2

LEVEL II

ARMY MATERIEL DEVELOPMENT AND READINESS COMMAND
NAVAL MATERIAL COMMAND
AIR FORCE LOGISTICS COMMAND
AIR FORCE SYSTEMS COMMAND

# PROCEEDINGS OF THE JOINT LOGISTICS COMMANDERS JOINT POLICY COORDINATING GROUP ON COMPUTER RESOURCE MANAGEMENT

## COMPUTER SOFTWARE MANAGEMENT SUBGROUP
## SECOND SOFTWARE WORKSHOP

1 NOVEMBER 1981

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

THIS REPORT IS NOT AN APPROVED JLC POSITION

DEPARTMENT OF THE ARMY, THE NAVY, AND THE AIR FORCE

UNCLASSIFIED/UNLIMITED

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| | | |
|---|---|---|
| **REPORT DOCUMENTATION PAGE** | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. AD-A109 441 | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) Proceedings of the Joint Logistics Commanders Joint Policy Coordinating Group on Computer Resource Management; Computer Software Management Subgroup-Second Software Workshop 22-25 Jun 81 | | 5. TYPE OF REPORT & PERIOD COVERED Final |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) | | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS HQ AFLC/LOEC Wright-Patterson AFB, OH 45433 | | 12. REPORT DATE 1 November 1981 |
| | | 13. NUMBER OF PAGES 503 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | | 15. SECURITY CLASS. (of this report) UNCLASSIFIED |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |
| 16. DISTRIBUTION STATEMENT (of this Report) Approved for Public Release, Distribution Unlimited | | |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) | | |
| 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) These proceedings contain the Panel Chairpersons reports and Preliminary panel recommendations that were produced during the Joint Logistics Commanders sponsored Embedded Computer Software Workshop, 22-25 June 1981, held at the Naval Postgraduate School, Monterey, CA. | | |

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

JOINT LOGISTICS COMMANDERS SOFTWARE WORKSHOP

TAB A
Software Documentation

# REPORT OF THE PANEL ON SOFTWARE DOCUMENTATION

TRW

81.6420.37-057

9 September 1981

LTC Casper H. Klucas
HQ, AFLC/LOEC
Wright-Patterson AFB, Ohio 45431

Dear LTC ~~Klucas~~: [illegible]

Enclosed is the Final Report of the Software Documentation Panel. It is the result of our work at Monterey II and a two-day follow-up meeting at Ft. Belvoir in August.

As you can see from the size of the report and the substance of our recommendations, the panel members contributed significant amounts of time and plain hardwork to this effort. We all look forward to the day when the fruits of our labors will become official.

Very truly yours,

[illegible]

Antonia D. Schuman
Panel Chairwoman

ADS:sll
Enclosure

cc: CSM Subgroup
Panel Members
Other Panel Chairmen

TABLE OF CONTENTS

## REPORT OF THE PANEL ON SOFTWARE DOCUMENTATION

APPENDICES:

FIGURE:

JOINT LOGISTICS COMMANDERS

JOINT POLICY COORDINATING GROUP ON COMPUTER RESOURCE MANAGEMENT

SOFTWARE WORKSHOP

(MONTEREY II)

22-25 JUNE 1981

# Report of the Panel on SOFTWARE DOCUMENTATION

10 September 1981

|  |  |
|---|---|
| Chairwoman: | Antonia D. Schuman<br>TRW Inc. |
| Co-Chairman: | Paul Shebalin<br>SDC |

## PREFACE

This panel was a continuation of the documentation panel of the first software workshop held in April 1979. The original panel examined the existing software documentation arena, and developed overview descriptions for what they considered to be a comprehensive set of documents that could be used for DoD contracts. Their recommendations for further action included:

1. Preparation of Data Item Descriptions (DIDs) for these documents.

2. Preparation of modifications to existing military standards (MIL-STDs), or the creation of a new one, from which the DIDs can be involved.

3. Preparation of guidelines to help program managers select an appropriate subset of documents for specific contracts.

Since the 1979 workshop, the JLC completed the first task via contract, and awarded a contract to accomplish the second task.

## SOFTWARE DOCUMENTATION

### 1. OBJECTIVE

During the JLC Software Workshop (Monterey II) held in Monterey, CA on 22-25 June 1981, Panel A, Software Documentation, pursued four objectives:

a) Provide recommendations to the JLC for developing project management guidelines for the selection of software documentation.

b) Clarify the relationship between the DOD acquisition life-cycle (milestones, phases) and the JLC-list of software documents.

c) Provide recommendations to the JLC concerning the addition, deletion or modification of documents in the JLC-list of software documents.

d) Provide recommendations to the JLC for implementing the standard set of software documents (JLC-list) within the OSD/JLC/Services.

The "JLC" referenced in the above objectives refers more directly to the Computer Software Management (CSM) Subgroup of the Joint Logistics Commanders Joint Policy Coordinating Group on Computer Resource Management (JLC-JPCG-CRM).

### 2. SCOPE

The objectives of the Software Documentation Panel initially provided by the CSM Subgroup were:

a) Develop guidelines for project managers to help select the appropriate subset.

b) Prepare draft implementation plan for the DIDs and recommend method of evaluation.

The pursued objectives differed from these initial objectives for two reasons. First, the time (three days) available at Monterey II was insufficient to meet the objectives. Secondly, it became apparent, at the beginning of the Panel session, that several issues needed to be clarified or resolved in order to meet the more long-term objectives of acquisition management guidelines and DID implementation plans.

It was felt that acquisition management guidelines are certainly necessary, but that Panel A's contribution to the CSM Subgroup should be to clarify the issues involved with selecting documents. The major supporting panel tasks decided upon were developing a consistent acquisition cycle model and, concurrently, reviewing the DID's produced as a result of the CSM Subgroup's documentation contract. The information resulting from panel deliberations on these tasks was felt to be an integral part of any guidelines. It was also felt that contractual support is necessary for guideline development and to this end a discussion of the required tasking was undertaken.

Preparing a draft implementation plan, per se, was not tackled. It was felt that the approach most beneficial to the CSM Subgroup would be to document some of the considerations involved in implementation of the DID's.

Two additional topics were considered for discussion but were dropped because they were not felt to be within the scope of documentation/DID specification, standardization and tri-service implementation. These topics were the automation of document generation and the proper specification of software requirements. These are important, related technological issues but may not have a place in JLC standardization.

Twelve members of the panel met at Ft. Belvior, VA in August to review the draft Final Report and complete the DID review. This report incorporates the results of the follow-up meeting.

## 3. APPROACH

Discussions at the initial panel meeting revealed several deep-seated concerns about the documents, the DIDs and the life cycle. These concerns led to the formation of four subpanels which addressed specific issues. The subpanels met independently throughout the remainder of the workshop. Periodic full panel sessions were held to coordinate the activity and develop group concensus.

The four subpanels were:

1. Guidelines for Acquisition Managers.

2. Life Cycle.

3. Document set and DIDs.

4. Implementation Plan.

Membership of the subpanels is given in Appendix A.

Two of the subpanels had extensive interaction with representatives of the CSM. The Life Cycle was a particularly controversial topic and, during one session, discussion between subpanel and CSM members was needed to clarify the intent of the new life cycle.

The implementation subpanel attended a CRM-CSM meeting on policy because the enforcement of the new DIDs requires promulgation of the new MIL-STD. This helped formulate the conclusions reached by this subpanel.

## 4. DISCUSSION

### 4.1 SUBPANEL 1: GUIDELINES FOR ACQUISITION MANAGERS

This subpanel concluded at the onset that there was not sufficient time during the workshop to develop a set of guidelines. Instead, they decided to recommend that the guidelines themselves be developed via contract, and directed their efforts at drafting a Statement of Work (SOW) for such an effort. (See Appendix C).

As part of their deliberations, subpanel 1 considered two major questions:

1. What flexibility should an acquisition manager have in tailoring individual DIDs?

2. What determines if an individual DID is needed on a particular project?

Regarding the question of tailoring, the subpanel agreed (after considerable discussion) that DID tailoring should be discouraged. The basis for this decision, which differed from that of the Monterey I panel, was that document <u>standardization</u> is the most important concept behind the use of a single DID set, and that prohibition of tailoring does not prohibit offerors or acquisition managers from stating particular sections are not applicable if such is the case.

Regarding the question of DID selection, the subpanel had three concerns. First, it was agreed that there was considerable overlap in information content of some of the documents. Second, it was felt that not all of the documents were necessary on every project. Third, it was feared that acquisition managers, when in doubt as to what documents to require, would tend to procure all of them.

In view of these concerns, the subpanel first decided that some sort of selection matrix which would list appropriate DIDs versus project characteristics was needed. However, in a later split decision, the panel concluded that a flow chart or selection algorithm would be better than a decision matrix. (Minority opinions are contained in Section 4.1.2). Despite the differences in opinion regarding the form of the selection technique, the subpanel unanimously agreed that the technique should be incorporated into a guidebook.

### 4.1.1 Guidebook

The subpanel believed such a guidebook should contain the following:

1. A selection matrix, flow diagram, or other algorithmic form.
2. General discussion of why documentation is necessary, with case histories and the effect of documentation on their success or failure.
3. Statements indicating how military standards and regulations relate to documentation and which are in effect.
4. Descriptions of the role of the Statement of Work (SOW) and Contract Data Requirements List (CDRL) in requiring DIDs and providing flexibility when appropriate.
5. Expanded overviews of each document.
6. Project characteristics to be considered in selection DIDs to be procured.

Some of the project characteristics which the subpanel felt should be included in the guidebook were

1. Number of project offices, services, and contractors involved
2. Complexity factors, such as multiple processors and sizing/timing constraints
3. Criticality of software to system operation

4. Interoperability requirements
5. Stability of requirements over system's life
6. Projected length of system life cycle
7. Budget constraints
8. Use of existing software and documentation
9. Procurement type (sole source vs. multiple)
10. Software support concepts (contractor vs. organic)
11. Development phase
12. Significance of software in overall system, i.e., Hardware intensive, Software intensive, Operator intensive

### 4.1.2 Minority Reports

Two Minority Reports were submitted.

#### 4.1.2.1 Minority Report 1

1. The subpanel is recommending that a task be let to a contractor to put together a Guidebook to aid Program Managers in the selection of DIDs/Documents in support of Software/Firmware development.

2. As part of the guidebook an Algorithm may be required to be developed that, based on characteristics of the project, will lead a Program Manager to his selection of DIDs.

   The subpanel feels that this type of Algorithm is needed because of the overlapping coverage of some DIDs.

3. The minority analysis of the problem leads to the conclusion that:

   a) The number of DIDs has been reduced to a workable set.

   b) A minor expansion of the DID Overview Descriptions would suffice in the place of an Algorithm and be simpler to use than a flow type algorithm. If all additional information were provided in the DID overviews such as:

      1. Under what conditions is the DID needed and why?

      2. Under what conditions is the DID not needed?

      3. Is there a paragraph/paragraphs from another DID that relate to this DID that could satisfy this need?

c) A contract may not have to be let to accomplish this - it may be within the scope of the JLC staff to accomplish in-house.

d) It is recommended that a complete set of software/firmware DIDs be attached to the Guidebook along with the "expanded overviews" to give the Program Manager all he needs to make the document selection.

e) It is felt the algorithm is not impossible to construct, but may be infeasible due to the estimated 6 months to develop it and then validate it and due to the difficulty in quantifying project characteristics in a way that can be applied to an algorithm and due to the number of variables involved.

### 4.1.2.2 Minority Report 2

A second minority report also disagreed with the development of an "algorithm" for DID selection. The implication of an "algorithm" is that DID selection can be reduced to a series of black and white decisions, the results of which totally determine what DIDs are "appropriate." However, as alluded in 1 to 3 above, the "number of variables" involved may be too great to render a neat, deterministic selection process feasible. The selection process thus becomes a complex one of synthesis, judgement, and in part, intuition.

Thus, the inclusion of an "algorithm" poses a real risk of leading the uninitiated acquisition manager to conclude the DID selection process is less complex than it really is. A "good" acquisition manager who cannot decide for himself/herself what DIDs are needed should delegate the authority to someone on the staff who has the background to make the decision. If the acquisition manager does not have access to someone who does have the background, he/she should hire someone who does, or get out of the software acquisition business. If the latter is not possible, then require all the DIDs. (From the perspective of risk and life cycle implications, the public interest is perhaps better served by ordering and paying for a few DIDs which are not really needed in the final analysis than by not ordering DIDs which eventually are needed.) In any event, let us not try to reduce what is often truly a complex and somewhat judgemental decision to an algorithm.

## 4.2 SUBPANEL 2: LIFE CYCLE

The DID set developed for the JLC is based on a new software life cycle which includes two new reviews: the Software Specification Review (SSR) and the Test Readiness Review (TRR). Initial deliberations by this subpanel centered on the necessity of these new reviews. After much heated discussion, the subpanel decided, in a split decision, that the question of whether or not these new reviews were necessary was not their charter. They then proceeded to examine the role of the documents in the life cycle, with the ground rule that the life cycle (including SSR and TRR) is "given."

The subpanel examined all documents in the JLC package and projected them against the life cycle chart, showing when each should be available in preliminary form, baselined, or modified. (This projection was accomplished under the handicap of not having a formal description of the life cycle available to the panel). The life cycle is shown in figure 1.

Additional conclusions drawn by the subpanel include:

1. The life cycle chart should reflect the parallel paths for hardware and system development. Software should always be considered a part of the overall system. The phases of Hardware and Software must permit consideration of decision processes.

2. The life cycle should be tailored to specific projects and phases of development. The life cycle figure indicates the software acquisition process which is key to a total system acquisition. This process may occur within any phase of the system development life cycle. However, the reviews, documents and phases may be tailored as required by the complexity of the development.

3. The life cycle model can be applied to any development process, whether "start from scratch", modification to an existing system or the maintenance phase.

4. A method for proving a life cycle phase has been completed needs to be devised.

5. A good definition of "design review" needs to be developed. It was felt that the reviews indicated on the life cycle chart represent culminations of the review processes which track on-going activities.

6. The new policy will reflect the standardization of existing practices. A means for providing evolution to incorporate new methodologies must be developed.

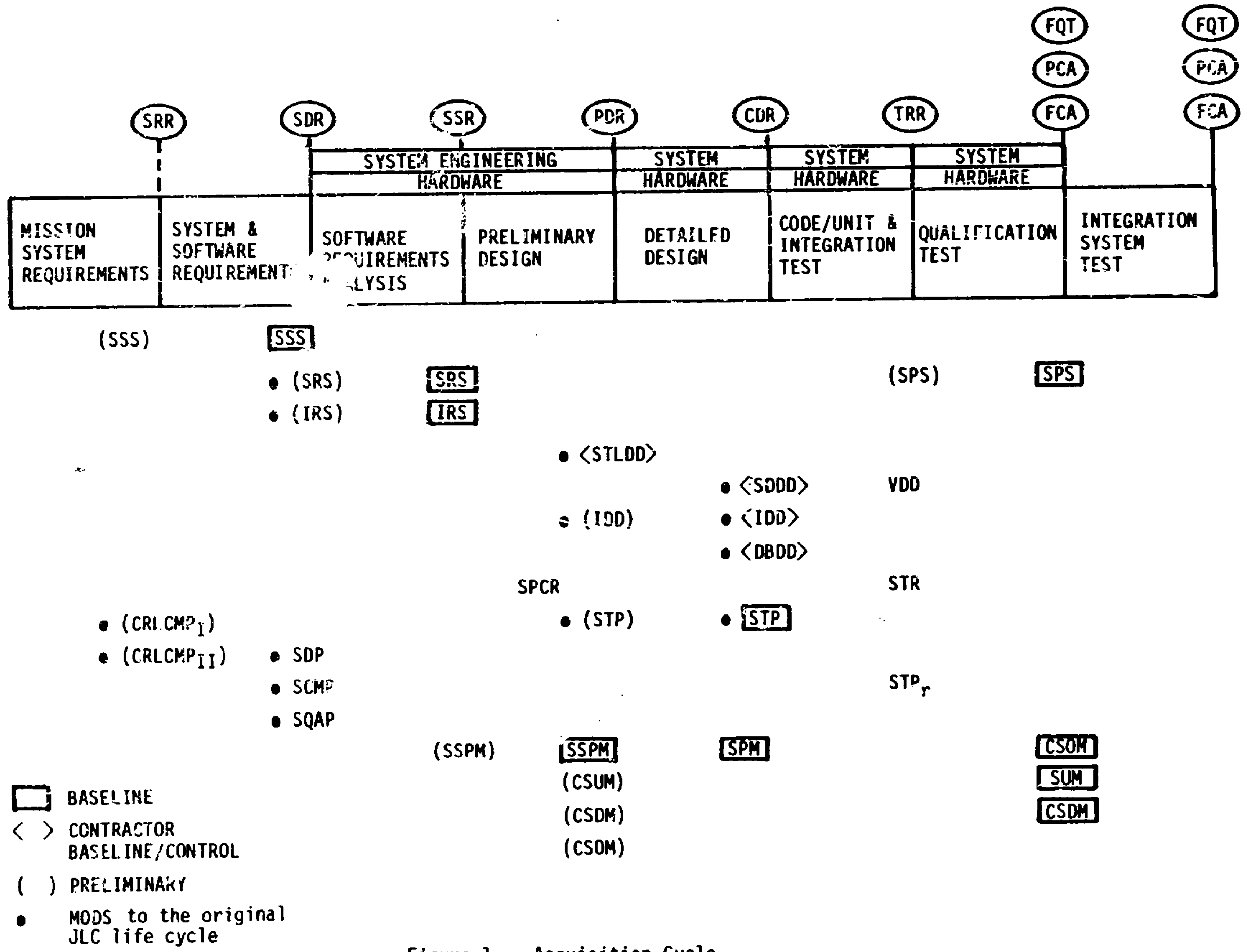


Figure 1. Acquisition Cycle

Page 1 of 2

Figure 1. Acquisition Cycle

ABBREVIATIONS

| | |
|---|---|
| CDR | Critical Design Review |
| CRLCMP (I) | Computer Resource Life Cycle Management Plan (Acquisition) |
| (II) | Computer Resource Life Cycle Management Plan (Support) |
| CSDM | Computer System Diagnostic Manual |
| CSOM | Computer System Operator's Manual |
| CSUM | Computer System User's Manual |
| DBDD | Data Base Design Document |
| FCA | Functional Configuration Audit |
| FQT | Formal Qualification Test |
| IDD | Interface Design Document |
| IRS | Interface Requirements Spec |
| PCA | Physical Configuration Audit |
| PDR | Preliminary Design Review |
| SCMP | Software Configuration Management Plan |
| SDDD | Software Detail Design Document |
| SDP | Software Development Plan |
| SDR | System Design Review |
| SPCR | Software Problem/Change Report |
| SPM | Software Programmer's Manual |
| SPS | Software Product Specification |
| SQAP | Software Quality Assurance Plan |
| SRR | System Requirements Review |
| SRS | Software Requirements Specification |
| SSPM | Software Standards and Procedures Manual |
| SSR | Software Specification Review |
| SSS | System/Segment Specification |
| STLDD | Software Top Level Design Document |
| STP | Software Test Plan |
| STP | Software Test Procedure |
| STR | Software Test Report |
| SUM | Software User's Manual |
| TRR | Test Readiness Review |
| VDD | Version Description Document |

## 4.3 SUBPANEL 3: DOCUMENT SET AND DIDs

This subpanel was tasked with reviewing the results of JLC DID development contract. The scope of this review included not only the JLC Software Documentation Report, but also existing DIDs and the Monterey I Final Report. This was supplemented by panel members' knowledge of existing software documents. Panel members received the DID package prior to assembling in Monterey, and each participant was asked to review in depth one or more specific DIDs and present his/her findings during the workshop.

The first task of the workshop was to review the JLC software documentation list for sufficiency. Open discussions were held on the necessity of adding, combining or deleting a document. The following general criteria were used to guide the discussion:

- Will this document alleviate or add to a problem?
- Is this topic adequately covered in another document?

After discussion on additions or deletions a vote was taken on the recommendation on a document if it was necessary. An open discussion on the DIDs was then held but was limited to general rather than specific comments for each DID due to time limitations. Specific comments were provided later to designated panel members who collated the comments. The follow-up meeting refined the comments which are included as Appendix F.

### 4.3.1 Document Set

The panel felt that three documents should be deleted from the list, some changes were necessary to those remaining, and three should be added. Descriptions of these are given in Appendix D. The documents to be deleted are:

| | |
|---|---|
| R-DID-102 | Computer Resource Life Cycle Management Plan, Vol. I, Acquisition |
| R-DID-103 | Computer Resource Life Cycle Management Plan, Vol. II, Support |
| R-DID-107 | Software Problem/Change Report |

The first two do not belong in the contractor set of documents or cited in the new MIL-STD because they are government-generated documents. If they are cited anywhere they might be appended to DODI 5000.29 or left to the services procurement practices.

The Software Problem/Change Report does not require a standard form and, in any case, the one suggested does not suffice.

A minority report on deleting certain additional documents is given in section 4.3.4.

4.3.1.1 <u>Firmware Support Data Document</u>

The panel was in almost total agreement that the design and development of software and firmware should be documented identically. The consensus was that size and complexity are the important factor in selecting documentation for firmware just as it is for software. It was recognized that there is a basic difference in the storage and modification of firmware and that the existing documents do not clearly address these differences from a firmware perspective. The Subpanel reviewed the Firmware Support Data DID (UDI-E-3937-ASD) and decided that a Firmware Support Data document should be added to the JLC list with this DID being used as a guide.

A related discussion was held on the need for a firmware document that addressed small developments such as the Non-Complex Computer Programs Specification found in Appendix XVI of MIL-STD-483. After discussion with the Guidebook subpanel this panel agreed that a subset of the JLC list would be sufficient to address the firmware requirements of a small project without creating a new document.

4.3.1.2 <u>Operational Concept Document</u>

One of the documents proposed during Monterey I which was subsequently dropped from the JLC list was the Operational Concept Document. This document was presented to the panel as being a potentially valid document in that it fed back to the government in plain English what the contractor understood about the proposed system, including man-machine interfaces. The proponents were willing to admit that, in theory, section 3.1.7 of System/Segment Specification

"Operational and Organizational Concept" addressed this area, but because it was part of a large specification it had never been adequately addressed in reality. The point was made that in a large system there was a potential need for a separate document covering this area. A vote was taken, and approximately 75% of panel thought that the addition of this document to JLC list would enhance the software documentation of some large projects; therefore it would be recommended that it be added to the JLC list.

4.3.1.3 Requirements Traceability Matrix

This document was proposed by the Air Force members. Its purpose is to tie together in one place the allocation of requirements to lower level documents. It was conceded that each of the B-level documents has a requirements traceability matrix back to the system specification and similarly C back to B. In large projects however, where there are several CIs and CPCIs, it is a tedious and time consuming process to dig these out of each document and collate all these individual matricies. In the discussion which followed, it was conceded that this document would be helpful to government personnel on large projects; thus, the panel decided to recommend that this document be added to the JLC list.

4.3.2 Data Item Descriptions

During the discussion on the adequacy of the DIDs, points were raised that the DIDs included policy requirements that should properly be contained in a MIL-STD. Members of the panel who had attended Monterey I discussed the difference between the actual implementation of their recommendations and how they had envisioned this implementation. It was the intent of Monterey I that the DIDs would be contractual documents that pointed to a MIL-STD for embedded computer systems that contained the required format and content for software documentation. As things turned out the MIL-STD development lagged considerably behind the development of the DIDs. As a result, the contractor who developed the DIDs put requirements into them since there wasn't any MIL-STD to reference. It was the general consensus of the panel that revision of these DIDs should be integrated with the development of the new MIL-STD for Embedded Computer Systems.

Specific comments on each of the DIDs are included in Appendix F. General comments are provided below:

- In revising DIDs the improvements made to MIL-STD 483 by NOTICE 2 should be considered.
- All relevant existing DIDs should be reviewed to determine if they can be superseded by the proposed DIDs.
- The word "paragraph" should be replaced by "section," and contractual language should be used (i.e., "shall" rather than "should").
- Include in review the DIDs not originally considered by the DID contractor (Appendix E).
- DIDs should not be service related or reference a service-unique regulation or guidebook.
- Technical Performance Measurement Report (DI-S-3619/S-153) should be replaced by Software Development Progress Report (UDI-A-21435).
- The DIDs should be reviewed to ensure that they are not too rigid, precluding flexibility in documentation.
- Review the EIA DIDs and see if they can be folded into the JLC's DIDs.
- Man-machine interfaces are not adequately covered.
- Interface Specification should be in the specification tree.
- SOFTWARE DIDs should be retitled SOFTWARE/FIRMWARE.
- In Block 9 of every DID add reference to MIL-STD-XXX.

## 4.3.3 Future

The new DID package, though solving many of our present problems, does not attempt to predict methodologies to solve our future problems such as the ramifications of ADA on software development and documentation. Thus, the subpanel felt that the JLC should periodically review and update MIL-STD-XXX (DID package) to take advantage of new trends in software development/documentation.

## 4.3.4 Minority Report

a. R-DID-105 "Software Configuration Management Plan

This plan should be incorporated into the overall configuration management plan to insure that configuration management is consistent, that the impact of changes on all parts of the system are covered. This can only occur if the handling of all ECPs are coordinated. The information in this DID should be added to that for the overall CMP. Software CM personnel should help develop a new CMP DID.

b. R-DID-111 Software Top Level Design Document

R-DID-112 Software Detailed Design Document

R-DID-115 Software Product Specification

The Software Top Level Design Document and Software Detailed Design Document should be combined to form the Software product spec, eliminating 2 DIDs. This document could be delivered in snapshot forms, as required, until the entire document is accepted. The Configuration Management and delivery problems with this scheme are what should be addressed.

## 4.4 SUBPANEL 4: IMPLEMENTATION PLAN

The objective of this subpanel was to examine the problem of bringing the revised DIDs developed under the DID study panel to the point of official approval.

"Official approval" was agreed to denote a combined joint service endorsement of the individual DIDs, followed by approval at the DOD level. With this approval accomplished, the DIDs would be entered into the AMSDL program and maintained within the procedures presently in place under DOD 5000.1-L.

The proper role of the JLC in this effort was examined, and it was concluded that the on-going OSD Standardization Program is the program within DOD which acts as the focal point for administering the approval of new documentation that is destined for the AMSDL program.

The coordination, then, must occur between the JLC and the Defense Material Specifications and Standards Office (DMSSO) which presently acts as the DOD agent for the standardization of documentation relating to Embedded Computer Resources. Several copies of the draft OSD Standardization Program Plan were made available.

The question arose as to how the JLC, as presently organized, can effectively interface with the OSD Standardization Program. In order to investigate this question, the subpanel was invited to join a combined meeting of the Joint Policy Coordinating Group and its Computer Software Management Subgroup.

At this meeting, an explanation of the DOD Standardization Program was offered, and several copies of its draft Program Plan was made available to the JPCG and the CSM Subgroup.

The responsibility for resolving the question of "how should the JLC organize itself to interface with the OSD Standardization Program as well as to administer their vertical service approval and their joint service integration" was accepted by the CSM Subgroup chairman. This task was seen to be relevant for two reasons:

a. The OSD Standardization Program Plan needs to be studied, particularly with regard to its data requirements.

b. There is no existing joint service documentation integration agency of the type envisioned for the JLC.

Considering these factors, it was thought advisable for the CSMS to examine possible document implementation plans at greater length before committing to firm milestones. The CSM subgroup chairman accepted the responsibility for resolving how the JLC should organize to interface with the Standardization Program as well as to administer inter and intra service approval.

It was agreed that DIDs and their corresponding source MIL SPECs/ Standards/Handbooks must be treated in an integrated fashion by the JLC in securing DOD approval.

It was also agreed that the ultimate plan for document implementation should validate the effectiveness by choosing on-contract pilot projects. Another desirable feature of the plan would be to consider phasing DID/ Standard sets by groups as they become available. In this way the number of new documentation requirements that are simultaneously mandated can be controlled. Sample documents should be developed as part of a guidebook.

## 5. RECOMMENDATIONS

1. The JLC-JPCGCRM-CSM should publish a document selection guidebook for use by acquisition managers based on the outline in Appendix C. .This guideline should address both software (operational, support, ...) and firmware.

2. Another guidebook should be developed to show how to apply the forthcoming MIL-STD to specific acquisitions. Specific items to be addressed should include the goal, conduct, and tailoring of reviews. The guidebook should clarify what is meant by successful attainment of each life cycle phase, and when documents are accepted. Emphasis in the life cycle should be placed on what is to be accomplished rather than who does it (as in current life cycle models). Three specific issues should be addressed:

   a. Insurance that hardware and software development are coordinated. This coordination should be addressed at all reviews, especially at non-system level reviews.

   b. Insurance that software and other system efforts do not duplicate each other.

   c. Firmware should be treated as software during development.

3. Add three new documents to JLC list: Requirements Traceability Matrix, an Operational Concept Document and a Firmware Support Data Document.

4. Delete three documents: CRLCMP Vol. I, CRLCMP Vol. II, Software Problem/Change Report.

5. Retitle SOFTWARE DIDs as "SOFTWARE/FIRMWARE".

6. Integrate revision of DIDs with the on-going development of a MIL-STD for Embedded Computer Systems. The new DIDs should be published in conjunction with the <u>proposed</u> DOD standards and industry and the services should have at least one year to respond with comments. Until key issues are resolved, existing DIDs should not arbitrarily be cancelled.

7. Address panel's comments (Appendix F) during revision of DIDs. Further, this panel should remain active until a published set of the new DIDs has been field tested and key issues have been resolved.

8. Initiate an effort to investigate the ramifications of ADA and other future innovations on software development and documentation.

9. Designate the CSM subgroup as the JLC agent responsible for integrating JLC standardization efforts with the OSD Standardization Program. Flow should be:

$$\begin{tikzcd}\text{CSM} \arrow[r] & \text{DMSSO} \arrow[r] & \text{OSD} \arrow[r] & \text{Service Secretary}\end{tikzcd}$$

10. Transition to the new MIL-STD and DIDs should be done using a pilot program in conjunction with a phased approach.

11. The life cycle should be reexamined and that current trends, lessons learned and practical aspects of software development be incorporated. This should include generalization of the life cycle, combining of phases and/or deletion of reviews. Someone (OSD?, JLC?,...?) must define what constitutes successful attainment of the objectives of life cycle phases, reviews and document deliveries.

12. A parallel activity should be started to evolve a successor policy and its implementation to reflect the evolution of existing practices and technologies such as utilization of automated documentation techniques.

13. Guidance should be provided for the preparation of documents according to the DIDs, including examples.

## 6. RECOMMENDATIONS FOR MONTEREY III

The Software Documentation Panel should be recalled to address the following issues:

1. Review MIL-STD-XXX

2. Review the Guidebooks

3. Examine evolving technology and methodologies

## APPENDIX A. PARTICIPANTS

Chairwoman: Antonia D. Schuman*
TRW/DSSG
Mail Stop 02-1796
One Space Park
Redondo Beach, CA 90278
(213) 535-3446

Co-Chairman: Mr. Paul Shebalin*
System Development Corp.
7929 West Park Drive
McLean, Virginia 22102
(703) 442-3815

### Subpanel 1: Guidelines

Chairman: Mr. Gene Sievert*
Teledyne-Brown Engineering
Mail Stop 202
Cummings Research Park
Huntsville, Alabama 30580
(205) 532-1500

2LT Julie Armstrong
ESD/TOEE
Hanscom AFB, Mass. 01731
(617) 861-2701
AV 478-2701

Mr. Reed Maynard
OO-ALC/ACDTA
Hill AFB, Utah 84056
(801) 777-7719

Mr. Antony Kunsaitis*
Director Joint Tactical
Communications Office (TRITAC),
Attn: TT-E-ED
197 Hance Avenue
Tinton Falls, NJ 07724
(201) 532-8292
AV 992-8292

Mr. William (Buddy) Gregory*
General Electric
50 Fordham Road
Wilmington, MA 01887
(617) 657-4652

### Subpanel 2: Life Cycle

Co-Chairmen: Mr. Joseph P. Ceran
Computer Sciences Corp.
10 State Hwy 35
Red Bank, NJ 07701

and

Mr. Jacob (Jack) D. Rosenbaum*
Higher Order Software, Inc.
Director, New York Office
131 Jericho Turnpike
Jericho, NY 11753
(516) 997-7825

Mr. Montie Felman*
Soft Tech
5630 Walston Ct.
Dayton, Ohio 45426
(513) 253-1522

Ms. Lorraine M. Duvall
IIT Research Institute
P. O. Box 180
Rome, NY 13440
(315) 336-2359

Mr. Edward Gallagher*
Commander, USA CORADCOM
Attn: DRCPM-PL-CC
Fort Monmouth, NJ 07703
(201) 532-3482
AV 992-3482

Mr. Gerald A. Bourdon
Dynamics Research Corp.
60 Concord Street
Wilmington, Mass. 01887
(617) 658-6100 (ext. 167)

Subpanel 3: Document Set and DIDs

Chairman: Mr. Patrick J. Ward
Director, AMSAA
Attn: DRXSY-CC
Aberdeen Proving Grounds, Maryland 21005
(301) 278-4030
AV 283-4030

Mr. Marvin Lubofsky*
Aerospace Corporation
P. O. Box 92957
Los Angeles, CA 90009
(213) 648-7440
AV 833-1966

Mr. Alfred W. Florence*
Martin Marietta Aerospace
P. O. Box 179
Denver, Colorado 80201
(303) 977-3000

Mr. Dave F. Southard*
Commanding Officer, FLTCOMBATDIRSSACT
Dam Neck
Virginia Beach, Virginia 23461
(804) 425-4903
AV 274-4903

Mr. Lee McCoy*
NOSC, Attn: Code 9133
San Diego, CA 92152
(714) 225-7409
AV 933-7409

Subpanel 4: Implementation Plan

Chairman: Mr. William A. Clark
Computer Science Corporation
Defense Systems Division
304 West Route 38
P. O. Box N
Mooretown, New Jersey 08057
(609) 234-1100 (ext. 2306)

Dr. Larry Lindley
Naval Avionics Center (NAC)
Attn: Code D/821
6000 East 21st Street
Indianapolis, Indiana 46218
(317) 353-3979
AV 724-3979

Mr. Burt D. Newlin, Jr.
DMSSO (OUSDR&E(SS))
Two Skyline Place, Suite 1403
5203 Leesburg Pike
Falls Church, Virginia 22041
(703) 756-2343
AV 289-2343

*Also attended second meeting 18, 19 August 1981

## APPENDIX B BIBLIOGRAPHY

1. Joint Services Software Acquisition Documentation Description Development Project Final Engineering Report, June 1, 1981.
2. OMB Short Study on the Management of Contractual Data.
3. DOD Directive 4120.21 "Application of Specifications, Standards and Related Documents in the Acquisition Process."
4. EIA draft DID on Software Development Plan.
5. DOD Embedded Computer Resources Standardization Program Plan, Draft June 1981.
6. Various Firmware DIDs.
7. Report of the Panel on Software Documentation, Monterey I, 2-5 April 1979.

## APPENDIX C - DRAFT SOW TO IMPLEMENT A TASK TO DEVELOP A GUIDEBOOK FOR DOCUMENT SELECTION

### 1. BACKGROUND

The Acquisition Manager (AM) has always had a difficult task deciding the amount of documentation required by a specific project. The selection of the required software/firmware documentation is extremely difficult (complex) due to such factors as system complexity, project resources, development stage, documentation overlap, etc. In the past, the AM has typically required too much, too little, or the wrong documentation because he/she had little or no basis for document selection.

The task, therefore, is to develop a guidebook which will select the minimum set of software/firmware documentation based upon project characteristics. This guidebook is applicable for Embedded Computer System (ECS) Software/Firmware (on-line and support software).

The total effort/task shall be broken up into three subtasks:

a. Generate the DID selection algorithm

b. Validate the algorithm generated

c. Write the DID selection guidebook

The subtasks are described in detail in Section 2 below.

### 2. TASK DESCRIPTION

a. Algorithm Development

The contractor shall develop an algorithm which takes as input various characteristics of a program and produces as output the minimum subset of software DIDs necessary for the proper documentation of the program.

The algorithm may take any form that the contractor deems appropriate; however, whatever form is developed shall be concise and easy to use.

The following characteristics shall be considered, as a minimum:

1) Will the program require interfacing between multiple services, multiple program offices, multiple contractors, or multiple agencies (acquiring, using, maintaining)?

