

• AFOTEC ORGANIZATION FOR SOFTWARE OT&E

• AFOTEC APPROACH TO SOFTWARE OT&E

• FUTURE EFFORTS IN SOFTWARE OT&E

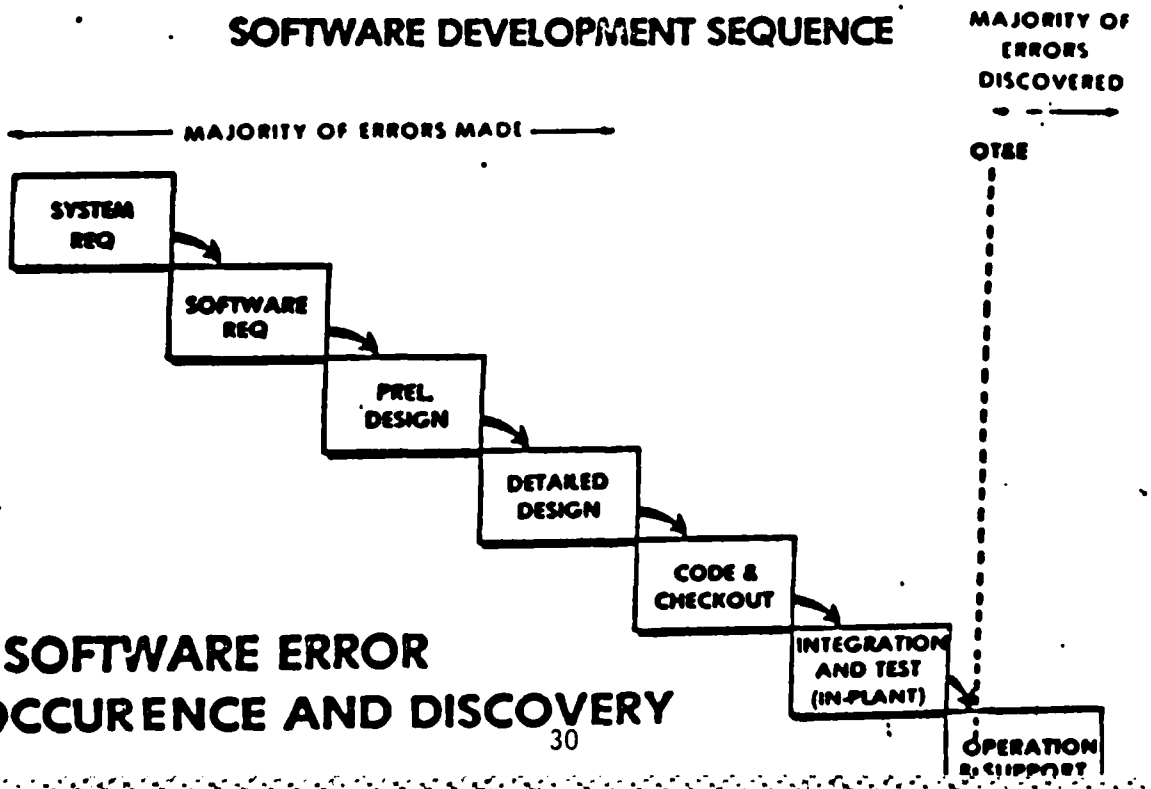
BACKGROUND

THE 1974 DEFENSE SCIENCE TASK FORCE ON TEST AND EVALUATION

THE TASK FORCE FOUND THAT:

"WHEREAS THE HARDWARE DEVELOPMENT WAS FOR THE MOST PART SCHEDULED, MONITORED, TESTED, AND REGULARLY EVALUATED, THE SOFTWARE DEVELOPMENT WAS NOT."

SOFTWARE DEVELOPMENT SEQUENCE



SOFTWARE ERROR OCCURENCE AND DISCOVERY

AFOTEC OT&E DIRECTION

DOD 5000.3 (26 DEC 79)

- PERFORMANCE OBJECTIVES SHALL BE ESTABLISHED FOR SOFTWARE DURING EACH SYSTEM ACQUISITION PHASE
- SOFTWARE SHALL UNDERGO OPERATIONAL TESTING.... UTILIZING TYPICAL OPERATOR PERSONNEL
- OT&E AGENCIES SHALL PARTICIPATE IN SOFTWARE PLANNING AND DEVELOPMENT TO ENSURE CONSIDERATION (OF THE) OPERATIONAL ENVIRONMENT AND EARLY DEVELOPMENT OF OPERATIONAL TEST OBJECTIVES

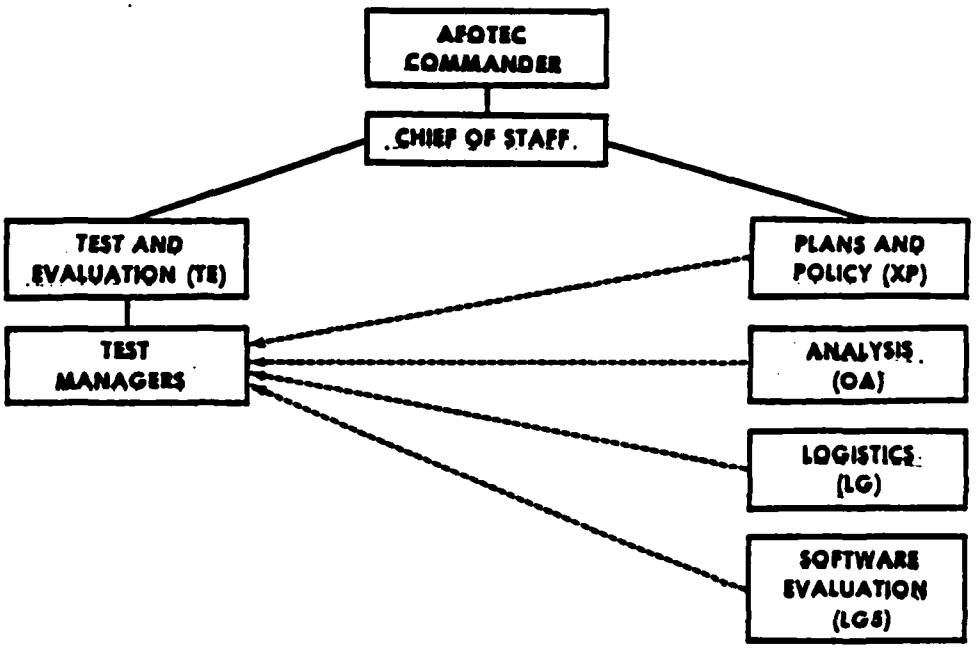
AFR 80-14 (12 SEP 80)

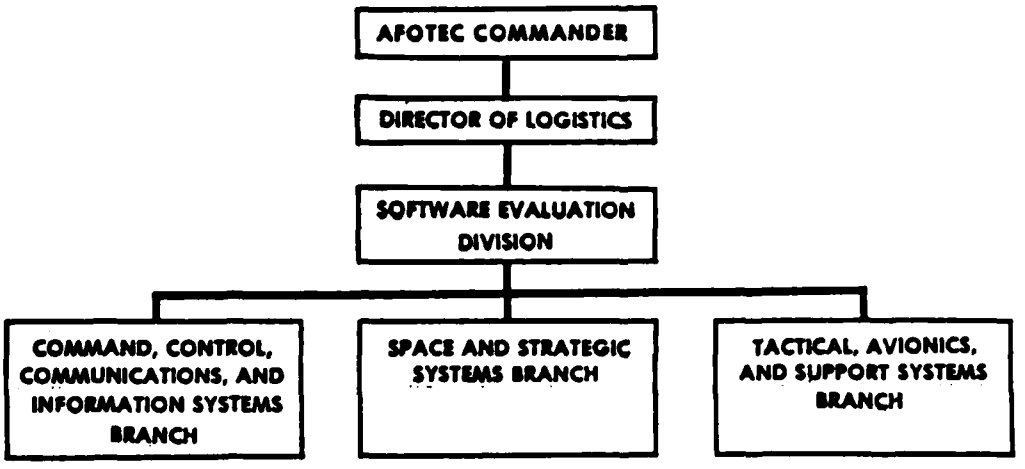
- OPERATIONAL TESTING OF SOFTWARE MUST EXAMINE ITS FUNCTIONAL PERFORMANCE, THE INTERFACE BETWEEN OPERATOR AND MACHINE, AND ITS MAINTAINABILITY.

SOFTWARE TEST AND EVALUATION

- BACKGROUND
- → AFOTEC ORGANIZATION FOR SOFTWARE OT&E
- AFOTEC APPROACH TO SOFTWARE OT&E
- FUTURE EFFORTS IN SOFTWARE OT&E

TEST MANAGEMENT APPROACH (AFOTEC HQ ELEMENTS)

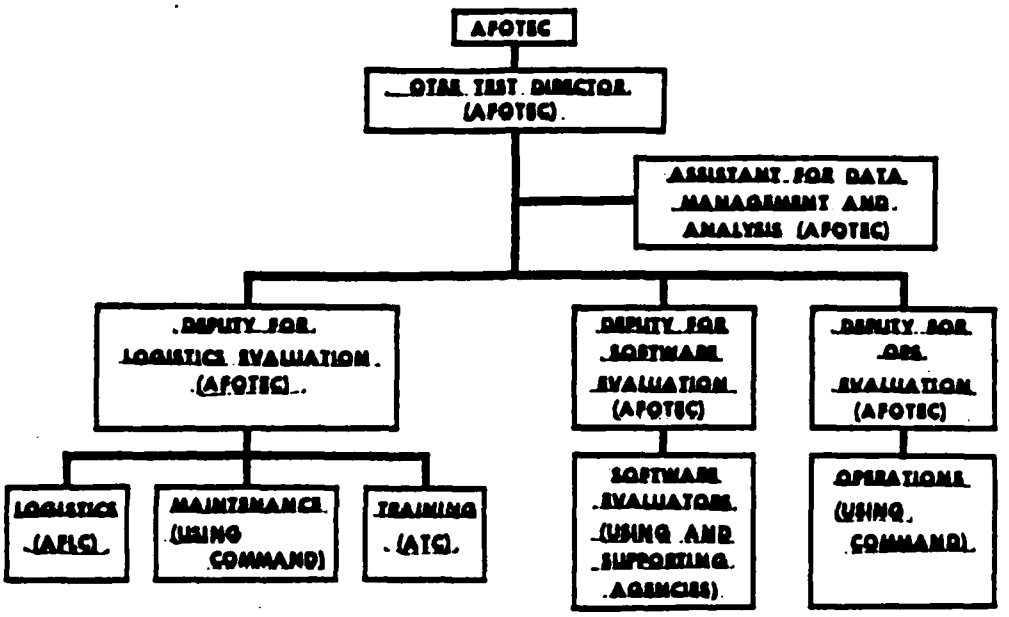




MISSION STATEMENT

RESPONSIBLE FOR PROVIDING SOFTWARE EXPERTISE FOR PLANNING, EVALUATING, AND REPORTING THE OPERATIONAL EFFECTIVENESS AND SUITABILITY OF AIR FORCE SYSTEMS WITH EMBEDDED COMPUTERS.

OT&E TEST TEAM COMPOSITION



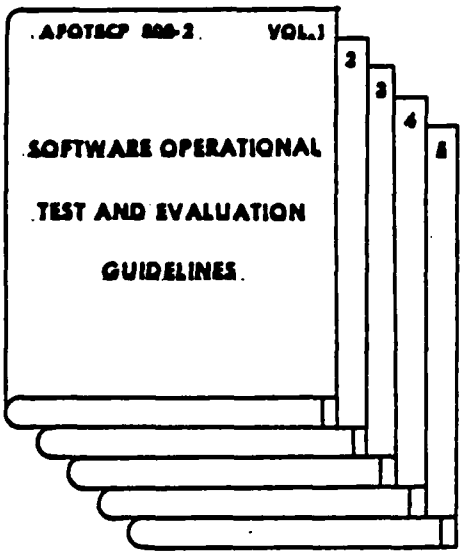
SOFTWARE TEST AND EVALUATION

- BACKGROUND
- AFOTEC ORGANIZATION FOR SOFTWARE OT&E
- → AFOTEC APPROACH TO SOFTWARE OT&E
- FUTURE EFFORTS IN SOFTWARE OT&E

SOFTWARE EVALUATION

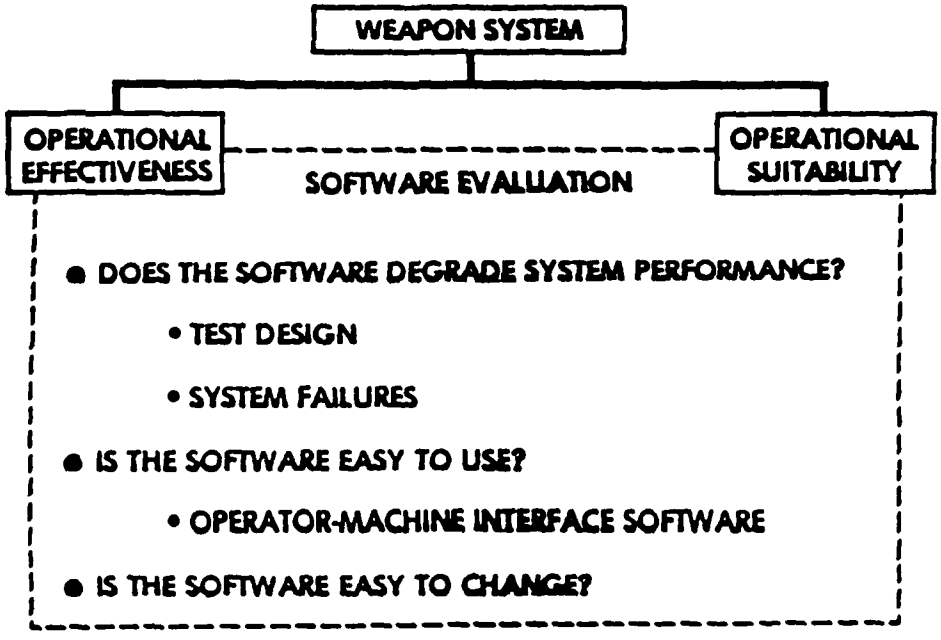
- TEST PREPARATION
  - EARLY PLANNING WITH IMPLEMENTING, USING, SUPPORTING AGENCIES
  - PREPARE OBJECTIVES, MEASURES, METHODOLOGY
  - DESIGN REVIEWS, CRWG, TPWG
- TEST CONDUCT
  - IN-PLANT TESTING
  - ON-SITE TESTING
- EVALUATION
  - TEST DATA ANALYSIS
  - TEST DATA EVALUATION
  - REPORT PREPARATION