2) How complex is the software? Will multiple processors be used? What size is the software? Will there be timing constraints? Is the technology being used already state-of-the-art, or beyond it?

3) How critical is the software to the correct and safe functioning of the system?

4) What significance does software have in the overall system? Is it hardware intensive, software intensive, operator intensive?

5) What are the interoperability requirements of the system?

6) For what application will the software be used, i.e., $C^3$, avionics, fire control, flight control, etc.?

7) Will the software requirements be firm during the life-cycle of the system? Is an upgrade of the system planned?

8) What is the projected lifetime of the software? Is it planned to be replaced in the near future? Is this program a fast, short term solution which will be replaced by a long term solution when it is completed?

9) What are the resource constraints? Is there sufficient manpower to manage and review the documentation? Is there enough money? Is the program manager working with a compressed schedule?

10) Is there existing software code (either off-the-shelf or from a previous program) which can be used?

11) Is there existing documentation which can be used?

12) Is the same contractor being used for different phases in the system acquisition life-cycle? Can his software development plan, configuration management plan, etc. from the last phase be used for this phase?

13) What are the anticipated maintenance requirements?

14) In what phase of the acquisition life-cycle is the program?

b. Validation of DID Selection Algorithm

The algorithm and its documentation shall be validated for correctness in selecting the critical subset of DIDs for a given ECS software/firmware development program and for usability by project manager's staff in clearly and directly guiding him in DID selection. The following shall be considered for validation methodology:

1) Coordinated review by a representative sample of Government and Industry software development and acquisition specialists and project manager's staff.

2) Analyses of past ECS developments comparing the results of the application of the DID selection algorithm to the actual DIDs used in those developments and their effectiveness.

3) Selection of pilot case developments to use the DIDs selected by the algorithm. Data shall be collected and evaluated on algorithm usability and effectiveness of selected DIDs.

c. Development of the DID Selection Guidebook

The Contractor shall prepare a DID selection guidebook to be used by program managers in the selection of DIDs/Documents in the acquisition of Embedded Computer System (ECS) Software/Firmware. The following items are to be included to aid the Contractor in his thought processes and provide guidelines. (These data are meant to be a minimum and may be expanded upon by the JLC.)

1) The guidebook shall include a tutorial on why Software/Firmware documentation is needed, including the following topics:

   - maintenance of software/firmware
   - high personnel turnover
   - planning and good design
   - agreement that design is acceptable
   - management/development overviews.

   The Contractor shall include case histories to reinforce the need for documentation.

2) The guidebook shall include a section which identifies all pertinent military standards (e.g., 1679, 483, 881, 490, XXX, etc.) effecting the use of DIDs and any appropriate comments on their applicability or use.

3) The Contractor shall study the DIDs and DID overviews (attached) and modify the DID overviews to state when each particular DID is needed or not needed. The expanded 'DID overviews" shall be included in the guidebook.

4) The use of the algorithm or selection process shall be demonstrated in the guidebook with examples/instructions on its effective use.

## APPENDIX D ADDITIONAL DOCUMENTS

### D.1 FIRMWARE SUPPORT MANUAL

The Firmware Support Manual provides information necessary to modify the Read-Only-Memory (ROM) components of an embedded computer system. Specifically, the Firmware Support Manual describes the physical ROM components, installation and repair procedures, ROM programming/testing equipment and procedures for ROM "burning," and any special handling or security requirements. In the above discussion, the term "ROM" encompasses ROM's, PROM's, EPROM's, UVPROM's, and other forms of ROM devices.

The Firmware Support Manual is not intended to provide information regarding the design and implementation of the computer programs represented by the bit pattern within the ROM. These computer programs are incorporated into the mainstream of software development (and documented) just as any computer program module intended to reside on read-write memory.

Relevant DIDs are:

1. USAF UDI-E-3937-ASD "Firmware Support Data."
2. NSA/CSS DI-H-5526 "Firmware Data Abstract and Document Record.
3. NSA U-H-5552 "Firmware Documentation Standards."
4. EIA draft "Hardware Intensive Firmware Description.

## D.2 OPERATIONAL CONCEPT DOCUMENT

The Operational Concept Document provides explicit description of the way in which the data processing system will appear to its users and the way in which it will interact with them. It describes assumptions being made about how the user will use the system to perform his operational tasks. It is derived from: System Operational Concept, System Specification, CI and CPCI identification, knowledge of current operations and environment, etc. Although the system is typically in an early stage of development and the exact production baseline configuration is unknown when the OCD is first written, this document serves to clarify and make explicit the intended appearance and functional capabilities of the future system. The OCD provides a basis for concensus and understanding between the various agencies and contractors involved and serves as a valuable source of information during design, implementation, test and acceptance of the system. The OCD is delivered prior to the System Design Review (SDR) and the approved Operational Concept is formally presented during the SDR.

## D.3 REQUIREMENTS TRACEABILITY MATRIX

The Requirements Traceability Matrix (RTM) explicitly identifies each requirement specified in the System/Segment Specification and provides a mechanism for tracing the allocation of these requirements to the hardware and software components of the system/segment (e.g., MIL-STD-483/490 CI's and CPCI's). In the case where a system is divided into segments, the RTM documents the allocation of system requirements to segments, and then to the lower level specifications and documents: Software Requirements Specifications, Software Preliminary and Detailed Design Documents, Software Product Specifications, and the appropriate hardware specifications and documents. The RTM, as it evolves, provides a basis for testing and acceptance.

Ideas to be contained in DID block 10 are given on the following page

## PROPOSED DID BLOCK 10

10. Preparation Instructions for Requirements Traceability Matrix

The content of the Requirements Traceability Matrix shall be as follows: The leftmost columns of the matrix shall contain paragraph and/or subparagraphs numbers of each system/system segment (or top-level) specification requirement and the test method(s) of that requirement from the test verification matrix. Adjacent column sets shall show the allocation of requirements to lower level specifications by listing corresponding paragraph and/or subparagraph numbers in those documents and shall show the test method of each allocated requirement from each document's test verification matrix. For traceability of requirements between the system specification and the segment specifications, there shall be a separate column for each segment specification. For traceability of requirements between a system or segment specification and lower level specifications, there shall be separate columns sets for each development (type B), product (type C), and interface specification (IFS) or interface control document (ICD). Note that test methods for C specs, IFS's and ICD's may not be applicable. Submatrices which repeat the development specification requirements in the left column set and the corresponding product specification paragraph(s) in the right column set are acceptable. Initial submittal of the RTM shall include the available segment information as well as interface specifications. Updates shall accompany subsequent new specifications submittals, revisions to approved specification, (e.g., via SCN's), and resubmittals of draft specifications which contain changes in their paragraph numbering relative to higher level specifications. Revisions to the RTM in response to comments made by the SPO shall be submitted by the contractor within fifteen (15) working days of receipt of comments.

3.x Requirements

Develop a means for requirements traceability which will indicate where, in each development, product and interface specification (or ICD), each of the requirements in the system/system segment (type A) specification is addressed and which will indicate the corresponding test method for each of those requirements.

## APPENDIX E

COMPUTER RELATED DATA ITEM DESCRIPTIONS
NOT ADDRESSED IN THE JLC DID STUDY DATED 1 JUNE 1981

| DID NUMBER | TITLE | SOURCE DOCUMENT NO. |
|---|---|---|
| DI-H-3277 | Training Equipment Computer Program Documentation | MIL-STD-876 |
| UDI-I-E-3935 | Firmware Development Plan (FDP) | AFR 800-14 |
| UDI-E-3936-ASD | Firmware Technical Description (Product Specification) | MIL-D-83468 |
| UDI-E-3937-ASD | Firmware Support Data | AFR 800-14 |
| DI-M-5085 | Commercial Computer and Peripherial Equipment Manuals | |
| UDI-H-5503 | Firmware Reprocurement Specification | |
| DI-H-5526 | Firmware Data Abstract and Document Record | |
| UDI-H-5552 | Firmware Document Standards | MIL-STD-100 |
| DI-A-30001 | Software Delivery Documentation | |
| DI-A-30023 | ADPE Systems Configuration Report (Schematics) | |
| DI-E-30102 | Computer Program Detail Specification | |
| DI-E-30112 | Computer Program Listings | |
| DI-E-30141 | Interface Specification | MIL-STD-483 |
| DI-E-30149 | Research & Development Computer Software | |
| DI-E-30150 | Visual Data Base Description Document | MIL-STD-490 |
| DI-F-30202 | Automatic Data Processing (ADP) Manpower and Cost | AFM-171-404 |

APPENDIX E CONTINUED

| DID NUMBER | TITLE | SOURCE DOCUMENT NO. |
|---|---|---|
| DI-L-30309 | Automatic Data Processing Equipment (ADPE) Cost and Utilization | AFM-300-6 |
| DI-M-30404 | Computer Program Configuration Item Users Manual (Operational Software) | AFSCM/AFLCM 375-7 |
| DI-M-30405 | Computer Programming Manual | |
| DI-M-30408 | Computer Program Configuration Item Maintenance Manual (Operational Software) | |
| DI-M-30410 | Computer System Operational Manual | |

## APPENDIX F DID COMMENTS

Table I of the JLC Report (reference 1), page 5 lists 29 documents recommended by the CSM Subgroup. Of these, new DIDs were developed for 23. The DID contractor recommended that existing DIDs be used for the other six.

The Documentation Panel reviewed the 23 new DIDs plus four of the existing DIDs. Comments on these are on the following pages. We ignored the Engineering Change Proposal and the Specification Change Notice.

## R-DID-101/SYSTEM/SEGMENT SPEC

Below are comments on the software aspects and on the general aspects of the System/Segment Spec DID. A considerable number of changes and improvements are recommended. Problem areas are also noted such as the level of detail called for on the interfaces (3.1.5). Their resolution will impact the life cycle model as depicted in Figures 1 and 2 of the DID package.

1. Spell out all abbreviations (e.g., DMA, ISA etc.).
2. Move Configuration Identification/Spec tree from 3.1 to 3.3.1.
3. Add Interface Requirements to Spec tree.
4. 3.1.3 After "shall contain" add "as a minimum."
5. 3.1.5.1, 3.1.5.2, 3.1.5.3 After "quantitive terms" add "such as."
6. 3.2.2 Delete g. "command and control requirements."
7. 3.3
   (1) Insert "and computer software" after "system equipment."
   (2) Don't understand last sentence - What does "specify criteria" mean?
8. 3.3.1 This section should state then an appendix may be used for the specification tree. The example figure should be updated to include an Interface Specification.
9. 3.3.2.3 Computer System Requirements
   Add "existing hardware and software items"
   Delete "etc."
10. Add a paragraph (3.3.2.4) on Computer Resources Utilization Measurement. This paragraph shall specify computer resources utilization monitoring requirements. Included shall be the requirement that resources be dynamically monitored during real time operations, the variability of data recording intervals, the functions under operator control, and the methods of displaying/

reporting the utilization data.

11. Add paragraph (3.3.2.5) on Firmware. This paragraph shall state the requirements for programming languages (3.3.2.1) and design and coding standards (3.3.2.2) which apply to firmware as well as software.

12. The subsections on maintenance and supply should be expanded to include software. It is important to provide guidance as to where software maintenance/modification will be done and how software upgrades will be distributed to the field.

13. System Spec/SW

    3.4.3.2 Computer Resources Support

    Change first sentence to:

    "This paragraph shall specify the computer resources facilities, equipment, and software to be provided for CSCI maintenance during the system's operational service life."

14. Section 4 should be titled "Test Requirements." There seems to be considerable overlap between the information asked for here and in the Test Plan.

15. System/Segment level verification cross reference matrix should be added to section 4.

16. Add a glossary in Section 6.

R-DID-102 CRLCMP VOLUME I, ACQUISITION

R-DID-103 CRLCMP, VOLUME II, SUPPORT

DELETE BOTH THESE DOCUMENTS

R-DID-104 SOFTWARE DEVELOPMENT PLAN

1. 1.2 - Expand somewhat to describe general implementation approach e.g., versions, languages, tools). This also might be the place to discuss categories of software and the level of attention to be applied to each category (e.g., how will operating systems from hardware vendors be treated?)

2. 6.1 - typo - "computational."

3. Section 6.5 "Preparation for Delivery" is in System Spec section 5, (DID 101).

4. Paragraph 7 — Should include techniques for managing the allocation of development resources, priorities who resolves conflicts etc.

5. 7.1 — Add firmware - unique facility.

6. Paragraph 9.0 — Development plan would be a logical place to document developer's understanding of what criteria must be fulfilled to satisfy each program milestone. For example:

   What constitutes data package for PDR? What documentation in package? To what level of detail? Whose approval must be obtained to hold PDR meeting? Can PDR's be segmented? etc.

7. Add the following items from existing DID 30567A:

   system support

   proprietary items

   long lead items

   Ilities treatment

## R-DID-105 SOFTWARE CONFIGURATION MANAGEMENT PLAN

1. This DID does not discuss or reference the concept of "incremental builds" or partial deliveries and the impact on software configuration control.
2. Add a section dealing with implementation similar to section 6 of Software Quality Assurance Plan, R-DID-106.

| | | |
|---|---|---|
| 3. | 1.1 | Add abbreviation to title "(SCMP)." |
| 4. | 1.3 | Replace "the approval channel for making changes." to "controlled" |
| 5. | 4.1.d | What does "Contractual Significance" imply? It is not a standard DOD term. |
| 6. | 9 | Delete reference to other DIDs. |

### MINORITY REPORT

This plan should be incorporated into the overall configuration management plan to insure that configuration management is consistent, that the impact of changes on all parts of the system are covered. This can only occur if the handling of all ECPs are coordinated. The information in this DID should be added to that for the overall CMP. Software CM personnel should help develop a new CMP DID.

R-DID-106 SOFTWARE QUALITY ASSURANCE PLAN

1. General Comments: The DID should be rewritten to reflect changes in MIL-S-52779A.

2. 9 Add MIL-S-52279 and MIL-S-5227A and MIL-STD-XXX.

3. 10.4.1 Add to last sentence "and test standards."

4. 4.3.4 Change title to "Critical Resource Factors Montoring."

5. 4.5 Add to section 4.5
   e. Method of documentation of problem, corrective action, and test results.

6. Add a new section. 4.6 Controls for Computer Program Library.

7. Add a new section.
   4.7 Work Certification. Describe the procedures and controls for the handling of source code and object code and related data in their various forms and versions, from the time of their initial approval or acceptance until they have been incorporated into the final media. This shall include controls to ensure that different computer program versions are accurately identified and documented, that no authorized modifications are made, that all approved modifications are properly incorporated, and that software submitted for testing is the correct version.

8. Block 7 "...internal plan or agreements..." imply that it is not a deliverable data item. Reword "It documents the contractor's quality assurance program as applied and, if necessary, or tailored to the specific contract."

9. 4.3.1 thru 4.6 This section might be the data that should be provided as Section 5.6 of the CRLCMP, Volume 1, as it relates to a management function. These paragraphs infer that the QA process is to perform all the listed activities. The QA process is to <u>assure</u> they <u>happen</u>, and in accord with contract promises or company practices. Change context.

| | | |
|---|---|---|
| 10. | | Section 5 of MIL-S-52779 Preparation for Delivery, has been deleted. We suggest that Section 5.6.3.1 of the CRLCMP, Volume 1, be placed in the SQAP and Sections 5.6.3.2 and 5.6.3.3 be part of the SCMP which are then monitored for compliance by the SQA group. |
| 11. | Blocks 3 & 7 | Add "or activity" wherever "contractor" is stated. |

R-DID-107 SOFTWARE PROBLEM/CHANGE REPORT

DELETE THIS DID

## R-DID-108 SOFTWARE REQUIREMENTS SPECIFICATION

1. Rewrite DID to use 483 Notice 2.
2. A subsection (or subparagraph) should be added to the DID wherein the computer resources requirements allocated to the particular CSCI are to be placed. This includes programming languages, design coding and standards, and computer system requirements of the System/Segment Spec. (3.3.2).
3. Although DID's can include an example when implemented for a specific system, the DID itself should not have general examples because they are misleading, e.g. FIGURE 2 has the design in Software Specification instead of concentrating on requirements.
4. Include definition of Inspection, Analysis, Demonstration and Test in MIL-STD-XXX. Add to Table III in this DID.

| | | |
|---|---|---|
| 5. | 3&7 | Delete the word "digital" from the phrase "digital computer software." |
| 6. | 3.1.1 | Interface Requirements. The word "contractor" should be deleted from the 4th line. The government also performs analysis.<br><br>Add "whenever separate interface specifications are to be used they will be referenced in this section." |
| 7. | 3.2 line 5 to 7 | Delete "and followed ... Table II"<br>Delete Table II. |
| 8. | 3.2.17 | The traditional B-5 format for functional requirements specification does not enhance the software engineering process. In fact, the use of separate input and output sections leads to confusion and conflicting definitions of the function-to-function interface.<br><br>Recommend that the traditional format be dropped in favor of a format that requires |

| | | |
|---|---|---|
| | | separate sections be created consisting of specific function-to-function interface definition subsections and a separate section created consisting of specific functional processing description subsections. |
| 9. | 3.3 | Add "whenever a separate specification for data base is written it will be referenced in this paragraph." |
| 10. | 3.4 | Delete this paragraph. |
| 11. | 4.0 | Change title to "Test Requirements." |
| 12. | Figure 2 | Delete figure and all references to it. |

MINORITY REPORT

There should always be an IRS if a CSCI interface, with another CSCI. Then there will only be one source for all design groups to use for interface details and there will only be one document to change when changes are made to an interface. For that reason paragraph 3.1.1 should be changed so that it only identifies the interfaces and references the IRSs. Also, the material in 3.1.2, 3.1.3, 3.1.4 should be deleted and the appropriate parts therein incorporated into the IRS.

## DID 109 - INTERFACE REQUIREMENTS SPECIFICATION

1. Block 3 delete Last sentence
2. Block 7, line 7, add after "contractors" "or activities" since government activities sometimes develop this document in-house.
3. Block 7, delete second paragraph.
4. The IRS does specify the physical portion of the interface. The DID does not presently call for that. Contractors on each side of an interface must have the electrical characteristics (signals) of the interface as well as the functional characteristics (messages and data elements) defined. This must be addressed.
5. Correct punctuation everywhere.
6. 1.1 — Is interface a configuration item? and does it have configuration item numbers and nomenclature as the DID calls for?
7. 1.1 line 5 & 7 — Delete quotation mark before "The nomenclaure"... and after ... numbers)."
8. 1.2 — Should be made to read plural (i.e., interfaces, ... their relationship ..., which they apply.)
9. 3.1 — Delete the underline.
10. Paragraph 3.2, line 2, after "interfacing equipment" insert "man to machine interface."
11. Paragraph 3.2, line 3, change "computer programs" to read "the software."
12. line 4 — Delete "where applicable." Phases such as these only lead to arguments between contractors and customers - give the government representative credit for recognizing when somethings are legitimately omitted from a document.
13. 3.2.1 3.2.2 — Change "Interface Signal Cross Index" in title to "Interface Data."

14. 3.2.1 Line 2 — Delete "if applicable."

15. 3.2.1.X.1 3.2.1.Y.1 — Delete quotation marks.

16. 3.2.1.X.5 3.2.1.Y.5 — Second line - last word - "related", question- "realted to what?"

17. 3.2.1.X.14 3.2.1.Y.14 — Replace "with its" with "and."

18. Paragraph 3.2.1.X.4 - The following sentence appears: "Each interface data requirement shall be realted to its corresponding Executive Logic requirements." Again, this requirement, makes no sense. What is required within this paragraph, with regards to content, is vague. This paragraph should be rewritten to make clear that interlock logic is being addressed.

19. Paragraph 3.2.1.X.7 - This paragraph is incomplete. The paragraph only requires that the "protocol for blocking, message switching, and handshaking" be stated. Error conditions and interface testing techniques or requirements with specific prose pertaining to the periodicity of tests and the type of data to be tested should be included.

20. 3.2.1.x.15 — Add a subsection calling for limits and/or ranges of values and accuracy/precision requirements. Or change 3.2.1.x.11 title to Quantitative Description of Data Elements" and combine material here with units of measure.

21. 3.2.2(all) — The specific comments on paragraphs 3.2.1.all apply to like paragraphs of 3.2.2.

22. Paragraph 6. — Delete last sentence.

23. 4. — Delete entire paragraph.

24. Delete sections 5, 7, 8 & 9.

MINORITY REPORT

3.2.1.x.2
3.2.1.x.4
3.2.1.x.6
3.2.1.x9
3.1.1.x.12

(1) The requirements for shared memory, executive logic control, interrupts, memory map, and shared variables apply only to interface between two or more CSCIs that share the same computer. They should be categorized as such and grouped together.

(2) The material called for in these paragraphs is at the design level not the requirements level and will not be known at the time the IRS is due, which is between SDR and PDR. It should be in the IDD.

R-DID-110 SOFTWARE STANDARDS AND PROCEDURES MANUAL (SSPM)

1. Delete Section 3. It is an example of a paragraph that increases the cost of a document without adding utility. It calls for material (guidelines, philosophy, or rationale used in defining the software performance requirements, and in allocating requirements to design elements) which can be obtained elsewhere and which is volatile in nature. Philosophy and rationale will change as the design evolves and as ECPs are made and therefore the text will become outdated unless the SSPM is changed. It probably won't be. Further, it's good to avoid situations where many documents have to be changed for a change to the CSCI(s).

| | | |
|---|---|---|
| 2. | 4.4.1 thru 4.4.3 | Delete these paragraphs. |
| 3. | 5.4 | This paragraph should include a discussion of when it will be permissible to use assembly language. |
| 4. | 6.1,6,2, and 6.3 | Delete the sentence in each of these sections, which defines the purpose of the activity. |
| 5. | 6.4.2 | Change the last sentence; for example to "Define the integration testing methodology, top-down, bottom-up etc.". |
| 6. | 6 | This information should be included in the QA Plan, except for section 6.6 which should be included in the Software Development Plan. |

## DID 111 - SOFTWARE TOP LEVEL DESIGN DOCUMENT

1. Paragraph numbers of DIDs 111 and 112 should match to allow traceability between the two levels of design; example: Data base is Section 3.5 in this DID but is Section 3.3 in DID 112.

2. All acronyms and abbreviations should be official JLC and should be found in a single glossary.

3. Frequent use of "may" vice "shall" in contractual statements.

4. Numerous typo's and incorrect use of capitalization should be corrected. Also, correct punctuation.

5. Block 7: Change "baselined" to "reviewed." Change "SE&TD, IV&V .." to "the GSE&I and IV&V Contractors..."

| | | |
|---|---|---|
| 6. | 1.1 | CSCI's should be allowed to be plural, since it is possible that one software Top Level Design Document could be the basis for more than one program (CSCI). |
| 7. | 1.1 | Third line, lead line with "identified as ..." |
| 8. | 3.2 | Need to have a section that addresses each of the CSCI's and indicate which CSC's apply to which CSCI (i.e., module to program allocation). Replace Figure 1 with the attached. |
| 9. | 3.3 | Review and rewrite sections. There is significant difference between information and control flow neither of which are adequately covered. |
| 10. | 3.3.2 and 3.3.4 | There should be a requirement to specify worst case timing requirements. What functions must be executable simultaneously, etc. Timing for individual CSCs does not give a complete picture in a multiprogramming/multitasking environment. |
| 11. | 3.4 | The $N^2$ chart is subject to controversy in regard to its utility. Delete the $N^2$ chart (figure 4) and all references to it in the paragraph. |

12. Paragraph 3.4.X, first line, the word "intent" is redundant when combined with purpose; delete "intent and."

13. Add a new paragraph, 3.4.x.5, to address error detection and recovery.

14. Need subsection (3.7) for CPU allocation to cover multiprocessing and multiprogramming applications.

15. Delete paragraphs 4, 5, 7, 8, 9.

Requirements

| | | CSC1 | CSC2 | ... | CSCN |
|---|---|---|---|---|---|
| 3.2.1 | | X | | | |
| 3.2.2 | | | X | | |
| 3.2.n | | | | | |
| . | | | | | |
| . | | | | X | |
| . | | | | X | |
| 3.2.m | | | | | |
| | | | | | X |

FIGURE 1 REPLACEMENT

MINORITY REPORT

The Software Top Level Design Document and Software Detailed Design Document should be combined to form the Software product spec, eliminating 2 DIDs. This document could be delivered in snapshot forms, as required, until the entire document is accepted. The Configuration Management and delivery problems with this scheme are what should be addressed.

## DID 112 - SOFTWARE DETAILED DESIGN DOCUMENT

1. Paragraph numbers of DID 112 should match DID 111 as much as possible to allow traceability and minimize confusion.

2. General - recommend that an optional section for programming guidelines be included to allow a small project to reduce the numbers of required documents.

3. Correct punctuation.

4. Paragraph 1.1, Need to allow CSCI's to be plural, since it is possible that one Software Detailed Design Document could be the basis for more than one program (CSCI).

5. Paragraph 3.1, Need to address each of the CSCI's and INDICATE which CSC's apply to which CSCI (i.e., module to program allocation).

6. Paragraph 3.1.1, Should be 3.1.X. When referring to routines, the term routines needs to be defined - some systems address them as subroutines, some as procedures. Believe this should be addressed as the lowest level program structure element.

7. Paragraph 3.1.1.1 should contain an example of a chart or table illustrating functional allocation.

8. Paragraph 3.1.1.2 should probably include an update to Table I of the DID 111.

9. Paragraph 3.1.1.3.Y - all bullets should be numbered sub-paragraphs.

10. Paragraph 3.1.1.Y Care must be taken to make certain we don't just verbalize code. This area must be kept at a level above coding. We have gotten into the trap of verbalizing code in the post and the document maintenance becomes overwhelming.

11. Paragraph 3.1.1.3.Y, first bullet, change "development" to read "description."

12. Paragraph 3.3, fourth line, recommendations to delete "so as to require a data base manager."

13. Delete paragraphs 4, 5, 7, 8, 9.

MINORITY REPORT

The Software Top Level Design Document and Software Detailed Design Document should be combined to form the Software product spec, eliminating 2 DIDs. This document could be delivered in snapshop forms, as required, until the entire document is accepted. The Configuration Management and delivery problems with this scheme are what should be addressed.

## DID 113 - INTERFACE DESIGN DOCUMENT

1. Recommend that the title be changed to "Interface Design Specification."

2. General, Data required at the IRS and IDD levels which are missing and should be added are error condition detection and interface testing requirements.

3. The "model" the IDD (and the IRS) is based on assumes that interfacing CSCIs will all execute in the same computer. Thus the IDD DID calls for things (executive control logic, shared variables) that will not apply to many projects. Most system to system interfaces will <u>not</u> have CSCIs executing in the same computer. The DID should be changed to accommodate both situations.

| | | |
|---|---|---|
| 4. | Block 3 | Relationship of the IDD to the IRS should be presented. |
| 5. | Block 7 | Line 3, delete "too." Between "or" and "several" insert "if." |
| 6. | 3.1 | Line 5, delete "it is" and insert "they are", Delete $N^2$ chart reference and Figure 1. |
| 7. | 3.2.1.X.4 | This section requires "detailed design of executive control logic." This requirement makes no sense. What is meant by "executive control logic" should be explained or the paragraph deleted. This paragraph is undoubtedly intended to cover methods for shared data and should be reworded to so state. |

### SPLIT DECISION

The panel was equally divided on accepting or rejecting the following comments.

| | | |
|---|---|---|
| 8. | 1.1 | Following the word "between" change remainder of sentence to "(insert nomenclatures and configuration item numbers of interfacing CSCIs |

and CIs)."

| | | |
|---|---|---|
| 9. | 1.2 | Delete this subsection. As is, direction in DID is vague, calling for "overall interface design" and "major design issues." Will only add to cost of IDDs. |
| 10. | 3.2.1 | Change 3.2 to a title only paragraph and delete text calling for brief overview and discussion of key design issues. Only adds to IDD's cost. (1) Change "Signal" to "data element" here and throughout IDD. Signals are the hardware-to-hardware realization of the interface. |
| 11. | 3.2.1.x.1 | Delete "and give an overview of the key design issues..." |

12. Define "design considerations" and provide details on what is expected from the contractor. Actually this is material only appropriate at design reviews and, if the customer feels the need, documented as study reports.

MINORITY REPORT

13. Block 7, the IDD should always be required when a computer to computer interface is involved.

14. Block 7, the IDD is required in order to adequately detail performance requirements in the SRS. It should be produced earlier in the cycle than shown.

15. The IDD must be a government controlled specification. It is both controlled and maintained by the government and changes can only be made via ECP once it is baselined.

16. Block 7, (1) Message presented should be changed to always calling for an IDD. If left to the contractor, pressure will be to define extremely complex information as being very simple. The major reason there should always be a separate IDD is the avoidance of the redundancy of repeating the interface material in two or more Detailed Design Documents

and to decrease the potential that the different design and coding groups will be working from different versions of the interface. Also, only the IDD will have to be changed when changes are made to the interface design.

(2) Design documents should be controlled by the contractor - delete last sentence starting with "It is controlled....."

17. Two panel members disagree with comment 3.

| | | |
|---|---|---|
| 18. | 3.1 | (1) How does the block diagram called for here differ from that called for in the IRS? Why have both of them? |
| 19. | 3.2.1.x.2 | Changes in formats should be clearly delineated in 3.2.x.1 not here. Delete sentence starting with "Indicate whether..." |
| 20. | 3.2.1.X.7 | Memory Maps are not required to detail an interface. Place it in the design. Relocatable modules and segments make this an impossible task. |
| 21. | 3.2.1.x3<br>3.2.1.x.4<br>3.2.1.x.5<br>3.2.1.x.6<br>3.2.1.x.7<br>3.2.1.x.10 | The material called for these subsections apply only to CSCIs which share the same computer (CPU and memory). A qualification should be added to the DID which makes this explicit - for those CSCIs which do not share the same computer, material such as executive control design, priority determination algorithms belong in each respective CSCI Detailed Design Spec,) |

## R-DID-114/DATA BASE DESIGN DOCUMENT

General Comment

DID-114, as currently written, was found inadequate and was rewritten. The rewritten version is attached under the title, "A Working Paper on DATA BASE DOCUMENTATION, June, 1981."

Selected references used for the rewritten version of DID-114 are as follows:

1) Atre, S. Data Base: Structured Techniques for Design, Performance, and Management. New York: John Wiley & Sons, 1980.

2) Date, C.J. An Introduction to Database Systems, 2nd ed. Reading, Mass.: Addison-Wesley, 1977.

3) Ullman, Jeffrey D. Principles of Database systems. Potomac, Maryland: Computer Science Press, Inc. 1980.

General Comments on R-DID-114 follow.

1. In general, the DID suffers from a lack of focus. The Data Base Design Document identifies itself as covering both the data base application system and the data base management system (DBMS). The delineation of documentation between each of these distinct products should be explicit. The DID allows for inclusion of standard documentation on the DBMS portion when documenting a data base application. But the DID confuses this by selecting out and interleaving parts of a DBMS description with the application data base description. For example paragraph 3.3 Data Base starts with the requirement for describing the terminology and relationship between the different groupings of data. These are elements of a DBMS, independent of the application. It continues with a requirement to name all files, records, fields, and items, which are dependent on the application. It then follows with the requirement to identify construction and reconstruction algorithms, and data access and

documentation.

2. There is a definite need to make the documentation required distinct for the two areas of data base applications and data base management systems. This provides the means to clearly identify implementation dependent issues distinct from application requirements.

3. There is no attempt to address any of the many additional issues in data base design resulting from implementation as a distributed application. Such issues include data concurrency, data distribution, distributed query processing, distributed control, and access and security in a distributed application. While the user of a specific application would not necessarily know the difference between a centralized or a distributed data base implementation, the designer of such an application would.

4. The meaning of section 3.2 is not clear. Does it refer to the programming source language? If so, than that information does belong here, but in a programmer reference manual. Does it refer to some data base descriptor language? If so, then it should be clarified. It is also not clear why section 3.2.2 is included as a subparagraph of 3.2 and why it is titled "Processor Design."

5. The description of the data base should not be a subsection of the section describing the data base management rules. The information asked for under section 3., 3.1, and 3.2.2 is redundant with information called for in section 3.4.

6. The description of the data base elements should follow the structure of the data base itself. The following outline for section 3.1 is suggested:

```
3.        Data Base Description
3.1       Data Description
3.1.1     File Design
3.1.1.W   File W
3.1.1.W.X Record X (description of a record of file W)
3.1.1.W.X.Y Field Y (field Y of record X)
3.1.1.W.X.Y.Z. Item Z (item of field Y)
   (identical structures should reference previous
```

descriptions rather than repeating them)

3.2 Management system

3.2.1 Data Base Construction

3.2.2 Data Base Reconstruction

3.2.3 Accessing and Manipulation

# A WORKING PAPER ON

# DATA BASE DOCUMENTATION

by

W. L. McCoy

Code 9133

of

Naval Ocean Systems Center

San Diego, California

Tel. 714-225-7409

June 1981

## TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

## INTRODUCTION

The attached Data Item Description was written in response to an assignment by the Chairman of Panel "A" of the Software Workshop held at Monterey, California 22-25 June 1981 at the Naval Post Graduate School.

The panel is specifically addressing Software Documentation on behalf of the Joint Logistic Commanders' Policy Coordinating Group on Computer Resource Management. The objective is to reduce the size and amount of "paper requirements" inherent in modern software development. This particular assignment is to review the Proposed Data Base Description (DID) which is denoted as R-DID-114.

## COMMENTARY ON RDID-114

Data bases have been a most neglected and misunderstood part of software development. It is seldom recognized that a database may be many magnitudes more complex than the "programs" or algorythms which drive the data base.

One of the basic problems is the definition of just what a data base is. Most definitions suffer from the "blind man syndrome," as described in the fable about a number of blind men asked to describe an elephant. Each one described the part he could feel. So it is with data bases. The data base requirements and definitions given by a Management Information System Specialist are entirely different from those given by an applications programmer who works only with tactical embedded computers.

Another complicating factor in dealing with data bases is the fact that originally the pioneer designers of the data base concept tried to make the operating environment entirely invisible to the user. This was a noble ideal, but as many data base designers have learned, it is very difficult to talk about a specific data base without considering the environment it is being used in. Failing to do this has caused some very rude suprises, such as the illfated WWMCCS system. WWMCCS failed because its data base design was tied to a hardware suite which stood still in time, while the user requirements grew, and grew, and grew. The same problem faces any data base designer today, for who can tell what surprise the technological explosion will give us next.

We have learned from experience that no matter how esoteric or sophisticated our data base system is, it ultimately will have to go on a suite of hardware that is limited in size, speed and throughput capacity, thus limiting the growth potential and use-ability of its data base. The users of small embedded computers have had to relearn what the users of big systems learned a decade ago, the fact that there are limits. The wise designer of a computer data base today scopes his design with the thought of data conversion constantly in the back of his mind.

The avowed purpose of the current workshop is to reduce the amount of paper and, thence, the cost of computer software development. This must not be done, however, at the expense of losing the far more valuable asset of clear and maintainable software documentation. It is already clear that the cost of not having this asset far exceeds the cost of the documentation. Of far greater cost saving significance is the establishment of a clear and commonly understood documentation standard which describes for all users what the documentation requirements are and allows him/her to select the subset which best suits the particular application. The DoD goal of life cycle development cost reduction can best be met by developing a good Data Base Documentation Standard which is clear, flexible enough and adequate to meet both current and forseeable needs and, thus, obviate the necessity for creating unique DIDs or by confusing the industry, DoD and the user by having a large, confusing and diverse set of "specialized" and/or general "standards."

This writer has reviewed both the currently proposed and the past DIDs which have been written on the subject of Data Base Design and does not find any of these adequate to fulfill the requirements that are described in this paper. A number of texts have also been reviewed on this subject, the last of which is included in a short bibliography attached to this paper. This research has revealed a thread of commonality which ties present concepts together in principle, but this thread is seldom recognized.

The result of these endeavors is attached to this paper. It is a rewritten version of R-DID-114 in the format which this writer believes best addresses the real issues of data base design and management. The writer does not entertain in the slightest way the concept that this document is anything more than a very rough draft or that it is sufficient in itself. The writer does feel, however, that it does address, on a broad spectrum, the real issues of data base design in a manner that would adequately meet the current and at least some of the future requirements, while at the same time deleting many of the older architectural "provincialisms" of the preceding dedades. The draft is intended to enlighten, stimulate the thinking processes, and to invite commentary.