S/W OT&E GUIDELINES



- SOFTWARE TEST MANAGER'S GUIDE
- GUIDE FOR THE TEST TEAM DEPUTY FOR SOFTWARE EVALUATION
- SOFTWARE MAINTAINABILITY EVALUATOR'S GUIDE
- SOFTWARE OPERATOR-MACHINE INTERFACE EVALUATOR'S GUIDE
- SOFTWARE SUPPORT RESOURCES EVALUATION GUIDE

SYSTEM OT&E



SOFTWARE OPERATOR-MACHINE INTERFACE EVALUATION

- USES STANDARD STRUCTURED QUESTIONNAIRE
- USES SYSTEM OPERATORS AS EVALUATORS
- DETERMINES PRESENCE/ABSENCE OF DESIRABLE ATTRIBUTES
- RESULTS ARE QUANTITATIVE VALUES

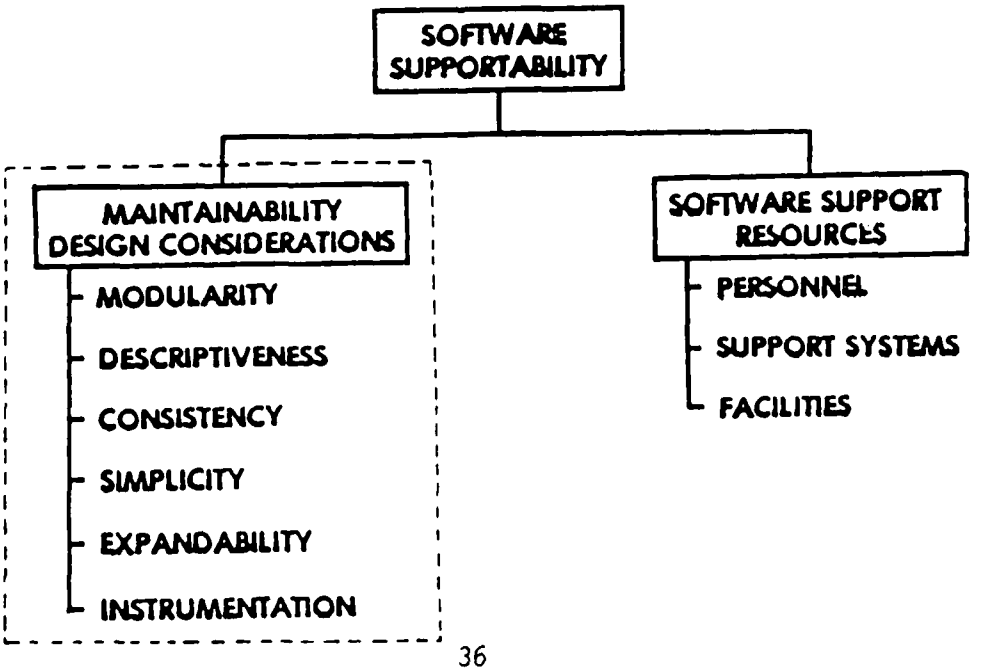
SAMPLE QUESTION

Q: 5 ..OPERATOR INPUT ERRORS ARE DETECTED, AND THE CAUSE OF THE ERROR IS DISPLAYED TO THE OPERATOR.

TEST FACTOR OR CHARACTERISTIC: ASSURABILITY.

RESPONSE (CHOOSE ONE).	POINTS.
COMPLETELY AGREE.	6.
STRONGLY AGREE.	5.
GENERALLY AGREE.	4.
GENERALLY DISAGREE.	3.
STRONGLY DISAGREE.	2.
COMPLETELY DISAGREE.	1

ELEMENTS OF SOFTWARE SUPPORTABILITY



SOFTWARE MAINTAINABILITY EVALUATION METHOD

APPROACH

- EVALUATE DOCUMENTATION AND SOURCE LISTINGS
- USE STANDARD QUESTIONNAIRES FOR ALL SOFTWARE
- USE EVALUATORS FROM EVENTUAL SUPPORT AGENCY

SOFTWARE MAINTAINABILITY EVALUATION METHOD

- QUESTIONS ARE ORGANIZED BY DESIRABLE CHARACTERISTICS:
  - MODULARITY
  - DESCRIPTIVENESS
  - CONSISTENCY
  - SIMPLICITY
  - EXPANDABILITY
  - INSTRUMENTATION
- 10 TO 15 QUESTIONS PER CHARACTERISTIC

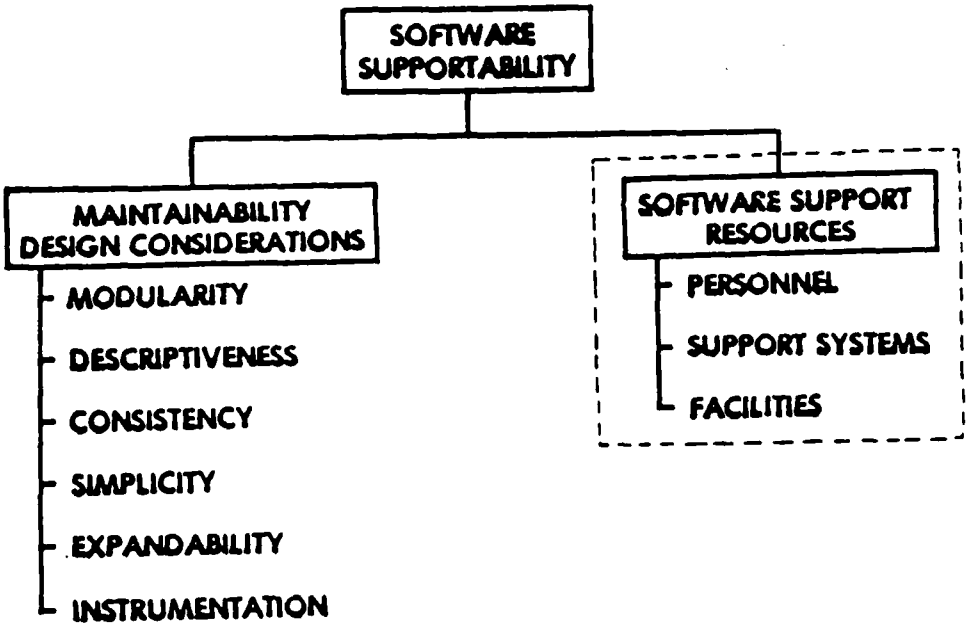
SAMPLE QUESTION

Q: S-65 THE NUMBER OF EXECUTABLE STATEMENTS IN THIS MODULE IS MANAGEABLE.

TEST FACTOR OR CHARACTERISTIC: SIMPLICITY

RESPONSE (CHOOSE ONE):	POINTS
COMPLETELY AGREE	6
STRONGLY AGREE	5
GENERALLY AGREE	4
GENERALLY DISAGREE	3
STRONGLY DISAGREE	2
COMPLETELY DISAGREE	1

ELEMENTS OF SOFTWARE SUPPORTABILITY

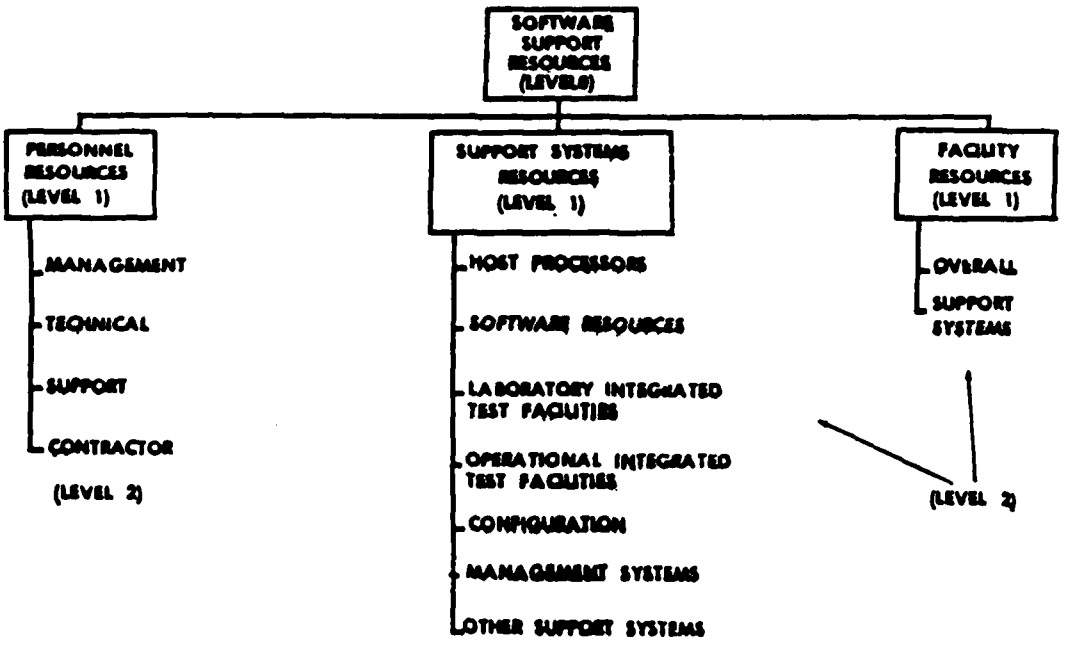


SOFTWARE SUPPORT RESOURCES EVALUATION METHOD

APPROACH

- EVALUATE EXISTING OR PLANS FOR SUPPORT FACILITY
- DETERMINE IF NECESSARY SUPPORT ASSETS ARE/WILL BE ALLOCATED
- USE A TAILORED QUESTIONNAIRE FOR EACH SYSTEM
- USE EVALUATORS FROM EVENTUAL SUPPORT AGENCY

SOFTWARE SUPPORT RESOURCES EVALUATION METHOD QUESTIONNAIRE HIERARCHY



SOFTWARE TEST AND EVALUATION

OTHER EVALUATION EFFORTS

- IV&V
- COMPUTER TIMING AND SIZING
- SOFTWARE MATURITY

SOFTWARE TEST AND EVALUATION

- BACKGROUND
- AFOTEC ORGANIZATION FOR SOFTWARE OT&E
- AFOTEC APPROACH TO SOFTWARE OT&E
- → FUTURE EFFORTS IN SOFTWARE OT&E

ACTIVE PARTICIPATION WITH SOFTWARE COMMUNITY

- RADC
- STEP
- ADA
  - ADA/JUG
  - AFWAL EAV
- STARS
- IEEE
- MC²S²

CURRENT INITIATIVES

- AFOTEC DEVELOPING METHODOLOGIES
  - SOFTWARE ERROR TRACKING
  - SUPPORTABILITY RISK ASSESSMENT
  - COMPUTER SYSTEM SECURITY
  - DIAGNOSTICS

SUMMARY

- SOFTWARE EVALUATION TOOLS EXIST
  - MAINTAINABILITY
  - COMPUTER SUPPORT RESOURCES
  - OPERATOR-MACHINE INTERFACE SOFTWARE

Effect of Testing Tools on Aegis Lifetime Support

P. Graham O'Neil

This report is on the use of tools in large scale system testing. This will include an overview of the system, the environment and the schedule. Attention will be given to areas in which improvements could be made. Cases where results can be compared to theory will be presented and assessments of tool utilization and productivity will be presented. The focus of the testing effort was in the integration phase and the goals were to provide a stable, highly reliable, well understood set of computer programs to control the system.

P. Graham O'Neil

P. Graham O'Neil received the BS degree with majors in mathematics, psychology and English from the University of Richmond. After graduation, he joined the Naval Surface Weapons Center at Dahlgren, VA. During this 17 years, he was responsible for generating six degree of freedom simulations for aircraft, missiles, and projectiles. For seven years, he was the Combat System Engineer for the AEGIS Combat System. He is presently Senior Principal System Engineer at Sanders Associates, Nashua, NH. His current interests are software tools, knowledge base system development and large scale system engineering.