Data Base Design Document

2. Description/Purpose

The Data Base Design Document establishes and defines the type of data base, the environment in which it is used, its design/architecture, its structure and the methods, rules and language required for the maintenance, update and use of the data base system.

See Section (7) on separate sheet (1a)

10. Preparation instructions

10.0 Legal preface, see DI-R-2174

10.1 Identification (See R-DID-114)

10.2 Introduction

10.2.1 Background Describe the type of system this particular data base is intended to be used on and the use/user environment. Clear distinction shall be made as to whether the data base is intended for use as a management information system, a scientific application, an embedded computer system, or other system of special design and purpose.

10.2.2 Purpose Describe the purpose and intent of the Data Base Document. Normally this should be the same as block 3 above, but tailored and enhanced to fit the particular application of this document.

10.2.3 Scope This paragraph shall describe the scope of the data base and its objectives in terms of the user requirements, mission requirements or system requirements and the manner in which these objectives are intended to be met and described by this document.

10.3 Type of Data Base This section shall describe in detail the type of data base that is intended. This description shall use the terminology and vernacular appropriate to the application of the system described, with the intent that it be clearly understandable by the practitioners of the discipline addressed.

10.3.1 Hardware Environment This section shall describe the basic hardware environment which the data base system is intended to be used on. It will describe the type of computer, the memory allocation for data, the I/O ports available, Data Base System, and types of peripherals intended. Where distributive processing is used, the number of processors, the network interconnect scheme, the"hardshake protocol" and specific constraints shall be addressed. When appropriate and convenient, these documents shall reference the resource project on other documentation which contains the cited information.

10.3.2 Software Environment This section shall describe the software environment which the data base system was designed to be used on. If the system is designed for general use, it shall so state, but shall specify the systems on which it has been tested and used and describe constraints, if any which are peculiar to each system.

10.3.2.1 Operating System Where appropriate, name and describe the operating system which the data base was designed for use with. This description should describe the setup procedures, job control language required, utility programs required and or other special configurations used, such as special library packages and the special user software required. This section shall state the High Order Language (HOL) or other which the user program is written in and the version/mod of the computer HOL required to compile these programs. Where the system is designed for use under a special run time executive written for an embedded computer, user constraints shall be specified

10.3.3 Data Base Design Type This paragraph shall address the particular design approach of the data base. In particular, it shall state the design approach used, whether the relational approach, the hierarchial approach, or the network approach or what other approach is being utilized.

10.3.3.1 Record Characteristics Describe the record and file characteristics of the data base design. The definition of such words as file, record, etc., shall be given. Variable or fixed length file/record schema shall be described. Also describe the basic record characteristics and type, such as byte, word, etc.. In general, the purpose and intent of this section is to fully describe all record data characteristics. The description shall include, but not be limited to, the following.

10.3.3.2 Record Access Methods Describe the type of record access methodology used, such as sequential or indexed. Describe the data reference dictionaries, data indices, sequence pointers, data dictionaries, and bibliographies. In particular, show a record map which describes where pointers are embedded into the basic record unit.

10.3.3.3 Data compression and Expression If data compression and expansion techniques are used, describe the methodologies and techniques used to compress and decompress the data.

10.3.3.4 Stacking Where stacking and stack processing are used, describe fully the stacking techniques used, and the stack table/ pointer structure.

10.4.1 Throughput/timing requirements This paragrpah shall specify the throughput requirements and constraints in terms of number of record sets per unit time. Where applied to an embedded computer, this section shall specifically state the access time of a unit record, as specified by the contract, from mass stroage, through buffer channels, and to the active memory use via the runtime executive or direct memory access used. This unit time shall then be used to show full access time in terms of system time available and number of records that can be processed in such a manner that totalthroughput times can be computed. The specific data base access time and reserves time shall be stated. Where random record access techniques are used and where applicable, this shall be stated in terms of a statistical equation which addresses systems design parameters, in such a way as to allow a determination of the effect of system expansion on timing and throughput. Where distributive systems are used the impact of ~~scheme~~ for prioritizing of data access, and the "handshake protocol" used shall be considered with regard to their impact and timing and throughput. Where the application addresses a management information system, this paragraph shall show the anticipated throughput with respect to the target user system. It is the intent of this paragraph to enable the user to assess the operational and cost impact of increasing data base size.

10.4.2 User Formating Requirements Where the data base described is for management information systems, or business type application, this section will be included to describe for the user the required user information.

10.4.2.1 Organization Overview Show an overall flow diagram.

Describe the hierarchy of data base management rules and how they are implemented.

10.4.2.2 User Language Describe the syntax and rules for the use of the full management language. A complete dictionary of user words and meanings shall be included. If a particular syntax, such asthe Bacus Naur (BNF) syntax is used, it shallbe completely described, or an appropriate reference document/shall be cited which will acccomplish this purpose. In any case, departures from standard syntax shall be fully explained.

10.4.2.3 Output Formats The format of all output data shall be described in terms of information fields and the purpose, meaning and intent of the information presented.

10.4.2.3 Data Input Formats This section shall describe the input data format in terms of the outputs they generate. Where the output format is variable, the controls and tags for varying the output and the expected result shall be described.

10.4.2.4 Data Access Where applicable and in the use of a specific environment, the methodology of calling and displaying the data via an interactive terminal shall be described.

10.4.2.4.1 Special Security Describe all data security requirements.

10.4.2.5 Storage Requirements This section shall describe all data storage requirements which shall include active memory, peripheral storage requirements, and backup storage requirements. Thos requirements shall be described in terms of the storage required and or used. Backup storage life expectancy and/or renewal cycles shall be recommended, and the arguments supporting these requirements presented.

10.4.2.6 Human Interface Requirements This section shall describe the method of collecting data, for organizing, the data collected and the method of inputing to and updating the data base.

10.4.2.6.1 Human Factors This paragraph shall address the complexity level, human comprehensibility, and training required for personnel support and use of the data base.

10.4.7 Data Conversion Requirements This section shall address specifically the methods and requirements for converting the data base or specific components of the data base to another data base. This section shall describe the characteristics of the addressed data base such as ANSCI code, Field data, EBCDIC, etc. and the requirements for converting from one type of data base to another. It shall include changes in media tags, identification and formating labels, such as tape leader formats, end of file marks, etc.. This section is mandatory and applies to the format of the total data delivered in the final program product package.

10.5 Special Data Requirements When called for in the contract this section shall address any special data requirements not covered in the preceding sections. It may also be used at the discretion of the contractor as specified in the contract to describe the internal active data memory structures and/or formats of internal data. The general format of the data shall include but not be restricted to the following sections.

10.5.1 Data Dictionary This shall give the definition and description ofthe internal use of all internal data. Data shall be grouped and defined as the following, as a minimum:

(a) Global Data: Defined as data used by a number of processors operating as a system;

(b) Common Data: Defined as data used commonly by a number of modules within one processor;

(c) Local Data: Local data is data used by a single module or routine.

All data shall be listed by its literal name and its symbolic name and/or the engineering mathematical symbol it represents. Data shall further be listed in accordance with its functional characteristics, such as whether it is ASCI data, Boolean logical data, or mathematical data formated for external transfer or internal usage by the CPU. If the data is arithmetic in nature the scale factor shall be specified and the computer word size requirements stated, along with field definitions within the data word, such as the mantissa and characteristic of an arithmetic double precision floating point word.

In addition, data shall be categorized as to whether it is constant data or variable data, and all variable data shall specifically state whether or not an initial value is required and what that value should be along with any special conditions which require its reinitialization.

10.5.2 Data Cross Reference Where required a Data Cross Reference shall be provided.

10.5.3 Memory Map A memory map shall be provided which shows the location in memory where all data are stored.

10.5.4 Table Structure This section shall describe the structure of all internal tables (meaning tables internal to a specific CPU memory structure). The structure of the table shall describe the table access dictionary, the basic memory storage unit (i.e., 1 byte, 2 byte, etc.). The indexing scheme, the table access dictionary, and table concatenation schemes. Where practical these shall be standardized throughout the system.

10.5.5 Special Word Formats This section shall describe any or all special word formats used in the data base. For example, see Figure I for an example of a data word used to depict a range and elevation word used in a tactical computing system.

(Use Figure I in DI-A-2140)

Other examples may be packed word instruction formats used in "table driven" control systems.

## APPENDICES

When specified in the contract the following appendices shall be used in the data base design system.

(a) Glossary: This is a glossary of terms used in the data base design system.

(b) Index: This is an index of all data symbols and is utilized in the data base.

(c) Data Cross Reference: This is a cross reference of data symbols and may reference data symbolic names to engineering symbols or common name labels. This may be used in lieu of similiar sections in Section 9.

(d) Bibliography: This is a bibliography of technical data which may enhance of support the maintenance of this data base system.

(7) Application/Interrelationships This document is based on the top level computer program specification documents and is intended to be used with large software development projects and with a variety of software applications. It is intended that it be tailored to fit the particular applications and is written to cover a variety of disciplines, from the large data bases to small embedded computer systems or distributive computer systems.

The elements of the DID are intended as a minimum set of data which the user may tailor to his/her specific requirements.

Block (9) MIL-STD-483
MIL-STD-1679
DoD Standard 7935.1-5

R-DID-115/SOFTWARE PRODUCT SPEC (SPS)

| | | |
|---|---|---|
| 1. | 2.3 line 3 | Add after disk, "or firmware memory device". |
| 2. | 3.2 | This section calls for listings if IDDs are generated. Listings are not associated with IDDs. The IDD only specifies an interface. The software to implement the interface is included in the interfaced CSCIs. |

The Panel was equally divided on the following comments.

3. The "model" upon which the IDD (and the IRS) is based is needed by a reviewer in order to appreciate what interface listings are (see 3.2 of the DID). An interface is the description of the functional (data transferred) and physical (electrical signals that go between) characteristics. Within each CSCI there will be programs, such as message handlers, that handle the data transferred from another CSCI. The listing of such programs would not go in an interface document but in the product spec of the CSCI in which the program is contained.

4. Source comment requirements should be put in the system/segment spec and then allocated to the SRS which is the "test to" document. That way the listings can be tested (by inspection) to see if they conform to the good workmanship requirements that were specified. This is no different than the workmanship for hardware that is already called for in the system spec (3.3.3, 3.3.3.2, and 3.3.3.3).

| | | |
|---|---|---|
| 5. | 2.3 | Delete from.."Compiler Source statements..." to end of page. |
| 6. | Page 2 | Delete from top of page thru "The Restrictions" sentence. |
| 7. | 3.2 | IDD should, if anything, be appended to IRS. |
| 8. | 2.3 | Cross reference listings specifically call for mnemonically labeled statements. Numbered |

statements should also be cross referenced. Change the bullets to sub-paragraph numbers.

9. A provision needs to be made for machine readable object code. In section 2.3, delivery on "card decks, magnetic tape, or disk" is too restrictive. What about direct transmission across remote data lines, or pluggable bubble memories, etc.? Perhaps a phrase like "media acceptable to the Government" could be used.

## R-DID-116 VERSION DESCRIPTION DOCUMENT

This DID is acceptable as is, although some panel members are in favor of the following change.

Paragraph 2, change "APPLICABLE DOCUMENTS" to "REFERENCE DOCUMENTS" since a Version Description Document does not have "applicable documents."

## R-DID-117 SOFTWARE TEST PLAN

1. A preliminary test plan is needed at PDR and the final plan should be submitted at CDR. (Reference 80.6451.2-316, Rev. 1, page 27).

| | | |
|---|---|---|
| 2. | Block 7 line 4 | Change "Quality Assurance" to "Test". |
| 3. | Block 7 line 6 | Change period following "Specification" to comma and make "Also" lower case. |
| 4. | 4.1 | This isn't the place to define a unit. It should be defined in the SDP or SSPM. |
| 5. | 4.2.2 (a) line 3 | Add "testing" after "integration". |
| 6. | 4.2.4 (a)<br>5.2.1 (a) | Define terms. |
| 7. | 4.2.5.b | This section should reference the SRS as the source of the software requirements. |
| 8. | 5.2.3 | The example of a tabular form to correlate tests versus requirements could be better. |
| 9. | 5.5.d | Shouldn't the "of control" be "to control"? |

### SPLIT DECISION

The panel was equally divided on the following comments.

| | | |
|---|---|---|
| 10. | Paragraph 3 Line 5 | Software Test Plan [illegible]es not account for system level testing and it should. |

11. The test plan should provide for both informal and formal Hardware/Software Integration Testing. These tests should be designed to verify that the software correctly functions with the interfacing hardware prior to system level test. In many cases system level tests do not adequately test the software/hardware interface from the software viewpoint.

12. This DID refers to interfaces between the contractor and the government (seems to use customer and government interchangeably). It should also specify an interface to contractor CM, QA, and independent test groups. There is also an Independent Verification and Validation contractor interface when used.

13. There should be one outline for all three types of testing (Unit, Integration and Qualification testing) a proposed outline would be:

    x.1 Purpose
    x.2 Resources Required
    x.3 Test Management
    x.4 Test Structure
    x.5 Test Requirements
    x.6 Schedules
    x.7 Retest
    x.9 Test execution QC

| | | |
|---|---|---|
| 14. | Block 3 Line 4 | "test requirements" are provided in SRS. |
| 15. | 1.1 Line 4 | "Authority" not called for in other DIDs. |
| 16. | Paragraph 2 | Some provisions should be made for reference documents. |
| 17. | Paragraph 3 Line 5 | The word "procedures" is used as part of the description or purpose of this document but it is not. Test Procedures in a separate DID. Remove the phase "test implementation procedures". |

|  |  |  |
|---|---|---|
| 18. | Paragraph 4 | If contractor testing is informal, i.e., not contractually deliverable, why is it so explicitly defined in a contractual document. |
| 19. | 4.1 a thru d | Move to Standards and Proceaures Manual. |

## R-DID-118 SOFTWARE TEST PROCEDURES

1. General Comment: An apparent conflict exists between this DID and R-DID-119, Test Report. 119 is geared to an integrated software system while DID 118 addresses module or task level type testing. DID 119 states that the test procedures applicable shall be part of test report. Is one test procedure meant to accommodate all levels of testing or are they oriented to a test level.

| | |
|---|---|
| 2. Paragraph 7 Line 5 | Change: "base" to "basis" |
| 3. Paragraph 2 | Change Title from "Appropriate documents" to "Reference Documents." |
| 4. Paragraph 10 Line 7 | Change "a test procedure" to "test procedures." |
| 5. Paragraph 1.1 Line 4 | Change: "base" to "basis" |
| 6. Paragraph 2.0 Line 2 | Change: "base" to "basis" |
| 7. Paragraph 9.2 | Change title to "Computer Preparation" because "Digital" excludes Hybrid or Analog Computers. |
| 8. Paragraph 10.0 | Change "a detailed procedure" to "detailed procedures." |

## R-DID-119 SOFTWARE TEST REPORT

| | |
|---|---|
| 1. Paragraph 1.1 Line 3 | Change: "test procedure and" "test plan, test procedure, and" to complete the definition of the test being reported |
| 2. Paragraph 1.1 Line 4 | Change: "Also identify" to "Identify" for clarity and to prepare for following commands |
| 3. Paragraph 1.1 Line 5 | Change: "participate" to "participated" |
| 4. Paragraph 1.1 | Add after last sentence: "Identify date and location of test." |
| 5. Paragraph 7.0 Line 2 | Change: "design or operation" to "design, operation, or additional testing" to provide capability to recommend the conduction of additional test to fulfill objectives for which results were not expected. |

R-DID-120 COMPUTER SYSTEM OPERATOR'S MANUAL

| | | |
|---|---|---|
| 1. | Block 7 and 10 (1.1) | Delete "host" |
| 2. | Block 9 | Reference should be made to MIL-STD -XXX being developed by the JLC's under contract that will supersede MIL-STD-1679. MIL-STD-1679 is a Navy document and not recognized by all services under the JLCs. |
| 3. | Block 10 Line 2 | Add "and" at end of line. |
| 4. | 1.2a | Change "by name and nomenclature" to "either by popular trade name or official nomenclature." |
| 5. | 3.1.b.3 | Lowercase bootstrap |
| 6. | 4.1 Line 2 | Delete "...if any..." |
| 7. | General | The manual should be verified. |

## DID 121 - SOFTWARE USERS MANUAL

1. This DID is written for Air-Force use and not joint service JLC use. It specifically references AFSCM/AFLCM 375-7 where it should reference MIL-STD-XXX being developed by the JLC.
2. Paragraph 5, obviously written with a batch/operating system in mind. Should be made flexible enough to cover tactical software and console operations of a complex nature.
3. Block 3, correct spelling software.
4. Correct punctuation throughout.
5. Paragraph 4.a.7., delete "formalized".
6. Add a paragraph to specify system termination procedures.

## DID 122 - COMPUTER SYSTEM DIAGNOSTIC MANUAL

1. The manual should also include a fault catalog listing error messages that are presented to the operator by built in test software. Such a catalog should include instructions for manual test using special equipment or replacement card numbers.

2. Block 3 of the DID says that the Diagnostic Manual is to identify hardware malfunction. This is consistent with the common meaning of a diagnostic manual. However Block 7 says that the manual is based on the SRS. It should be based on the Software Programmer's Manual (R-DID-123).

3. Block 7 states that a Software User's Manual is not required when a diagnostic manual is written. These documents are not related; one shouldn't influence the other. Users and maintenance personnel are usually different people with different responsibilities and background.

4. Paragraph 3. Circled Name/Nomenclature. Question What's the difference - redundant.

### Minority Opinion

There is no need for a separate DID for a Diagnostic Manual. Applying the Software User's Manual DID to the diagnostic CSCI will achieve the same thing.

## DID 123 - SOFTWARE PROGRAMMERS MANUAL

1. Block 7 - meaning of "Interpretive Computer Simulation (ICS)" is questioned.

2. Block 9. This DID is written for AF use and not joint service JLC use. It specifically references AFSCM/AFLCM 375-7 where it should reference MIL-STD-XXX being developed by JLC.

3. Paragraph 1.0 - The scope of subject manual is specified in paragraphs 1.1 and 1.2 conflicts with text contained in Blocks 3 and 7 which appear to require a computer-dependent manual vice a system-dependent manual. Since a given computer may be implemented in many systems it is recommended that subject manual be system-independent. This will tend to preclude the time-honored technique of re-documenting the system-independent material within system dependent documentation. Suggest documentation be structured to prohibit the redocumentation practice. This will permit single-source documentation with a resultant capability of effective configuration control.

4. Paragraph 5.2.C.10 - Inflating terminology should be defined if used (e.g., "supercode").

### MINORITY REPORT

5. The DID for the Software Programmers' Manual is not very useful as it is written, because it is only applicable to the macro programming level. This type of manual will be less and less needed with the increase in use of standard architectures and higher order languages. There is a definite need for a programming manual which is oriented from the microcode level, down to algorithms implemented in discrete logic and driven by PROM date. (AYK-14 is currently dealing with the former case, and LAMPS is dealing with the latter case.) Attached is a proposed Microcode Programmers Reference Manual DID written for the AYK-14. Whether it is better to combine this with the existing SPM DID or leave them separate, I do not know.

PROPOSED

DATA ITEM DESCRIPTION

1. Title:

Microcode Language Reference Manual

3. Description/Purpose

The Microcode Language Reference Manual (MLRM) provides instructions to enable microcode programmers to prepare, interpret, and alter microcode programs. These microcode programs will be written in a particular machine, assembly or compiler language, usually for a specific computer. The manual is based, in part, on the product performance and design specifications which describe the particular machine.

7. Application/Interrelationship

This Data Item Description is used to develop a microcode language reference manual for a specific microcode language being developed where no suitable commercial manual exists. Microcode is machine code which controls the elementary parts of the computer to define the instruction set architecture. The MLRM will be used for evaluating, generating, correcting, or updating one or more microcode programs. This manual together with associated microcode documentation should provide a sufficient basis for microcode maintenance. It is also used to evaluate commercial manuals to ensure they meet Navy requirements.

This Data Item may be used in conjunction with DIDs:

DI-S-2141 Program Package
DI-E-2136 Program Design Specification
DI-E-2138 Program Performance Specification
DI-S-2139 Program Description Document

## 10. Preparation Instructions

The MLRM shall contain sufficient information to enable the microcode programmer to fully understand the microprogramming aspect of the particular machine, assembler or compiler language. The manual shall be applicable to any microcode program and any equipment configuration of the computer in which that microcode program may be utilized. Information presented in the MLRM shall be that necessary to enable the programmer to (1) produce a microcode source program suitable for the particular machine language or assembler/compiler, ;(2) interpret an existing microcode program for that machine language or assembler/compilier, and (3) effect corrections, additions, and deletions in microcode programs. Information shall be in the form of text, supporting illustrations, tables, diagrams, and appendixes.

1. Arrangement. The microcode language referenc manual shall contain the following in the order indicated:

Title Page
List of Effective Pages
Table of Contents
List of Illustrations
List of Tables
Introduction
Chapter 1, General Information
Chapter 2, Programming Features
Chapter 3, Bus Control and Internal Memory Usage
Chapter 4, Program Instruction Statements
Chapter 5, Micro-instruction Timing
Chapter 6, FPLA/PROM Code Generation
Chapter 7, Error Detection/Diagnostic Features
Glossary
Appendixes
Index

2. Introduction. The introduction shall identify the specific microcode language by Government or manufacturer's designation and shall contain a brief statement of the language's use. The scope of the manual shall be outlined briefly by description of the contents of each chapter so that the user can readily locate needed information. In addition, the introduction shall list the publications, Government specifications, etc., needed for clarification of symbols, designations, and abbreviations used in preparation of the manuals.

3. Chapter 1, General Information:

a. Purpose. This portion of chapter 1 shall contain a description of the purpose of the manual and the manner of its intended use.

b. Equipment Organization. This portion of chapter 1 shall consist of a brief description of the specific target computer system, including a general description of the basic modules comprising the equipment. The information may be given in tabular form. A figure shall be provided to illustrate the functional equipment organization. A block diagram of the computer system shall also be included in chapter 1. Examples of basic modules are the following:

(1) ALU
(2) Micromemory
(3) Internal memory and FPLAs other than micromemory
(4) Registers used by microcode other than the internal ALU registers
(5) Busses used or controlled by the microcode
(6) Micro sequencer
(7) External interfaces such as UARTS

c. Operational Structure. This portion of chapter 1 shall consist of a description of the operating characteristics, capabilities and limitations of the target computer system as it relates to the programming function. The information should include:

(1) Machine Cycle Time
(2) Word Length

(3) Memory Capacity and Characteristics
(4) Arithmetic Operations
(5) Interrupt/Event Capabilities
(6) Micro Command Word Format
(7) Micro Command Word Bit Pattern Definitions
(8) Module Intercommunication
(9) Operational Registers
(10) Internal Flags (parity, overflow, timeout, etc.), Flag Registers, and Control Registers
(11) Input-Output Descriptions
(12) Communication With Support Equipment

4. Chapter 2, Programming Features:

a. This chapter shall contain descriptions of the programming features of the particular microcode language. Complete detail should be included if the feature is unique to the equipment, machine language or assembler/compiler involved and could not be expected to be within the scope of knowledge of the experienced programmer. Features described should include:

(1) Type of instructions/statements
(2) Word structure
(3) Command list
(4) Pseudo-instructions
(5) Operand stack
(6) Indexing
(7) Relative direct addressing
(8) Indirect addressing
(9) Branching
(10) Subroutine control
(11) Interrupt/Events

b. This section shall include illustrations of typical instruction word formats and data word formats and flow charts of unusual programming features, as applicable.

5. Chapter 3, Bus Control and Internal Memory Usage

a. Section I, Bus Control. This section shall describe the way in which the microcode controls the computer busses. Detailed descriptions of the use of control registers and flags (set or cleared) shall be provided. These descriptions shall be supported by illustrations and Tables.

b. Section II, Internal memory Usage. This section shall list all internal memory locations for which the microcode has a designated use and indicate their use. This information may be provided in the form of a Table.

6. Chapter 4, Program Instruction Statements:

a. Section I, Microcode Word Format. This section shall describe in detail the word/statement formats used. An illustration of each word/statement format shall be included.

b. Section II, Instruction Description. This section shall describe all the micro-program instructions/statements used with the computing equipment. Examples of instruction and statement/group are:

(1) Fixed-point arithmetic instructions.
(2) Floating point arithmetic.
(3) Shifting operations.
(4) Hardware Control operations.
(5) Index instructions.
(6) Logical operations (if applicable).
(7) Input-output operations.
(8) Branching operations.

The execution of the instructions by the computer shall be described and illustrated as necessary, with flow charts.

This information shall include

(1) Instruction Mnemonic
(2) Description of What The Instruction Does
(3) Micro Command bits set or cleared by the instruction
(4) Special Conditions When This Instruction Can't Be Used

Information provided shall be supported by tables giving descriptions, abbreviations, and codes used in the presentation. Examples illustrating the use of the instruction shall also be provided. This section shall also include a table of instructions with variable cycle time, a table of operational registers and their functions and table of arithmetic properties of registers.

7. Chapter 5, Micro-instruction Timing. This chapter shall indicate the time it takes to execute a complete micro-instruction. Any special conditions which cause this time to be lengthened or shortened will also be stated. This chapter shall also explain all hardware operations not completed during the same micro-instruction in which the operation was initiated (i.e., memory references where the data are not available until halfway through the next micro-instruction). This chapter will also illustrate the timing within a micro-instruction (i.e., if the micro-instruction can perform more than one operation, indicate the order in which the operations are performed).

8. Chapter 6, FPLA and PROM Code. This chapter shall contain information required to enable the programmer to write, understand, and modify the portions of microcode which generate code for FPLAs and PROMs other than the code generated for micromemory. This shall include an explanation of all instructions used that are not defined in chapter 4, illustrations of the format required for these instructions, and examples which will illustrate how to generate FPLA/PROM code. Diagrams and tables shall be added as needed for clarification.

9. Chapter 7, Listing Interpretation. Each error detection/diagnostic feature of the compiler/assembly language will be clearly described. The exact printout provided by the system will be shown with concise descriptions

of what the printout means and what conditions generate the printout. An explanation of any cross reference list and memory maps generated by the compiler/assembler shall be provided.

10. Glossary. The glossary shall list, in alphabetical order, all terms and their accepted meaning as used by programmers, operators, and other personnel associated with the target computer and peripherial equipment. It shall include definitions of registers, components, and characteristics of the particular equipment which the manual supports.

11. Appendices. The appendices shall contain information to support the programmer in the preparation, use, and maintenance of microcode programs. Only those appendices shall be included which are required for the particular computer which the manual supports. Applicable appendices shall be numbered consecutively. The appendices may include:

| | |
|---|---|
| Appendix 1. | Mnemonic listing of operation codes (if applicable). |
| Appendix 2. | Alphabetical listing of operation codes. |
| Appendix 3. | Numerical listing of operation codes. |
| Appendix 4. | Instructions by operating group or micro command subfield. |
| Appendix 5. | Listing of control inputs required to effectively use the machine language or the assembler/compiler. |
| Appendix 6. | Magnetic tape BCD codes (if applicable). |
| Appendix 7. | Input-output typewriter codes (if applicable). |
| Appendix 8. | Flexowriter (or equivalent) codes (if applicable). |
| Appendix 9. | Punched card codes (if applicable) (should include an example of card). |
| Appendix 10. | Papertape codes (if applicable) (should include an example of tape). |

12. Index. If the completed manual will consist of more than 50 pages, an alphabetical index shall be provided.

13. Arrangement and Content:

a. Abbreviations. Standard acronyms and abbreviations may be used provided they are first defined in the text. They shall also be defined in the glossary.

b. Page Numbers and Titles:

(1) Pages, paragraphs, figures, and tables shall be numbered separately and consecutively within each chapter by Arabic numerals.

(2) Chapters, numbered paragraphs, figures, and tables shall have brief descriptive titles. Chapter titles shall be centered horizontally on the page.

c. Space Conservation. Layout shall conserve space without lessening usability or clarity of material. Blank pages and spaces shall be avoided except to meet basic format requirements. For example, the first page of each chapter must always start on a right-hand page, even though this may require a preceding blank page.

d. Blank Page Numbering. All blank pages shall be shown on the succeeding printed page, under the number of that page.

Example: 4-13
(Page 4-12 blank)

e. Number Identification of Manual. Identifying numbers on each manual shall be as specified by the procuring activity and/or user activity.

f. List of Effective Pages. This page shall provide the current status of each page of the manual with regard to its letter code and date of publication. If a manual is to be changed periodically, the list of effective pages shall serve as a control device by specifically identifying the latest changed pages as well as showing the exact page breakdown of the entire manual.

g. Table of Contents. The table of contents shall list chapter titles and numbered paragraph headings. When applicable, a list of illustrations and a list of tables shall follow the table of contents.

h. Collating, Drilling, and Binders. Collating, drilling, and the type of covers shall be as directed by the procuring activity and/or user agency.

i. Style of Writing. The text shall be written in a style that is clearly understandable to the users of the manual. The instructions shall be concise, specific, and clearly worded. Illustrations and tabular data (tables) shall be used as necessary to clarify or supplement the written text.

j. Authorization for Relaxed Format and Reproduction Requirements. Relaxed format and reproduction methods shall be permitted for the handbooks in the interest of economy and expeditious availability. Areas in which requirements may be relaxed are:

(1) Continuous type across page.
(2) Use of standard typewriter to prepare reproduction copy.
(3) Use of office-type reproduction equipment.
(4) Uniform lettering size on final copy not required. However, lettering shall not be smaller than 6-point type.

k. Illustrations and Diagrams. The illustrations and diagrams for users manuals shall be prepared under relaxed format style and shall be used in lieu of halftones whenever possible.

SOFTWARE REQUIREMENTS ANALYSIS REPORT DI-S-3606/S-128-1

DESIGN ANALYSIS REPORT DI-S-3606

These two reports reference different version of the same DID. Only DI-S-3606/S-128-1 is approved for use in the Acquisition Management System and Data Requirements Control List (AMSDL). There are many things wrong with this DID.

1. It is written for the Air Force and references AFSCD 375-7 is block 9.
2. Block references DI-S-3618/S-152 and block 10 paragraph 1.d references DI-S-3605/S-127-1.
3. It does not define the differences in system development being reported in the two reports.

POSITIONAL/STATION HANDBOOK DI-M-3409/H-109-1

This DID is written for Air Force use, not joint service use under JLC. It specifically references AFR 8-2, AFSCM/AFLCM 375-7.

Paragraph 3.b references AFM 35-1.

DI-S-3619/S-153 - TECHNICAL PERFORMANCE MEASUREMENT REPORT

1. The proposed DID is primarily hardware oriented and does not address the software aspects of the development.

   Replace this document with attached Software Development Progress Report (UDI-A-21435).

   Delete the Navy-specific references.

| DATA ITEM DESCRIPTION | 2. IDENTIFICATION NO(S) | |
|---|---|---|
| | AGENCY | NUMBER |
| 1. TITLE<br>Report, Software Development Progress (SDPR) | NAVY-AS | UDI-A-21435 |
| 3. DESCRIPTION/PURPOSE<br>This report presents the status of the software during its development. | 4. APPROVAL DATE<br>20 JAN 1976 | |
| | 5. OFFICE OF PRIMARY RESPONSIBILITY<br>AIR-5331 | |
| | 6. DDC REQUIRED | |
| | 8. APPROVAL LIMITATION | |
| 7. APPLICATION/INTERRELATIONSHIP<br>7.1 Current status is based upon the information provided in the Software Development Management Plan (SDMP).<br>7.2 Do not order this data item unless the Software Development Management Plan, DID UDI-A-21434 is cited on the CDRL, DD Form 1423. | 9. REFERENCES (Mandatory as cited in block 10)<br>SECNAVINST 3560.1 | |
| | MCSL NUMBER(S) | |

10. PREPARATION INSTRUCTIONS

10.1. This report shall provide the status for software designed and documented in accordance with SECNAVINST 3560.1 and shall include as a minimum:

(a) The progress of the milestones defined in the Software Development Management Plan and causes for deviations, if any.

(b) Accounting of functions or threads that are in system analysis, in simulation, in design, in code, in module test, in system test, and in flight test.

(c) Hardware changes and discrepancies that affect processing.

(d) Major difficulties anticipated or encountered and plans to overcome them, including:

(1) Events that are currently behind schedule (or have anticipated schedule changes), their effect on completion of the project, and steps being taken to remedy schedule delays.

(2) Other information which defines cause and effect of significant changes on the contract schedule.

(3) Problems which actually or potentially will cause deviation from contractual requirements.

DD FORM 1 JUN 68 1664 S/N 0102-019-4000 PLATE NO. 19444 PAGE 1 OF 2 PAGES

(e) Revised schedule with projected completion dates.

(f) Manpower incurred versus planned manpower with projected manpower to completion.

(g) Total costs incurred versus planned costs with projected costs to completion.

(h) Core and timing estimates per module versus planned core and timing.

(i) Actual computer usage versus predicted.

10.2 The report shall conform to the following:

(a) All pages, including attachments shall be typewritten or clearly lettered with non-fading ink on standard letter size paper or on a standard size engineering drawing paper.

(b) The first page shall be a title page on which the following data shall be contained, located three inches from the top of the page, and two inches from its unfastened edge:

(1) Type of report, e.g., monthly, interim, final.

(2) Title as indicated on the data item description.

(3) Contract number.

(4) Dates of the reporting period.

(5) Contractor's name.

Other necessary information may be included elsewhere on the title page.

(c) All figures, tables, appendices, attachments, etc., will be identified in the Table of Contents and within the text of the report.

(d) Security classification and distribution limitation markings shall conform to the requirements contain in the contract to which the status report applies.

10.3 Unless otherwise indicated herein, documents cited in this block of the issue in effect on the date of invitation for bids or request for proposals or quotations form a part of this DID to the extent specified herein.

## REPORT OF THE PANEL ON HARDWARE/SOFTWARE/FIRMWARE CONFIGURATION ITEM SELECTION CRITERIA

TAB B
Hardware/Software/Firmware

TRW

18 September 1981

LTC Casper H. Klucas
HQ, AFB/LOEC
Wright-Patterson AFB, OH 45431

To: Panel B Participants of the JLC Software
Workshop - June 1981

Enclosed is the final report of the panel on Hardware/Software/Firmware Configuration Item Selection Criteria. Although a total concensus on several complex issues was not achieved, I feel the panel attacked the problem in a structured manner and have offered the JLC CSM/CRM several specific recommendations and have outlined a course of action for additional effort. I know I speak for Ralph San Antonio and all the panel members in expressing our graditude at being selected to participate in a very interesting and productive workshop.

Please call on us again for continued support.

Sincerely,

[illegible]

R. A. Maher
Chairperson
Panel B

RAM:rd

Attachment

cc: R. Dunn
P. Mauro
T. Schuman
D. Hartwick

## PANEL B

## HARDWARE/SOFTWARE/FIRMWARE

## CONFIGURATION ITEM

## SELECTION CRITERIA

CHAIRPERSON: R.A. MAHER
TRW

CO-CHAIRPERSON: R. SAN ANTONIO
DRC

TABLE OF CONTENTS

## REPORT OF THE PANEL ON HARDWARE/SOFTWARE/FIRMWARE CONFIGURATION ITEM SELECTION CRITERIA

Draft Report — Panel B — Hardware/Software/Firmware Configuration Item Selection Criteria

1. OBJECTIVE: The objective as defined in Appendix C-1, JLC Charter, was to examine the general problem of hardware/software/firmware CI/CPCI partitioning and specification and develop a set of criteria to aid in the selection and documentation process. The problem has become exceptionally complex for microcomputers and computer software embedded in reprogrammable hardware (firmware). The difficulty lies both in defining the "hardware intensive" vs "software intensive" nature and also in the style and complexity of management control documentation. A secondary objective therefore was to recommend an approach for defining firmware/software categories and support documentation requirements.

2.0 SCOPE: Reprogrammable CIs have a varying scale of software content that range from simple ROM/PROM devices, utilized as design solutions, never intended to be changed or have attendant support system requirements; to software intensive systems, requiring full CPCI treatment and very complex support systems, tools and maintenance CI/CPCIs. The current CI/CPCI allocation process does not recognize the "hardware intensive" or "software intensive" nature of firmware CIs nor is adequate criteria available to guide in the CPCI selection process or scope the required support documentation and data. The panel attacked the problem in the following manner:

a. Conduct a top down analysis of technical and management considerations important in the selection of systems hardware, software and firmware components.

b. Establish technical, programmable and management guidelines/criteria for CI/CPCI selection and treatment.

c. Test the criteria against representative hardware/firmware/software architecture for adequacy and clarity.

d. Define sensible categories of reprogrammable CIs for treatment of their software nature as less than full CPCIs.

e. Review recommended firmware DIDs and define documentation requirements for the different reprogrammable CI categories.

The panel effort was scoped by first establishing an initial cut at tasks a. and d. and utilizing the remaining time for related tasks b., c. and e. considerations.

3. <u>APPROACH</u>: The panel conducted a general review session of selected reference material from the bibliography. Several of the references have been included in the Appendices for completeness of this report. Each panel member discussed CI/CPCI issues, problems and important considerations from their technical/management experience point of view. Several action items and questions were generated and grouped together as best matched the five task areas defined in Section 2.0 (Scope). Two subpanels were established in which one group (subpanel B-2) would attack the basic selection criteria problem (Tasks a., b., and c.), while another subpanel (B-1) would jump in the middle of the "what do-I-do-now?" problem of firmware definition and documentation (Tasks d., and e.).

4. <u>DISCUSSION</u>: Sub Panel B-1 divided firmware into four general categories and reviewed the impact to MIL-STDs, DIDs and CPCI selection. References used included recent EIA findings and recommendations (Reference 8). Dr. Sylvesters (USAF/ASD) Hardware Intensive Treatment white papers (Reference 9-11) and four candidate sets of firmware DIDs (Reference 4-8). It was concluded that reprogrammable CIs could be sensibly divided into four general categories with CPCI impact as follows:

Category #1 - <u>Software Intensive</u> - Full CPCI and support system (Documentation tailoring as necessary)

Category #2 - <u>Moderate Software Attributes</u> - Non-complex specification treatment for definition and documentation. Adequate CPCI and support system

Category #3 - <u>Hardware Intensive -</u> CI with Product Specification - identification and new firmware DID. Partial software support system

Category #4 - <u>Slightly Reprogrammable</u> - CI with reprogramming data and attributes included in DoD D100/D1000 drawing release. Some support system definition.

Recommendations for a new DID or DID application matrix were made to Panel A in a working session.

Subpanel B-2 reviewed the DRC recommended selection criteria (Reference 14) and several of the source material references as a starting point initiated systematic process of issue statement, criteria identification decision flow development and test criteria application. The available time did not allow completion of the full sequence, but a baseline set of selection criteria were established as an excellent continuation point for this effort.

Section 4.1

DRAFT

FINAL REPORT

SUB-PANEL B-1

# FIRMWARE DOCUMENTATION AND DOCUMENTATION

SUB-PANEL CHAIRMAN: A. MAHER
SINGER-KEARFOTT DIVISION

RECORDER: J.L. RAVELING
SPERRY UNIVAC

## 4.1 SUB-PANEL B-1, FIRMWARE DOCUMENTATION

### 4.1.1 Background

We are experiencing an explosion of microprocessor technology that will yield thousandfold improvements in computer resource's cost, size, and performance during each of the next two decades. Microprocessors are proliferating at an exponential rate, and they are performing tasks never before imagined for military systems. This unprecedented expansion of computation embedded in military systems has created an entirely new set of management concerns for today's military managers and defense contractors.

In early applications of microtechnology, the software aspects of the application often were not visible to the military program manager. Concerns were expressed over real and perceived problems in managing this new technology area. Do you manage it as hardware, or software, or both? We found ourselves caught up in a debate on definitions. What is a "microprocessor", "microprogram", "microcomputer", "microcode", or "firmware"? Something had to be done — but what?

The military services determined that the most significant problem lay in establishing a mechanism for the control of the software aspects, i.e., firmware, of the microtechnology area. As an interim measure, the Military Services determined that firmware should be handled as software and the microprocessors and microcomputers would be controlled as hardware items.

While this solution did solve the immediate management and control problem, it also introduced a potential for significant growth in acquisition costs. If all firmware was to be handled as software, then must it not also be managed as a Computer Program Configuration Item (CPCI)? As a CPCI, the full rigors of a structured development process would apply: requirements and product specifications, design reviews and audits, development and management plans, and test plans and procedures. Further use of an approved higher order language in developing the CPCI would be required, data rights to the software would be acquired, and the support tools required to develop and test the firmware would be deliverable. For many applications of firmware,

4.1.1 (Continued)

implementation of the described requirements would clearly be a case of "over-kill".

The Air Force Systems Command, under the guidance of Dr. Richard J. Sylvester, developed, in 1979, a draft policy which described either a "hardware intensive" or "software intensive" classification for firmware (ref. 9 and 10). The intent of this policy was to allow the identification and description of the hardware intensive firmware as part of the hardware documentation hierarchy, and to manage and control hardware intensive firmware as part of the hardware; thereby avoiding the costs inherent in designating an item as a CPCI.* At the same time, provisions were made to designate an item as being software intensive, and thus subject to all the management and control provisions inherent to the CPCI designation. The hardware or software intensive classification seemed to offer a possible solution to the problem of cost effective management and control of firmware.

A problem, of course exists in defining what comprises a hardware intensive versus a software intensive application. Technology is moving so fast, it is questionable whetheranyone could, in fact, define a set of classification which would remain valid for an extended period. Dr. Sylvester's most recent report on this subject (ref. 11) does provide a good discussion of cases where the distinction can be made. However, many other cases are less clear. Figure 1 diagrams the "fuzziness" which occurs when attempting to determine what classification to assign to firmware. Conceivably, all microprocessor/firmware applications falling to the left of point "a" might be excluded from most software management requirements, and all applications to the right of point "c" would be subject to the normal software control practices. Applications falling between "a" and "c" would be considered a case-by-case basis (ref. 12).

*Sylvester Report (ref. 11) page 3, "For hardware intensive applications, policy . . . does not exclude in the imposition of full MIL-STD documentation..."

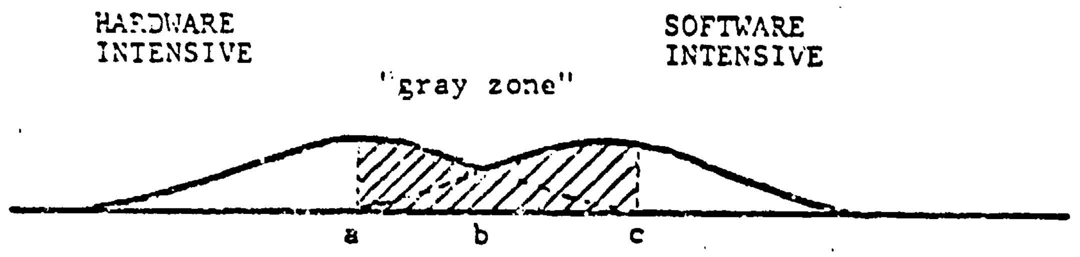


Figure 1. Firmware Categories for Management - The "Gray Zone"

4.1.2 Sub-Panel Charge

Against the preceding background, the basic problem assigned to our sub-panel was to determine what methods currently exist, or must be developed, to establish required management and control of firmware. Both hardware intensive and software intensive firmware was to be considered.

4.1.3 Sub-Panel Deliberation Summary

a. Assumptions

The sub-panel began their deliberations using the following basic assumptions, or precepts:

- All firmware has both software and hardware aspects.
- The development, release and maintenance of firmware must be controlled following basic project management and configuration management practices.
- While the charter which must be developed to determine whether firmware should be classified as hardware or software intensive is extremely important, it was neither within the purview of the sub-panel to establish this criteria nor was it required to accomplish our purpose.

4.1.3 (Continued)

- Maximum advantage should be taken of existing, or proposed specifications and/or Data Item Descriptions (DIDs) in developing recommendations for firmware identification and documentation.

b. General Discussion

All government panel participants and the Mitre representative expressed significant concern over the management of firmware, and the need to provide increased visibility into its (firmware) design, development, test, operation, and maintenance. Each government representative was able to provide examples of current or past projects wherein lack of effective control of firmware or related support tools had led to, or was leading to, difficulties in its management and control. Major emphasis was placed during these discussions on the need for providing improved visibility and control of the software aspects of firmware. However, it was also agreed that designation of all firmware as CPCI's was not required for effective management, nor was the CPCI designation always defensible from a cost effectiveness standpoint. Judicious use of the hardware and software intensive classification for firmware was generally endorsed as a means of effecting the proper level of management visibility and control.

Concern was also expressed on the need to ensure that the appropriate degree of firmware support tools (those tools required to develop, test, enhance, and maintain the firmware) be acquired as part of the basic procurement. The level of support tools required is closely oriented to the planned application of the firmware and the planned maintenance concept.

While the discussion of support tools, or environment, is closely allied with the firmware management and control problem addressed by the sub-panel, it was eventually agreed that the subject was not specifically germane to our chapter.

4.1.3 (Continued)

The panel's industry representatives described their current policies and practices related to firmware. Systems have evolved which can, perhaps, be best described as being of a hybrid nature, e.g., a combination of hardware and software identification and control criteria and processes. These policies and practices have been developed in response to the perceived need by industry to properly identify and control their microtechnology products . . . both hardware and software. Although these practices have been effective for the firms involved, there are perceived weaknesses in the process:

- Each firm has established systems which are unique unto themselves,
- the customer has no, or limited, visibility into these internally developed and implemented policies and practices, and
- no convenient, or formal, contractual vehicle exists for the government to request the firmware data.

There was considerable discussion of the adviseability of treating the documentation of the software aspects of firmware in the same fashion as conventional software, without the stipulation that it be designated as a CPCI. If this path were pursued, the minimum documentation for the software aspects of firmware would have to be established well below the full CPCI level. The concept of a tailored CPCI was introduced during these discussions to designate those diminished documentation requirements. The concept was finally dropped by the panel in favor of the recommendations in paragraph 4.1.4 but a minority report has been included in paragraph 4.1.5 to record (and, hopefully, more clearly identify) the issue on which so much fruitful discussion was based.

The panel also considered the adviseability of imposing directives on the use of HOL's for firmware and on the use of specific computer Instruction Set Architectures (ISA's). The unanimous conclusion was that the use of HOL's for firmware should be

4.1.3 (Continued)

encouraged but should not be made mandatory by directives, especially for hardware intensive applications. That is, the use of specific HOL's and/or the use of HOL's in general should not be edicted. This conclusion was based on the conviction that such edicts would often create costs which are not recovered in the life cycle. Where the use of HOL's is, in fact, practical - the designer will likely elect to use the HOL since that will simplify his/her task. A similar conclusion was reached with respect to ISA's, although it was not unanimous in that case.

c. The Problem

The sub-panel's deliberations pointed to the need to establish an efficient and effective method of identifying both hardware and software intensive firmware. Once identified, there is a need to tailor the acquisition process to accommodate the unique characteristics of firmware.

The sub-panel further expressed the opinion that current DoD military standards, and related Data Item Descriptions (DIDs), do not adequately address firmware management and control. Shortcomings identified in the MIL-STDs/DIDs were:

- MIL-STD-483/490: Provisions for inclusion of the software elements of firmware, or for providing traceability to these elements, is not provided in the Type A, B, and C hardware specification detailed requirements appendices. The reverse situation is true for the hardware elements of firmware in the related software specification requirements appendices. Such provisions are required to support the respective classification of firmware as hardware or software intensive.

- MIL-STD-1679: Provisions for inclusion of the hardware elements of firmware, or for providing traceability to these elements, is not provided in the documentation DIDs related to this MIL-STD. These provisions are required to support the classification of firmware as software intensive.

4.1.3 (Continued)

Similar findings would also apply to the documentation formats found in DOD-7935.1 S.

Since the above referenced specifications are key to establishing the configuration identification of all CI/CPCIs, the sub-panel believed it was critical that these MIL-STDs/DIDs be modified to effectively accommodate firmware. Good configuration management practices dictate that whenever feasible, a single configuration identification (CI or CPCI) be applied to firmware.

d. Proposed Documentation

During the past several years, the military Services, industry, and many of the industry associations have devoted significant time and energy in discussing the firmware issue, and in developing documentation schemes for firmware. Our sub-panel reviewed several specific documents which have been developed as part of this overall effort, and which appeared to provide a possible solution to the problems with the existing MIL-STDs/DIDs cited in c. above.

(1) Panel D, "Tailoring CM Practices for Microprocessors and Firmware", Electronics Industries Association (EIA), Report of the Fourteenth Annual Data and Configuration Management Workshop; October 20-24, 1980

The report prepared by Panel D at the 1980 EIA Workshop reviewed configuration management requirements for microprocessors and firmware. Two recommendations made by this panel, which were of specific interest to our sub-panel:

- Define a new element of a hardware Configuration Item (CI) entitled a Computer Program/Hardware (CP/H). The CP/H is microprocessor firmware which is identified and controlled as a portion of a hardware configuration item. (Appendices C-3 and C-5.)

4.1.3 (Continued)

- Prepare a Data Item Description (DID) for the Computer Program/Hardware (CP/H). The DID would be in the format of a Book Form drawing and would be called for as a CDRL. The DID would reference DOD-STD-100 and DOD-D-1000. See Attachment 1 for detail.

(2) UDI-E-3935, "Firmware Development Plan"; September 14, 1979.

This proposed DID provides format and content guidance for a Firmware Development Plan (FDP). The FDP is the engineering management plan which identifies the contractor's efforts required to develop and deliver computer resources and the associated firmware, processes, documentation, and necessary support resources. See Appendix C-6 for detail. The plan would be appropriate when a procurement does not require a Software Development Plan/Computer Program Development Plan to be developed.

(3) UDI-E-3936 — ASD, "Firmware Technical Description (Product Specification)"; March 30, 1979.

This proposed DID provides a complete and detailed technical description of each functional implementation of firmware. The document(s) serve as an instrument for acceptance, modification, trouble diagnosis, maintenance, and reprocurement of firmware. See Appendix C-7 for detail.

(4) UDI-E-3937 — ASD, "Firmware Support Data"; March 9, 1979.

This proposed DID provides a description of the data required for the maintenance and modification of firmware ROMs (Read Only Memories) and PROMs (Programmable Read Only Memories), microprocessors, and other firmware devices. See Appendix C-8 for detail.

(5) Draft AFSC/ASD paper, "Hardware Intensive Application of Computer Resources", Dr. Richard J. Sylvester; June 4, 1981 (Reference #11).

4.1.3 (Continued)

This paper provides a full discussion of hardware intensive firmware: the advantages and disadvantages which are inherent in the classification, criteria for selection of the classification, documentation required, testing, and SOW guidance. Further, a draft DID, "Configuration Item Product Fabrication Specification" was provided. See Appendix C-9 for DID detail.

### 4.1.4 Sub-Panel Recommendations

The sub-panel recommended the following actions be considered by the Joint Logistics Commanders, Computer Resources Management Committee, and Computer Software Management Sub-group:

a. Consideration be given to the adoption of the 1980 EIA Panel D recommendations for identification of a Computer Program/Hardware (CP/H) component of a hardware Configuration Item (CI). A related recommendation is for the publication of a DID referenced to DOD-STD-100-DOD-D-1000, describing the Book Form drawing which encompasses the software elements of firmware (see Appendix C-5).

   The documentation provided under this DID would serve to satisfy required configuration identification data for selected current and future perceived hardware intensive applications of firmware. Selection of this DID would be appropriate when minimal visibility into the software aspects of firmware is required.

b. Consideration be given to the adoption of the Configuration Item Product Fabrication Specification DID found in Dr. Sylvester's draft paper on "Hardware Intensive Application of Computer Resources" (See Appendix C-9).

   The documentation provided under this DID would serve to satisfy required configuration identification data for selected current and future perceived hardware intensive applications of firmware. Selection of this DID would be appropriate when greater detail and visibility into the software aspects of firmware is required.

4.1.4 (Continued)

c. Consideration be given to the adoption of the Firmware Technical Description (Product Specification) DID developed by AFSC/ASD. This DID should be published under MIL-STD-483 and/or 490 (see Appendix C-7).

   The documentation provided under this DID would serve to satisfy required configuration identification data for selected current and future software intensive applications of firmware. Selection of this DID, which would define what the sub-panel referred to as a "tailored CPCI", would be appropriate when it is deemed feasible to: combine the performance, detailed design requirements, test and evaluation requirements, and hardware aspects of firmware in one specification.

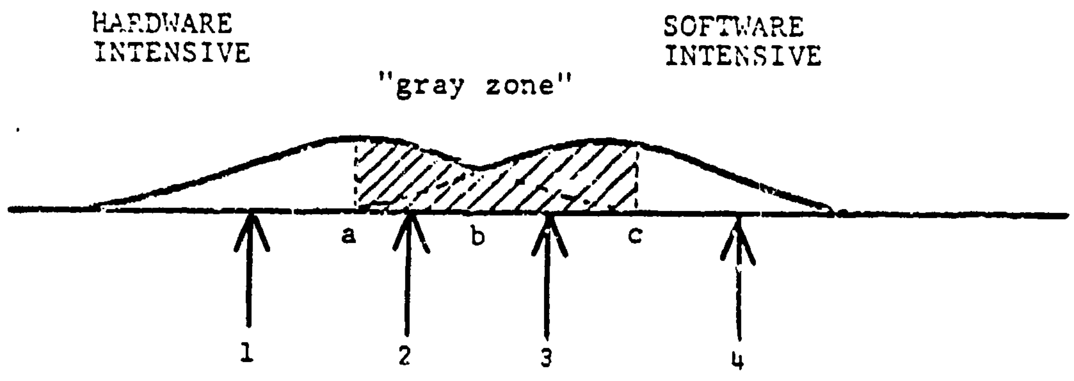
d. Consideration be given to the issue of a DID referenced to MIL-STD-483, 490, and 1679, which would modify the software specification formats contained, or referenced therein, to include data in the form of an appendices, on the hardware aspects of a software intensive firmware application. The appendices would be similar in intent to that which has been provided in Appendix C-9, except it would be for hardware versus software.

   The documentation provided under this DID would serve to satisfy required configuration identification data for those firmware applications determined to be fully software intensive.

e. The preceding recommendation, since they are all keyed to existing MIL-STD's, should be considered for implementation as soon as practicable.

   To ensure a longer term solution to the firmware documentation problem, it is requested that the provisions of the documentation changes requested in a. through d. above be considered by the members of Panel A when defining their final recommended DID list.

   To provide an overview of our recommendation for firmware documentation, refer to Figure 2.



1. - AFSC/ASD CI Product Fabrication Specification (paragraph 4.1.5b)
2. EIA Book Form Drawing for CP/H (Paragraph 4.1.5a)
3. AFSC/ASD Firmware Technical Description (paragraph 4.1.5c)
4. Sub-Panel recommended modification to MIL-STD-483, 490, and 1679 (Paragraph 4.1.5d)

Figure 2. Recommended Firmware Documentation Scheme

4.1.4 (Continued)

NOTE: The subpanel did not have sufficient time to review the recommended DIDs in detail (Appendices C-6, C-7 and C-8), therefore, a full review of the DIDs should be accomplished to insure appropriateness and correctness for the recommended application.

f. It is further requested that Panel A's consideration be extended to the DIDs found in Appendix C-6 (Firmware Development Plan), and 4 (Firmware Support Data). This consideration should ensure that the data encompassed in thse DIDs has been provided for in their final recommeded DID list. It is our understanding that Appendices C-6. C-7 and C-4 are intended to be utilized as a package.

g. Do NOT impose HOL direction on firmware developed (especially hardware intensive firmware) but do encourage its use.

4.1.4 (Continued)

h. Do NOT impose specific ISA direction on firmware controlled computers (especially for hardware intensive applications) but do encourage use of common ISA's.

## 4.1.5 Minority Viewpoint

When considered from the viewpoint of the software practitioner, there would appear to be no difference between conventional software and firmware, except for the requirement to make a hardware change whenever the associated firmware program is changed. Why then shouldn't the existing software documentation technology be perfectly adequate for covering the software aspects of firmware? After much discussion, the sub-panel decided not to pursue this approach but rather to support efforts to define special documents to contain coverage of the software aspects of firmware (see recommendations above). While the final vote on this issue was clear-cut, the issue merits recording here since it does not receive explicit treatment in the current literature and since it is likely to elicit support from the software community if the spectre of "full CPCI" treatment can be disspelled.

In the fourth subparagraph of 4.1.1, the reasons for avoiding the use of "conventional software documentation" for firmware are well presented. We are all familiar with attempts to apply full CPCI treatment to a small firmware program with little prognosis for change during its life cycle. This certainly is overkill but it does not directly follow that a more suitable approach cannot be found within the software documentation technology. In fact, the existing MIL-STD's permit the tailoring of the CPCI concept to be a subset of "full CPCI" treatment. However, this approach is not often used in practice – perhaps due to ignorance of the existence of such an option.

Many firmware programs are developed by design engineers with a dominantly hardware background. Their natural reluctance to get involved in the apparent maze of "software documentation" is another reason for avoiding this approach. However, if one believes that there is no

4.1.5 (Continued)

technological difference between a firmware program and a software program of comparable complexity, then avoiding the existing development and documentation technology for software will doom these new firmware designers to relearn the lessons which were painfully taught to today's software practitioners over many years of evolution and refinement. It would be more effective to introduce them to the least complex forms of software documentation, which will likely prove adequate for most of their work.

From this perspective, the identification of a plethora of new DIDs (as recommended in 4.1.4 and elsewhere) to cover the software aspects of firmware is redundant, wasteful, and is likely to be ineffective in many cases. To illustrate the point, suppose two engineers develop the identical program. But one is implemented in Read Only Memory (ROM) and is considered firmware, while the other is implemented in a writeable memory and is considered software. Why should these programs be documented in two different ways? Their software design aspects clearly call for uniform treatment! The plethora of firmware-unique DIDs would force such disparity.

To pursue the thesis: uniform documentation for programs of comparable complexity — it would be necessary to clarify the available options for handling the case of minimum complexity. A minimum documentation package for such a case might be defined to be a Version Description Document and a Listing Document (an approach which has been successfully used by at least one manufacturer). Sub-Panel #1 elected not to pursue this path so no further specific recommendations are available as a result of their deliberations.

Section 4.2

DRAFT

FINAL REPORT

SUB-PANEL B-2

# CPCI SELECTION CRITERIA AND EVALUATION

SUB-PANEL CHAIRMAN & RECORDER: R. SAN ANTONIO
DRC

## 4.2 SUBPANEL B-2: CI/CPCI SELECTION CRITERIA

4.2.1 Objective. The objective of Subpanel B-2 was to identify the technical and management needs that should be considered in selecting the configuration items from the hardware, software and firmware components of the system.

4.2.2 Scope. This effort was limited to developing criteria which can be used by a program manager to select the minimum set of CIs/CPCIs for a given system considering management, technical, and programmatic issues. These criteria, along with appropriate guidelines, will assist the program manager in deciding how the individual components of a system should be managed. Outside the scope of this panel were the tradeoffs which can be made between the management visibility and control provided by a given CI/CPCI structure and the attendant cost. (Subpanel B-1 dealt with the specific issue of how these tradeoffs should be made for the firmware components of a system.)

4.2.3 Approach. The approach followed by Subpanel B-2 was to: first, state the issues; second, identify the list of appropriate criteria and subcriteria; third, develop guidelines (decision flows, text) for applying the criteria; and, last, test the criteria, subcriteria, and guidelines against some real-world examples.

4.2.4 Discussion. CI/CPCI designation provides a level of management visibility and control identified as follows:

- Formal configuration identification
- Development/product specifications
- Government approval of changes
- Configuration status accounting records
- Individual design review activity
- Individual qualification testing
- Individual physical and functional configuration audits
- Direct ECP handling
- Separate identification, marking
- Separate operating and user manuals
- Standardization requirements.

The "cost" associated with a given CI/CPCI structure is measured not only in dollars, but also in the administration burden, personnel resources, and complexity introduced into a program by the above requirements.

Historically, there has been very little guidance available to acquisition managers for selecting the hardware, software, and firmware configuration items of a system. For many years this lack of guidance resulted in very little visibility and control over the software elements of the system as the software was treated as data (a disposable commodity) procured in conjunction with the hardware elements of the system. This period was followed by one in which there was a tendency to overspecify the number of CPCIs (to maximize visibility and control), which resulted in excessive cost and administration burden, increased program complexity, and a drain on persennel resources. The net result was to decrease visibility and control over the software.

A major step forward was achieved with the addition of Appendix XVII to MIL-STD-483. This appendix identifies the importance of and provides guidelines for proper CI/CPCI selection: it was the starting point for the work undertaken by this subpanel. Other reference documents reviewed by the subpanel are identified in Table 4.2-2. These references were used to develop the "strawman" list of CI/CPCI selection criteria contained in Table 4.2-1.

Subpanel B-2 identified the following issues related to CI/CPCI selection criteria:

- For a given system, what are appropriate criteria for selecting the configuration items from the hardware, software, and firmware components of the system? or
- When is an item important enough to be considered a configuration item?

To address these issues, the subpanel developed subcriteria and guidelines for each of the criteria listed in Table 4.2-2. (See Appendix C-4.) These are contained in the following sections. In practice, the acquisition manager would apply each criterion to the program element under consideration, weigh the relative importance between criteria, and decide if the program element merits treatment as a CI/CPCI, "tailored" CI/CPCI, or as a component of one of the above!

Table 4.2-1. Panel B Reference Material

| | | |
|---|---|---|
| MIL-STD-483 | March 21, 1979 | Configuration management practices for systems, equipment, munitions, and computer programs |
| ESD-TR-77-254 | August 1977 | An Air Force guide to computer program configuration management |
| AFSCP 800-7 | December 1, 1977 | Configuration management |
| UDI-E-3935 | September 14, 1979 | Firmware development plan data item description (DID) |
| UDI-E-3936 | March 30, 1979 | Firmware technical description (product specification) DID |
| UDI-E-3937 | March 9, 1979 | Firmware support data DID |
| EIA G-33 Panel D Report | October 1980 | Tailoring CM practices for microprocessors and firmware |

Table 4.2-2. CI/CPCI Selection Criteria

Criticality

Function

Maintenance Concept

Supplier

Interfaces

Use

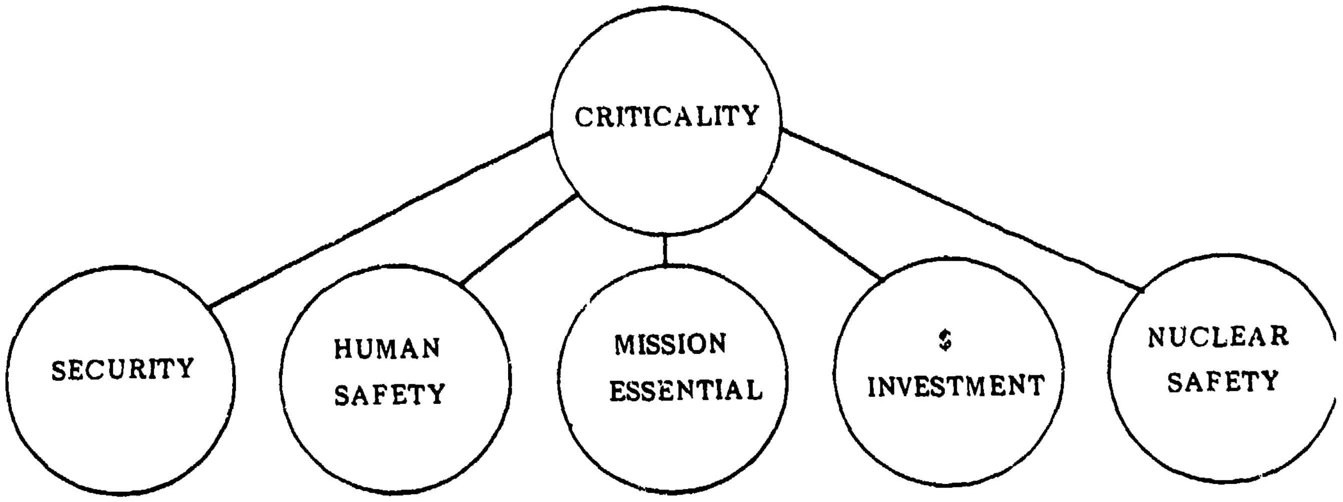
Location

Size

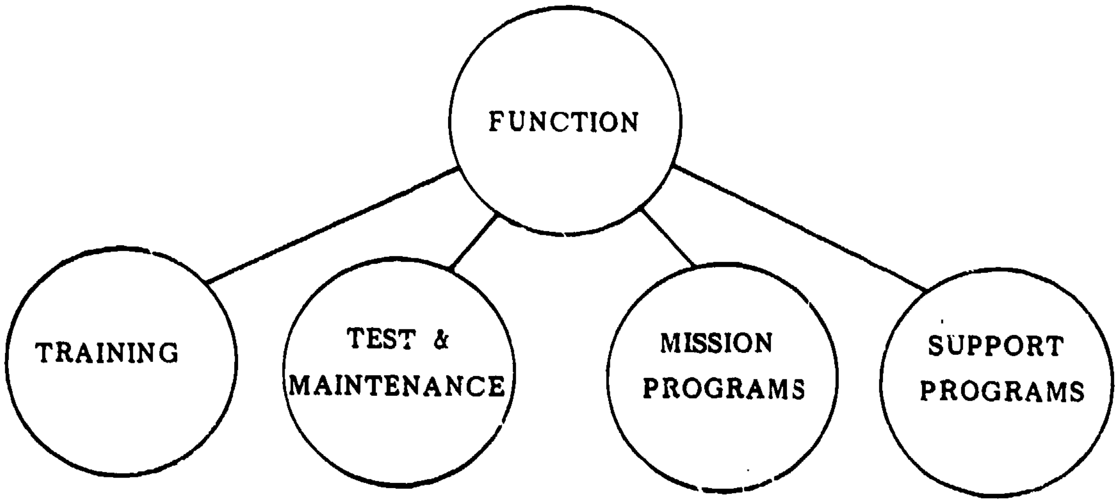
Schedule/Phasing

4.2.4.1 CI/CPCI Selection Criteria. The information contained in this section is intended to assist program managers in deciding how the individual components of a system should be managed. Detailed guidelines for the following criteria are provided in order: location, criticality, function, maintenance concept, use, supplier, interfaces, size, and program schedule/phasing. To use this information, the program manager should apply each criterion to all program elements under consideration, weigh the relative importance between criteria, and decide if each element merits treatment as a CI/CPCI, "tailored" CI/CPCI, or as a component of one of the above.

4.2.4.1.1 Criticality. If failure of the element would adversely affect security, human safety, the accomplishment of a mission, or nuclear safety, or would have a significant financial impact, strongly consider identifying the element as a separate CI/CPCI.

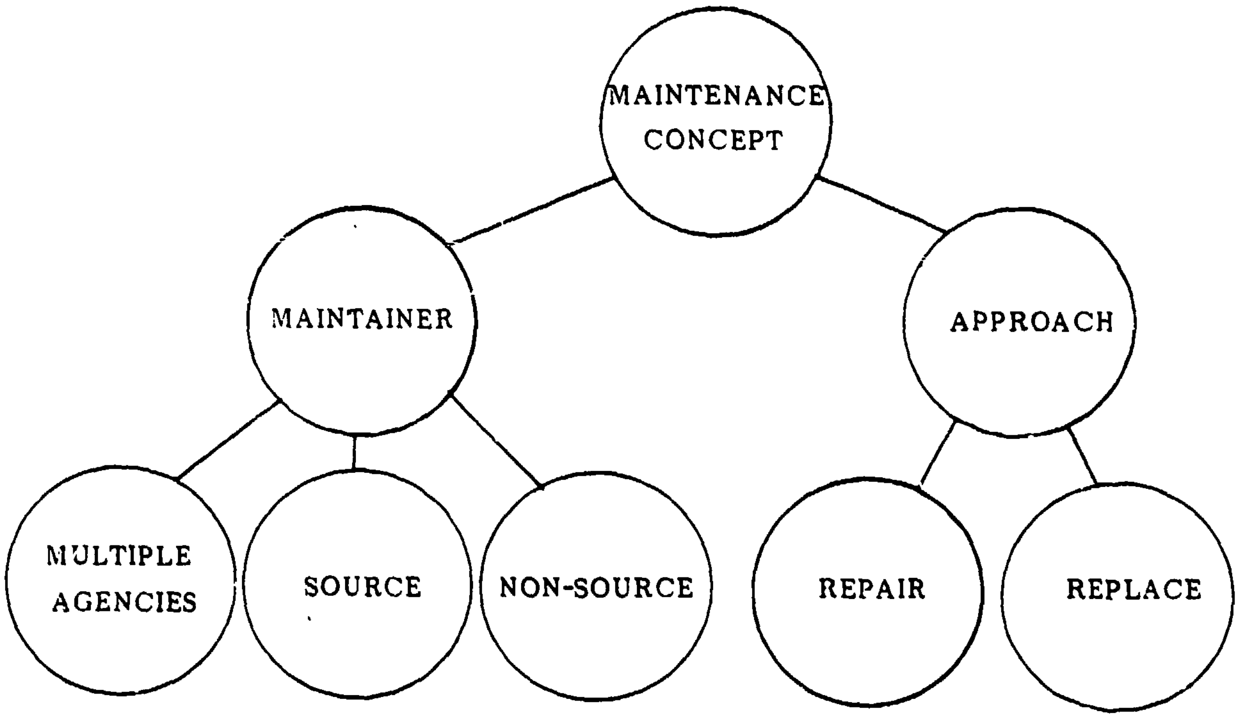


4.2.4.1.2 Function. The program manager should avoid mixing training, mission (including initialization, normal operation, and back-up or degraded operation), test and maintenance, and support functions within a given CI/ CPCI.

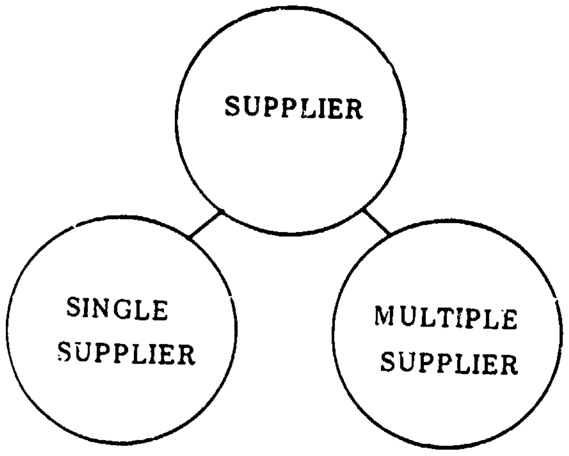


4.2.4.1.3 Maintenance Concept. When different agencies have responsibility for maintaining parts of an element, the program manager should consider breaking the element into separate CI/CPCIs.

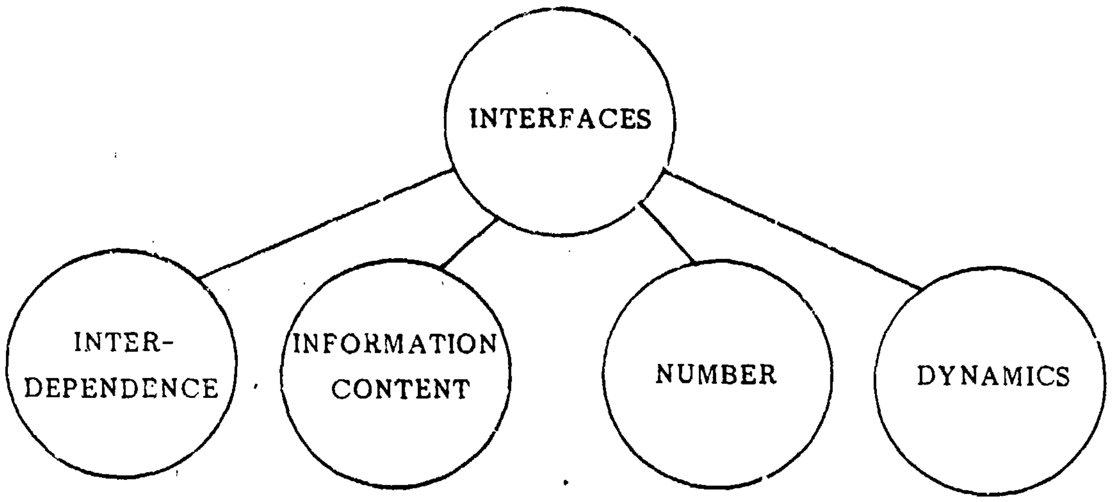
When the source of an element will maintain it, or when it is intended that the element be replaced rather than repaired, the program manager can safely consider tailoring the CI/CPCI or including the element as part of a larger CI/CPCI. Additionally, the support tools need not be identified as CI/CPCI to be delivered.