EFFECT OF TESTING TOOLS

ON

AEGIS LIFETIME SUPPORT

GRAHAM O'NEIL
ITEA-84
8 NOV 1984

OUTLINE

I. INTRODUCTION

II. CHARACTERISTICS OF EACH PHASE

III. NET RESULTS FOR TWO SHIPS

IV. SHORTFALL EXAMINATION

V. ASSESSMENT OF EFFORTS

VI. SUMMARY

ELEMENT	SIZE (K WORDS)
ADS	60
ACTS	60
CND	205
FCS	40
ORTS	64
SPY	200
NCS	195

TOOLS	SIZE (K WORDS)
ISS	350
ISA	260+
ADAR	200
PMTA	30
IVG	180

TACTICAL PROGRAM AND SUPPORTING TOOL SIZE

OUTLINE

I. INTRODUCTION

II. CHARACTERISTICS OF EACH PHASE

III. NET RESULTS FOR TWO SHIPS

IV. SHORTFALL EXAMINATION

V. ASSESSMENT OF EFFORTS

VI. SUMMARY

REQUIREMENTS REVIEW CHARACTERISTICS

1. USED PSL/PSA TO CONSTRUCT AS-SPECIFIED AND AS-BUILT DATA BASES.
2. DATA BASES REFLECT LOGICAL AND PHYSICAL STRUCTURE.

RESULTS

1. UNCOVERED OMISSIONS IN REQUIREMENTS.
2. WAS USED TO DETERMINE TESTING REQUIREMENTS OF THE SIMULATOR SYSTEM.
3. SUPPORTED TESTING EFFORTS.
4. BEING USED FOR LIFETIME SUPPORT

DESIGN CHARACTERISTICS

1. TOP-DOWN DESIGN OF THE TACTICAL PROGRAMS.
2. SIMULATOR FAMILY USED SKELETON DESIGN.

RESULTS

1. DESIGN INTEGRITY OF SYSTEM.
2. EASE OF ANALYSIS TO A DESIRED LEVEL.
3. SIMULATOR FAMILY PROVIDED EASILY TRANSPORTABLE/REUSABLE SOFTWARE.
4. SIMULATOR FAMILY PROVIDED COMMON CONTROL FROM SCRIPT PROCESSOR.

CONSTRUCTION CHARACTERISTICS

1. SIMULATOR USAGE COMBINED WITH REAL WORLD STIMULUS.
2. TESTING SUPPORT SYSTEM USED IN CONJUNCTION WITH SIMULATOR.
3. STRESS TESTING FOCUSED ON SATURATION.
4. DATA DICTIONARIES.
5. COUNTER SOLUTION PROGRAMS NOT AVAILABLE.

RESULTS

1. SIMULATION COMPLEMENTED REAL WORLD FOR STRESS TESTING.
2. TESTING SUPPORT SYSTEM USED FOR ARCHIVAL OF TEST SETS, RESULTS AND REPORTS.
3. TESTING SUPPORT SYSTEM USED FOR TEST DATA GENERATION AND EXPECTED RESULTS.
4. HIGH LEVEL OF AUTOMATION, BUT MORE WAS POSSIBLE.

INTEGRATION CHARACTERISTICS

1. REAL EQUIPMENT.
2. AUTOMATED SCHEDULING AND COMPLETION PROGRAMS FOR TRACKING PROGRESS.
3. ACCESSIBLE PROBLEM REPORT DATA BASE.

RESULTS

1. SOME SURPRISES AND SHORTFALLS WITH REAL EQUIPMENT.
2. SIMULATOR USED AS WORK AROUND UNTIL CORRECTIONS WERE SUCCESSFUL.
3. SOME INTUITIVE TESTERS HIGHLY EFFICIENT.
4. GROWING PAINS.

OPERATION AND MAINTENANCE CHARACTERISTICS

1. FOCUS ON REGRESSION TESTING AND PROGRAM RELIABILITY.
2. MULTIPLE USERS/SITES AND BASELINES.
3. AUTOMATIC DATA COLLECTION SYSTEM FOR SOURCE CODE.

RESULTS

1. UNINTENTIONAL ERROR SEEDING AND PROGRAM MUTATIONS.
2. NEW TOOLS TO ANALYZE RELIABILITY.

OUTLINE

I. INTRODUCTION

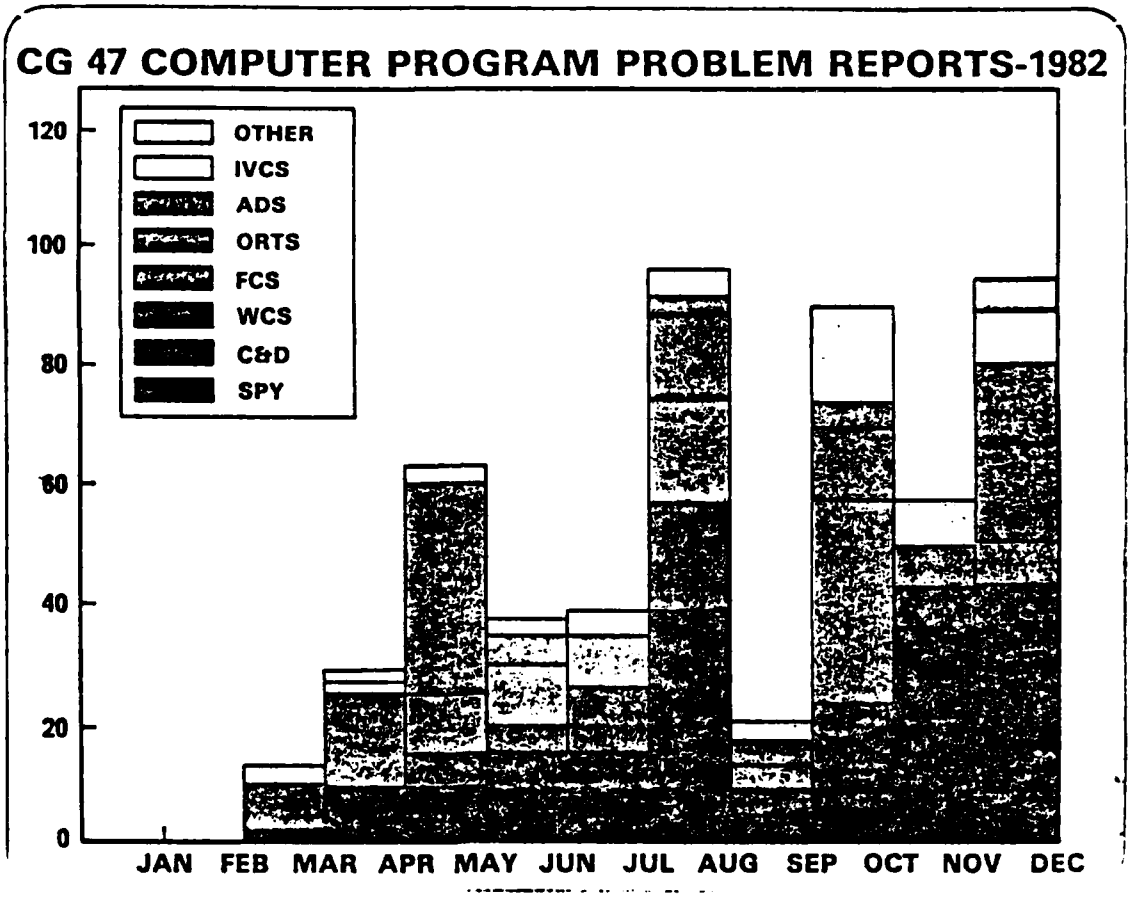
II. CHARACTERISTICS OF EACH PHASE

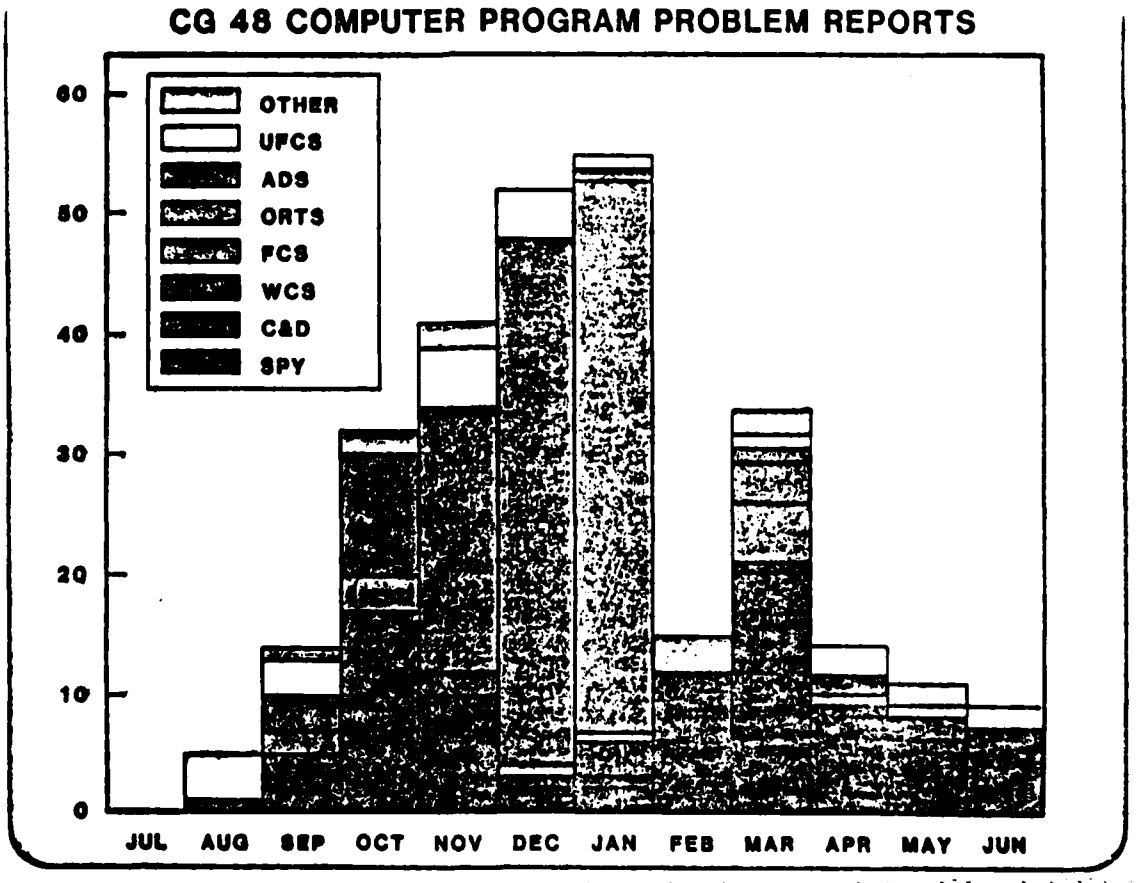
III. NET RESULTS FOR TWO SHIPS

IV. SHORTFALL EXAMINATION

V. ASSESSMENT OF EFFORTS

VI. SUMMARY


CG 47 COMPUTER PROGRAM PROBLEM REPORTS-1982
120
100
80
60
40
20
0
OTHER
IVCS
ADS
ORTS
FCS
WCS
C&D
SPY
JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC


CG 48 COMPUTER PROGRAM PROBLEM REPORTS
60
50
40
30
20
10
0
OTHER
UFCS
ADS
ORTS
FCS
WCS
C&D
SPY
JUL AUG SEP OCT NOV DEC JAN FEB MAR APR MAY JUN

OUTLINE

I. INTRODUCTION

II. CHARACTERISTICS OF EACH PHASE

III. NET RESULTS FOR TWO SHIPS

IV. SHORTFALL EXAMINATION

V. ASSESSMENT OF EFFORTS

VI. SUMMARY

INTEGRATION SHORTFALLS - FIRE CONTROL

PROBLEM CONTRIBUTORS

1) INCOMPLETE AND INCONSISTENT REQUIREMENTS/DESIGN.

2) LATE INTRODUCTION OF REAL EQUIPMENT.

3) DECEPTIVE SIMULATORS.

4) LACK OF EXCEPTION TESTING.

5) MASSIVE AMOUNTS OF UNORGANIZED DATA.

RESOLUTION CONTRIBUTORS

1) FUNCTIONAL DESIGN INTENT CAPTURED.

2) STATE DIAGRAM KEYED EXCEPTION TESTING.

3) SHORTFALL WAS NO SURPRISE.

4) DEDICATED PEOPLE.

INTEGRATION SHORTFALLS - READINESS DATA BASE

PROBLEM CONTRIBUTORS

1) BLURRED SUBSYSTEM BOUNDARIES.

2) LATE INTRODUCTION OF COMPLETE EQUIPMENT SUITE.

3) SIMULATOR LACK.

4) COMPUTER IS TIME, I/O, AND CORE BOUND.

5) TASK UNDERESTIMATED.

OUTLINE

I. INTRODUCTION

II. CHARACTERISTICS OF EACH PHASE

III. NET RESULTS FOR TWO SHIPS

IV. SHORTFALL EXAMINATION

V. ASSESSMENT OF EFFORTS

VI. SUMMARY

AEGIS INTEGRATED DATA BASE SYSTEM (AIDS)

1. ANALYZED REQUIREMENTS/SPECIFICATIONS BY EVALUATING TRACEABILITY, COMPLETENESS, CONSISTENCY AND DATA USAGE.

2. DETECTED TEST ANOMALIES BY:

   ANALYZING FUNCTIONAL PATHS.

   IDENTIFYING MISSING THRESHOLDS OR TOLERANCES.

3. DESIGNED TEST CONDITIONS TO REVEAL PROBLEMS BY:

   DETERMINING AFFECTED PROCESSES/PROCEDURES.

   IDENTIFYING DATA USED OR MODIFIED.

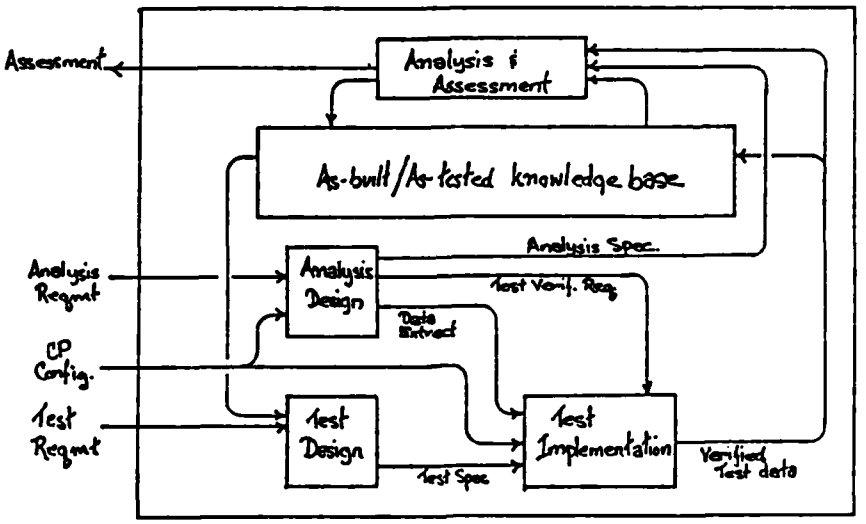
AIDS PROBLEM ANALYSIS

ISOLATES CAUSE BY IDENTIFYING

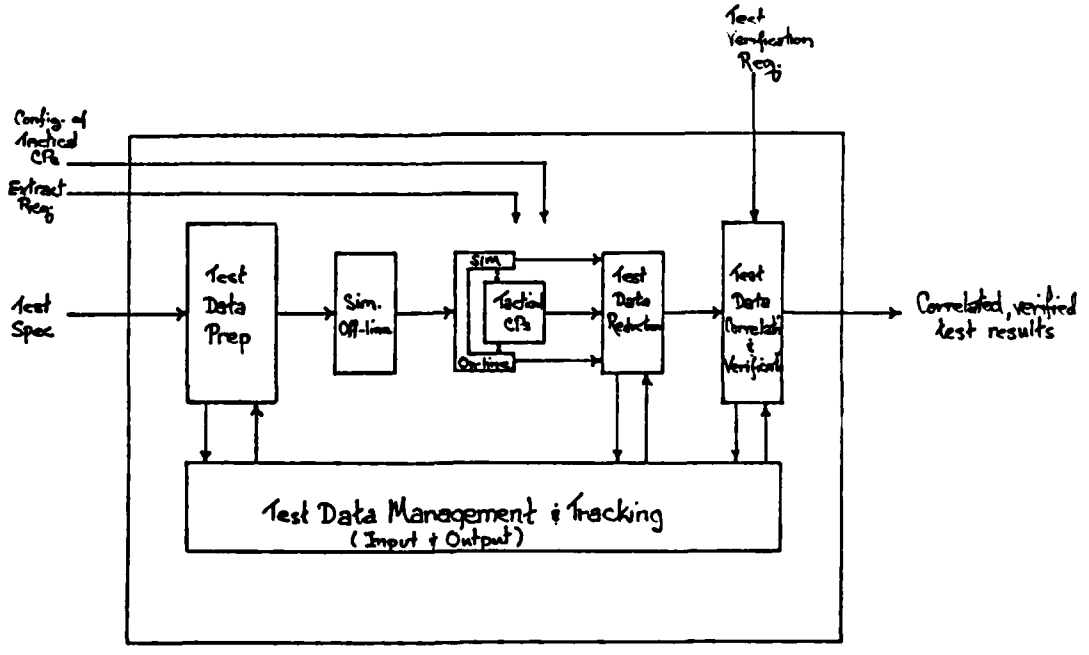
- FUNCTIONAL PATH.
- TRIGGERED EVENTS.
- DATA SET/USED.
- INTERACTION OF DATA AND PROCESS.

AIDS IN RESOLUTION BY

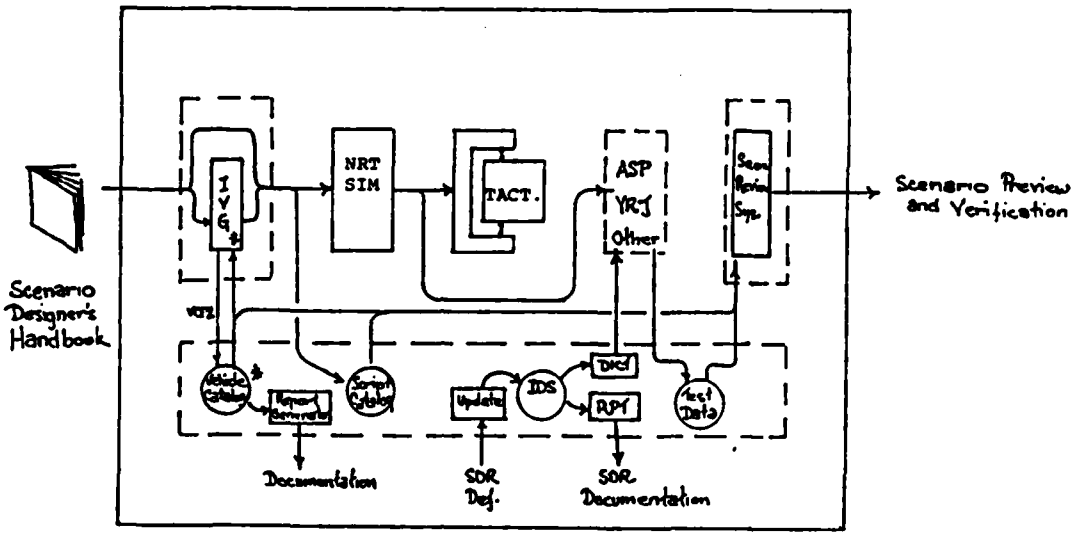
- EVALUATING PROBLEM REPORT BY POSSIBLE CAUSE.
- DETERMINING EFFECTS OF POSSIBLE SOLUTIONS.
- IDENTIFIES RE-TEST REQUIREMENTS.



AEGIS EXPERT SYSTEM



TEST IMPLEMENTATION BREAKOUT



TESTING SUPPORT SYSTEM - AS BUILT

RELIABILITY EFFORT FOCUS

1. USEFUL TO MANAGE THE DELIVERY/DEVELOPMENT OF QUALITY SOFTWARE.
2. IDENTIFY MODULES WITH INTRINSIC WEAKNESS.
3. IMPROVE MTBM/CE.
4. IMPROVE PROGRAMMER/TESTER PRODUCTIVITY.
5. DUAL STUDY OF PROBLEM REPORT LOG AND CODE CHANGES.

FIELD	SELECT	SORT	PRINT
CPPR Number			X
Log Date	X	X	X
Originating Site	X	X	X
Element	X	X	X
Problem Type	X		X
Priority	X	X	X
Baseline	X	X	X
Level	X		X
Technical Evaluation	X		X
Code Version	X		X
SCCB Status	X		X
Element Code Status	X		X
Affected Function			X

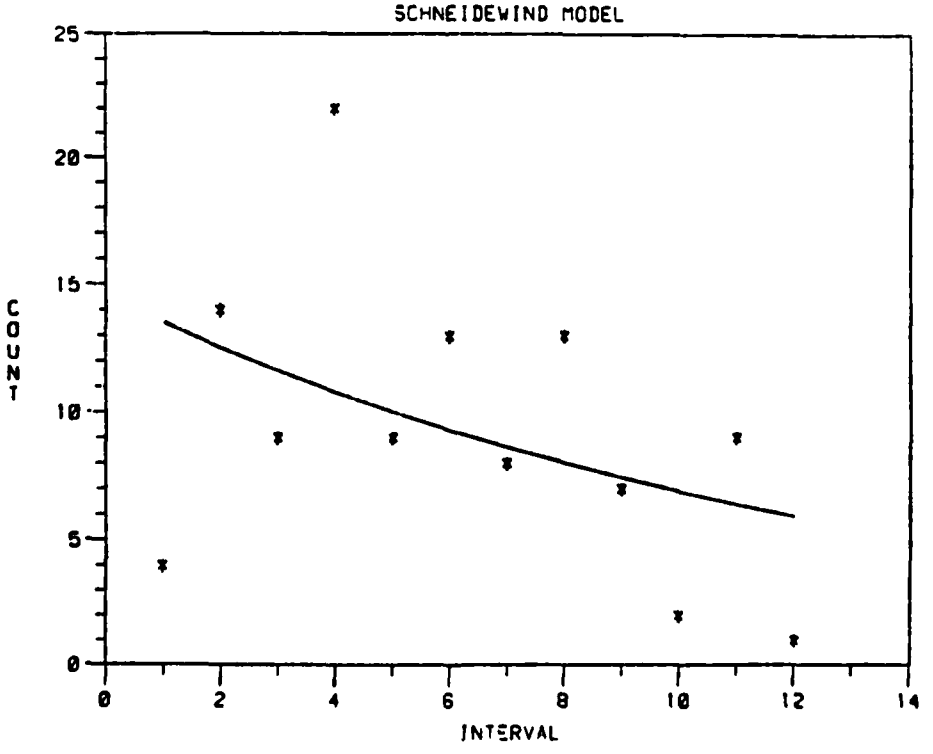
MINIMUM DATA COLLECTION CAPABILITIES FOR

RELIABILITY ASSESSMENT

CPFR ERROR COUNT DATA

CG-47 Fire Control System

DATE	PRIORITY			TOTAL
	HIGH	MEDIUM	LOW	
APRIL 1982	1	1	2	4
MAY 1982	8	4	2	14
JUNE 1982	5	3	1	9
JULY 1982	14	5	3	22
AUGUST 1982	8	1	0	9
SEPTEMBER 1982	11	1	1	13
OCTOBER 1982	3	1	4	8
NOVEMBER 1982	4	7	2	13
DECEMBER 1983	6	1	0	7
JANUARY 1983	2	0	0	2
FEBRUARY 1983	3	5	1	9
MARCH 1983	0	1	0	1
	65	30	16	111



EXPECTED NUMBER OF ERRORS IN FUTURE TESTING INTERVALS

FUTURE INTERVAL	MODEL			
	GENERALIZED POISSON	NONHOMOGENEOUS POISSON PROCESS	BROOKS & MOTLEY	SCHNEIDEWIND
1	4.92	5.51	5.67	5.51
2	4.26	5.11	5.18	5.11
3	4.04	4.74	4.74	4.74
4	3.91	4.40	4.33	4.40
5	3.81	4.08	3.95	4.08
6	3.73	3.79	3.62	3.79

RELIABILITY RESULTS

I. PROBLEM MODULES COULD BE PREDICTED BY SEVERAL METRICS.

II. LITTLE CORRELATION BETWEEN MODULES WITH NUMEROUS PROBLEMS REPORTED AND THOSE WITH NUMEROUS PROBLEMS FIXED.

III. RELIABILITY PREDICTORS APPEAR ACCURATE ENOUGH TO USE FOR DECISIONS.

IV. STATISTICS ARE REQUIRED AS COMPENSATION FOR EVALUATION VARIANCES.

V. USING THESE STATISTICS AS A BASIS FOR TAKING ACTION TO IMPROVE PROGRAMMER/TESTER PERFORMANCE HAS NOT BEEN ACCOMPLISHED.

OUTLINE

I. INTRODUCTION

II. CHARACTERISTICS OF EACH PHASE

III. NET RESULTS FOR TWO SHIPS

IV. SHORTFALL EXAMINATION

V. ASSESSMENT OF EFFORTS

VI. SUMMARY

SUMMARY

1. PROBLEM REPORTS FROM EACH SHIP ARE HALF THE PREVIOUS NUMBERS.
2. ANALYSIS OF REQUIREMENTS AND DESIGN PROVIDES A GOOD INDICATOR OF FUNCTIONS/MODULES WITH SIGNIFICANT PROBLEMS.
3. EXTENSIVE COUNTER SOLUTION DATA IS NOT AN ABSOLUTE REQUIREMENT.
4. IN AN EXTENSIVE SYSTEM, CONSISTENT ERROR REPORTING IS UNLIKELY TO BE ACHIEVED.
5. DEFINITE NEEDS EXIST FOR AUTOMATED TOOLS.
6. USAGE OF THESE TOOLS MUST BE PLANNED AND ENCOURAGED TO HAVE EFFECT.
7. WHERE SUFFICIENT LEVEL OF SIMULATION FIDELITY CANNOT BE ACHIEVED OTHER ACTION IS REQUIRED.
8. ERROR SEEDS IN PROGRAMS WERE DISCOVERED IN SHORT TIME SPANS.

A-7E Test Program

Paul Clements

The paper will describe the Naval Research Laboratory's Software Cost Reduction (SCR) project, a project to investigate the application of advanced software engineering techniques to real-time embedded software systems with tight requirements and resource constraints. Under the direction of Dr. David Parnas, SCR is providing a model of software development, including model documents, specifications, code, and procedures, for other software projects to emulate. The example application chosen is the Navy's A-7E aircraft.

SCR testing philosophy is that design and specification of software is as important to testing as is the execution of after-the-fact test cases. SCR testing embodies the following areas:

specification in the software requirement of required exception-handling;

specification of undesired events in all module interface design documents;

black box module testing, based on the module interface specifications;

clear box module testing based on module implementations;

subset testing; and

hand-over to standard Navy Test procedures.

Finally, the paper will describe promising indications of drastically reduced integration and testing time achieved as a result of implementing a small subset of the system.

Paul Clements

Paul Clements is the project manager for the Naval Research Laboratory's Software Cost Reduction (SCR) project. Under the guidance of Dr. David Parnas, SCR is investigating the application of advanced software engineering techniques to real-time embedded software systems with tight requirements and resource constraints. Mr. Clements graduated from the University of North Carolina at Chapel Hill in 1977 with a BS degree in Mathematical Sciences, and in 1980 with an MS degree in Computer Science.