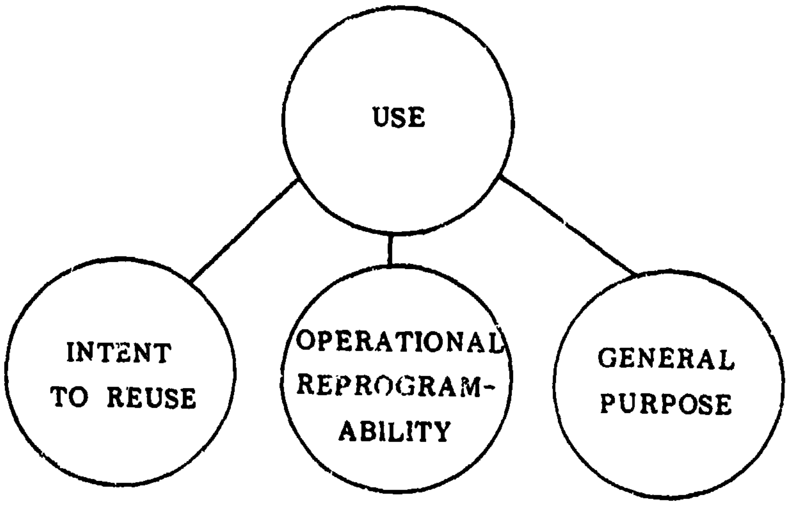
4.2.4.1.4 Supplier. Elements provided by different suppliers should be assigned to separate CI/CPCIs.



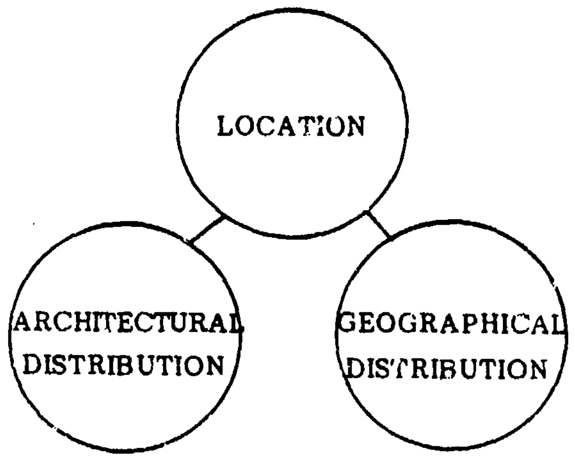
4.2.4.1.5 Interfaces. Interfaces among CIs and CPCIs should be simple. Functions which are highly data or control interdependent should be allocated to the same CI/CPCI. Functions which exhibit a high disparity between input and output data rates should be allocated to separate CI/CPCIs.



4.2.4.1.6 Use. Elements which are general purpose in nature, require the capability to be operationally reprogrammed, or are intended to be reused in another system should be considered as separate CI/CPCIs.



4.2.4.1.7 Location. The functions allocated to a CI/CPCI should not be partitioned among separate geographic areas. Functions allocated to physically distinct processors in a distributed environment should be considered as separate CI/CPCIs.



4.2.4.1.8 Size. CI/CPCI selections which cannot be made on the basis of other criteria should be made to keep the CI/CPCI to manageable proportions.

4.2.4.1.9 Schedule/Phasing. Elements scheduled for development, testing, and delivery at different times should be assigned to separate CIs/CPCIs.

4.2.4.1.10 Criteria Development Status. Although the subpanel textual descriptions of how the criteria and subcriteria should be applied to a given program element, there was insufficient time to develop decision flow diagrams which would lead the acquisition manager through the CI/CPCI selection process. In addition, the subpanel was not able to vigorously test the criteria, subcriteria, and guidelines against actual system configurations.

4.2.5 Recommendations. Subpanel B-2 recommends that the JLC-CSM subgroup, support continued work in this area to accomplish the following:

1. Expand and refine selection criteria and accompanying guidance
2. Test criteria and guidance against actual system configurations.
3. Document expanded guidance in existing and planned acquisition management guidebooks.
4. Revise MIL-STDs to include new requirements as appropriate.
5. Coordinate the effort in recommendations closely with the on-going JLC documentation and standards development activities.

## 5.0 Panel B Recommendations

The following are general recommendations based on Panel B review of the C/1/CP/C1 selections criteria problem. Detailed, specic recommendations are contained in Subpanel B-1 and B-2 reports (Silicons 4.1.4 and 4.2.5).

1. The JLC should recognize and adopt a policy of reprogrammable C.I. categories and take appropriate action to revise MIL-STD-490, 483, 480 and DOD-STD-100/D-1000 necessary.

2. JLC obtain a contractive effort to refine the firmware treatment categories further develop selection criteria; establish test criteria, cases and test architectures; and perform allocating during system aquisitions activities.

3. Panel A and Panel B meet as a follow-on activity to jointly review and develop a matrix DID firmware appliciability matrix and/or review proposed firmware DIDs (Appendix C-6 thru C-9).

ATTACHMENTS

A. Participants:

Chairperson — R. Maher — TRW

Co-Chairperson — R. San Antonio -- DRC

| SUB-PANEL B-1 | SUB-PANEL B-2 |
|---|---|
| Chairman: A. Maher - Singer-Kearfott | Chairman: R. San Antonio* - DRC |
| R.F. Anderson - USAF/AFLC | M. Levin - USAF/AFSC |
| J. Medea - USA/ERADCOM | W. Grimshaw - USAF/AFLC |
| V. Skullman - USN/NASC | R. Calland - USN/NOSC |
| G. Bargero - USAF/AFSC | N. Taupeka - USA/CECOM |
| M. Reece - MITRE | R. Marchand - USA/ERADCOM |
| J. Raveling* - Sperry Univac | |

*Responsible for subpanel report preparation.

B. Bibliography

1. AFSCP 800-7, "Configuration Management of Computer Programs," Ch. 6, dtd: 1 September 1977.

2. MIL-STD-483 (USAF), "Guide for Selecting Configuration Items (CIs)," Appendix XVII, dtd: 21 March 1979.

3. TRW Report No. 30323-6018-TU-00, "Airborne Systems Software Acquisition Engineering Guidebook for Microprocessors and Firmware," dtd: February 1980.

4. DD-1664, Proposed DID "Firmware Development Plan (FDP)," dtd: 14 September 1979. (UDI-E-3935 ASD/M2.)

5. DO1664 Proposed DID, "Firmware Technical Description," dtd: 30 March 1979. (UDI-E-3936 ASD.)

6. DD 1664 Proposed DID, "Firmware Support Data," dtd: 9 March 1979. (UDI-E-3937 ASD.)

7. DD 1664 Proosed DID, "Configuration Item Product Fabrication Specification," dtd: (draft).

8. Electronics Industry Association (EIA) Panel write-up, "Tailoring CM Practices for Microprocessors and Firmware," Panel D.

9. Sylvester, Dr. R.J., "White Paper on AFSC Microprocessor Policy," 1 May 1979. Air Force Systems Command, Aeronautical Systems Division, Wright-Patterson AFB, OH.

10. Sylvester, Dr. R.J., "Revised White Paper on AFSC Microprocessor Policy," 18 January 1980. Air Force Systems Command, Aeronautical Systems Division, Wright-Patterson AFB, OH.

11. Hardware Intensive Application of Computer Resources (DRAFT), Richard J. Sylvester, dtd: 4 June 1981.

12. Herrelko, Capt. D.A., "Managing Firmware for Weapons Systems — An Air Force View," 14 January 1980. Proceedings of the IEEE Society Workshop in Microprocessors in Military and Industrial Systems.

13. Extract from AF/ESD Guidebook on Computer Program Configuration Management, Section 2 - "Selection of Computer Program Configuration items."

14. DRC Presentation "JLC Monterey II Workshop: Panel B: Hardware/Software/Firmware Configuration Item Selection Criteria.

## BACKGROUND MATERIAL

A. "Management of Microprocessors," Proceedings of the 12th Annual G-33 Committee Configuration and Data Management workshop — 23-28 October 1978.

B. "Management and Control for Microprograms, Microprocessors, and Microcomputers," Report of the Thirteenth Annual Data and Configuration Management Workshop — 8-12 October 1979.

C. "Tailoring CM Practices for Microprocessors and Firmware," Report of the Fourteenth Annual Data and Configuration Management Workshop — 20-24 October 1970.

## C. APPENDICES

### INDEX OF APPENDICES

## APPENDIX C-1 JLC PANEL B CHARTER

### HARDWARE/SOFTWARE/FIRMWARE CONFIGURATION ITEM SELECTION CRITERIA

1. Rationale: From the government viewpoint, few criteria are available for the selection of computer software configuration items. MIL-STD-483 (USAF), Appendix XVII provides some top level guidance for configuration item (CI) selection. The Air Force Systems Command Pamphlet (AFSCP) 800-7 provides some limited criteria, primarily as do nots, for selecting computer software CIs. Various guidebooks have also been developed by the services which discuss computer software selection but provide few criteria to guide selection.

With the advent of microcomputers and computer software embedded in firmware, it has become more difficult to predetermine how CIs (for hardware) should be distinguished from computer software CIs. What is needed is a complete set of criteria for guiding the selection process. These criteria should take into account both technical and management considerations. In addition, the criteria must consider procurement, developer support, and user needs. Criteria must be specified which allow for reasonableness in selection so that overhead costs such as documentation and control mechanisms are not restrictive in getting the software developed.

2. Approach. A panel should be formed to address the issues associated with computer software CI selection and to develop a set of criteria to aid the selection process.

## APPENDIX XVII

### 170. Guide for Selecting Configuration Items (CIs)

170.1 Purpose. This appendix provides guidance to government and contractor personnel responsible for selecting CIs. As used herein, CIs also encompasses Computer Program CIs (CPCIs).

170.2 Scope. The criteria of this appendix shall be used in the CI selection process whenever it occurs during the life cycle, however, the most beneficial results from its application will be realized when used at the beginning of the acquisition cycle.

170.3 Applicability. Each contractor to the government shall be responsible for his compliance with this appendix as well as the compliance of his subcontractors, vendors, and suppliers in accordance with paragraph 1.3 of this standard.

170.4 General Considerations.

170.4.1 Selection of CIs is based on the definition contained in DODD 5010.19, "an aggregation of hardware/software, or any of its discrete portions, which satisfies an end use function... CIs are those specification items whose functions and performance parameters must be defined and controlled to achieve the overall end use function and performance."

170.4.2 The selection of CIs is normally a function of anticipated design and should be independent of the concept for future reprocurement. The selection process is one of separating the elements of a system into individually-identified sub-sets for the purpose of managing their development. The CI should be regarded as a deliverable entity to which certain system functions have been allocated. CI selection reflects an optimum management level during acquisition. This level is one at which the procuring activity specifies, contracts for, and accepts individual elements of a system.

170.4.3 The selection of items to be managed as CIs should be determined by the need of the government to control an item's inherent characteristics or to control that item's interface with other items. The selection is a management decision normally accomplished through the system engineering process in conjunction with configuration management and with the participation of logistics.

Selecting CIs should be with a full view of the life cycle cost and management impacts associated with such a designation. Choosing too many CIs increases the cost of control; choosing too few or the wrong elements as CIs runs the risk of too little control through lack of management visibility. "What does program/project management need to know and control?" The government must determine what control it needs to exercise in light of cost/benefit trade-offs. The CI selections are made accordingly.

170.4.4 On development programs for subsystems or support equipment that will be common to more than one system, the basic CI should be that assembly that is common to all applications. An assembly part that is required to meet interface or other requirements peculiar to one of the systems should be identified as a separate CI in that system.

170.4.5 The major elements comprising the system should be identified as CIs during the Demonstration/Validation Phase. Early selection of CIs is important since management emphasis becomes greater as development progresses. As development continues and logistic or technical considerations surface, additional items can be designated CIs. Usually, the CI selection process should be essentially complete by PDR.

170.5 Specific Considerations. The following are some of the considerations upon which the decision shall be based:

170.5.1 Level of Government Control. The CI must be a manageable level of assembly. The CI is the basic element for configuration control and, for example, is usually limited to major subsystem levels of the Work Breakdown Structure or to a critical item of a lower level, when so identified.

170.5.2 Engineering Release System. The CI must allow the contractor to release engineering changes at an assembly level which is reportable and which enables verification of change incorporation, i.e., does not preclude change incorporation verification in a lower level assembly.

170.5.3 Safety. If the operation of an item is critical to other operations, flight safety or ground safety, the item will be more susceptible to being classified as a CI.

170.5.4 Existing or modified existing design items. Existing items that are not CIs developed at government expense, should not generally be candidates for reidentification as new CIs on new programs. Existing/modified design, commercial off the shelf equipment/computer program(s), should not necessarily be excluded from CI selection. The considerations identified in paragraph 170.5 and its subparagraphs should be addressed prior to making a decision.

170.5.5 New or modified design. Careful consideration shall be given new or modified design items, wherein more than a modest degree of complexity, utilization of new materials, processes or technology is involved; and, where the government wants direct control over the performance requirements for that item, at a specific time, i.e., when the government is directly concerned with the detail development.

170.5.6 Interface with GFE. The higher the degree of interface with Government Furnished Equipment (GFE), the higher the likelihood for selection as a CI.

170.5.7 Susceptibility to change. The higher the anticipated or estimated degree of change or modification which might be expected after the item is operational the higher the likelihood for selection as a CI.

170.5.8 Repairability/Maintainability. An item which is clearly designated as "Repairable" is much more a CI candidate than one which is not repairable. Eventually logisticians must deal with the Line Replaceable Units (LRUs) which comprise the principal components of the subsystem. However, designating CIs at the LRU level at the onset of full scale engineering development (FSED) would add significant cost to the development effort, especially in the area of change management. The LRU level is usually too low a level for effective government control during development.

170.5.9 Support Equipment Considerations. Without proper planning, minor items of support equipment could swell the list of CIs. Minor in this context refers to items such as individual hand tools, as compared to hydraulic torque wrenches, engine build-up tools, etc. There will usually be little or no change activity on many of these minor items. It may be sufficient to list these items as "support equipment in paragraph 3.2.4 of the CI Part I specification per MIL-STD 490, paragraph 20.3.2.4.c.

170.5.10 Subassembly Characteristics. Subassemblies (within a CI) should have a common mission relationship; should have common installation and deployment requirements (ground and airborne segments would be separate CIs); should have a cycle of changes dependent on the CI; and should not be the subject of separate test or formal acceptance by the procuring activity (should be accomplished as part of a CI). If these conditions are not met, the subassembly should be either part of another CI or a separate CI.

170.5.11 Computer Program CI (CPCI) Considerations. CPCI selection is usually a technically driven decision made by the developer. The decision(s) is/are based upon system trade-offs and the natural decomposition of the software. Premature partitioning must be avoided because to a certain extent it may preordain the design.

Generally, the executive/supervisor, functional/applications, input/output, test and support programs should be individual CPCIs. CPs with potential use in multiple systems should be separate CPCIs. Highly interrelated CPs should be combined as one CPCI. CPCIs should be established to their largest functional element (i.e., Operational Flight Program, Flight Simulator CP, etc.).

170.5.11.1 Source of Assemblies. Assemblies of CP elements to be acquired from a single contractor are potentially a single CPCI. (Separate sources usually supply separate CPCIs for contracting and delivery purposes.)

170.5.11.2 Separate Applications. CPs to be designated for operation in different models of computers should be separate CPCIs. Separate CPCIs may also be indicated for computer programs when a given installation uses a number of computers of the same type/model, each performing different functions in the system as a whole and having different sets of interfaces with other system elements.

170.5.11.3 Separate Schedules. CPs scheduled for development, testing, and delivery at different times may be separate CPCIs. When indicated by interrelationships and intended use, however, consideration should be given to such alternatives as: expansion of the earlier-developed CPCI via ECP; or development of the later CPCI to incorporate and replace the earlier item.

170.5.12 Types. If there are different configurations due to different adaptation data for each operating location, the different configurations should be identified by types (MIL-STD-490, paragraph 4.1.2 and 4.3b) within a single CI.

170.6 Effects of CI Selection. CI selection affects cost, schedule and/or performance for the government, prime contractors, subcontractors and suppliers. The effects of CI selection should not be permitted to occur automatically upon selection of an item as a CI. The effects which are unnecessary or premature can be tailored out for each CI by means of an appropriate contractually recognized vehicle, e.g., Program Plan, Statement of Work, CM Plan, Exceptions and Deviations. Selection of an item as a CI for manageability may be based on its administrative complexity, technical (engineering) criticality or maintenance (logistics) criticality. The following is a listing of the usual effects of CI designation:

a. Formal preparation of discrete configuration identification - most often in the form of a specification(s).

b. A discrete development specification and a companion product specification.

c. Government approval of changes over the configuration identification governing the item.

d. Continuing and accurate recording of the exact configuration status of the CI, including providing field activities precise data dealing with impending or completed modification actions.

e. Providing traceability of detailed design for follow-on activity, including historical data and individual status information for accident investigations, failure analysis, etc.

f. Individual design review activity (PDR, CDR, FQR, etc.) during development.

g. Individual qualification testing and reporting.

h. Individual physical and functional audits (PCA and FCA) at the conclusion of development.

i. Discrete and separate "related" ECP development preparation, review, approval and negotiation (for changes to CIs).

j. Separate identification indices and qualification records.

k. Separate nameplates and discrete CI identifiers (i.e., CI number, type, model, series, etc.).

l. Preparation of separate operating and user manuals.

m. Too many CIs may result in effects hampering visibility and management rather than improving it. These effects include:

(1) Increased administrative burden in preparing, processing, and status reporting of engineering changes which tends to be multiplied by the number of CIs.

(2) Increased development time and cost as well as possibly creating an inefficient design.

(3) Possible increase in management effort, difficulties in maintaining coordination and unnecessary generation of paper work.

n. Too few CIs may result in costly logistics and maintenance difficulties. The following may result:

(1) Loss of identity through separation of affected portions of a CI during field or depot maintenance or modification installation activity, i.e., an autocollimator and its mount.

(2) Inability to control like individual remove/replace items when CI identification and control is at the "set" level, e.g., a storage battery set.

(3) Loss of operational use of one function because required maintenance on another function requires action against the CI level, e.g., a CI having separate VHF/UHF functions loses both when maintenance must be done on either function.

170.7 CI Selection Checklist. The following questions should be used in selecting CIs tailored to individual program/project requirements. If most of the questions can be answered NO, the item probably should not be a CI. If most questions can be answered YES, the item probably should be a CI. If the questions can be answered with approximately equal numbers of YESs and NOs, additional judgment is needed to determine if the item should be a CI. The selection of CIs is a management decision based on experience and good judgment. It should be kept in mind that some of the factors such as serialization and nameplates will be required, regardless of CI selection, e.g., part of a higher level assembly.

a. Is it a critical high risk, and/or a safety item?

b. Is it readily identifiable with respect to size, shape and weight (hardware)?

c. Is it newly developed?

d. Does it incorporate new technologies?

e. Does it have an interface with hardware or software developed under another contract?

f. With respect to form, fit or function, does it interface with other items whose configuration is controlled by other entities?

g. Is there a requirement to know the exact configuration and status of changes to it during its life cycle?

APPENDIX C-3

## ELECTRONICS INDUSTRY ASSOCIATION (E/A) PANEL WRITE-UP

### PANEL D

### TAILORING CM PRACTICES FOR MICROPROCESSORS AND FIRMWARE

### INTRODUCTION

Panel D explored alternatives in managing the production configuration of firmware and microprocessor software. Concerns were raised with respect to the implementation of programs in several read-only-memory devices, the type of airplane programs are installed in, and the mission assigned to each airplane. As a result of these concerns, it was concluded that any change to software in read-only-memory devices will be treated as a functional change. The rationale for this approach is based on the fact that all software changes require some retest. It was further concluded that read-only-memory device changes will be identified and handled in accordance with DoD-STD-100C. To accomodate these conclusions, the logistical aspects of read-only-memory devices need to be reviewed and considered as a part of the Preliminary Design Review (PDR).

To form a basis for the panel's deliberations, the following definitions were adopted.

Firmware is computer software, resident in hardware read-only-memory devices, that cannot be modified under program control.

Microprocessor Software is computer software[1] used in conjunction with microprocessors. It may be delivered as firmware or as code and data stored on media (tapes, disks, card decks, etc.).

Computer Program/Hardware (CP/H) is microprocessor firmware identified and controlled as a portion of a hardware configuration item.

Prior to the start of deliberations a presentation, which provided background information on microprocessor software, was made to Panel D participants. Subsequent to this presentation, the participants were subdivided into five subpanels; namely:

| Subpanel | Topic |
|---|---|
| D-1 | Identification |
| D-2 | Configuration Control |
| D-3 | Configuration Accounting |
| D-4 | Configuration Audits |
| D-5 | Library Controls |

A summary of the background information presentation and the report of each subpanel is provided hereinafter.

1. Computer software is a collection of associated computer programs and computer data required to enable the computer equipment to perform computational or control functions. NOTE: It is the abstract of tapes, disks, card decks, and firmware (Reference EIA Bulletin 4A, dated April 1979).

## BACKGROUND INFORMATION SUMMARY

PANEL D

MAJOR LARRY FRY

AIR FORCE SYSTEMS COMMAND

The increased application of software in microprocessor read-only-memory devices (commonly referred to as firmware) is having a major impact on defense systems. As more and more microprocessors and associated firmware appear in devices such as displays, controllers, etc., configuration management practices need to be reviewed and revised, as necessary, to accommodate management and control requirements of the acquisition and support environment. This is the charter of Panel D; Tailoring CM Practices for Microprocessors and Firmware.

Imposing extensive software configuration management procedures on microprocessor firmware could prove to be unnecessarily expensive. For example, it is hard to visualize treating a small computer program, which is hardwired (i.e., stored in a read-only-memory device) into a display terminal, as one would treat a more sophisticated Computer Program Configuration Item (CPCI). The costs associated with conducting comprehensive reviews,audits, testing, and documentations, using such an approach, could prove to be more costly than the development cost for the small computer program.

Assuming that the display terminal is a hardware Configuration Item (CI), it is more cost effective to treat the computer program as a part of the display terminal CI and not as a separate CPCI.

On the other hand, treating a larger hardwired computer program as part of a hardware CI is probably not a prudent move either. This is certainly the case if this program is expected to change once the CI is fielded. Changes to fielded hardwired software implies a capability to generate a changed computer program, support facilities to burn-in and verify these changes into read-only-memory devices, and procedures to identify these changes in the hardware implementation environment. When changes are anticipated, there may also be need for a requirement to program the software in a higher order language to facilitate the change process. All of this implies that some hardwired microprocessor software should be treated as a CPCI with the attendant reviews, audits, testing, and documentation.

The overall governing directive within DoD for computer resources is DoDD 5000.29, "Management of Computer Resources in Major Systems". This directive stipulates that computer software will be specified and treated as configuration items. In a recent draft rewrite of 5000.29, additional words were added; namely: "A hardware implementation of computer software (so called firmware) may be subject to configuration management as either hardware or software depending on the need for post-development support".

The standards covering software configuration management (e.g., MIL-STD-483, MIL-STD-490, etc.) are silent concerning the treatment of microprocessor firmware. However, since firmware is, in fact, computer software that is hardwired, one can assume that the controls utilized for computer programs would equally apply to firmware.

# SUBPANEL REPORTS

## D-1: IDENTIFICATION

Subpanel members were as follows:

| | |
|---|---|
| Anderson, Jack V. | Sperry Univac |
| Bryan, William L. | CTEC, Inc. |
| Cardin, L. P. Roland | Garrett Manufacturing Company |
| * Egan, Leo G. | ITT, Federal Electric Corporation |
| Ellison, Richard | Medtronic, Inc. |
| Finch, Frederick D. | Norden Systems |
| Hill, Charles A. | Magnavox |
| Kolsti, Weston G. | Analytical System Engineering |
| Lewis, Joseph L. | Westinghouse |
| Liberatore, Domenic | Raytheon |
| Lillo, Clifford L. | Hughes Aircraft |
| Wallin, Joe D. | Hughes Aircraft |

* Subpanel Chair

The objectives of this subpanel were as follows:

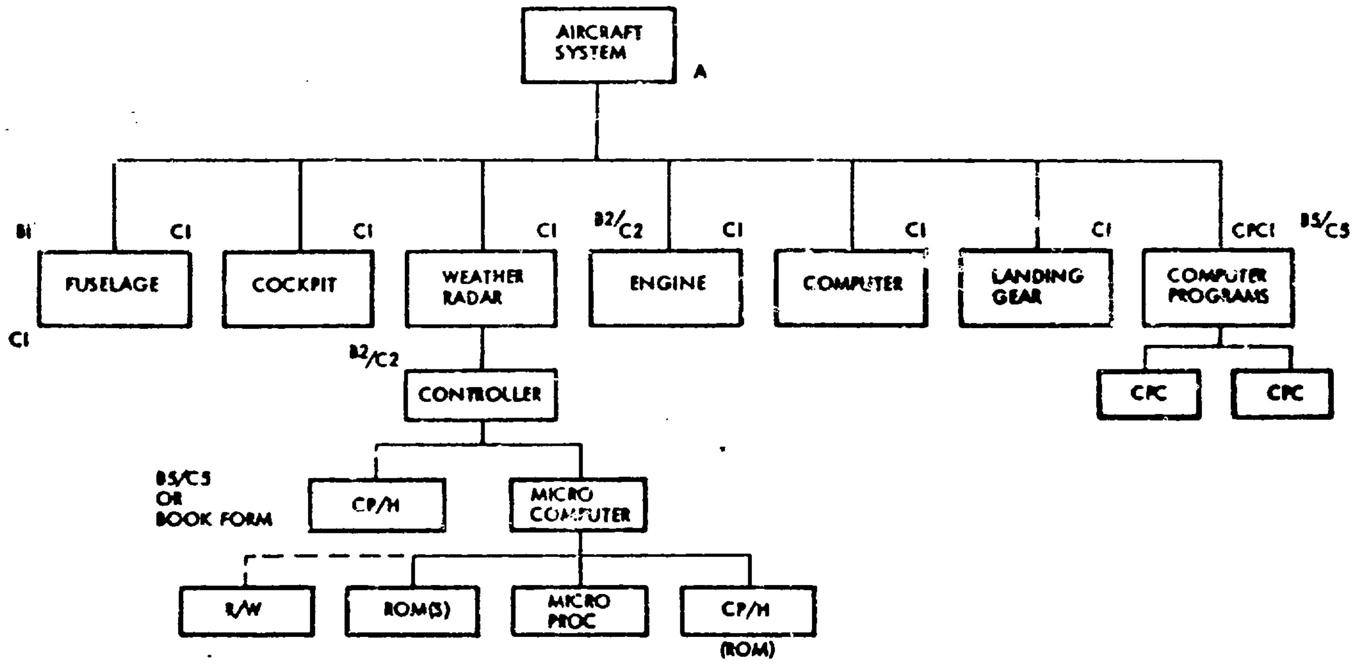
(a) Prepare a typical specification tree and design/test documents.
(b) Prepare a flow chart of the activities required to achieve microprocessor software.
(c) Prepare checklists that can be used to assure the achievement of configuration identification objectives for microprocessor software.

The panel examined documentation approaches for microprocessor software when it is treated as a:

(a) Configuration Item (CI),
(b) Computer Program Configuration Item (CPCI), or
(c) Software within a hardware CI.

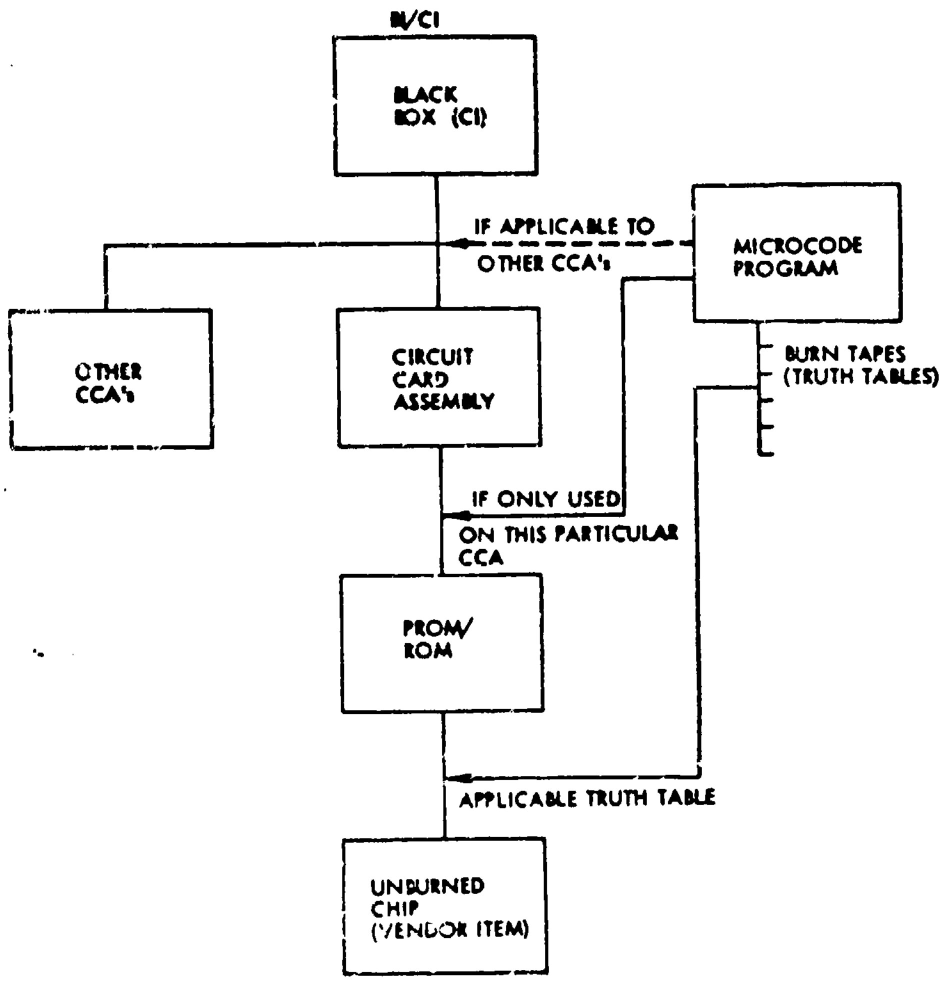
When treated as a CI there was no problem. The identification of a CI would be documented via a set of specifications, typically a Type B1/C1 or B2/C2. The Type C specification would call out a top assembly drawing, which would identify all of the detail engineering drawings. When microprocessor software is treated as a CPCI, no apparent problems result. The identification of a CPCI would be through the B5/C5 set of specifications. When microprocessor software is treated as software within a hardware CI, many problems arise. This is a "firmware" problem. As an aid to resolving this problem, the microprogram or software within a microprocessor device was called CP/H (Computer Program/Hardware).

In keeping with the objectives stated earlier, we developed a specification tree which is depicted in Figure 1-1. We also developed documentation trees for the design/test documents of a CP/H. These are shown in Figures 1-2, 1-3, and 1-4. A time line chart, for the typical events that would take place in the development of a CI that includes a CP/H, was developed as shown in Figure 1-5. To support this time line, a flow chart was developed and is presented in Figure 1-6. Finally, a checklist (Table 1-1) was prepared which could be used to assure the achievement of the objectives that relate to software within a CI.