PROBLEM

- NAVY SOFTWARE IS EXPENSIVE TO MAINTAIN
  - Changes Required but Risky
  - Difficult to Understand
  - Poorly Documented
  - Poorly Structured
  - Difficult to Validate
  - Difficult to Train New People

PROBLEM

- NAVY SOFTWARE IS UNRELIABLE
  - Delivered software contains errors
  - Changes often introduce errors
  - No good theory for software QA

MAJOR PROBLEM CAUSES

- Many Arbitrary Details
  - Syntax & semantics of programming languages
  - Procedures needed to communicates with devices; e.g., radar
  - Format of input data; e.g., message format
  - Communications protocols
- Rapidly changing requirements
- Long lifetime of systems
- Lack of good models
- Lack of appropriate formalisms

SOFTWARE COST REDUCTION

TEST & EVALUATION

SCR OBJECTIVE: REBUILD A-7E OPERATIONAL FLIGHT PROGRAM (OFP) USING RECENT SOFTWARE ENGINEERING TECHNOLOGY

- PRODUCE MODEL DOCUMENTS
- PRODUCE A WORKING OFP
- FLY THE PLANE
- COMPARE OFPs

USEFUL PRINCIPLES

- Separation of Concerns
  - A principle for isolating independent problems
  - Worker should only have to think about one thing at a time
  - Examples: reading input data, detecting events, allocating resources (scheduling)

USEFUL PRINCIPLES

- Abstraction
  - A principle for exploiting commonalities
  - Abstract means conceived apart from special cases
  - Many-to-one mapping
  - Examples: differential equations, graphs, data types

COROLLARIES

- Process Structuring (Cooperating Sequential Processes)
  - Separation of concerns to permit scheduling independent of function
- Undesired Events
  - Separation of concerns to permit appropriate error handling

REQUIREMENTS SPECIFICATION - COMPLETENESS

- Computer Characteristics
- Input and Output Interfaces
- System States and History
- Output Values
- Timing
- Accuracy
- Likely Changes
  - Exception-handling
  - Subsets
  - Other likely changes
- Dictionary

SOFTWARE DESIGN

- The (Information-Hiding) Modular Structure
- Separate Concerns of:
  - **Hardware details (Hardware-Hiding Module)**
  - **Requirements (Behavior-Hiding Module)**
  - **Software design decisions (Software Decision Module)**

FIRST-LEVEL MODULES

- Hardware-Hiding Module
- Behavior-Hiding Module
- Software-Decision-Hiding Module

SECOND-LEVEL MODULES

- Hardware-Hiding Module
  - Extended Computer Module
  - Device Interface Module
- Behavior-Hiding Module
  - Function Driver Module
  - Shared Services Module
- Software-Decision-Hiding Module
  - Physical Models Module
  - Data Banker Module
  - Application Data Types Module
  - System Generation Module
  - Software Utility Module

THIRD-LEVEL MODULES

Extended Computer Module
- Data Type Module
- Input/Output Module
- Parallelism Control Module
- Sequence Control Module
- State Module
- Test Module
- Timer Module
- Virtual Memory Module
- Interrupt Handler Module

Device Interface Module
- Air Data Computer
- Angle of Attack Sensor
- Audible Signal Device
- Computer Fail Device
- Doppler Radar Set
- Flight Information Displays
- Forward Looking Radar
- Head-Up Display
- Inertial Measurement Set
- Panel
- Projected Map Display Set
- Radar Altimeter
- Shipboard Inertial Navigation
- Slew Control
- Switch Bank
- TACAN
- Visual Indicators
- Waypoint Information System
- Weapon Characteristics
- Weapon Release System
- Weight on Gear

Function Driver Module
- Air Data Computer Functions
- Audible Signal Functions
- Computer Fail Signal Functions
- Doppler Radar Functions
- Flight Information Display Functions
- Forward Looking Radar Functions
- Head-Up Display Functions
- Inertial Measurement Set Functions
- Panel Functions
- Projected Map Display Set Functions
- SINS Functions
- Visual Indicator Functions
- Weapon Release Functions
- Ground Test Functions

Shared Services Module
- Mode Determination Module
- Stage Director Module
- Shared Subroutine Module
- System Value Module
- Panel I/O Support Module
  - Input
  - Display format
  - Configuration

Application Data Type Module
- Numeric Data Type Module
- State Transition Event Module

Physical Models Module
- Earth Characteristics Module
- Aircraft Motion Module
- Spatial Relations Module
- Human Factors Module
- Weapon Behavior Module
- Target Behavior Module
- Filter Behavior Module

Data Banker Module
- (one submodule per producing submodule

System Generation Module
- System Generation Parameter Module
- Module Version Selection Module
- Subset Selection Module
- Support Software Module

Software Utility Module

SOFTWARE DESIGN

- Module Interface Specification
  - **Apply abstraction to produce abstract interface**
  - **Users see (use) only externally visible functions**
  - **Redundancy for error checking (review)**
  - **Formalism for precision**
  - **UE specification to highlight significant error handling decisions**

EXAMPLE OF ABSTRACT INTERFACE SPECIFICATION

- Chosen from Device Interface Module

DI.DRS
DOPPLER RADAR SET

DI.DRS.1. INTRODUCTION

The Doppler Radar Set (DRS) is a sensor that measures aircraft ground speed and drift angle during flight.

DI.DRS.2. INTERFACE OVERVIEW

DI.DRS.2.1 ACCESS PROGRAM TABLE

Program Name	Parm type	Parm info	Undesired events
+G_DRS_MODE+	pl:drs_mode;0	!+DRS mode+!	None
+G_DRS_DRIFT_ANGLE+	pl:angle;0	!+drift angle DRS+!	ZWrong DRS modeZ
+G_DRS_GROUND_SPEED+	pl:speed;0	!+gnd speed DRS+!	
+G_DRS_RELIABILITY+	pl:boolean;0	!+DRS reliable+!	
+G_DRS_TEST_RESULTS+	pl:boolean;0	!+DRS test result+!	
+START_DRS+	none		
+STOP_DRS+	none		

Program Effects

+START_DRS+	Changes the mode from OFF to one of the other modes. The mode entered depends on conditions beyond the control of the software.
+STOP_DRS+	Sets the mode to OFF.

DI.DRS.2.2. EVENTS SIGNALLED

@T(!+DRS mode changed+!)
@T/@F/=T/=F(!+DRS reliable+!)

DI.DRS.2.3 MODES OF THE MODULE

The modes of this module are OFF, OPERATE, MEMORY, STNDBY, and TEST. They are mutually exclusive. The following table defines what program calls are legal in what mode; L=legal, NL=not legal.

SCR TEST & EVALUATION

DEVELOPMENT OVERVIEW

* DESIGN THE MODULAR STRUCTURE
* SPECIFY THE MODULE INTERFACES
* DESIGN SUBSETS
* CODE SUBSET FUNCTIONS
* TEST (PARTIAL MODULES)
* TEST SUBSET ON TC-2 SIMULATOR
* DELIVER COMPLETE OFP FOR FLIGHT TEST
* COMPARE

APPLICATION OF PRINCIPLES

DOCUMENT / REVIEW

DEFINE IMPLEMENTATION-INDEPENDENT REQUIREMENTS

ORGANIZE SYSTEM INTO MODULES

DEFINE MODULE INTERFACES — DESIGN PROCESS STRUCTURE

WRITE PSEUDO-CODE — DEFINE USES

WRITE IMPLEMENTATION CODE

TEST ON FLIGHT SIMULATOR

SCR TEST & EVALUATION

MODULE TEST PHILOSOPHY

* BLACK BOX TESTS

  - USE ONLY INTERFACE SPECIFICATIONS IN CONSTRUCTING TESTS

  - TEST INTERFACE FUNCTIONS OVER INPUT DOMAIN

  - CAUSE FUNCTIONS TO PRODUCE RESULTS OVER OUTPUT RANGE

  - (PSEUDO) CONTINUOUS DOMAIN INPUT

    = USE BOUNDARY VALUES & TYPICAL VALUES

  - DISCRETE INPUT DOMAIN

    = USE EVERY INPUT VALUE, IF POSSIBLE, OTHERWISE SPECIAL VALUES & AT LEAST ONE NON-SPECIAL VALUE

  - (PSEUDO) CONTINUOUS OUTPUT

    = PRODUCE BOUNDARY VALUES & TYPICAL VALUES

  - DISCRETE OUTPUT

    = PRODUCE EVERY VALUE, IF POSSIBLE, OTHERWISE SPECIAL VALUES & AT LEAST ONE NON-SPECIAL VALUE

SCR TEST & EVALUATION

MODULE TEST PHILOSOPHY

* CLEAR BOX TESTS
  - CODE READING
    - ORGANIZE PROGRAM INTO EQUIVALENCE CLASSES OF STATES

      2 STATES ARE IN THE SAME CLASS IFF ONE CAN PROVE THAT IF THE PROGRAM WORKS CORRECTLY WHEN STARTED IN ONE STATE, IT WILL WORK CORRECTLY WHEN STARTED IN THE OTHER STATE
  - SELECT ONE TEST CASE FROM EACH EQUIVALENCE CLASS

SCR TEST & EVALUATION

SUBSET TESTING

* SUBSET CAN BE VIEWED AS A MODULE; APPLY MODULE TEST PRINCIPLES
  - IF SUBMODULES OF THE SUBSET ARE ALL EXTERNALLY-VISIBLE, SUBSET TESTS ARE THE SAME AS MODULE TESTS (SUBSET INTERFACE IS THE UNION OF THE SUBMODULE INTERFACES)
  - IF THERE ARE HIDDEN SUBMODULES, THEN THE SUBSET INTERFACE IS DIFFERENT THAN THE UNION OF THE SUBMODULE INTERFACES; SUBSET TESTS MAY BE DIFFERENT FROM MODULE TESTS

WHAT WAS TESTED AT NWC IN OCTOBER

- PRELIMINARY DEMONSTRATION SUBSET
  - EXERCISE EVERY 2nd LEVEL MODULE IN THE OFP RUNNING ON THE TC-2
  - TAKE INPUTS AND MOVE SYMBOLS ON HUD
- SUPPORT TOOL SUBSET
  - TRANSLATOR GENERATOR
  - EC TRANSLATOR

INTEGRATION AT NWC

- INTEGRATED IN 3 NIGHTS
- TOTAL OF 9 ERRORS
- NO HAND-PATCHES - ALL WERE CORRECTED IN SOURCE

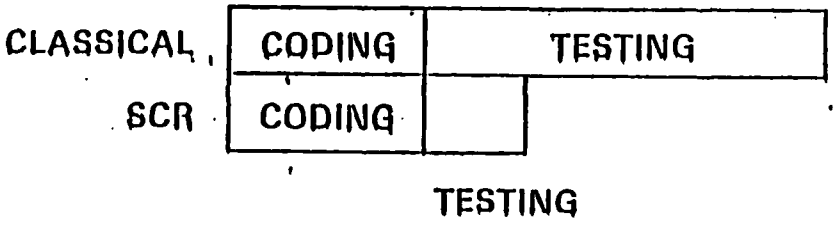
NO ERROR CROSSED A MODULE BOUNDARY

CODING AND TESTING EFFORT

- CLASSICAL: 1 WORK HOUR OF TESTING FOR EVERY 1/2 WORK HOUR OF CODING

  (WOLVERTON (1974), DALY (1977), BOEHM (1981))

- IN PDS WE FOUND: 10 WORK MINUTES OF TESTING FOR EVERY 1/2 WORK HOUR OF CODING



DoD-STD-SDS and SQS

LCDR Michael T. Gehl

In 1977, the Joint Logistics Commanders chartered a Joint Policy Coordinating Group on Computer Resource Management (CRM). The mission of the CRM was to coordinate and ensure consistency in the preparation of new and revised regulations and standards, to provide recommendations on critical resource areas, and to provide a focal point for coordinating standardization programs. The CRM subsequently chartered a subgroup for Computer Software Management (CSM) to review policies, procedures, regulations, and standards relating to computer software and forward specific recommendations to the CRM on critical areas related to software acquisition and management, including software development, quality, testing, and post-development support.

The CSM has structured their activities into three projects; the software development project, the software quality project, and the post-development software support (PDSS) project. A software development cycle model was developed and is the fundamental framework for the CSM projects. The CSM has hosted three joint industry/government workshops. The first two, held in Monterey, California in 1979 and 1981, dealt with issues concerned with software development and software quality. The third workshop, held in Orlando, Florida, in 1983, dealt exclusively with PDSS issues. The results of the workshops are being incorporated into the products of the CSM projects.

The software development project consists of a Joint Regulation on the Management of Computer Resources in Defense Systems, a tri-service coordinated DoD Standard on Software Development(DoD-STD-SDS), a collection of 25 Data Item Descriptions, proposed changes to several existing Militalry Standards, and a guidebook for program managers on implementing the new standard. The software quality project consists of a Joint Regulation on the Software Quality Program, a DoD Standard on Software Quality (DoD-STD-SQS), and a guidebook for program managers. The post development software support project has only recently been initiated, and currently consists of an action plan which, when approved, will start several projects in the area of PDSS.

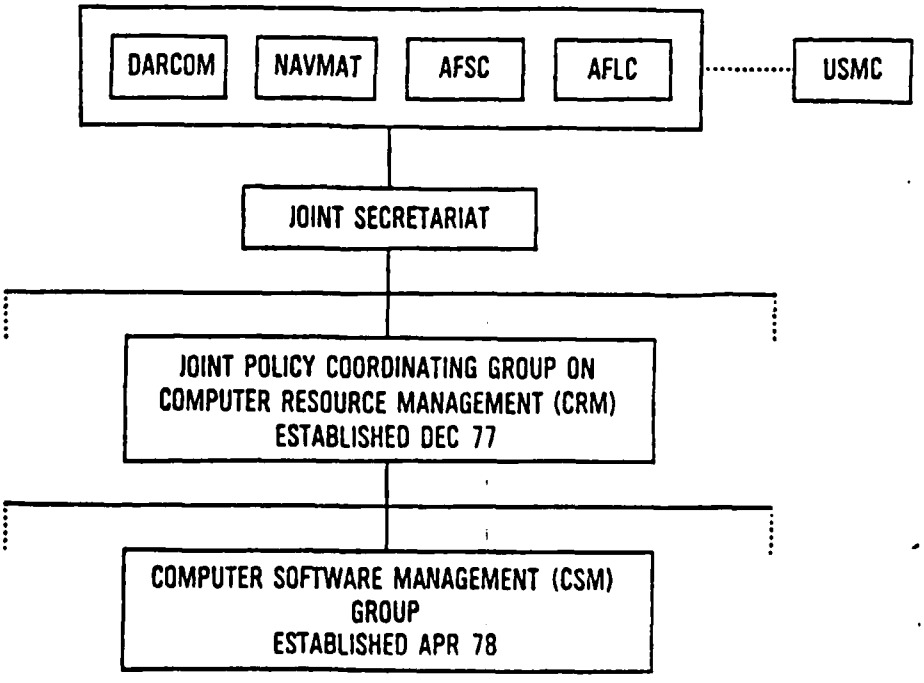
LCDR Michael T. Gehl

Lieutentant Commander Gehl is currently the Assistant Division Director for the Computer Resources Division at the Naval Electronic Systems Command (NAVELEX). He is the primary interface with the Embedded Computer Program Office at the Naval Material Command (NAVMAT) for matters relating to the Navy Standard Computer Program, and for NAVMAT ECR policies.

LCDR Gehl received a Master of Science degree in Computer Science from the Naval Postgraduate School in Monterey, California, prior to reporting to NAVELEX in September of 1982. Prior to his postgraduate education, he served on USS Henderson (DD 785), USS Jouett (CG 29), USS Lynde McCormick (DDG-8), and COMDESRON TWO SEVEN. He graduated from Iowa State University in 1971 with a BS degree in Electronic Engineering.

JOINT LOGISTIC COMMANDER'S SOFTWARE STANDARDIZATION PROGRAM

JOINT LOGISTICS COMMANDERS' [JLC] ORGANIZATION



MAJOR EVENTS

1977 — CRM FORMED

1978 — CSM FORMED

1979 — MONTEREY I

1981 — MONTEREY II

1982 — DRAFT POLICY, STANDARDS, DIDs

1983 — DRAFT QUALITY POLICY, STANDARDS

1983 — ORLANDO I

JOINT POLICY COORDINATING GROUP (JPCG) ON COMPUTER RESOURCE MANAGEMENT (CRM)

• U.S. ARMY MATERIAL COMMAND (AMC)	COL H. ARCHIBALD
• NAVAL MATERIAL COMMAND (NMC)	CAPT D. BOSLAUGH (CHAIRPERSON)
• AIR FORCE LOGISTIC COMMAND (AFLC)	LT COL J. HARRINGTON
• AIR FORCE SYSTEMS COMMAND (AFSC)	COL K. NIDIFFER
• U.S. MARINE CORPS (USMC)	MAJ K. PTACK

CRM MISSION

- TO COORDINATE AND INSURE CONSISTENCY IN THE PREPARATION OF NEW AND REVISED REGULATIONS AND STANDARDS
- TO PROVIDE RECOMMENDATIONS ON CRITICAL RESOURCE AREAS
- TO PROVIDE A FOCAL POINT FOR COORDINATING STANDARDIZATION PROGRAMS

COMPUTER SOFTWARE MANAGEMENT [CSM] SUBGROUP

• AMC	C. OGLESBY
• NMC	LCDR M. GEHL
• AFLC	D. KVENVOLD
• AFSC	CAPT L. COOPER (CHAIRPERSON)

CSM SUBGROUP MISSION

TO REVIEW POLICIES, PROCEDURES, REGULATIONS, AND STANDARDS RELATING TO COMPUTER SOFTWARE AND FORWARD SPECIFIC RECOMMENDATIONS TO THE JPCG-CRM ON CRITICAL AREAS RELATED TO SOFTWARE ACQUISITION AND MANAGEMENT, INCLUDING SOFTWARE DEVELOPMENT, QUALITY, TESTING, AND POST-DEVELOPMENT SUPPORT.

MONTEREY I AREAS OF CONCERN

- SOFTWARE ACQUISITION POLICY
- SOFTWARE ACQUISITION AND DEVELOPMENT STANDARDS
- SOFTWARE DOCUMENTATION STANDARDS
- SOFTWARE QUALITY ASSURANCE STANDARDS
- SOFTWARE ACCEPTANCE CRITERIA

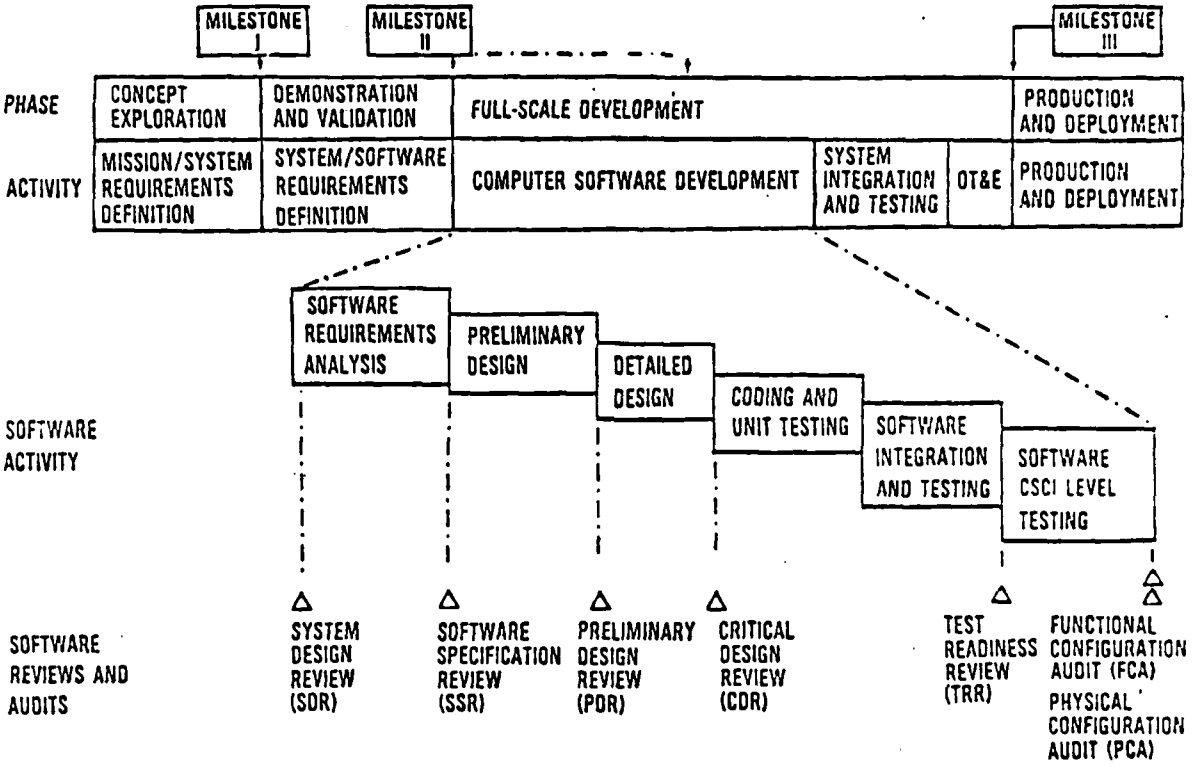
MONTEREY II AREAS OF CONCERN

- DRAFT DID REVIEW
- HARDWARE/SOFTWARE/FIRMWARE CONFIGURATION ITEM SELECTION CRITERIA
- STANDARDIZATION AND ACCREDITATION OF COMPUTER ARCHITECTURES
- SOFTWARE COST ESTIMATING
- SOFTWARE REUSEABILITY

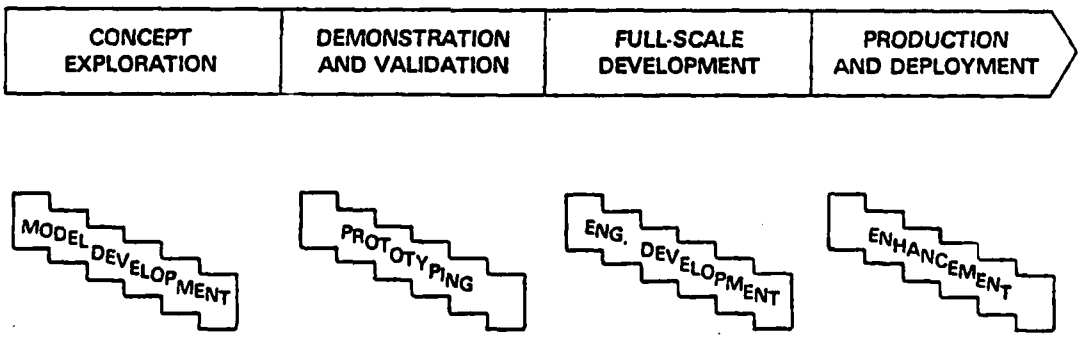
ORLANDO I AREAS OF CONCERN

- INDUSTRY/GOVERNMENT WORKFORCE MIX
- IV & V BY SUPPORT PERSONNEL
- COST OF OWNERSHIP
- SOFTWARE SUPPORT ENVIRONMENT
- CHANGE IMPLEMENTATION
- CONFIGURATION MANAGEMENT

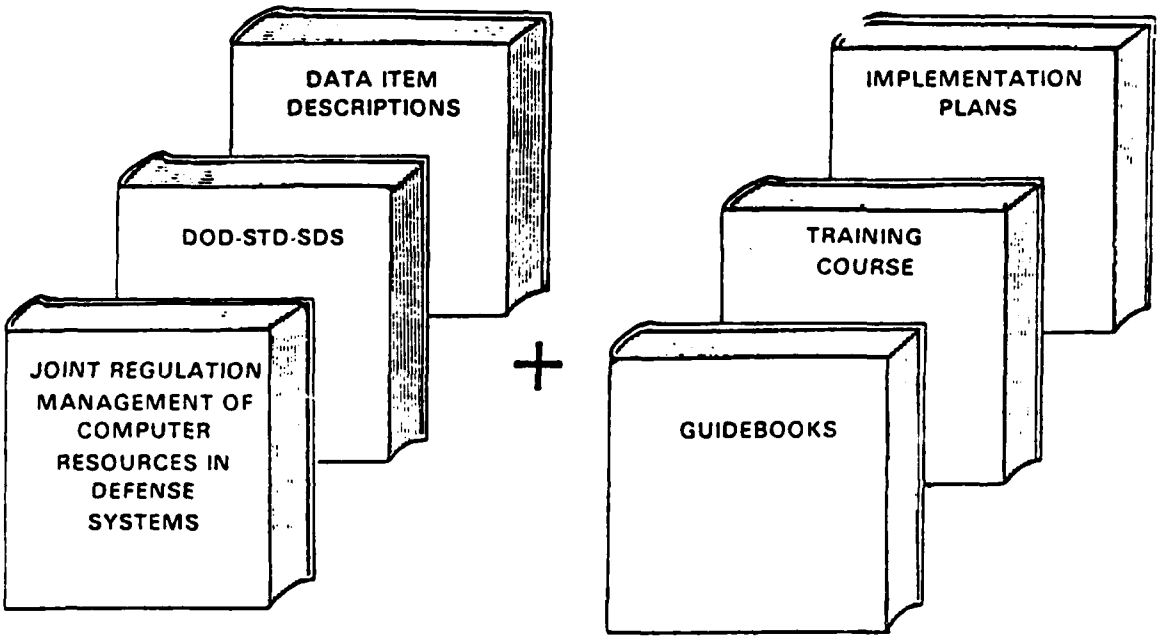
COMPUTER SOFTWARE DEVELOPMENT CYCLE



COMPUTER RESOURCE DEVELOPMENT



SOFTWARE DEVELOPMENT STANDARDIZATION PROJECT



JOINT SERVICE SOFTWARE STANDARDS: PROGRAM OBJECTIVE

- PRODUCE A COMPLETE, CONSISTENT SET OF SOFTWARE ACQUISITION AND DEVELOPMENT STANDARDS WHICH
  - ESTABLISH A WELL-DEFINED AND EASILY UNDERSTOOD SOFTWARE ACQUISITION AND DEVELOPMENT PROCESS
  - PROVIDE ADEQUATE VISIBILITY AND CONTROL MECHANISMS THROUGHOUT ALL PHASES OF ACQUISITION
  - REDUCE CONFUSION AND ELIMINATE CONFLICTS IN EXISTING SOFTWARE STANDARDS

CSM SUBGROUP APPROACH

- DEFINE SOFTWARE DEVELOPMENT FRAMEWORK
- ESTABLISH JOINT POLICY
- DEVELOP TRI-SERVICE DEVELOPMENT/DOCUMENTATION STANDARDS
- DEVELOP SOFTWARE QUALITY ASSESSMENT/MEASUREMENT STANDARDS
- DEVELOP TAILORING GUIDEBOOKS AND TRAINING COURSES
- COORDINATE/IMPLEMENT ON NEW SYSTEMS

SOFTWARE DOCUMENTATION STANDARDS

DIDS DEVELOPED FOR

- **MANAGEMENT PLANS**
- **ENGINEERING SPECIFICATIONS/DOCUMENTATION**
- **TEST PLANS/PROCEDURES/REPORTS**
- **SUPPORT DOCUMENTATION**

WILL PROVIDE COMMON DOCUMENTATION SET

SOFTWARE QUALITY STANDARDIZATION PROJECT

PROGRAM THRUST

- END-USE ORIENTED
- NOT ORGANIZATIONALLY LIMITED
- PLACES EMPHASIS ON PROCESS AS WELL AS PRODUCT
- PLACES EMPHASIS ON PROJECT FRONT END
- TOTAL PROGRAM
  - SOFTWARE QUALITY PROGRAM EMBEDDED WITHIN MANAGEMENT OF SOFTWARE PROJECT
  - PLACES RESPONSIBILITY FOR PRODUCT QUALITY ON PROGRAM MANAGER
- SOFTWARE QUALITY PROGRAM ≠ SQA