CONTROLLER B2/C2 SPECIFICATION IS SUFFICIENT TO DESCRIBE THE COMPUTER PROGRAM(S) USED WITHIN THE HARDWARE CONFIGURATION ITEM (CI)

FIGURE I-I: SPECIFICATION TREE FOR A TYPICAL WEAPON SYSTEM



B/CI
BLACK BOX (CI)
IF APPLICABLE TO OTHER CCA's
MICROCODE PROGRAM
OTHER CCA's
CIRCUIT CARD ASSEMBLY
BURN TAPES (TRUTH TABLES)
IF ONLY USED ON THIS PARTICULAR CCA
PROM/ ROM
APPLICABLE TRUTH TABLE
UNBURNED CHIP (VENDOR ITEM)

FIGURE 1-2: FIRMWARE DOCUMENTATION STRUCTURE (SPERRY-UNIVAC METHOD)

SOFTWARE INTENSIVE

CPCI

B5 SPEC
C5 SPEC
} MIL-STD-490

CPCI TEST PLAN
TEST PROCEDURE
TEST REPORT

HARDWARE INTENSIVE

CP/H

DOD-D-1000 {
BOOK FORM DRAWING
1. DESIGN DESCRIPTION
A. NARRATIVE
B. FLOWS
C. TRUTH TABLES
2.. LISTING

CI ACCEPTANCE TEST SPEC
ACCEPTANCE TEST PLAN
ACCEPTANCE TEST PROCED
ACCEPTANCE TEST REPORT

OTHER
PRINTED CIRCUIT ASS'Y DWG
SPEC OR SOURCE CONT DWG

FIGURE 1-3: DOCUMENTATION FOR SOFTWARE WITHIN A CONFIGURATION ITEM

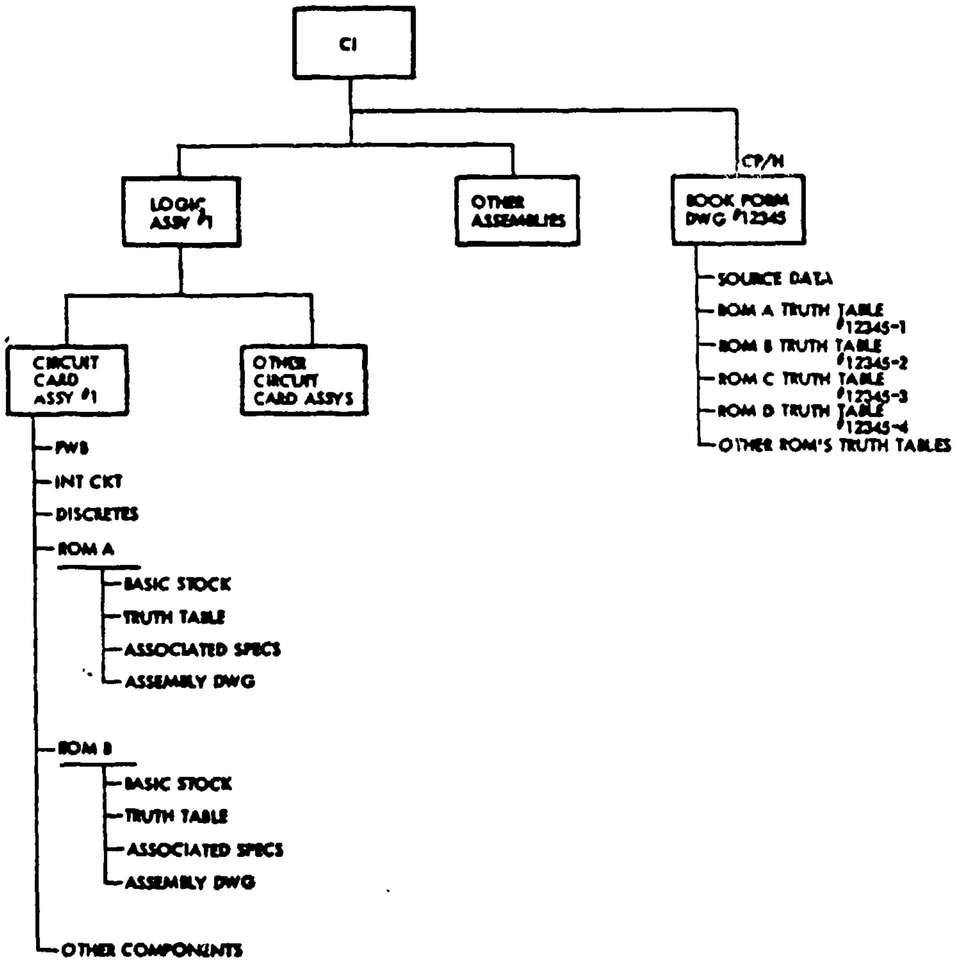
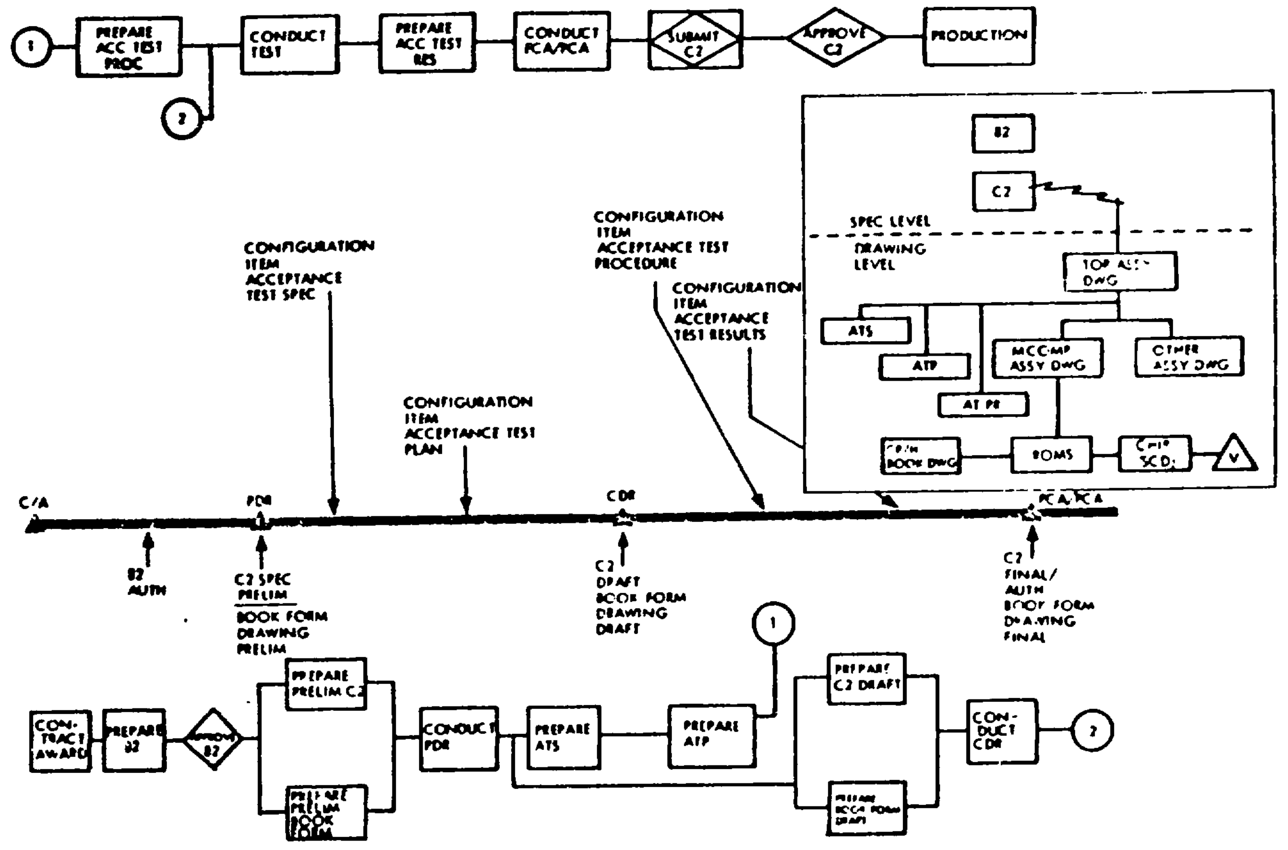


FIGURE 1-4: TYPICAL BREAKDOWN OF FIRMWARE & HARDWARE WITHIN A CONFIGURATION ITEM

FIGURE 1-5: TIME-LINE CHART FOR SEQUENCE OF EVENTS IN CP/H DEVELOPMENT



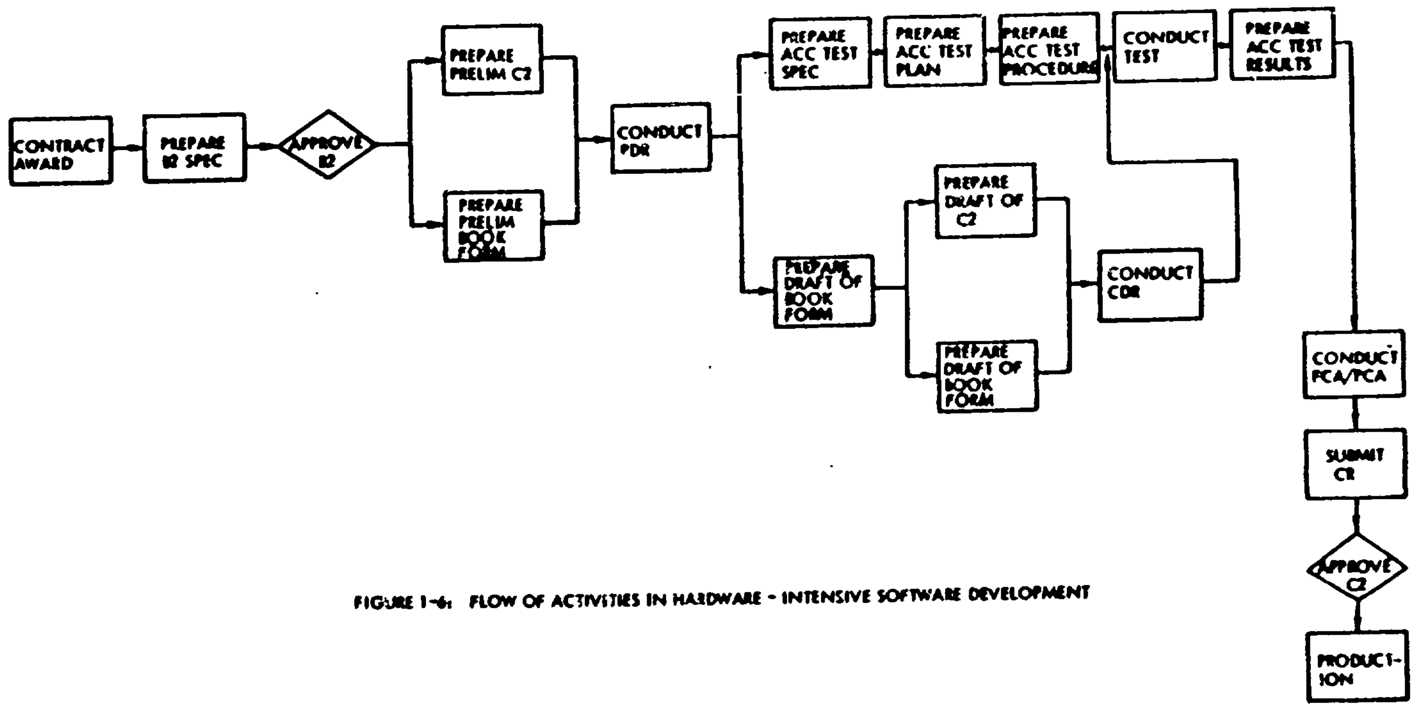


FIGURE 1-6: FLOW OF ACTIVITIES IN HARDWARE - INTENSIVE SOFTWARE DEVELOPMENT

Recommendations:

1. Provide for allowing the (Type A) Segment Specification to be used when a "software intensive" CPCI is embedded within a CI revision to MIL-STD-483 is needed.

2. Prepare a Data Item Description (DID) for the Computer Program/Hardware (i.e. Book Form Drawing) and call for it on the CDRL. This DID should not require authentication since it is a part of the drawing system defined by DOD-D-1000.

3. MIL-STD-130 should be revised to include part number marking of firmware media.

4. Develop procedures that would be applicable when both hardware-intensive and software-intensive microprocessor software reside in the same CI.

TABLE 1-1 CHECKLIST FOR MICROPROCESSOR SOFTWARE IDENTIFICATION

| | | Yes | No |
|---|---|---|---|
| 1. | Does a work breakdown structure/specification tree exist for the program? | | |
| 2. | Where do hardware-intensive software applications exist? Have they been identified? | | |
| 3. | Are these applications mutually agreed to between contractor and customer? | | |
| 4. | Has a maintenance philosophy at each level of maintenance been defined for these applications? | | |
| 5. | Is the intended application an existing design? Is this a proprietary design? | | |
| 6. | Have the firmware developmental requirements been adequately defined? | | |
| 7. | Have configuration management requirements for customer design reviews and audits been established? | | |
| 8. | Has a functional baseline been established? | | |
| 9. | Has an allocated baseline been established? | | |
| 10. | Has a preliminary design review been conducted? | | |
| 11. | Has the software development environment been identified and documented? (what versions of assemblers, compilers, linking loaders will be employed?) | | |
| 12. | Has an engineering release schedule for the firmware under development been prepared? | | |
| 13. | Have design documentation guidelines been established? | | |
| 14. | Have adequate internal controls been established for version-tracking of the application program(s) under development? | | |
| 15. | What physical media have been chosen for the firmware in question (ROM, EPROM, PROM, EAROM, etc)? Have documentation requirements for that medium been adequately defined? | | |
| 16. | Have firmware media marking requirements been defined? | | |
| 17. | Has a Critical Design Review been conducted on the Computer Program Hardware (CP/M) in question? | | |
| 18. | Have the production firmware master tapes been released through the engineering release system? | | |
| 19. | Have firmware quality assurance requirements been identified? | | |
| 20. | Does the test documentation call for both read/write and read-only-memory test phases? | | |
| 21. | Is final validation testing performed with read-only-memory as configured for delivery? | | |
| 22. | Has a Functional Configuration Audit/Physical Configuration Audit been performed? | | |
| 23. | Has a product baseline been established? | | |

## SUBPANEL REPORTS

### D-2 Configuration Control

Subpanel members were as follows:

| | |
|---|---|
| DeBerry, Robert D. | Sperry Univac |
| Heim, William J. | Naval Weapons Center |
| Hurwitz, Martha H. | Strategic Systems |
| Kessel, Robert | Raytheon |
| Martin, Daniel | Emerson Electric |
| * McKinney, Arthur | Sanders |
| Ottone, Michael N. | Fairchild |
| Paslosky, Paul | TRW DSSG |
| Sabb, Elijah | GTE Sylvania |

* Subpanel Chair

The objectives of this subpanel were as follows:

(a) Define Class I/II changes for microprocessor software, using EIA Bulletin 4A as a guide.

(b) Prepare a flow chart which depicts the activities required to control microprocessor software changes.

(c) Prepare a checklist that can be used to assure that microprocessor software changes are adequately controlled.

To satisfy these objectives, the subpanel investigated configuration change controls for microprocessor software that is treated as a

(a) Configuration Item (CI),

(b) Computer Program Configuration Item (CPCI), or

(c) Software within a hardware CI.

Subpanel members concluded that

(a) Configuration control need only be applied to the microprocessor device and the software (code and data).

(b) Once a microprocessor device is released (Engineering Release) for use, changes should be processed via the methodology utilized for hardware CIs.

(c) Changes to microprocessor software should be processed using the existing methodology for CPCIs.

(d) Formal, internal change control for microprocessor software should occur at the completion of unit test and prior to the start of subsystem integration test.

Figure 2-1 depicts the "Sequence of Events for A Generic Project/Program". It should be understood that for a specific project/program, events may deviate from those depicted. The triangles ($\Delta$s) are used to identify items released for use after specific events occur. After release, either internal (company) or external (customer) change control procedures should be applied.

Figure 2-2 is a flow chart that illustrates the change control activities required for microprocessor software after engineering release. Responsibility for change processing ranges from a single control point consisting of the Engineering Manager to a formal Change Control Board (CCB). Detailed change recommendations are submitted to the formal CCB for final review and acceptance or rejection. The "Analyze Required Change; Assess Impact" rectangle recognizes a need to also review baseline documentation for necessary changes.

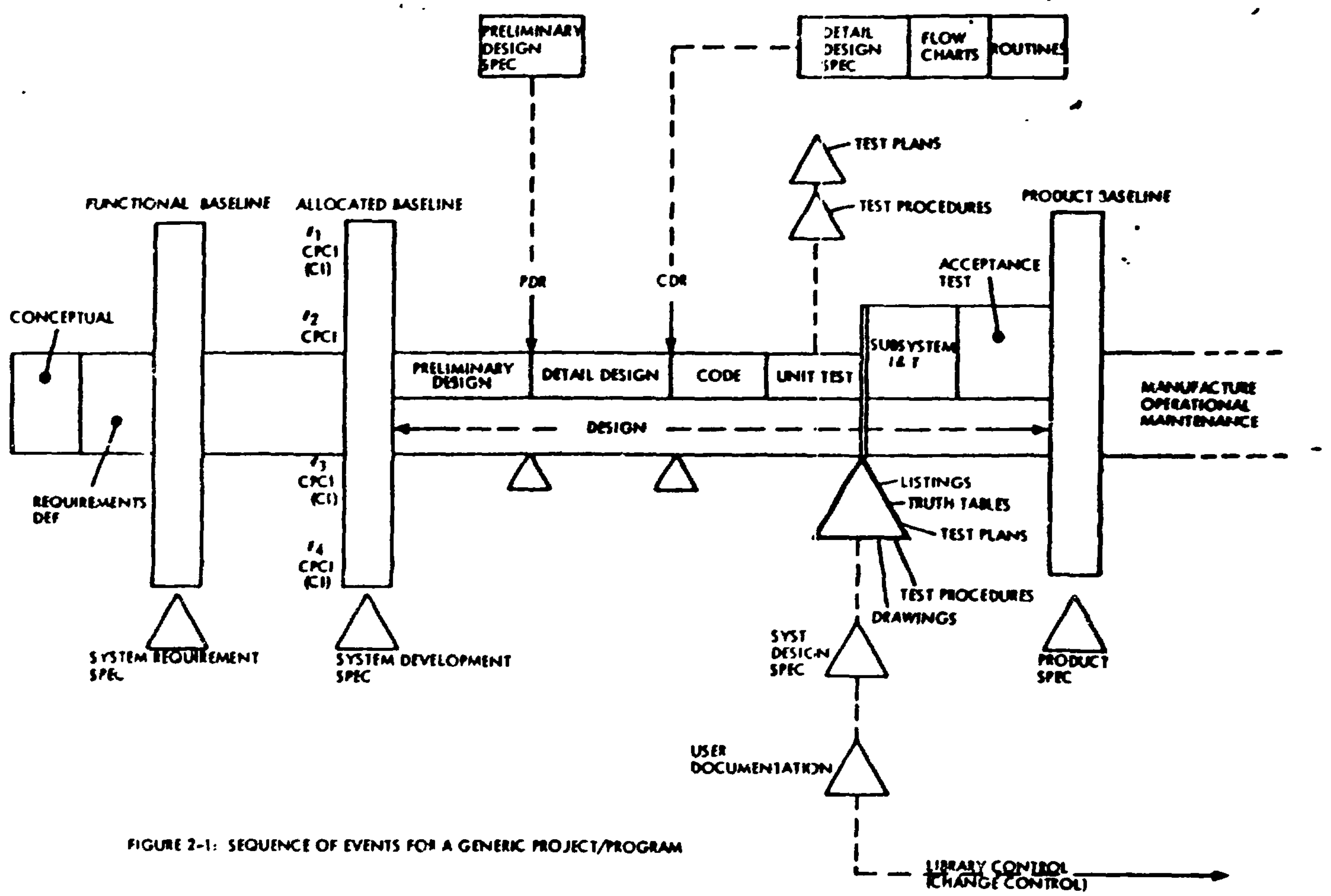


FIGURE 2-1: SEQUENCE OF EVENTS FOR A GENERIC PROJECT/PROGRAM

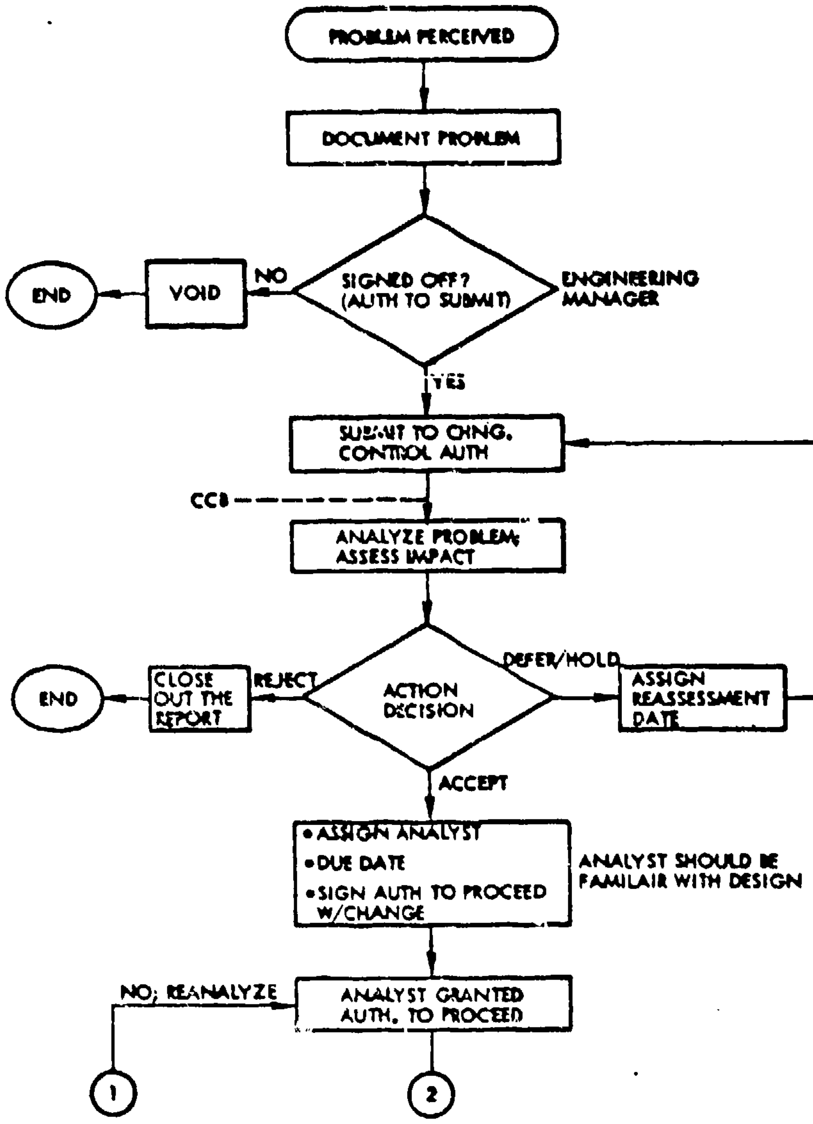


FIGURE 2-2: MICROPROCESSOR SOFTWARE CHANGE CONTROL FLOW CHART

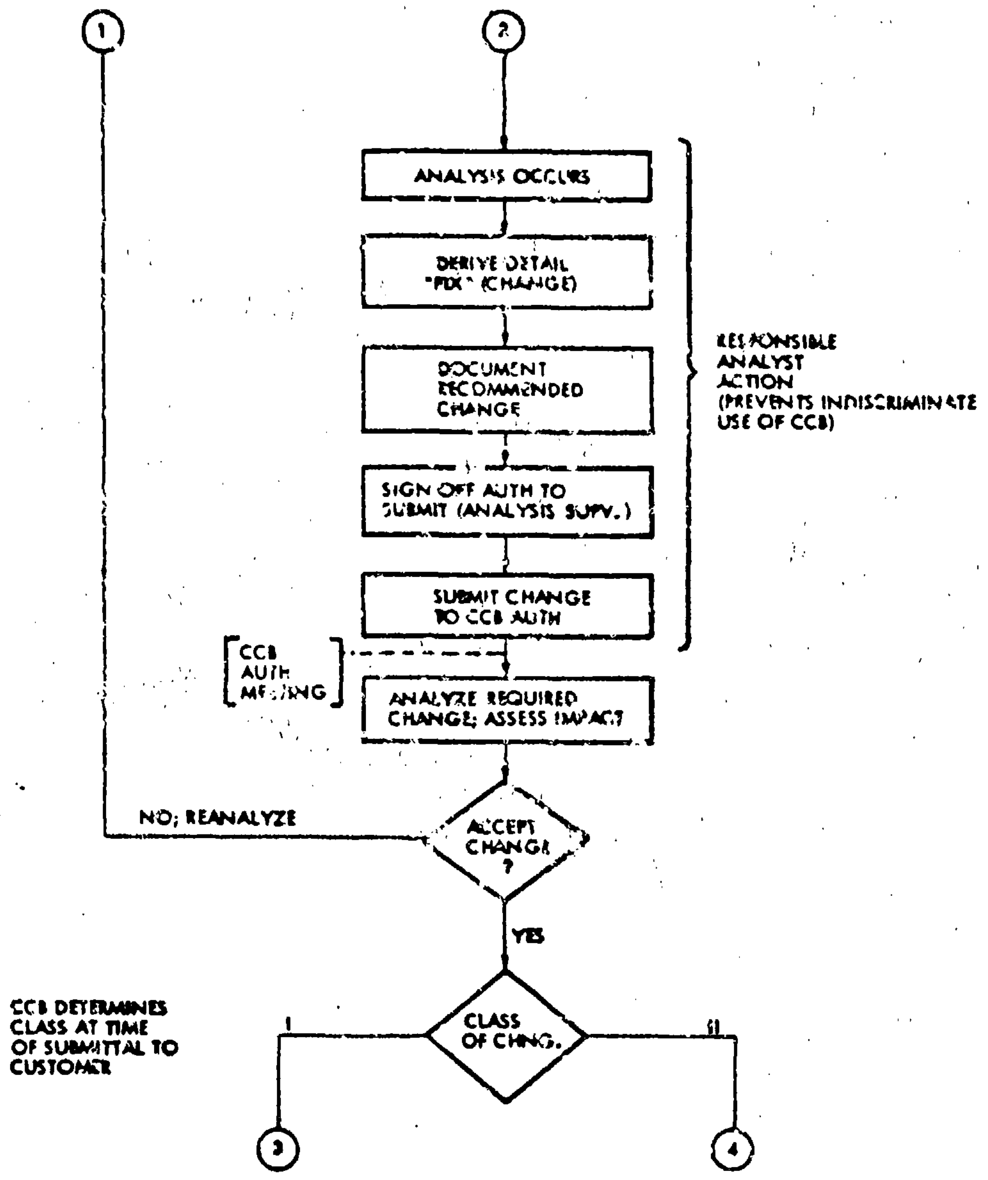

1
2
ANALYSIS OCCURS
DERIVE DETAIL "FIX" (CHANGE)
DOCUMENT RECOMMENDED CHANGE
SIGN OFF AUTH TO SUBMIT (ANALYSIS SUPV.)
SUBMIT CHANGE TO CCB AUTH
RESPONSIBLE ANALYST ACTION (PREVENTS INDISCRIMINATE USE OF CCB)
CCB AUTH MEETING
ANALYZE REQUIRED CHANGE; ASSESS IMPACT
ACCEPT CHANGE ?
NO; REANALYZE
YES
CLASS OF CHNG.
I
II
CCB DETERMINES CLASS AT TIME OF SUBMITTAL TO CUSTOMER
3
4

FIGURE 2-2 (CONTINUED) PAGE 2 OF 3

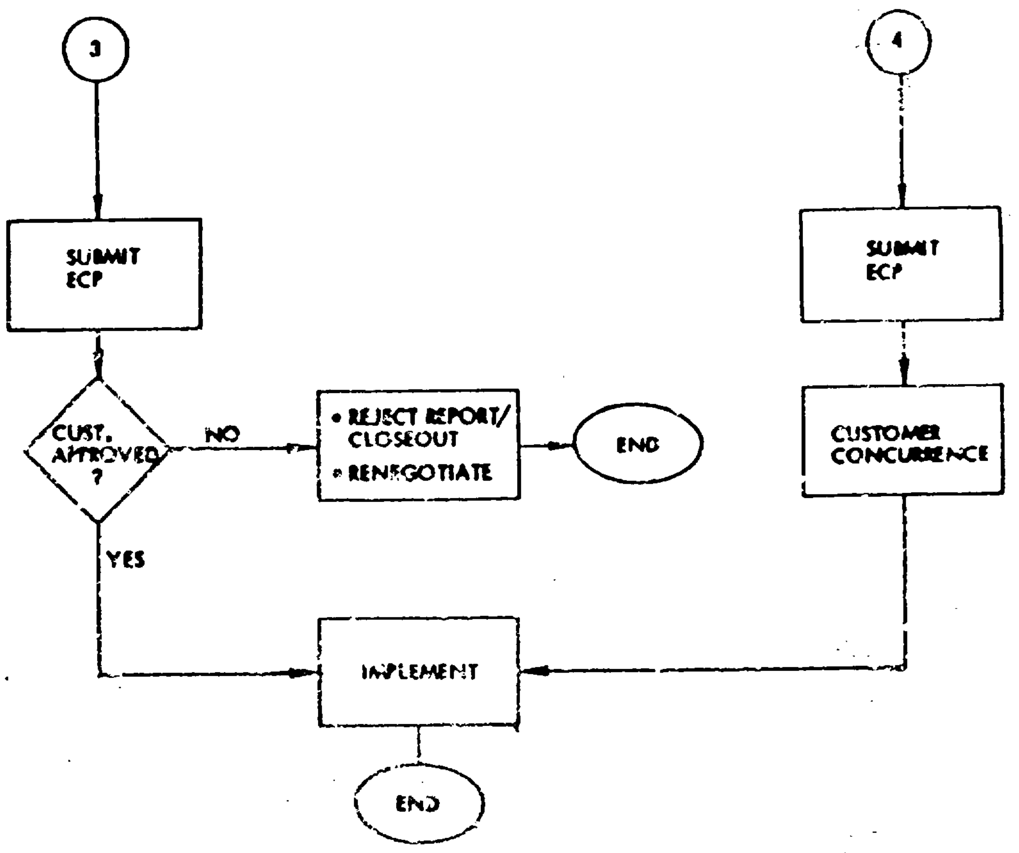


FIGURE 2-2 (CONTINUED)

TABLE 2-1 CHECKLIST FOR MICROPROCESSOR SOFTWARE CHANGE CONTROL

| | Yes | No |
|---|---|---|
| 1. Has a configuration identification system been established? | ___ | ___ |
| (a) Are release dates available? | ___ | ___ |
| (b) Is a document/specification tree available? | ___ | ___ |
| 2. Has a problem reporting system been defined? | ___ | ___ |
| (a) Have forms and format been defined? | ___ | ___ |
| (b) Have change control approval authorities been identified? | ___ | ___ |
| (c) Has a problem number/tracking system been established? | ___ | ___ |
| (d) Has a means of software problem classification been categorized? | ___ | ___ |
| 3. Has the implementation/verification procedure been documented? | ___ | ___ |
| 4. Have the procedures for distributing release, problem and change documentation been developed? | ___ | ___ |
| 5. Have logistic requirements been identified? | ___ | ___ |

## D-3: CONFIGURATION STATUS ACCOUNTING

Subpanel members were as follows:

|  |  |
|---|---|
| Gainey, Ann M. | Naval Weapons Support Center |
| Hartman, L. Alvin | Lockheed |
| * Roggero, Vincent R. | Trident Command |
| Yorba, Michael | NAVSHIP |

* Subpanel Chair

The objectives of this subpanel were as follows:

(a) Define internal and contractual requirements for microprocessor software configuration status accounting.

(b) Prepare a flow chart depicting the activities required to achieve adequate configuration status accounting for microprocessor software.

(c) Prepare a checklist that can be used to assure the achievement of configuration status accounting objectives for microprocessor software.

The subpanel investigated configuration status accounting requirements for microprocessor software that is treated as a:

(a) Configuration Item (CI),

(b) Computer Program Configuration Item (CPCI), or

(c) Software within a hardware CI.

Subpanel members concluded that:

(a) When microprocessor software is treated as a CI, configuration status accounting methods used for hardware are applicable.

(b) Configuration status accounting methods applicable to software should be used when microprocessor software is treated as a CPCI or a component within a hardware CI.

The flow chart shown in Figure 3-1 reflects the division of activity between contractual and internal requirements for configuration status accounting as development progresses through specific milestones.

Figure 3-2 depicts the flow of activities that are required to achieve adequate configuration status accounting for microprocessor software. It is imperative that the accounting tasks, designated by triangles ( $\triangle$s), be started at the time indicated. Hence, the status accounting task increases as documents, hardware and software items, and integration assemblies are added.

A "Configuration Status Accounting Checklist for Microprocessor Software" is presented in Table 3-1.

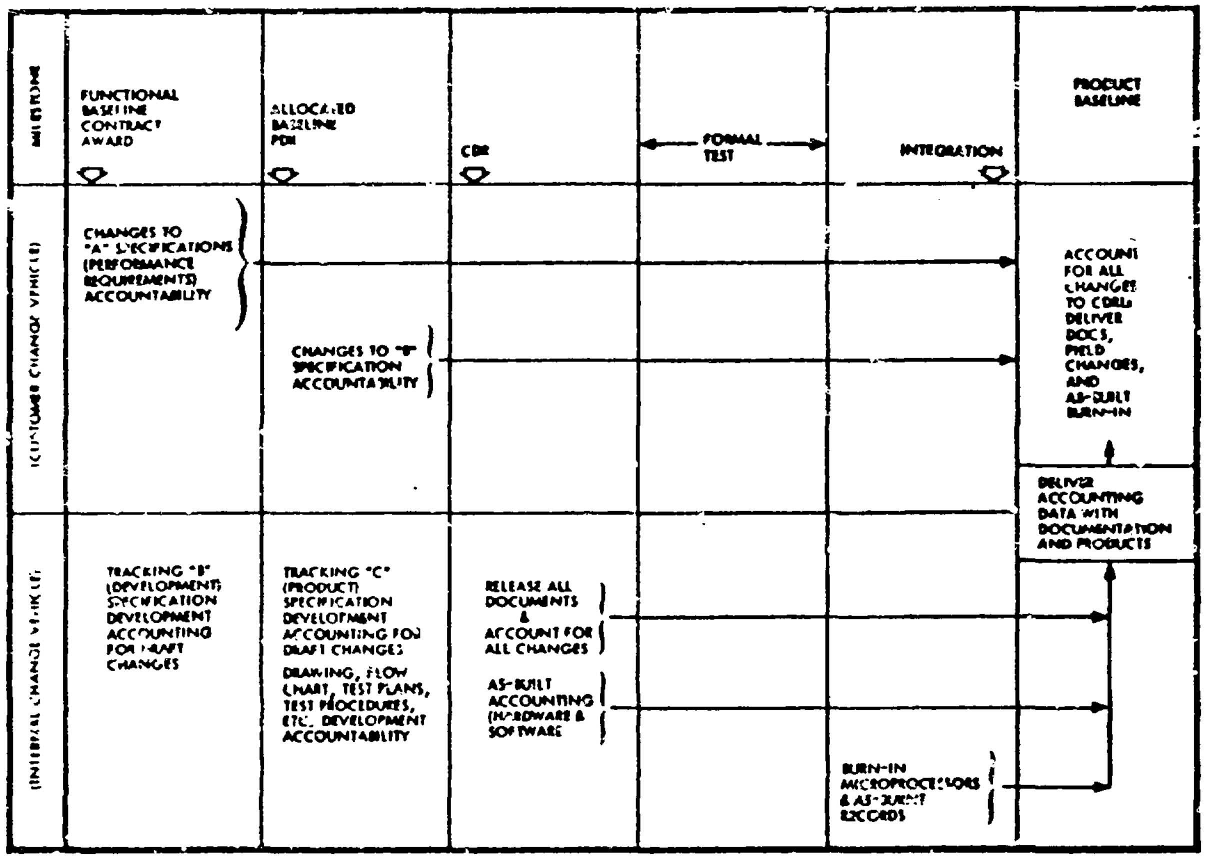

MILESTONE
FUNCTIONAL BASELINE CONTRACT AWARD
ALLOCATED BASELINE PDR
CDR
FORMAL TEST
INTEGRATION
PRODUCT BASELINE
(CUSTOMER CHANGE VEHICLE)
CHANGES TO "A" SPECIFICATIONS (PERFORMANCE REQUIREMENTS) ACCOUNTABILITY
CHANGES TO "B" SPECIFICATION ACCOUNTABILITY
ACCOUNT FOR ALL CHANGES TO CDRL DELIVER DOCS, FIELD CHANGES, AND AS-BUILT BURN-IN
DELIVER ACCOUNTING DATA WITH DOCUMENTATION AND PRODUCTS
(INTERNAL CHANGE VEHICLE)
TRACKING "B" (DEVELOPMENT) SPECIFICATION DEVELOPMENT ACCOUNTING FOR DRAFT CHANGES
TRACKING "C" (PRODUCT) SPECIFICATION DEVELOPMENT ACCOUNTING FOR DRAFT CHANGES
DRAWING, FLOW CHART, TEST PLANS, TEST PROCEDURES, ETC. DEVELOPMENT ACCOUNTABILITY
RELEASE ALL DOCUMENTS & ACCOUNT FOR ALL CHANGES
AS-BUILT ACCOUNTING (HARDWARE & SOFTWARE
BURN-IN MICROPROCESSORS & AS-BUILT RECORDS

FIGURE 3-1: DIVISION OF ACTIVITY BETWEEN CONTRACTUAL AND INTERNAL REQUIREMENTS FOR CONFIGURATION STATUS ACCOUNTING

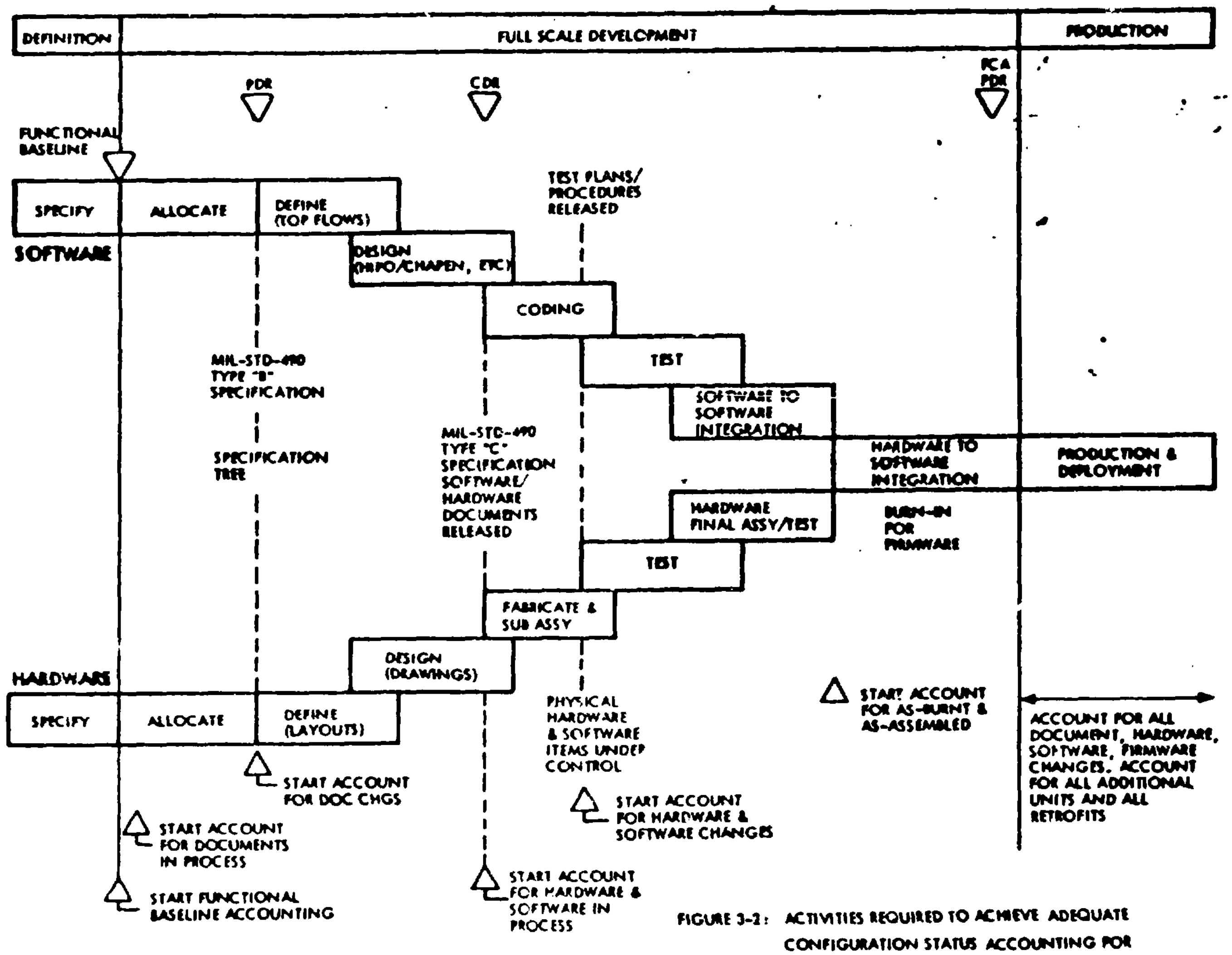

DEFINITION
FULL SCALE DEVELOPMENT
PRODUCTION
PDR
CDR
FCA
PDR
FUNCTIONAL BASELINE
SPECIFY
ALLOCATE
DEFINE (TOP FLOWS)
SOFTWARE
DESIGN (HIPO/CHAPEN, ETC)
TEST PLANS/ PROCEDURES RELEASED
CODING
TEST
SOFTWARE TO SOFTWARE INTEGRATION
MIL-STD-490 TYPE "B" SPECIFICATION
SPECIFICATION TREE
MIL-STD-490 TYPE "C" SPECIFICATION SOFTWARE/ HARDWARE DOCUMENTS RELEASED
HARDWARE TO SOFTWARE INTEGRATION
PRODUCTION & DEPLOYMENT
HARDWARE FINAL ASSY/TEST
BURN-IN FOR FIRMWARE
TEST
FABRICATE & SUB ASSY
DESIGN (DRAWINGS)
HARDWARE
SPECIFY
ALLOCATE
DEFINE (LAYOUTS)
PHYSICAL HARDWARE & SOFTWARE ITEMS UNDER CONTROL
START ACCOUNT FOR AS-BURNT & AS-ASSEMBLED
ACCOUNT FOR ALL DOCUMENT, HARDWARE, SOFTWARE, FIRMWARE CHANGES. ACCOUNT FOR ALL ADDITIONAL UNITS AND ALL RETROFITS
START ACCOUNT FOR DOC CHGS
START ACCOUNT FOR DOCUMENTS IN PROCESS
START ACCOUNT FOR HARDWARE & SOFTWARE CHANGES
START FUNCTIONAL BASELINE ACCOUNTING
START ACCOUNT FOR HARDWARE & SOFTWARE IN PROCESS

FIGURE 3-2: ACTIVITIES REQUIRED TO ACHIEVE ADEQUATE CONFIGURATION STATUS ACCOUNTING FOR MICROPROCESSOR SOFTWARE

TABLE 3-1 CONFIGURATION STATUS ACCOUNTING CHECKLIST FOR MICROPROCESSOR SOFTWARE

| | | Yes | No |
|---|---|---|---|
| 1. | Have policies and procedures for configuration management been published? | ___ | ___ |
| 2. | Does a procedure for internal change control exist? | ___ | ___ |
| 3. | Has a Configuration Change Board (CCB) been established (single CCB for hardware and microprocessor software)? | ___ | ___ |
| 4. | Have different levels of change approval/authority been established for different states of development/production? | ___ | ___ |
| 5. | Does a numbering/identification system exist for: | | |
| | Microprocessor Software | ___ | ___ |
| | Related Documentation | ___ | ___ |
| 6. | Does a system for approval/quality control, prior to release/acceptance, exist for: | | |
| | Microprocessor Software? | ___ | ___ |
| | Related Documentation? | ___ | ___ |
| 7. | Has a library control procedure been published for microprocessor software and documentation? | ___ | ___ |
| 8. | Has a configuration management accounting system been established for recording and reporting the status of proposed, approved, and implemented changes for hardware, microprocessor software, and related documentation? | ___ | ___ |
| 9. | Does the configuration status accounting system provide for timely and accurate reporting of the information needed by management or procuring activity? | ___ | ___ |
| 10. | Does a procedure exist that provides for for an audit/quality control check of microprocessor software to insure that it matches configuration documentation prior to integration into higher level assemblies, burn-in of microprocessor devices, or acceptance by the procuring activity? | ___ | ___ |

## D-4: CONFIGURATION AUDITS

Subpanel members were as follows:

| | |
|---|---|
| * Hawks, Kenneth | HQ ASO/RWCC |
| Hoyt, Richard | Tracor |
| McAlister, Clyde O. | Sperry Univac |
| Parks, Frederick | Naval Electronics System |
| Stees, Mae S. | Magnavox |
| Teixeira, Stephen M. | Naval Underwater Systems Center |
| Whitlock, James A. | General Electric |

* Subpanel Chair

The objectives of this subpanel were as follows:

(a) Prepare a typical Cross Reference Test Verification Matrix for microprocessor software.