SOFTWARE QUALITY STANDARDIZATION PROJECT

SOFTWARE QUALITY PROGRAM

– A SOFTWARE QUALITY PROGRAM SHALL BE IMPLEMENTED AND INCLUDE ACTIVITIES NECESSARY TO ACHIEVE A DEFINITION OF QUALITY REQUIREMENTS, THE METHODS TO BUILD QUALITY INTO THE PRODUCT AND MAINTAIN THAT LEVEL OF QUALITY, AND THE DETERMINATION OF THE QUALITY LEVEL ATTAINED IN THE PRODUCT

SOFTWARE QUALITY STANDARDIZATION PROJECT

SOFTWARE QUALITY REQUIREMENTS (STANDARDS)

SDS	SQAM	REQUIREMENTS \ STANDARD	FAR	MIL-Q-9858A	MIL-S-52779A	DOD-STD-1679A
4.4	4.8	SUBCONTRACTOR CONTROLS				
4.4	4.8	– ESTABLISH	–	5.2	3.3	4.6
4.4	4.8	– EVALUATE	–	5.1	3.3	4.6
		CORRECTIVE ACTION SYSTEM				
5.7.1.4	–	– ESTABLISH	–	3.5	3.2.9	5.8.4(p)
5.7.1.4	–	– IMPLEMENT	–	3.5 6.5	3.2.9	5.8.4(p)
-	4.5	-- EVALUATE/ASSESS	–	3.5	3.2.9	5.9.10
–	5.7.1	PERFORM ACCEPTANCE INSP	46.5 46.202-1	6.3		5.10 (p)
		LIBRARY CONTROLS	--	–	3.2.6	
–	–	– ESTABLISH	–	–	–	5.5.4
–	–	– PERFORM	–	–	–	5.5
–	5.8.1	– EVALUATE/ASSESS	–	–	–	–

SOFTWARE QUALITY STANDARDIZATION PROJECT

EXPECTED RESULTS

- ESTABLISHES A FULLY INTEGRATED SYSTEMATIC AND WELL-DEFINED SOFTWARE QUALITY PROGRAM
- INSTITUTIONALIZES THE TRAINING PROGRAMS AND GUIDANCE ESSENTIAL TO IMPLEMENTATION
- PROVIDES COMMONALITY AMONG SERVICES
  - FEWER REGULATIONS AND STANDARDS
  - FEWER DATA ITEM DESCRIPTIONS
- REDUCES COST TO CONTRACTORS AND GOVERNMENT
  - IMPROVES CONTRACTOR PRODUCTIVITY
  - ALLOWS CONTRACTORS TO STANDARDIZE AND AUTOMATE PROCESSES
- PROVIDES MORE ACCURATE VISIBILITY INTO SOFTWARE DEVELOPMENT STATUS

JOINT POLICY
MANAGEMENT OF COMPUTER RESOURCES IN DEPENSE SYSTEMS

TEST PLANNING

"TEST & EVALUATION PLANNING SHALL
BEGIN AS EARLY AS POSSIBLE AND SHALL
CONTINUE THROUGHOUT THE SYSTEM
ACQUISITION LIFE CYCLE"

"QUANTITATIVE & DEMONSTRABLE PERFORMANCE
OBJECTIVES AND EVALUATION CRITERIA
SHALL BE ESTABLISHED..."

"TEST PLANNING SHALL INCLUDE BOTH
FORMAL & INFORMAL TESTING"

TEST PLANNING/CONDUCT

GOVERNMENT TEST PLANNING

- TEST PLANNING WORKING GROUP (TPWG)
- TEST & EVALUATION MASTER PLAN (TEMP)
- COMPUTER RESOURCES LIFE CYCLE MANAGEMENT PLAN (CRLCMP)

CONTRACTOR TEST PLANNING

- SOFTWARE TEST PLAN (STP)
- SOPTWARE TEST DESCRIPTION (STD)
- SOFTWARE TEST PROCEDURES (STPR)

TEST READINESS REVIEW

A TEST READINESS REVIEW SHALL BE CONDUCTED FOR EACH CSCI TO DETERMINE THAT THE SOFTWARE TEST PROCEDURES ARE COMPLETE AND TO ENSURE THAT THE CONTRACTOR IS READY FOR FORMAL TESTING.

LEVELS OF TESTING

FORMAL TESTING

- A TEST WHICH IS CONDUCTED IN ACCORDANCE WITH TEST PLANS AND PROCEDURES APPROVED BY THE PROCURING AGENCY AND WITNESSED BY AN AUTHORIZED PROCURING AGENCY REPRESENTATIVE.
- CSCI TESTING
- MAY INCLUDE SOME UNIT AND/OR CSC TESTING
- SYSTEM TESTING
- DOCUMENTED IN SOFTWARE TEST REPORT

INFORMAL TESTING

- ANY TEST WHICH DOES NOT MEET ALL THE REQUIREMENTS OF A FORMAL TEST
- UNIT TESTING
- CSC INTEGRATION TESTING
- DOCUMENTED IN SOFTWARE DEVELOPMENT FOLDERS

Transcript of Panel Discussions

Panelists: Donald R. Greenlee

Richard A. DeMillo

Francis Bartosik

Major Frederick Foster

Paul Clements

LCDR Michael T. Gehl

DeMillo: Mike read a list of issues this morning and I would like to get to some of them during this panel discussion. To follow the format that Mr. Watt laid out during the Panel discussion yesterday, we should lead off with discussions from the audience. I know that Dr. Phil Dickinson had a comment about an hour ago that would be appropriate for the Panel. The issue was whether or not MIL-STD-SDS or the methodologies that are growing up around it should handle concerns on the right hand side of the life cycle. The issue that Phil was talking about, in particular, was that he sees in the operational test community the need for hooks in the software and, more generally, design for testability issues in the software.

Gehl: I think that we agree with the requirement that the entire system life cycle needs to be looked at, back end as well as the front end. All of the life cycle has to be looked at when you are defining the requirements and developing the system. We think that is fundamental. I don't see that as an immediate goal in any effort that I know of that is ongoing. I talked to you already about the proposed revisions to SDS. Even though we don't have it on the street yet, there is a proposed revision already started and we do plan on looking at several issues that we just can't fix in the time frame for getting the first version out. One of those issues is what we have been calling the system-engineering concept: for example, the question of how the software that we are developing fits into the entire system. That is where most of these issues of testing and post development software support come in. Those issues come in when you realize that it is not just software, not just a black box, but a system.

DeMillo: Let me see if I can carry Phil's comments a little farther. It seems to me that from the testing community's point of view that we are spending a lot of time validating Ada compilers, talking about standard instruction set architectures, and doing a lot of very careful funneling of the design process along lines that we think are going to help the system. Yet there is one area - the area of making sure that the software that gets put into operational systems can be instrumented, tested and evaluated in a rational way - that is not being addressed.

Gehl: We would certainly like to see that. But you have to understand that we four members of the CSM cannot solve all the problems. We can't say here's how to fix the problems for testing, configuration management, and quality. We need input from you, the test community, so that when it is appropriate, you can say we think that you can incorporate this particular technology into your document. We would welcome input about how to incorporate that either into SDS or into MIL-STD-499 "System Engineering Requirements" or, in general, wherever you think it should best be placed. I think that is what needs to be done. We try to do too much and now we need to get the rest of the community involved.

DeMillo: Any other followups on that? Yes, Ronnie?

Martin: I'd like to hear responses from the other members of the Panel on that.

Clements: Shall I start? As I understand the question, we are talking about pieces of software such that you can tell things about the operation of the software at runtime to help you test and debug. The undesirable events are taken into account or designed by flags that we can raise when they occur. Now, it turns out that we cannot support that kind of software in our application in the production version. There is not enough room. A lot of systems have that problem, there is just not enough room to have those kinds of frills. I am sorry that is the only word that comes to my mind and I don't mean to trivialize the matter by calling it frills, but they are extras. So what we are going to do is to build our system, with the undesired event handling code present. We are going to implement in subsets so we have the room and then piece by piece we will take that out as we are convinced that those undesired events can no longer occur. We handle the problem up front by putting it in our software requirements document. If you want hooks in the software that is delivered to you, you had better require it. That is how SCR handles that issue.

Dickinson: It has implications at the build level, at the OT level, and at the fielded level so that when it does hiccup, as Rich says, you can tell what is going on. It's probably also useful for training because you can start getting human interactions. You can tell what's going on and sort out whether a human screwed it up or the system is really doing something strange.

Foster: At AFOTEC we try and do it by getting testability put in early in the various groups that we work with. We try to encourage them to build testability into the system. We did review the SDS document and we expect to see testability put into it as part of the development standards. It is something that you can't just stick on the end, it has got to be built in from the start.

DeMillo: Don, do you have anything to add?

Greenlee: I certainly go along with all the comments relating to early-on hooks, flags and other support functions for testability but I am going to pick up on something that Phil Dickinson said which is the need to discriminate between different types of errors. We see numerous examples of systems coming up for a major milestone review in which growth parameters such as realiability and maintainability are less than expected but the developer is unable to distinguish whether those are hardware or software errors. A specific example is a communication system in which the MTBF is less than the minimum acceptable value and some of the errors which were presented during the course of the review were software errors. These were promised to be eliminated in subsequent versions of the software when "it matured". Later on incidentally, it turned out that an appreciable number of those were actually hardware errors. There is a tendency to ascribe problems to the software. So in addition to moving up earlier in the process and not leaving everything to testing, it is important to think about the problem of trying to discriminate between hardware and software and the user errors.

DeMillo: Yes, we have a question from the audience.

Bartosik: TECOM has commented on the SDS document and I must admit that I was not one of the guys who contributed to that effort. Phil Dickinson and I have been involved in the Army C^3I Studies over the past year. One issue has been the software hooks and hardware monitor points on systems. These should be designed in from the very beginning from the system inception and, as Paul said, made a requirement. We at TECOM certainly support that. We have been fighting for that ourselves for a long time but we haven't had any more success with it than the operational testers have had so far.

Gehl: I think I would like to bring up a problem with that; it's the same one that has been mentioned: the size of the production version. Usually, what I think of as OT&E is the stuff done by OPTEVFOR. It is not the OT&E that's done as part of the development but it is the evaluation that is done after the system is ready to go and we are ready to go for a production decision and the system is essentially what is going to be put in the fleet. In the Navy, we have a lot of programs that were constrained by size. I don't know what my bottom line is, but I think it would be hard to come up with a standard way of putting hooks in the software except for the kind of hook that says, "do it earlier, maybe at OT1 or OT2, rather than OT3". Possibly that could get into the standards, but even SDS and MIL-STD 1679 don't talk about OT1 and OT2 or the different types of operational testing. But I think we do need to distinguish between the phases when that kind of thing should occur.

DeMillo: I think that the kind of test environments that I was talking about and Phil was talking about is when, during an operational test, you have an integrated system, you are driving it around on the desert, and instrumenting all kinds of things that happen in the system, except the software. That is fine when the software works the way it should, but when the system hiccups you really don't have any visibility into the system software. The problem that I see is that you can't really get the visibility that you want unless you have the software architected to allow it.

Gehl: One way of doing that, which in the hardware corresponds to built-in test, is to require or specify built-in software tests so that you can isolate things down to that level.

capacity standpoint, the physical size of the memory may not be such a limiting factor. The thing that will have to be carefully engineered will be the timing requirements on the execution of the software. I can see where having the routines in the software that will pump out the data for testing might slow the execution down to an unacceptable level. The time line will have to be looked after very carefully to see that the speed that is needed to get bits of information from one point of the architecture to another isn't impeded to the point where the mission is jeopardized.

DeMillo: Isn't it a mistake to argue about systems and technologies that are 20 years old in light of what we can do today? It is very unlikely that you will find 16 bit limitations in technologies where you are packing highly dense circuits.

Dickinson: We have to keep in mind that you may be grandfathering existing systems but I agree with the general principal. In a newer system you should do it better.

Audience: I have an alternative for testers that might be useful. Under the old way of doing things, when you charged for errors during a test, you wouldn't charge errors to software unless you could prove the software was at fault. An alternative is that the software developers would be responsible for errors until you can prove that it wasn't the software that was at fault.

DeMillo: That comment leads into one of the issues that Mike McCracken mentioned earlier this morning. Specifically, I would like to ask if the grass is always greener. If you knock around software conferences long enough, we see a lot of concern over the status of software. Your suggestion indicates to me that the software is more suspect than the hardware and I would like to spend just a few minutes getting the feeling of the panel as to whether or not things are really as bad as we have been led to believe.

Greenlee: Yes.

DeMillo: Thank you.

Bartosik: Yes.

Audience: (Laughter)

AD-A163 236 PROCEEDINGS OF THE INTERNATIONAL TEST AND EVALUATION ASSOCIATION 1984 SYM..(U) GEORGIA INST OF TECH ATLANTA SCHOOL OF INFORMATION AND COMPUT.. APR 85 2/2

UNCLASSIFIED GIT-ICS-85/05 F33657-82-G-2083 F/G 9/2 NL

[redacted]

Audience: I would like to disagree with the entire panel. Now that you have all answered it seems that the best answer to this question is it would be nice to have the capacity to judge these errors. I am a member of the test community, and we require in the case of mechanics, for example, that there be access ports to mechanical equipment so it can be observed during tests. In the case of electronics you have test measurement and diagnostic equipment that is required. Now what is it about software that makes it an exception to this? Is software some kind of magic so that we can't require this? It seems to me that the test and evaluation community position should be that these things should be a requirement.