(b) Propose method(s) for directing/determining the physical identification of microprocessor software.

(c) Prepare a flow chart which depicts the activities required to conduct audits for microprocessor software.

(d) Prepare one or more checklist that can be used to insure that microprocessor software audits are conducted properly.

The subpanel investigated configuration audit requirements for microprocessor software that is treated as a:

(a) Configuration Item (CI),

(b) Computer Program Configuration Item (CPCI), or

(c) Software within a hardware CI.

Subpanel conclusions are as follows:

(a) The Verification Cross Reference Matrix (VCRM) shown in Figure 4-1 was developed for microprocessor software. It is a modification of the MIL-STD-483 version. This VCRM is applicable because microprocessor software must undergo the same tests as any other software.

(b) Identification of microprocessor devices is important; however, a definitive conclusion did not result. Several labeling methods were discussed, but each one had drawbacks. The methods discussed are summarized in Table 4-1.

(c) A flow chart depicting the activities required to conduct audits for microprocessor software was not prepared for lack of time.

(d) It was determined that a FCA and preliminary PCA should be performed for microprocessor software prior to integration into the microprocessor device. The preliminary PCA would involve examination of the microprocessor software on a sampling basis without establishing a product baseline. Once system level testing is completed, the microprocessor software PCA and FQR would be held jointly with the microprocessor device (hardware) PCA and FQR. (See Figure 4-2).

(e) Microprocessor software audit checklists were prepared for the FCA, PCA, FQR, and minutes of these audits/reviews, and are presented in Tables 4-2, 4-3, 4-4, and 4-5, respectively. Some of the items listed may not be applicable for a given procurement. Users should tailor the list to their needs.

Side issues that were discussed during panel deliberations are as follows:

(a) The difference between a software version and revision.

(b) Traceability of microprocessor software via the engineering release system.

(c) Declassification of non-eraseable microprocessor devices.

Recommendations:

As part of the military qualification process, the responsible agency should:

(a) Provide potential users with suggested method of identifying/labeling microprocessor devices which contain software, and

(b) Provide guidance on applicable constraints with respect to the erasure of classified information contained within microprocessor devices.

VERIFICATION CROSS REFERENCE MATRIX

METHOD LEGEND: 1 - INSPECTION (CODE REVIEW) 2 - REVIEW OF ANALYTICAL DATA
3 - DEMONSTRATION 4 - TEST
N.A. - NOT APPLICABLE

TEST CATEGORY LEGEND: A - ENGINEERING TEST & EVALUATION (DEBUG/PROGRAMMER CHECKOUT)
B - PRELIMINARY QUALIFICATION
C - FORMAL QUALIFICATION
D - RELIABILITY TEST & ANALYSIS
E - ENGINEERING CRITICAL COMPONENT QUALIFICATION
F - SYSTEM TEST

| SECTION 3.0 REQUIREMENT | VERIFICATION METHOD (S) | | | | | TEST CATEGORY | | | | | | TEST REPORT | | | | | | SECTION 4.0 VERIFICATION REQUIREMENT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REFERENCE | 1 | 2 | 3 | 4 | NA | A | B | C | D | E | F | B | C | D | E | F | | |
| 3.0 | | | | | | | | | | | | | | | | | | |
| 3.1 | | | | | | | | | | | | | | | | | | |
| 3.1.1 | | | | | | | | | | | | | | | | | | |
| 3.1.1.1 | | | | | | | | | | | | | | | | | | |
| 3.2 | | | | | | | | | | | | | | | | | | |
| 3.2.1 | | | | | | | | | | | | | | | | | | |
| 3.2.1.1 | | | | | | | | | | | | | | | | | | |
| 3.2.2.1 | | | | | | | | | | | | | | | | | | |
| 3.2.2.2 | | | | | | | | | | | | | | | | | | |
| 3.3 | | | | | | | | | | | | | | | | | | |
| 3.4 | | | | | | | | | | | | | | | | | | |
| ETC | | | | | | | | | | | | | | | | | | |

FIGURE 4-1: VERIFICATION CROSS REFERENCE MATRIX

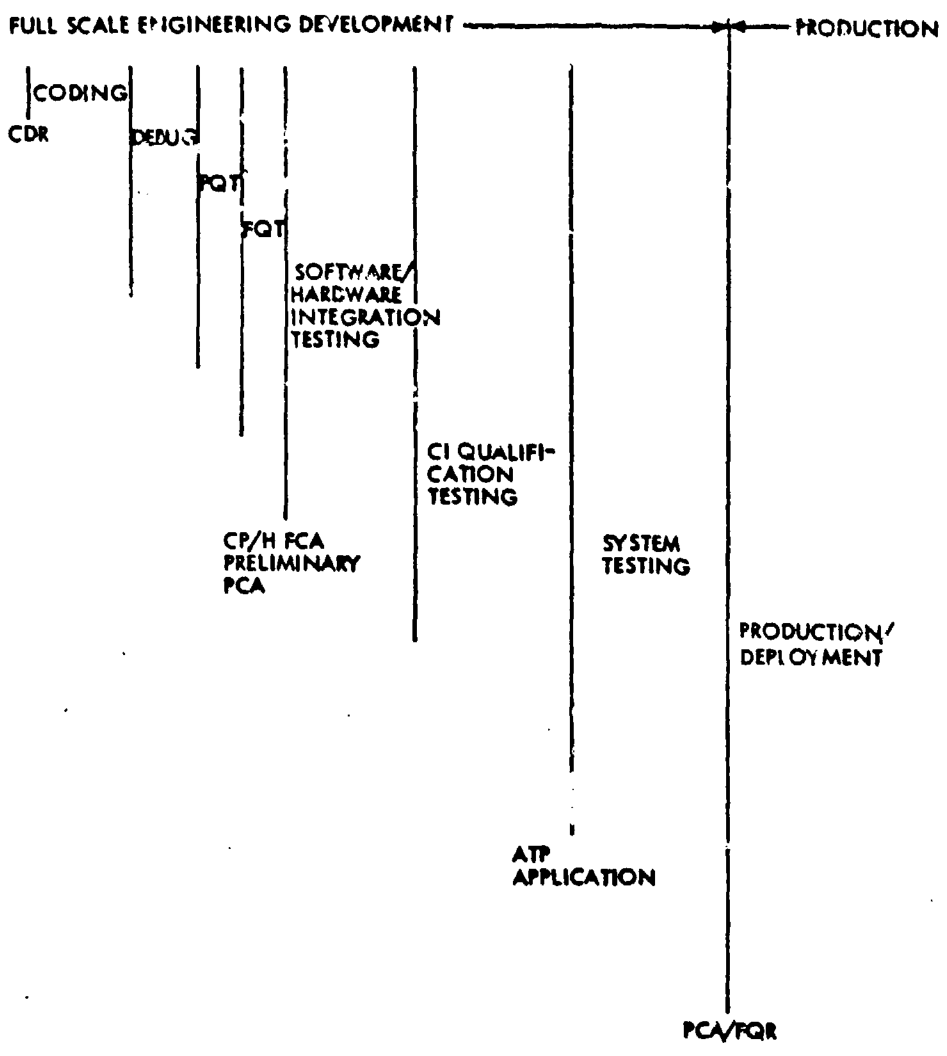


FIGURE 4-2: PROPOSED AUDIT/REVIEW EVENTS FOR MICROPROCESSOR SOFTWARE

TABLE 4-1 IDENTIFICATION METHODS FOR MICROPROCESSOR DEVICES

| Method | Remarks |
|---|---|
| 1. Stick-on Tape | 1. Relatively easy to use.<br>2. Usually not harmful to microprocessor device.<br>3. Dissolves or slides off when heated. |
| 2. Ink Numbering | 1. Easy to read and rub off.<br>2. Some inks produce static charges.<br>3. Difficult to record numbers in space provided. |
| 3. Burn and Bag | 1. Unlabeled microprocessor device is placed in a container which is subsequently labeled.<br>2. If item is separated from container, identity is lost. |
| 4. Color Coding | 1. Resistor color code can be used.<br>2. Color code is confusing.<br>3. Government agencies object to it. |
| 5. Decals | 1. Easy to apply and read.<br>2. Can produce static charge.<br>3. Few vendor sources. |
| 6. Higher Assembly Labeling | 1. Unlabeled microprocessor device is installed on printed circuit board which is subsequently identified. |

TABLE 4-2 FCA CHECKLIST FOR MICROPROCESSOR SOFTWARE

1. Review minutes and corrective action from previous design reviews.
2. Evaluate Contractor's briefing of functional requirements and test results.
3. Audit Preliminary Qualification Test (PQT) and Formal Qualification Test (FQT) reports against actual test data and procedures.
4. Validate test report(s).
5. Review approved ECPs for implementation status.
6. Review interface requirements and related test results.
7. Identify parameters not verified during PQT and FQT.
8. Identify support software used during development and test.
9. Review deliverable documentation for compliance with applicable Data Item Description (DID).
10. Insure that pre-FCA checklist is completed.
11. Review failure modes and effect verification data.
12. Review PCA procedures and requirements, if FCA and PCA are not held jointly.

TABLE 4-3 PCA CHECKLIST FOR MICROPROCESSOR SOFTWARE

1. Review minutes and corrective actions from previous design reviews and audits.
2. Compare microprocessor software listing with narratives (HIPOs, flow charts, etc) in C5 specification.
3. Compare narratives with requirements in B5 specification.
4. Review deliverable documentation for compliance with applicable Data Item Description (DID).
5. Review interface control documentation for currency.
6. Review approved ECPs for implementation status.
7. Review applicable deviations and waivers.
8. Review DD250.
9. Reveiw Acceptance Test Plan (ATP) for currency.
10. Review nomenclature/numbering scheme.

TABLE 4-4 FOR CHECKLIST FOR MICROPROCESSOR SOFTWARE

1. Review the minutes of previous reviews and audits.
2. Review system test results to insure that imposed requirements were satisfied.

TABLE 4-5 CHECKLIST FOR MINUTES OF MICROPROCESSOR SOFTWARE AUDITS AND REVIEWS

1. Corrective action completion dates are established and recorded for all discrepancies revealed.
2. All requirements not tested are recorded.
3. Revision level of all deliverable documents are recorded.
4. Unacceptable recommendations and the reason for rejection are recorded.

## D-5: LIBRARY CONTROLS

Subpanel members were as follows:

| | |
|---|---|
| * Hall, Harold D. | Texas Instruments |
| Lett, Alva F. | General Dynamics |
| Negrelli, Thomas J. | HQ AFLC/LOEE |
| Reinish, Peter | Hughes Aircraft |
| Schnackenberg, Walter | General Electric |

* Subpanel Chair

The objectives of this subpanel were as follows:

(a) Review EIA Bulletin 4C, Computer Software Libraries, and determine its applicability to microprocessor software.

(b) Prepare a flow chart which depicts the activities required to implement library controls for microprocessor software.

(c) Prepare a checklist that can be used to assure the adequacy of library controls for microprocessor software.

The subpanel examined library control requirements for microprocessor software that is treated as a:

(a) Configuration Item (CI),

(b) Computer Program Configuration Item (CPCI), or

(c) Software within a hardware CI.

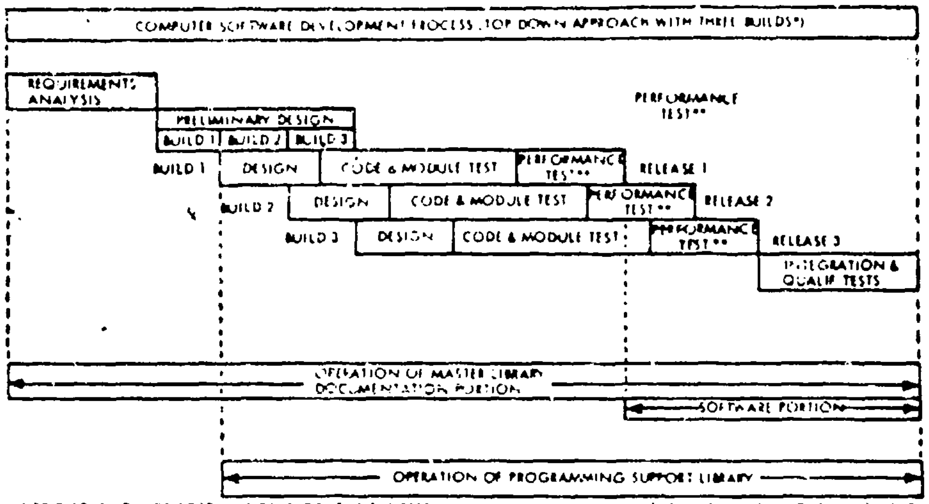
Subpanel members concluded that:

(a) EIA Bulletin 4C is not applicable when microprocessor software is treated as a CI. Hardware disciplines would apply.

(b) When microprocessor software is treated as a CPCI or software within a CI, EIA Bulletin 4C is applicable.

(c) Figure 5-1, Master Library and Programming Support Library Phasing, and Figure 5-2, Computer Software Libraries and Repository Relationships, depict the activities required to implement library control for microprocessor software. These flows were extracted from EIA Bulletin 4C.

(d) Table 5-1 presents a checklist that can be used to assure the adequacy of microprocessor software library controls.



*FOR DISCUSSION OF SOFTWARE DEVELOPMENT PROCESS SEE EIA BULLETIN 48, "CONFIGURATION IDENTIFICATION FOR DIGITAL COMPUTER PROGRAM"

** IN MANY ORGANIZATIONS, A "TURNOVER" FROM THE DEVELOPMENT TEAM TO A TEST TEAM OCCURS AT THE BEGINNING OF THIS TEST PERIOD. "RELEASE" SHOULD TAKE PLACE BEFORE THE TEST TEAM STARTS ITS PERFORMANCE TESTING.

FIGURE 5-1: MASTER LIBRARY AND PROGRAMMING SUPPORT LIBRARY PHASING

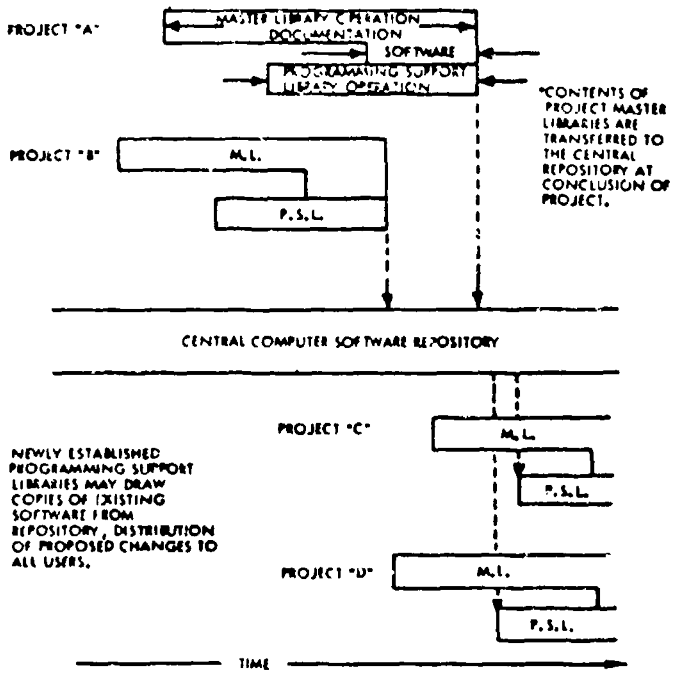

PROJECT "A"
MASTER LIBRARY OPERATION
DOCUMENTATION
SOFTWARE
PROGRAMMING SUPPORT LIBRARY OPERATION
*CONTENTS OF PROJECT MASTER LIBRARIES ARE TRANSFERRED TO THE CENTRAL REPOSITORY AT CONCLUSION OF PROJECT.
PROJECT "B"
M.L.
P.S.L.
CENTRAL COMPUTER SOFTWARE REPOSITORY
PROJECT "C"
M.L.
P.S.L.
NEWLY ESTABLISHED PROGRAMMING SUPPORT LIBRARIES MAY DRAW COPIES OF EXISTING SOFTWARE FROM REPOSITORY, DISTRIBUTION OF PROPOSED CHANGES TO ALL USERS.
PROJECT "D"
M.L.
P.S.L.
TIME

FIGURE 5-2: COMPUTER SOFTWARE LIBRARIES AND REPOSITORY RELATIONSHIPS

TABLE 5-1 CHECKLIST FOR MICROPROCESSOR SOFTWARE LIBRARY

Does the microprocessor software library contain:

| | | Yes | No |
|---|---|---|---|
| (a) | Source Code, both for application software and, as applicable, support software? | ___ | ___ |
| (b) | Object Code, both for application software and support software? | ___ | ___ |
| (c) | Load module? | ___ | ___ |
| (d) | Source listing, both for application software and, as applicable, support software? | ___ | ___ |
| (e) | Object listing, both for application software and support software? | ___ | ___ |
| (f) | Test plan, test procedures, test case source media and test reports associated with the qualification of the application software? | ___ | ___ |
| (g) | Software Standards and Procedures Manual? | ___ | ___ |
| (h) | User manuals for both target and host computers (if not contained in another corporate repository)? | ___ | ___ |
| (i) | Configuration identification (specifications and drawings) if not contained in another corporate repository? | ___ | ___ |
| (j) | Version Description Document or equivalent (identifies all items which make up the version plus all utility and/or support items/modification levels which are required to operate, load or regenerate the application software)? | ___ | ___ |
| (k) | Identification of equipments/modification levels, including both target and host computers, which were used for testing of the version (if not contained in test documentation or VDD as discussed in item j above)? | ___ | ___ |
| (l) | Release Procedure? | ___ | ___ |
| (m) | Change Control Procedure? | ___ | ___ |
| (n) | Final Release Procedure? | ___ | ___ |
| (o) | Backup File Procedure? | ___ | ___ |

## PANEL RECOMMENDATIONS

1. For Industry

   (a) Develop procedures that would be applicable when both hardware-intensive (Firmware) and software-intensive (microprocessor software) reside in the same CI.

   (b) Develop guidelines for the erasure of microprocessor devices containing classified software.

2. For Industry Associations

   (a) Prepare a Data Item Description (DID) for the computer program/hardware (i.e., book-form drawing, that can be called for as a Contract Data Requirements List (CDRL) item). This DID should not require authentication since it is a part of the drawing system defined by DoD-D-1000.

3. For Government

   (a) Revise MIL-STD-483 to allow the use of a Type "A" specification when software-intensive CPCIs are embedded within a hardware CI.

   (b) Revise MIL-STD-130 to include part number marking requirements for microprocessor devices containing software.

APPENDIX C-4

# JOINT LOGISTICS COMMANDERS

## MONTEREY II WORKSHOP

## PANEL B: HARDWARE/SOFTWARE/FIRMWARE

## CONFIGURATION ITEM SELECTION CRITERIA

DYNAMICS RESEARCH CORPORATION
WILMINGTON, MASSACHUSETTS 01887

## OUTLINE

- ISSUES
- OBJECTIVES
- REFERENCE MATERIAL
- SELECTION GUIDELINES (MIL STD 483)
- CI/CPCI DESIGNATION IMPLICATIONS
- AFFECTS OF IMPROPER SELECTION
- JLC DEFINITIONS
- JLC POLICY
- SOME CRITERIA

## PANEL B: ISSUES

- WHAT IS AN APPROPRIATE LEVEL OF CONFIGURATION ITEM IDENTIFICATION (HARDWARE/SOFTWARE/FIRMWARE) FOR A GIVEN ACQUISITION PROGRAM?

- WHEN SHOULD FIRMWARE BE TREATED AS SOFTWARE AND WHEN SHOULD IT BE TREATED AS HARDWARE?

## PANEL B: OBJECTIVES

- CHARACTERIZE PROBLEM SO IT CAN BE SOLVED.
- DEVELOP CRITERIA TO AID THE SELECTION PROCESS.
- *INFLUENCE ACQUISITION STANDARDS UNDER DEVELOPMENT.*

## PANEL B: REFERENCE MATERIAL

| | | |
|---|---|---|
| MIL STD 483 | MARCH 21, 1979 | CONFIGURATION MANAGEMENT PRACTICES FOR SYSTEMS, EQUIPMENT, MUNITIONS, AND COMPUTER PROGRAMS |
| ESD-TR-77-254 | AUGUST 1977 | AN AIR FORCE GUIDE TO COMPUTER PROGRAM CONFIGURATION MANAGEMENT |
| AFSCP 800-7 | DECEMBER 1, 1977 | CONFIGURATION MANAGEMENT |
| UDI-E-3935 | SEPTEMBER 14, 1979 | FIRMWARE DEVELOPMENT PLAN DATA ITEM DESCRIPTION (DID) |
| UDI-E-3936 | MARCH 30, 1979 | FIRMWARE TECHNICAL DESCRIPTION (PRODUCT SPECIFICATION) DID |
| UDI-E-3937 | MARCH 9, 1979 | FIRMWARE SUPPORT DATA DID |
| EIA G-33 PANEL D REPORT | OCTOBER 1980 | TAILORING CM PRACTICES FOR MICROPROCESSORS AND FIRMWARE |

## PANEL B: SELECTION GUIDELINES (MIL STD 483)

- CI SELECTION REFLECTS OPTIMUM MANAGEMENT LEVEL DURING ACQUISITION . . . ONE AT WHICH PROCURING ACTIVITY SPECIFIES, CONTRACTS FOR, AND ACCEPTS INDIVIDUAL ELEMENTS OF A SYSTEM.

| <u>CI CONSIDERATIONS</u> | <u>CPCI CONSIDERATIONS</u> |
|---|---|
| LEVEL OF GOVERNMENT CONTROL | DIFFERENT FUNCTIONAL TYPES |
| ENGINEERING RELEASE SYSTEM | MULTIPLE SYSTEM USAGE |
| SAFETY | INTERRELATIONSHIP WITH OTHER PROGRAMS |
| EXISTING OR MODIFIED DESIGN ITEMS | SEPARATE SUPPLIERS |
| INTERFACE WITH GFE | SEPARATE TARGET COMPUTERS |
| SUSCEPTIBILITY TO CHANGE | SEPARATE DEVELOPMENT SCHEDULES |
| REPAIRABILITY/MAINTAINABILITY | DIFFERENT OPERATING LOCATIONS |
| SUPPORT EQUIPMENT CONSIDERATIONS | DIFFERENT SUPPORT AGENCY |
| SUBASSEMBLY CHARACTERISTICS | |

## PANEL B: IMPLICATIONS OF CI/CPCI DESIGNATION

- FORMAL CONFIGURATION IDENTIFICATION
- DEVELOPMENT/PRODUCT SPECIFICATION
- GOVERNMENT APPROVAL OF CHANGES
- CONFIGURATION STATUS ACCOUNTING RECORDS
- INDIVIDUAL DESIGN REVIEW ACTIVITY
- INDIVIDUAL QUALIFICATION TESTING
- INDIVIDUAL PHYSICAL AND FUNCTIONAL CONFIGURATION AUDITS
- DISCRETE ECP HANDLING
- SEPARATE IDENTIFICATION, MARKING
- SEPARATE OPERATING AND USER MANUALS

## PANEL B: AFFECTS OF IMPROPER SELECTION

| TOO MANY | TOO FEW |
|---|---|
| INCREASED ADMINISTRATIVE BURDEN | LACK OF VISIBILITY |
| INCREASED DEVELOPMENT TIME AND COST | INADEQUATE CONTROL |
| INCREASED MANAGEMENT EFFORT | LOGISTICS AND MAINTENANCE DIFFICULTIES |
| UNNECESSARY PAPERWORK | |
| [INCREASED COMPLEXITY OF INTERFACE SPECIFICATION] | |
| [TOO MUCH DETAIL EARLY ON] | |

## PANEL B: PROPOSED JLC DEFINITIONS

FIRMWARE: ANY LEVEL OF COMPUTER PROGRAMS AND/OR COMPUTER DATA THAT CANNOT BE MODIFIED UNDER COMPUTER PROGRAM CONTROL; THAT IS, READ ONLY. THE DEFINITION ALSO APPLIES TO READ-ONLY DIGITAL DATA THAT MAY BE USED BY ELECTRONIC DEVICES OTHER THAN DIGITAL COMPUTERS.

HARDWARE-INTENSIVE: A COMPUTER RESOURCE SUPPLIED BY THE CONTRACTOR TO SATISFY A HARDWARE SPECIFICATION. IT IS NOT REPROGRAMMED BY THE GOVERNMENT AND IT IS REPLACED (POSSIBLY BY A REPROGRAMMED UNIT) RATHER THAN MODIFIED IN RESPONSE TO A CHANGE IN THE SPECIFICATION.

SOFTWARE-INTENSIVE: A COMPUTER RESOURCE SUPPLIED IN RESPONSE TO A SPECIFICATION IN WHICH REPROGRAMMABILITY IS A SYSTEM REQUIREMENT. THE ENTIRE SYSTEM (HARDWARE PLUS FIRMWARE) IS MAINTAINED BY THE GOVERNMENT. FULL CSCI DOCUMENTATION IS PROVIDED BY THE CONTRACTOR.

## PANEL B: PROPOSED JLC POLICY

CRWG DETERMINES ACQUISITION AND SUPPORT APPROACH FOR FIRMWARE

- HARDWARE INTENSIVE – MANAGED SIMILAR TO HARDWARE: CHANGES AFFECTED BY REPLACING PHYSICAL UNIT CONTAINING FIRMWARE

- SOFTWARE INTENSIVE – MANAGED SIMILAR TO SOFTWARE:

  LIFE CYCLE SUPPORT EMPHASIZED

  HOL AND ISA REQUIRED

## PANEL B: SOME CRITERIA

- LOCATION
- CRITICALITY
- FUNCTION
- MAINTENANCE CONCEPT
- INTERRELATIONSHIP WITH OTHER PROGRAMS
- SUPPLIER
- USE

APPENDIX C-5

PROPOSED

EIA

CP/H BOOK FORM DRAWING

| SOFTWARE INTENSIVE | | | HARDWARE INTENSIVE |
|---|---|---|---|
| CPCI | | | CP/H |
| B5 SPEC<br>C5 SPEC | MIL-STD-490 | DOD-D-1000 | BOOK FORM DRAWING<br>1. DESIGN DESCRIPTION<br>A. NARRATIVE<br>B. FLOWS<br>C. TRUTH TABLES<br>2.. LISTING |
| CPCI TEST PLAN<br>TEST PROCEDURE<br>TEST REPORT | | | |
| | | | CI ACCEPTANCE TEST SPEC<br>ACCEPTANCE TEST PLAN<br>ACCEPTANCE TEST PROCED<br>ACCEPTANCE TEST REPORT |
| | | | OTHER<br>PRINTED CIRCUIT ASS'Y DWG<br>SPEC OR SOURCE CONT DWG |

FIGURE 1-3: DOCUMENTATION FOR SOFTWARE WITHIN A CONFIGURATION ITEM

## APPENDIX C-6

**PROPOSED**

**FIRMWARE DEVELOPMENT PLAN**

**DID**

| | | |
|---|---|---|
| **DATA ITEM DESCRIPTION** | 2. IDENTIFICATION NO(S) | |
| | AGENCY | NUMBER |
| 1. TITLE<br>Firmware Development Plan (FDP) | USAF | UDI-E-3935 ASD/M2 |

3. DESCRIPTION/PURPOSE

The Firmware Development Plan (FDP) is that engineering management plan which identifies the contractor's efforts required to develop and deliver computer resources and the associated firmware, processes, documentation, and necessary support resources. The plan is used by the contracting activity to monitor and evaluate the contractor's firmware computer resource development effort and to plan the support of these resources.

4. APPROVAL DATE

14 September 1979

5. OFFICE OF PRIMARY RESPONSIBILITY

AFSC/ASD/ENETC

6. DDC REQUIRED

8. APPROVAL LIMITATION

For use at ASD/ENETC and ASD/YW

7. APPLICATION/INTERRELATIONSHIP

This Data Item Description applies to system development and acquisition contracts involving computer resources during the phases of validation, full scale development and production. This plan is related to the Computer Program Development Plan (CPDP), configuration management plan and test plan. The FDP and CPDP together cover the full spectrum of computer resources being acquired. Furthermore, this plan is closely related to contractual requirements covering the System Engineering management requirements and documentation. UDI-E-3936-ASD and UDI-E-3937-ASD are companion data items.

9. REFERENCES (Mandatory as cited in block 10)

AFR 800-14
MIL-STD-1521
MIL-D-83468
DI-S-30567A
UDI-E-3936-ASD
UDI-E-3937-ASD

MCSL NUMBER(S)

10. PREPARATION INSTRUCTIONS

10.1.a The FDP is a documented engineering management plan that shall define the contractors efforts to develop, document, and deliver computer resources in accordance with the contract terms. As a management plan, the FDP shall analyze and structure the work into manageable elements (components, modules, tasks, sub-systems, etc.), schedule the progress of these elements, define milestones and measureable products for each milestone or activity during the development process through delivery of the product (system). The FDP shall identify the organizational structure to perform the tasks, the interface with other groups, and the resources allocated to the work effort. Thus, the FDP shall provide effective management visibility, monitoring and control of the product development, test, integration, acceptance and delivery.

10.1.b The FDP shall be coordinated with the CPDP (DI-S-30567A) in terms of resources and schedule. This FDP shall apply to computer programs, computer data, microprocessor, digital electronic processor, and firmware developments and applications or implementations defined herein as Hardware Intensive. The balance of these applications and developments defined as Software Intensive shall be addressed in the CPDP.

10.2 As a minimum, the plan shall include the following:

10.2.a The contractor shall provide definition of the microprocessor, Digital Electronic Processor (DEP) and firmware intended to be developed and covered in this FDP (that is hardware intensive applications) as a supplement to the definitions provided herein. These definitions shall be used in identifying the applications covered in this FDP (as required by section 10.2.b).

10.2.b Firmware Identification. All planned/known application of microprocessors, DEP, and firmware elements in the system shall be identified and described as to functions performed and implementation technology. An overall block diagram and table shall be provided which identifies all microprocessor, DEP, and firmware implementations in the system.

10.2.c Firmware Development Process. The FDP shall include a complete definition and description of the step-by-step process of implementations or applying microprocessors, DEPs, and firmware in the development of this system development. A diagram of phased development/process steps shall be included. Each development/process step shall be completely described including the input baseline from the previous development/process step, the output to be generated by the activity, and the exact nature of the activity (see Figure 1). Identify specific milestohes, integral to this process, which you will use to manage and track the schedule (required by 10.2.e). Where appropriate the milestones identified in the CPDP (DI-S-30567A, Sec 10.4.a) may be used herein. Identify your method of integrating and verifying the status of these milestone products into your management control system and cost performance.

10.2.d The structure of the organization responsible to develop the Hardware Intensive application shall be provided. This manning structure shall identify all personnel responsible for and contributing to the definition/development of the applications. The structure, authority, responsibilities, interfaces and lines of communications for all participants shall be indicated. Each element of Hardware Intensive application should be size estimated in terms of instructions and/or data. The total manpower allocated to each element of the firmware shall be identified in terms of man months per month and by skill levels such as analyst, engineer, programmer, etc. Recognizing this development effort to be closely integrated with the total system development effort, the interrelationship and interfaces between the Hardware Intensive application development organization and the other development organizations shall be defined.

10.2.e The plan shall contain a schedule which defines the total development effort. This schedule shall identify the development tasks, milestones, and written documentation product associated with each implementation/application of microprocessors, DEP and firmware. These tasks shall include:

(1) Definition of the functional performance requirements to be implemented (allocation).

(2) Verification of allocated functional performance requirements.

(3) Algorithms, equations, and data structures development and their derivations.

(4) Verification of the derived algorithms, equations, and data against the functional performance requirements.

(5) Derivation of the design through detail flow process, specific data content, and data addresses (implementation of the design).

(6) Definition of Test criteria and test procedures to verify correct performance once implemented.

(7) The process the coding of digital data (manual or otherwise conversion of the data, i.e., micro instructions into bit patterns and the allocation of the data to firmware).

(8) Implementation of the writing of information, the code and/or data into the firmware device.

(9) Verification that the programmed device correctly performs the intended function.