Clements: If you have an absolute weight limitation on your vehicle would you rather throw away the test equipment or the weapon?

Audience: That strikes me as the same kind of objection we encounter in the hardware world when someone says, "it's too hard to put the ports or the hand holds in that place", but it is important to realize that you can require that sort of thing.

Clements: I am glad to hear that. I think it is important to realize that those requirements can be placed on software too. It is also important to realize that we have to have enough resources and capabilities so that we are able to do that. Sometimes the world is not like that.

Bartosik: If I can just add something. The theme of this ITEA conference is the impact of high technology on test and evaluation. It seems to me that with VLSI, VHSIC, and Crays being readily available, the smaller physical size of processes, lower power requirements, lower heat output, and the multi-millions of bits of storage you can put on boards these days, that I don't think the capacity of memory might be as much of a limiting factor as it would have been five years ago. I think that technology is going to permit us to do the kinds of things, at least in software, that we have been talking about. Understand that I am not an electronics engineer, so that I don't understand all the intricacies of what might happen to a system when you attach a probe at one point or another but I do think that from a software standpoint and from a memory

DeMillo: Let me tell you what motivated the question to begin with. I was at a Design Automation Conference a couple of years ago on a Panel that had to do with testing. And it turns out that I was the only software guy on the Panel. And the question was put to me, "How do you software guys do it? How come your testing of reliability is so much better than ours." When I picked myself up off the floor and thought about that and I wondered if you tend to look at those other technologies through slightly rose colored glasses. I think there are examples of systems where the software gets a lot of blame laid on it. The software gets a lot of blame because you are dealing with case analysis, because you don't really know or have the technology to know what is going on inside the system. In those cases, certainly the software ends up looking quite bad. Now if there is no controversy over that, I will go on to something else, but I offer that as a maybe unconventional view of the world.

Audience: I think maybe there is some truth to that and the purpose of my comment was to point out how difficult it is to really check.

Foster: I'd like to throw something out here. Just as a thought. We have seen that recently as hardware becomes more redundant where the software incorporates some fault tolerance that these problems can be pushed off onto the software. The failure really was in the hardware and the software is supposed to make a switch and it doesn't, then it is now a software problem.

O'Neil: One of the things that bothers me is the level of complexity involved in some of this. I am thinking particularly in the case of radar. Instead of using a computer to keep track of a file of several targets, we are using some mechanical device to do that. Don't you have a lot of problems with realiability and maintainability doing that sort of thing? You end up putting your data in a slightly different format and the problem becomes more difficult and that is what is responsible for a lot of maintainability problems.

DeMillo: I think there is a fundamental difference at work here. And it has to do with the manufacturing process. The manufacturing process for software is purely a design process. The platforms are manufactured. They are built on assembly lines and are subject to reliability measures that you get from engineering physical materials. We all talk about the cost of software but if you take any reasonably large program, put in hundreds and sometimes even thousands of platforms, the software represents a tiny portion of the capitalization on that system. I don't know where that leads but..,

French: We hear a lot of talk around the Government that when you go to a contractor to buy something, you get exactly what you pay for or you get exactly what you ask for. So, training government contract people about how to ask software contractors for good products is an important concern. In particular, how do you ask for maintainable software, and the state-of-the-art in software design. Presumably if we can solve that problem, we can avoid some of the difficulties.

DeMillo: I think we heard about that already. One of the purposes of SDS for instance, is to aid that problem.

Gehl: Well, SDS merely carries on what the RADC group is doing in their tech reports and what MIL-STD-1679 is doing in terms of their attempts to get good design principals in the software in the first place. The training of program managers is being addressed by several isolated efforts right now. There is no consolidated effort that I know of. We were asked by the tri-service test commanders about a year ago to come up with a training program to implement SDS. We are going to be faced with a lot of problems; for instance, how do we reach everyone that needs it. The contractors need it, the government program managers need it, the DCAS people need it to monitor it. The magnitude of the task is bigger than all resources can accomplish. We can develop a training course but we are going to need lots of help from the service training communities to get that implemented.

French: Do you think that would be one of the most effective ways to improve the state of the situation?

Gehl: Yes.

Gehl: I think that as a first rule the OT&E guy shouldn't have much to do because the DT&E guy should have done it for you for software. By the time the software comes through the design process which is also the construction process, it has been produced. By the time you give it to the operational tester, it should have completed all of its tests except for possibly system integration and stuff like that. The first rule for OT&E is to make sure that the DT&E guys did their job.

Clements: Precise, unambiguous and complete specifications of the requirements. I am next to last so I get to look very safe and wise. I have had a lot of time to think about it but that is my bottom line.

Dickinson: Precise, unambiguous and complete, but does it fit the needs?

Clements: So you would have the operational tester review the requirements to make sure that user needs are properly represented.

Dickinson: Does that include the verbage on the screen that talks to the sailor?

Clements: We have some pretty clear ideas about how you write a specification like that. When you say verbage I cringe. We try to use as little English as possible.

Audience: (Laughter)

Clements: We have some forms. Our software requirements are specified using tables and the tables have such properties that you can do completeness checking and consistency and all those things. We found that a document like that which is divided up into parts and each part is divided into parts lends itself very well to test case generation. Once the system is thrown over the fence, you can generate test cases for it from the tables to tell you what the system ought to do and you can find out whether or not it really does.

Greenlee: You guys said exactly what I was going to say, But the fifth principle is to decide how much testing is enough. Just like in the case of hardware testing, it relates to a very good point that you raised before and

Greenlee: I don't have an answer to that question but I just thought of an answer to another question. Actually, it relates because specifying what you want software to do is related to testing. Going back to the genesis of STEP, it was not uncommon to find when the software testing was elaborated for a decision point review, that the PM had done the best he could, literally. If, for some reason, that were held in question then the PM had the right to turn around to us and say, "well, what did you want me to do"? This is in terms of evaluating the software testing, validating or whatever you want to do. It seems to be a cultural phenomena: people are comfortable with sonar gear, jet engines, or many other things which are more physically tangible. Perhaps, carreer wise we have grown up with more familiarities with the hardware. Maybe this is something that will change with time but it is just basically a mysterious area of endeavor. That is why we are trying to improve the state-of-the-art in T&E guidelines. I think that will ultimately play back in terms of specifying software and specifying the properties of what you want it to do.

Audience: I would like to ask those panel members that failed to respond what is the very first thing that you should keep in mind as far as testing software goes?

Bartosik: I would say that one of the best things that came out of the Army study on C3I was the suggestion that we get the operational tester into the contractor's plant just as soon as they start coming up with something that looks like the system so that that tester can sit down and interface with that prototype - I am going to use the word loosely. The tester should see the system prototype at a very early stage, when the thing is first getting off the ground. And if you want to transfer that again into good requirements definition, feel free to do that but that is the first thing that came to my mind.

Foster: From my standpoint, I think it is important to plan for test. You have to make sure that you have the resources and that everything is there together. That you have the hardware to run the testing on, that you have the right kind of people and that they are ready to go into test. Don't frustrate yourself by sitting there with partial pieces of hardware and software while you are getting no data whatsoever. So be ready for the test and plan for the test.

Bartosik: Do you mean the developers or the operational tester?

Grover: I mean the contractors.

Bartosik: Into the test plans?

Grover: Yes, I mean to offer insight into those areas that should be tested and help DoD personnel in making decisions about what should go into a test.

Gehl: I personally don't see anything wrong with that. I would guess that if I were an operational tester, I would like having that kind of input to help me design my test. That's a cheaper way than having me sit in the plant everyday watching the code being built. I would like that kind of input. I don't know that it should be required. At least definitely not on every contract.

Clements: It would be nice to run some experiments. I personally have no problem with that approach. I would like to see an experiment where you have a test plan with that input and try to discover the kinds of errors that you uncover with the developer's input that you wouldn't have found out otherwise.

Foster: We see quite a bit of that in the hand off between DT and OT. What happens is that we get that information indirectly by seeing what the DT development tester discovers. We take that data and if it is usable in the operational environment, that is not just a laboratory environment, you could both have some direct application in the operational environment when that good operational test data. In other words, you don't run that test but we make inferences from it.

Bartosik: I think if you have a real team concept such as the one being tossed around in the Army these days, it is possible to closely coordinate everyone from the outset. Now I understand that there are a lot of vague terms here like "close coordination". Just what does that mean? For example, how many people are willing to go to all these meetings and reviews? How much documentation is everybody going to read? How much are they going to understand? But those problems aside, the team concept leads to good test plans. Those players that are now involved to that extent are going to know a lot more about that system than they currently do. From that standpoint, I think it will happen as a matter of course. Now to what degree you will want it to happen even more, is another question.

that is how much are you willing to invest in testing. Just like we really don't have an answer to how much testing is really enough for the hardware, I don't think we have even that degree of certainty for the software. That is one reason that we are interested in trying to develop techniques for risk assessment models which will tell you how much testing we have to do. I think the passion for built-in testing is probably over emphasized in most of the systems. I think that we are learning now that in a lot of cases it is not worth the effort and one fools oneself because of a false sense of security. How much do you want to pay for that extra increment of testing? I would like to have Georgia Tech look at that, and I know AFOTEC is looking at risk assessment. I think it is fine for us to be whipped into a lather in righteous indignation over the problems of software testing, but we also have to recognize there will never be enough resources on any given program to do as much testing of the software as we want. So I think that principal five ought to be started early on and decide how much testing is commensurate with the overall objective of the program.

DeMillo: Since I am dead last, I get to agree with everything. I think the only thing I would add to what has been said is the term "goals and thresholds". As much as possible, what the software should do should be quantified. That is a requirements issue. I really strongly disagree with Dan McDonald's idea yesterday of munging around with the software to see what happens. Software has the peculiar property that for every test that you design, there is a little twittle to the software that will make it pass that test. It's like conducting an experiment in a chemistry lab. You should say what you are going to do before you do it so that you can tell whether or not you have done it. Are there any questions from the audience?

Grover: Do you feel designers and implementers should have some role designing tests and test plans? The hardware gives some idea of what engineering principles are involved in putting the system together. With the software it is much more difficult to do unless you have more detailed knowledge about the architecture and design of the software. In general as you delve deeper into the systems the black box approach seems to be less acceptable.

Clements: I hear you attributing that to the size of the team. The small size of the team from beginning to end.

Passafiume: That sounds like the Marine Corps.

McCracken: Yes, you are probably doing the same thing on the A7 project. I think that is probably one of the reasons your success is so great there. That is a situation where you don't have all those communications problems to worry out.

Audience: Isn't there a more general principal at work here like the less government help, the better.

McCracken: No, absolutely not. The point I was trying to make is that these six guys were technically motivated. They questioned things, they wanted to understand the system, they wanted to make sure it was done right. They weren't motivated by filling out forms. They were technically interested. I think that, in fact, they were more involved in the program than their U.S. counterparts would have been. When you deal with our own Government contracting, thousands descend and thousands leave and you have a certain amount of time to submit written comments and then thousands descend and thousands leave. By contrast, we established a personal rapport with the engineers.

Bartosik: What was their penalty if they failed?

McCracken: I don't think they looked at it from that perspective. They wanted the system to succeed.

Bartosik: That wasn't my question. My question was, what was their penalty if they didn't succeed?

McCracken: I don't know. They were charged with delivering a radar system. If they hadn't succeeded, I am sure they would have suffered. But the point is they didn't and they knew what they were going to do.

Gehl: I have a couple comments on that. I don't think it is feasible for us to do. For one thing, the way we are structured in the Navy with seven program managers, each of which is staffed with 30 people who have responsibility for over 125 projects, we don't have the resources to be able to do that on every program. Secondly, the DoD acquisition process requires you to

Dickinson: As Rich DeMillo knows, that happened to some extent this past summer on Patriot. We had Raytheon and a number of contractors supporting our follow-on evaluation and supporting the program office. Even Georgia Tech.

DeMillo: We are running up against quitting time here.

Martin: Don Greenlee hasn't responded to that last question yet.

Greenlee: OT&E is supposed to be independent but that doesn't mean innocent or ignorant. What you say is, in fact, common place. The operational tester is obligated to get as much mileage as possible out of all prior testing information whether it comes from the developer or whether it comes from the contractor or other source. He doesn't have to believe it, he doesn't have to build a test around it but I think he is obligated to assess and consider and evaluate it for all for what it might bring to his own team. In fact, it is absolute T&E policies to promote concurrency to the degree possible without sacrificing the integrity of the operational tester.

McCracken: I was wondering if the panel would like to comment on a recent experience I had during my previous life as a contractor. The question is whether or not there is anything to be learned from buying systems with a very small dedicated team. I had that experience with a radar contract for a foreign country. The customer placed in our plant a small group of engineers who were responsible for the total program, i.e., systems engineering, design, software, testing, logistics and even operational testing when the system was fielded. The point is that with a small dedicated team, the communications problem was non-existent.

Bartosik: I would like to know how to manage that.

McCracken: Pardon me?

Bartosik: I'd like to know how they do business over there. You paint quite a contrast for us.

McCracken: Yes, I agree. When I experienced it there were six people who were very well trained, and their responsibility was to get this radar system into production as soon as possible. This group was concerned about the system requirements, the software requirements, logistics, training, development testing, operational testing. I think it was important that they had personal responsibility for the system from beginning to end and they got an outstanding product.

justify your money and your people every year over a seven or eight year period. That means that you can't behave in the way that you describe which, as I see it, is essentially like the commercial contractor.

McCracken: John Passafiume made a comment "It wasn't the Marine Corps, was it?" I think that's significant. It turns out the Marine Corps buys systems in a very similar fashion.

DeMillo: I would like to thank the Panel for their energy and help this afternoon. I would like to thank the audience for their endurance and enthusiasm. Thank you.

END

FILMED

2-86

DTIC