(10) Integration (incremental build-up) of the firmware elements into the subsystems and system (including measurable criteria for determining completion).

The scheduled tasks shall include the manhours planned to do the task and the specific planned personnel by skill level assignments. These tasks shall be scheduled and tracked with start and completion milestone dates. It is essential that these milestones be finite and measurable (written product). This schedule shall be consistent with the cost/schedule/reporting information required by the contract. Information identified for each item in this schedule shall include scheduled completion of the documentation milestones (e.g. detail design solutions must be documented at the Critical Design Review milestone). The contractor's approach to track and report progress relative to this development schedule shall be identified.

10.2.f The planning and procedures for firmware cost/schedule reporting shall be identified. Specific methods, reporting formats, lowest level planning status reporting tools shall form the baseline for cost performance reporting.

10.2.g Planning for testing shall be identified to include specific plans for all Hardware Intensive applications. Test plans shall cover both contractor verification methods and scheduled activities and Air Force verification. These plans should be integrated with the simulator system level test plans, but must cover the Hardware Intensive applications performance.

10.2.h Methods and procedures planned to develop firmware product specification information to meet the requirements identified in the Contract Data Requirements List shall be provided. The successive milestone build-up of the firmware development documentation in terms of design solutions is critical and must be planned and achieved in order to meet the scheduled design reviews. Detail design solutions, flow charts, process flows, or equivalent information processing designs must be planned to be complete for Critical Design Review.

10.2.i Methods and procedures to accomplish verification of each development step prior to preceeding to the next development step shall be identified. For example, the flow process or equivalent design must be verified against the algorithm and/or Part I specification prior to coding. Techniques used for design control to assure completeness, validity, traceability to requirement, testability, modularity and compliance with internal standards shall be identified.

10.2.j Resources, including computer programs, equipment, documentation, and personnel skills required to support the development, modification, and test of the Hardware Intensive applications shall be identified and scheduled.

10.2.k Planning which describes methods to develop and control digital data base information shall be identified. This planning shall include methods to baseline and control source and operational loads and to document these data bases. Levels of baseline control used throughout the development shall be identified.

10.2.1 A description of engineering practices, standards, conventions, design reviews, etc., as they apply to the development and how these standards will be maintained shall be provided. In addition, the methods to be employed to assure adherence to these disciplines shall be described.

10.2.3 Definitions:

Firmware: Solid state electronic devices, i.e., ROM, PROM, EPROM, etc., which contain binary bit patterns (digital information) that cannot be readily modifiable, i.e., read only.

Digital Electronic Processor: An electronic device whose functions or characteristics are programmable at multiple levels, i.e., programmable at the primative microinstruction or higher level (machine instructions). These microinstructions combine to make the microprograms which define the instructions set (macroinstructions or machine instructions) at the user level.

Embedded: (1) physically incorporated into a larger system or subsystem, i.e., integral.

(2) the larger system function is not general purpose data processing.

(3) the application may be hardware intensive or software intensive.

HW Intensive: Those microprocessor/digital electronic processor applications in which the function performed is fixed and for any change to occur a redevelopment of the application function would be required.

[illegible]

**SW Intensive:** Those microprocessor/digital electronic processor applications in which the function can vary or is capable of performing multiple functions within the embedded application (real-time function, calibration function, built-in test, diagnostic).

**Microprocessor:** An integrated circuit chip or chip set which has a fixed instruction set at the micro or machine instruction level and which has a fixed architecture.

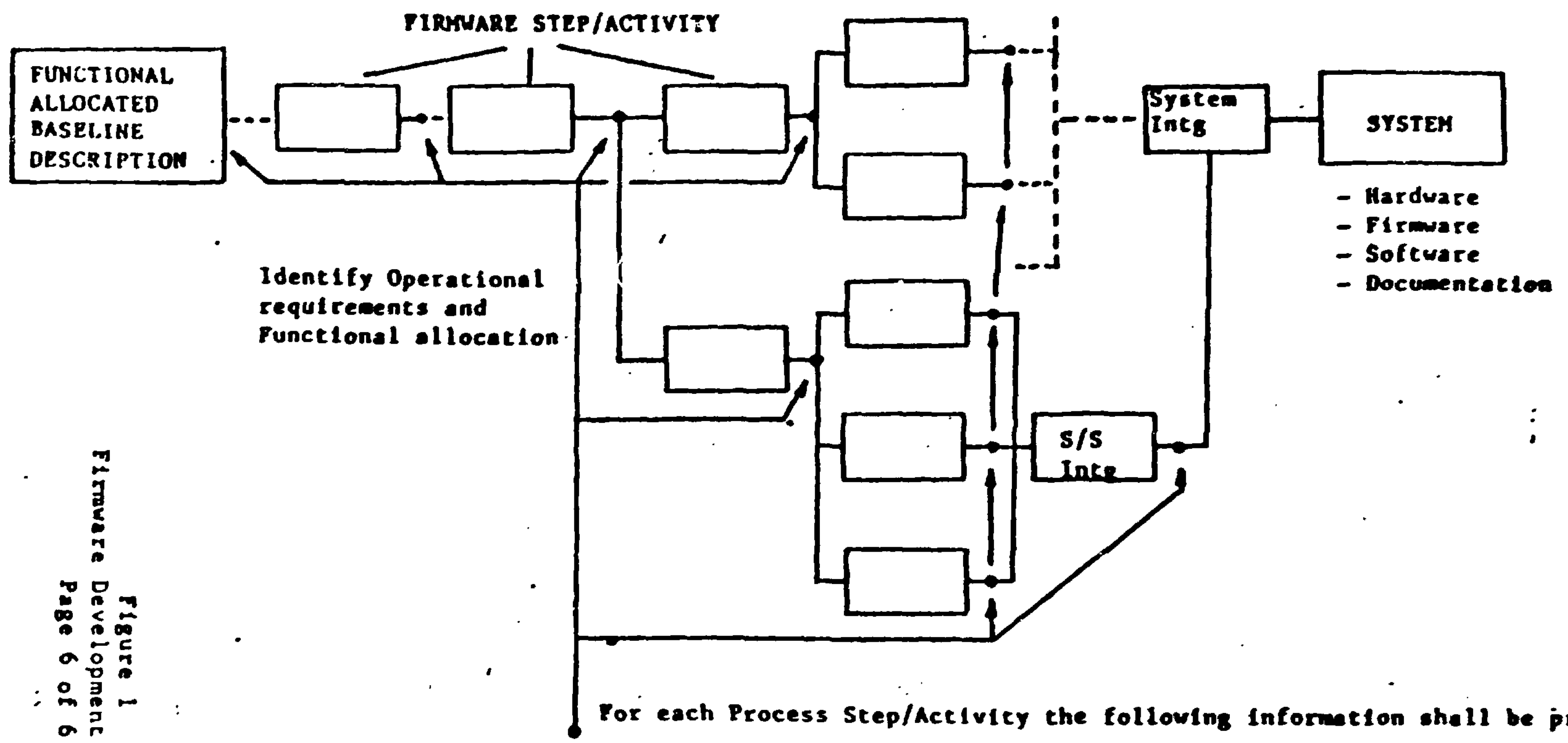
**Programmable microprocessor:** A microprocessor where the instruction set is programmable (modifiable) at the microinstruction level and whose architecture is therefore defined by those microinstructions.

**Microinstruction:** A bit pattern stored in high speed memory (normally nonvolatile) which controls the processor hardware logic.

**Machine instruction:** A bit pattern that is interpreted by the executing control hardware of a processor and is made up of a sequence of microinstructions.

**Macroinstruction:** A nmemonic instruction which causes an HOL computer program to generate one or more machine instructions.

**HOL:** Higher Order Language.


FIRMWARE STEP/ACTIVITY
FUNCTIONAL ALLOCATED BASELINE DESCRIPTION
Identify Operational requirements and Functional allocation
System Intg
SYSTEM
- Hardware
- Firmware
- Software
- Documentation
S/S Intg

For each Process Step/Activity the following information shall be provided

- Input Baseline (identify required inputs, design documents/ specifications/test data, etc.)
- Product of Process Step/Activity (Measurable, i.e., identify documentation, Test results, etc.)
- Traceability (of the process or product to the requirement/ allocation)
- Scheduled duration (see paragraph 10.2.e)
- Verification (How? identify reviews conducted, Who? Responsibility for Sign-Off)
- Test Activity (identify planning for subsequent testing, unit/ sub-system/system tests)

Figure 1
Firmware Development Process
Page 6 of 6

APPENDIX C-7

PROPOSED

FIRMWARE TECHNICAL DESCRIPTION

DID

**DATA ITEM DESCRIPTION**

| | | |
|---|---|---|
| 1. TITLE<br>Firmware Technical Description (Product Specification) | 2. IDENTIFICATION NO(S).<br>AGENCY<br>USAF | NUMBER<br>UDI-E-3936-ASD |

3. DESCRIPTION/PURPOSE

Provides complete and detailed technical description of each functional implementation of firmware. The document(s) serve as an instrument for acceptance, modification, trouble diagnosis, maintenance, modification and reprocurement of the firmware.

4. APPROVAL DATE

30 March 1979

5. OFFICE OF PRIMARY RESPONSIBILITY

AFSC/ASD/SD24E

6. DDC REQUIRED

8. APPROVAL LIMITATION

For use at ASD/SD24 and ASD/ENETC

7. APPLICATION/INTERRELATIONSHIP

This description serves to fully document all available information for each functional implementation of firmware to be delivered. It is developed simultaneously with the firmware and is used incrementally to measure development process at the various milestones. It includes flow charts and detailed descriptive text needed by the system programmer and data analyst in support of the delivered firmware. DI-E-3120, UDI-E-3935-ASD and UDI-E-3937-ASD are companion Data Items.

9. REFERENCES (Mandatory as cited in block 10)

AFR 800-14
MIL-STD-1521
MIL-D-83468

MCSL NUMBER(S)

10. PREPARATION INSTRUCTIONS

10.1 Prepare a complete comprehensive technical description of the firmware to be delivered under this contract.

Definitions: Firmware, most commonly defined as computer programs and computer data at the microprogram level, also applies to any level of executable computer programs and computer data that cannot be readily modified under program control, that is read only. For the purpose of this data item description, firmware is defined to include the above definitions and, in a broader sense, will include all information processing implementation technologies, programs, digital data, and devices not included under the definition of digital computers and associated computer programs, and not included under hardware. Firmware includes microprocessors, Read Only Memories (ROMS), Programmable Read Only Memories (PROMS), and any other programmable logic elements. The contractor shall expand from these generic definitions to specific definitions of the firmware intended to be covered by the documentation technical descriptions in response to this data item description.

10.2 Each firmware implementation (e.g. individual or grouped PROM(s), ROM(s)) shall be identified and entitled separately by function performed and described in relation to the associated configuration item of which it is a part. The technical data shall be comprised of the following:

Preparation Instructions (Cont'd)

a. Identification of the performance requirements and functions to be implemented. This shall include test criteria and requirements.

b. A narrative description of the functions to be performed, the inputs and outputs associated with the functions, and the hardware and software interfaces.

c. A description of the functional characteristic of the devices to be programmed and the types of information structuring, e.g. microinstruction/data formats and instruction type functions. As appropriate, device micro instructions shall be defined.

d. A complete description of the firmware device(s) to include memory size (length, width in bits), operating characteristics (access time, power supply/requirements, logic levels, etc.), pin functional descriptions and logical interfaces, and manufacturers part number.

e. A complete definition of the algorithms, equations and data structures to be implemented. Flow processes and data formats of implementation shall be described at the detailed functional level.

f. Logic layout schematic type block diagrams. These shall include timing, cycle, and clock information.

g. Detail process flow diagrams which support coding as the next step. Data structures, e.g. look up tables, shall be completely detailed including addresses and hex/decimal equivalents. The derivation of table entries shall be provided including descriptions of the equation/algorithms which generated the data.

h. The source code of instructions and data. Listings of the firmware contents, sequentially by address, in binary, hex or decimal and mnemonic representation or descriptions shall be provided. Listings shall contain comments which accurately correlate to the flow process diagrams used to generate the contents. These listings may take the form of parenthetical drawings of the contents of the firmware.

i. A listing of the firmware contents (i.e., the logic/program contained in memory) in either a binary or mnemonic representation referenced to a logic diagram graphical

Preparation Instructions (Cont'd)

representation. Comments shall be included in the listings except for listings of data tables.

j. Test requirements, procedures and test results.

k. Identification and description of the compiler/assembler systems used to develop the loadable object tapes (or other media). (Vendor format acceptable.)

APPENDIX C-8

# PROPOSED FIRMWARE SUPPORT DATA DID

| DATA ITEM DESCRIPTION | 2. IDENTIFICATION NO(S) | |
|---|---|---|
| | AGENCY | NUMBER |
| 1. TITLE<br>Firmware Support Data | USAF | UDI-E-3937-ASD |

**3. DESCRIPTION/PURPOSE**

Provides the data needed for the maintenance and modification of firmware (Read Only Memories (ROMs) Programmable Read Only Memories (PROMs)) microprocessors, and other firmware devices.

**4. APPROVAL DATE**

9 March 1979

**5. OFFICE OF PRIMARY RESPONSIBILITY**

AFSC/ASD/SD24L

**6. DDC REQUIRED**

**8. APPROVAL LIMITATION**

For use at ASD/SD24 and ASD/ENETC

**7. APPLICATION/INTERRELATIONSHIP**

This data serves to fully document all available information for the maintenance and modification of firmware. It is to be used as: (1) a maintenance manual by the system programmer in support of the delivered firmware, and, (2) engineering documentation for depot level support. DI-E-3120, UDI-E-3935-ASD and UDI-E-3936-ASD are companion Data Items.

**9. REFERENCES (Mandatory as cited in block 10)**

AFR 800-14
MIL-STD-1521
MIL-D-83468

**MCSL NUMBER(S)**

**10. PREPARATION INSTRUCTIONS**

Unless otherwise stated, the following information will be provided separately for each functional implementation (i.e., individual or grouped ROMs/PROMs and other devices) of firmware.

a. The object code for each ROM, PROM or other device (as input to a PROM (device) programmer) on punched paper tape, magnetic tape or other specified medium supplemented with a listing of the object code and a narrative description of the complete contents of the tape or other specified medium (to include the leader, etc.). If the listing is on perforated forms, the pages shall not be separated. The listing shall contain both address and content (in binary, octal or hexadecimal) to each individual (for PROMs and other programmable devices) ROM, PROM or other device. A listing of source language statements/code used at each stage of development shall be provided. All listings shall be annotated to identify computer program identification member (CPIN), version, date of listing system memory location, card memory location, and chip memory location.

b. A Programming Guide (for PROMs and other programmable devices) ROM, PROM or other device shall be provided in contractor format. This guide shall identify all hardware/software and describe the procedures, used for maintenance and support of both the PROMs (devices) and the information content of the PROMs (devices). The content of this guide shall include:

(1) A list of all equipments for each functional implementation

Preparation Instructions (Cont'd)

of firmware including computers and computer peripherals used by the contractor for software development/data base generation, PROM loading, PROM burn-in and PROM test (including verification that the proper content is stored). Each equipment shall be identified by manufacturer, manufacturer's designation, and any special features. Any unique equipments used shall be so specified.

(2) A list of all computer software for each functional implementation of firmware used by the contractor for firmware logic development/data base generation, device, e.g., PROM loading, burn-in and test. All cross-compilers, cross assemblers and utility programs used will be included in this list. Each computer software item shall be identified by vendor, vendor's designation, version, and any special features. Any unique software used shall be so identified.

(3) A sequential list for each functional implementation of firmware of those detailed procedures used by the contractor for firmware logic development/data base generation, device, e.g., description of PROM loading, burn-in, and test. All equipment and computer software used in each procedures shall be identified. All burn-in schedules and conditions used by the contractor shall be included.

(4) A list for each functional implementation of firmware of those tests used by the contractor to validate firmware logic/data base, compatibility of the firmware logic/data base with the system, device (e.g. , PROM) specification and device (e.g., PROM) content after programming. A brief overview of each test will be included. The test method and criteria for acceptance or rejection will be included for each test. Those equipments and computer software used for each test will be identified.

c. Vendor/Implementation Information.

(1) Vendor data as supplied by original supplier and/or vendor which, as a minimum describes in detail, the capabilities and methods of achieving these capabilities for each device.

(2) Pin layout and logic diagram relation to each pin shall be provided.

APPENDIX C-9

## PROPOSED

## CONFIGURATION ITEM PRODUCT FABRICATION SPECIFICATION

## DID

APPENDIX A

| | | |
|---|---|---|
| **DATA ITEM DESCRIPTION** | 2. IDENTIFICATION NO(S) AGENCY: USAF | NUMBER: (DRAFT) |
| 1. TITLE<br>Configuration Item Product Fabrication Specification | | |
| 3. DESCRIPTION/PURPOSE<br>The product configuration identification documentation (specifications and referenced drawings) establish the requirements for manufacture and acceptance of the configuration items (CI) to be delivered under the terms of the contract. | 4. APPROVAL DATE | |
| | 5. OFFICE OF PRIMARY RESPONSIBILITY<br>AFSC | |
| | 6. DDC REQUIRED | |
| | 8. APPROVAL LIMITATION | |
| 7. APPLICATION/INTERRELATIONSHIP The product fabrication specification is used to identify the contractual requirements (product baseline) for the configuration item. The product specification is the documentation to which production/operational engineering change proposals (ECPs) are addressed. The product fabrication specification will normally be prepared as Part II of a two part specification. Part I of the specification will always be the overriding part of the specification. For Computer Program Configuration Items (CPCIs), use DID-E-3120B. If other than Form la specifications(MIL-STD-490) are required, the specific form from MIL-S-83490 will be called out in block 16 of the Contract Data Requirements List (CDRL). For configuration items that contain embedded microprocessor applications, whose computer programs are not separate computer program configuraiton items, the additional requirements contained below in Bock 10 (para.2), Preparation Instructions, shall be included in the specification. For Prime Items that are training equipments, the additional requirements contained below in Block 10 (para 3), Preparation Instructions, shall be included in the specification. | 9. REFERENCES (Mandatory as cited in block 10)<br>MIL-STD-483<br>MIL-STD-490<br>MIL-S-83490 | |
| | MCSL NUMBER(S) | |

10. PREPARATION INSTRUCTIONS

1. The contractor shall prepare a product fabrication specification for each configuration item in accordance with the requirements of MIL-STD-490, Appendices VIII, X and XI, as applicable and as stated in the contract or work statement. When other than Form la specifications are called out in Block 16 of the CDRL, Appendices of MIL-STD-490 will be used as a guide in the preparation of specifications. The specifications cover page shall be in accordance with MIL-STD-483, Figure 1.

2. When the configuration item contains embedded microprocessor applications, an appendix for each microprocessor application shall be included in the configuraiton item product fabriacation specification. This appendix shall contain additional information about the microprocessor application. The section in the product specification which contains the chip schematics and bit pattern drawings shall reference this appendix. Title and numbering of the appendix shall be in accordance with MIL-STD-490. Detailed preparation instructions shall be implemented by the following instructions.

Section 1, Scope: This section shall identify the microprocessor and describe its function within the configuration item.

Section 2, Applicable Documents: References which may be required and which relate to this appendix shall be listed in Section 2 of the basic specification and shall be listed by the title in this section as a minimum. If there are no applicable references the following shall appear below the section heading:

"This section is not applicable to this appendix."

Section 3, Requirements: All of the requirements which are allocated to the microprocessor application shall be clearly stated, either directly or by reference, in thi section. Any requirements derived as a consequence of the application of a microproce in the configuration item shall be included in this section. Derived requirements may be allocated to the digital processing equipment, the microcode, computer program or data, and may include processing rates, memory usage, programming language(s), data structures, etc.

Paragraph 3.1, Instruction Set: Identification of the programming language and instruction set shall be included either directly or by reference in this paragraph. If the microprocessor is a user programmable microprocessor and an instruction set was developed, a complete description of what happens with the execution of each instructi shall be included either directly or by reference, along with a memory map for the instruction set in this paragraph and subparagraphs. If the microprocessor is a user programmable microprocessor, but an instruction set was not developed (i.e., the microprocessor is used as a hardware controller), then a complete description of the control flow sequencing and timing shall be included in this paragraph and subparagraphs. If the microprocessor is user programmable, but an instruction set was not developed, paragrpahs 3.2 through 3.5 of this appendix will not be applicable.

Paragraph 3.2, Functional Flow: This paragraph shall describe the major operation(s performed by the computer program. This paragraph and subsequent subparagraphs shall show the general flow of both data and control within the computer program. Paragraph 3.2 shall graphically portray the operations performed by the computer program. Graphical representations shall be in accordance with MIL-STD-483, Appendix VI.

Paragraph 3.2.1, Interfaces: This paragraph shall describe in detail, either directly or by reference, the as-built interface design between the computer program and the hardware with which it must operate. It shall provide a detailed logical description of all data, messages, and control signals. Sources for all inputs and destinations for all outputs shall be described.

Paragraph 3.2.2, Program Interrupts: This paragrpah shall describe all program interrupts. Each interrupt shall be described as to source, purpose, type, priority, and the required response. The probable rate of occurrence of interrupts shall also be given. If the computer program is loop driven, then loop rates (inner loop rate(s), outer loop rate(s), etc.) shall be described. The effect of interrupts on those loop rates shall also be described.

Paragraph 3.3, Design Description: The design description provided shall be to a level of detail sufficient to permit computer program modification and/or adaptation by the government. For applications specified by the government as Hardware Intensive, the level of detail shall be sufficient to provide for the theory of operation of the Configuration Item and to facilitate applicable maintenance at the hardware level.

Paragraph 3.3.1, Data Base Definition: This paragraph shall include the name, definition, variable type, and range of values for each variable, table, or array used in the computer program.

Paragraph 3.3.2, Storage Allocation: This paragraph shall include a memory map which describes the utilization of all portions of memory. This memory map shall include a description of how memeory has been partitioned and allocated to different section of the computer program(s) and/or data.

Paragraph 3.4, Object Code Creation: This paragraph shall describe the computer programs, digital processing equipment, hardware and procedures required to be utilized to generate the object code.

Preparation Instructions (Continuation)

a. Computer programs required, including but not limited to host operating system, compilers, assemblers, link editors, loaders, and translators.

b. Digital processing equipment including host computer hardware minimum configuration, vendor(s), part numbers, and other pertinent data such as model capacity and special capabilities.

c. Procedures required to generate object coce in the specified media suitable for operation in the target microprocessor.

Paragraph 3.5, Listings: This paragraph shall include source and object code listings. Comments of explanation shall be included and considered part of the listing. The level of detail required in the listings shall, when used with other information contained within this appendix, permit support, modification and/or adaptation of the computer programs and data utilized within the microprocessor.

Section 4, Quality Assurance Provisions: Requirements for formal verification of the microprocessor and associated computer program requirements of Section 3 and Section 5 shall be included in this section. This section shall reference each performance,design or delivery requirement of Section 3 and Section 5 and specify the specific verification method. The methods of verification to be specified herein may include inspection, review of analytical data (analysis), demonstration(s), and test (witnessing test results and review of qualitative and quantatitive test data resulting from formal procedures, controlled environment and instrumentation). This section shall specify requirement(s) for verification to the level of detail necessary to establish the scope and accuracy of the verification method, and shall clearly identify each verification requirement with the applicable performance/design/ delivery requirement in Sections 3 and 5.

Paragraph 4.1 Microprocessor/Computer Program Testing: This paragraph describes testing at the microprocessor/computer program-level where the results are intended to be the only source of data to verify specific requirements of Section 3 or 5. These tests may be conducted prior to integration of the microprocessor into the configuration item.

Paragraph 4.2, Configuration Item Testing: This paragraph shall identify requirements specified in Section 3 which will be verified during Configuration Item testing and which therefore must be listed as CI test requirements.

Paragraph 4.3, System Test Program: This paragraph shall identify requirements specified in Section 3 which cannot be verified until system testing (or equivalent) and which therefore must be listed as system test requirements.

Paragraph 4.4, Verification Cross-Reference Matrix: This paragraph shall include a Verification Cross-Reference Matrix identifying each Section 3 requirement with the Section 4 paragraph where the qualification requirement(s) is/are specified.

Section 6, Preparation for Delivery: When applicable, this paragraph shall be used to describe special handling requirements such as: packaging for delivery (e.g., to ships or to remote sites), which may require special labels, etc. This paragraph shall also specify the media or delivery, e.g., cassettes, cartridges, magnetic tape, punched paper tape, punched cards, disks, firmware, etc. Also included shall be the general or specific characteristics of the media as required for qualification testing and verification. If special or unique packaging is required to avert possible compromise of CI performance, then packaging requirements shall be included.

Section 6, Notes: This section of the specification is not contractually binding, and should not contain contractual requirements. It may include information or particular importance to the procuring activity in using the specification as a contractual instrument or administrative or background information (e.g., ordering instructions for technical data pertianing to the computer program, or specific information related to the use of the program in future assembly and integration testing). It shall not include requirements which constrain design, development, or qualification. The text may be preceded with the statement "Administrative Information Only - Not Contractually Binding." Background information or rationale which will be of assistance in understanding the specification itself, may be included herein. The procuring activity must specify material which is mandatory for inclusion in Section 6.

3. When the prime item is for training equipment, the prime item product fabrication specification preparation instructions in Appendix VIII, MIL-STD-490 shall be implemented by the following instructions:

APPENDIX VIII

80. TYPE, Clb, PRIME ITEM PRODUCT FABRICATION SPECIFICATION.

80.1 Section 1, "Scope." No Change

Example:

1. Scope.

1.1 "Scope." This section shall also briefly describe the intended use of the trainer, by whom, and where it is intended to be used; e.g., contractors' plant, ATC base, etc.

1.2 "Classification." No change.

80.2 Section 2, "Applicable documents." No change.

80.3 Section 3,"Requirements."

80.3.1 Paragraph 3.1, "Item definition." No change

80.3.1.1 Paragrpah 3.1.1, "Major component list." Add "This paragraph shall list the major items of equipment required to operate with a trainer in an instruction situation by the following categories:

"Contractor furnished system operational equipment."

"Government furnished equipment (by expected soruce)."

80.3.1.2 Paragraph 3.1.2, "Government furnished property list." No change

80.3.2 Paragraph 3.2, "Characteristics." No change.

80.3.2.1 Paragraph 3.2.1, "Performance." No Change.

Preparation Instructions (Continuation)

80.3.3 Paragraph 3.3, "Design and construction." No change.

80.3.3.1 Paragraph 3.3.1, "Production drawings." No change.

80.3.3.2 Paragraph 3.3.2, "Standards of manufacturer." No change.

80.3.3.3 Paragraph 3.3.3, "Workmanship." No change

80.3.4 Paragraph 3.4, "Reproduction sample." Not required.

80.4 Section 4, "Quality Assurance Provisions." No change.

80.4.1 Paragraph 4.1, "General." Add, "For trainers requiring extensive installation and checkout, the acceptance inspection may be accomplished at the installation site. This does not necessarily preclude an interim acceptance for specified purposes at the place of manufacturer."

80.4.1.1 Paragraph 4.1.1, "Responsibility for inspection." No change.

80.4.1.2 Paragraph 4.1.2, "Special tests and examination." No change.

80.4.2 Paragraph 4.2, "Quality conformance inspections." No change.

80.5 Section 5, "Preparation for delivery." Add, "Markings for items appearing in the Training Equipment List (TEL) shall coincide with their identification in that document. Interior packages and exterior shipping container shall be marked in accordance with specification MIL-STD-129, "Marking of Shipments," and MIL-STD-130, Identification Marking of U. S. Military Property," and the words "Training Equipment" shall be stenciled thereon. Interior packages shall also bear the:

a. TEL item number

b. Stock number

c. Condition status (indicate if item is reject)

Training equipment, including part of trainers, furnished in reject status, shall be permanently marked by stencil, stamp, engraving, or decalcomania: Factor Production Reject.

80.6 Section 6, "Notes." No change.

80.6.1 Paragraph 6.1, "Intended use." No change.

80.6.2 Paragraph 6.2, "Ordering data." Not required.

80.7 Section 10, "Appendix I." No change.

# REPORT OF THE PANEL ON STANDARDIZATION AND ACCREDITATION OF COMPUTER ARCHITECTURE

TAB C
Standardization & Accreditation

ITT

*International Telephone and Telegraph Corporation*
***Avionics Division***
*Telecommunications and Electronics Group-North America*

*100 Kingsland Road*
*Clifton, New Jersey 07014*
*Telephone (201) 284-0123*

September 14, 1981

LTC Casper H. Klucas
HQ, AFLC/LOEC
Wright-Patterson AFB, Ohio 45431

Dear Cas:

Enclosed is the final report of the panel on Standardization and Accreditation of Computer Architecture. It has been most gratifying to work with you, the other persons of the CSM and CRM, and all those who participated in the Monterey workshop, and I shall be pleased to continue support of the CSM at your discretion.

Best regards,

[illegible]

Robert H. Dunn
Staff Consultant

encl.
/da
cc: Mr. R. Hartwick
Dr. I. Hecker w/o attachment
Mr. R. Maher
Mr. P. Mauro
Ms. A. Schuman

## PANEL C

## STANDARDIZATION AND ACCREDITATION OF COMPUTER ARCHITECTURE

CHAIRPERSON: R. Dunn
ITT Avionics Division

CO-CHAIRPERSON: I. Hecker
ROLM Corporation

TABLE OF CONTENTS

## REPORT OF THE PANEL ON STANDARDIZATION AND ACCREDITATION OF COMPUTER ARCHITECTURE

## 1. OBJECTIVE

Agency policy frequently dictates the use of standardization. Given that this is a requirement, evaluate the potential for utilizing accreditation of computer architectures as a viable tri-service computer acquisition strategy. Determine the role and impact of MIL-STD-1750A, MIL-STD-1862 and DoD Instruction 5000.5X within the overall defense computer acquisition standards.

## 2. SCOPE

Each service has been acquiring embedded computer systems (versus commercial ADP-type computers) utilizing a myriad of selection criteria. The end user is often required to implement a system with an approved standard, yet technologically obsolete, computer. Accreditation has been promoted as an alternative computer acquisition strategy.

In 1978, the Navy's Office of the Assistant Secretary for Research, Engineering and Systems (ASN, RE & S) assembled a panel to review the status and trends of embedded computer applications and to recommend alternatives to existing acquisition policy. One of the panel's suggestions was the concept of accreditation, which had the following goals:

-- Stimulate competition in production (vs. sole source production)

-- Ease technology insertion

-- Increase flexibility of choice for program managers

-- Shorten the acquisition cycle

-- Minimize cost of ownership

With this as background, the issues relevant to the objective include:

-- The architectural levels at which accreditation can apply

-- Hardware and software logistics support

-- The effect of multiple vendors on the attainment of standards

-- Industry's willingness to support an accreditation policy

-- The capability of an accreditation policy to meet service operational and technological needs

-- Other influences on the flexibility of choice and the length of the acquisition cycle.

-- Administration of evaluation and selection under an accreditation policy

-- The introduction of new architectures as standards

As interpreted by Panel C, all computer acquisitions related directly to weapons systems, operational support, ATE, and the like, fell within its scope. Hardware-intensive processors were, however, excluded. The only HOL considered at the source level of architectural standards was Ada. Where relevant, a given standard was assumed to apply to a single performance envelope (e.g., mini, maxi, large scale).

## 3. APPROACH

Presentations by Lt. Col. Vance Mall of the Ada Project Office, Doug Stapp of ASD/ENASD, and William Smith of the Office of the Assistant Secretary of the Navy (R E & S) established a background for the discussion of standardization and accreditation. See Appendices C thru E.

Early in the panel's discussions, it became evident that the concept of accreditation went beyond the definition originally used by the Navy (see second sheet of Appendix E), which has viewed it in contradistinction to its policy of standardization. For acquisitions directed toward achieving architectural standards at other than the lowest replaceable unit level, accreditation could take on new meanings. Accordingly, the panel decided it must provide a definition to frame the balance of its deliberations. This definition (anticipating the next section of this report) was

"Accreditation is an acquisition strategy by which a product is certified to be suitable for service use in accordance with documented criteria."

With this definition, it became clear that there were, in fact, several accreditation approaches that needed consideration. Moreover, the panel decided that there were two distinct standardization methods that have been used and had to be included in further discussion. These acquisition strategies are defined in para 4.1.

As discussion evolved, it became evident that there were a large number of issues concerning the relative merits of each of the acquisition strategies that had been identified. To better focus on each of these, the panel decided to evaluate each of the acquisition strategies against specific criteria. These criteria (several of which are of more than one part) were

1. Reliability and maintainability
2. Effect on logistics
3. Effect on personnel
4. Sources for development and qualification funding
5. Promotion of a competitive environment
6. Effect of shortening the acquisition cycle
7. Life cycle cost
8. Product availability -- marketplace
9. Capability of achieving technological currency

Response to specific weapons systems requirements was not used as a criterion.

The panel wished, also, to consider the effect of these criteria at four levels of standards: source, object, box, and LRU. These are defined in Section 4 of this report. Although it was recognized that some criteria would apply equally to two or more of these levels, it was clear that the evaluation of the acquisition strategies vs. criteria vs. standard level would be a formidable task. Accordingly, to make the evaluation process more manageable, the panel divided into three sub-panels, with each responsible for a group of related criteria. Sub-Panel 1 tackled criteria 1 thru 3, sub-Panel 2 dealt with criteria 4 thru 6, and sub-Panel 3 was organized to handle criteria 7 thru 9.

The special task of each sub-panel was to evaluate the identified :quisition strategies (plus a sixth, "laissez faire," representing the antithesis of standardization) against each of the criteria assigned to it and at each applicable standard level (Source, et at). This would, in effect, provide an in-depth overview of the entire issue of standardization and accreditation independently of unique service needs.

he last day of the workshop, each of the sub-panels reported its findings to the others. This enabled all panel members to provide input to the identification of the key issues, and to draw attention to the need for clarification of the assumptions underlying each sub-panel's conclusions. Although the resultant three-dimensional array of mini-position papers is, in itself, a set of conclusions, before adjourning, the panel proceeded to synthesize the work accomplished to draw some general conclusions and recommendations.

## 4. DISCUSSION

The bulk of this section is formed by the matrix of acquisition strategy/criteria/level of standard. As will be seen, this matrix consists of a quick reference table and, of greater importance, the premises and remarks attending all but the most obvious entries in the table. Some definitions and additional background which are required for interpretation of the matrix follow in paragraphs 4.1 thru 4.4.

### 4.1 Acquisition Strategies

The following strategies are those evaluated by the panel. Strategies 4.1.2 thru 4.1.5 are accreditation techniques, although strategy 4.1.5 has also been regarded as a standardization method, and is so labeled.

4.1.1 Laissez faire ... So named by the panel because it permits each Program Manager (PM) or his contractor to select whatever computer architecture and implementation he chooses. This is the de facto method of acquisition currently used for many programs. Consideration of this strategy was not within the scope of the panel's task. It is included because it serves as a reference (at the pole opposite that of 4.1.6) against which all other strategies may be compared.

4.1.2 Accreditation I ... In this strategy, the PM has the responsibility for verifying (validating) the implementation of a standard architecture. This technique has been employed by the Air Force.

4.1.3 Accreditation II ... As in the operation of a qualified parts list, computer manufacturers develop implementations of standard architectures and submit them to the government for validation as machines qualified to join others on a published list.

4.1.4 Accreditation III ... In this strategy, the government buys a large number of machines for application to one or more programs; in essence, acting as a central supplier of a computer commodity. It is expected that the development costs of each such implementation of a standard architecture would involve at least partial government funding. This strategy, which has been considered by the Navy, lends itself to multiple developers and producers, with the number of developers expected to be greater than the number of producers.

4.1.5 Standardization I ... Still an accreditation method, in this strategy several potential suppliers are competitively selected and funded to develop an implementation of a given architecture. One of these is subsequently awarded a production contract. A second source production contract is also possible. This strategy is now employed by the Army and Navy.

4.1.6 Standardization II ... A single developer is funded to develop and produce machines conforming to a given architecture. That is, a single supplier is awarded a contract, normally on a competitive basis, for both development and production. Second sourcing is possible here, also.

## 4.2 Architectural Levels to Which Standards Apply

Four levels were defined. It should be noted that for certain criteria, the level was immaterial, as will be seen in the matrix, while for others, it was of critical importance.

4.2.1 Source ... This describes the ability to deliver a computer and (if not capable of direct execution of HOL programs) a compiler such that HOL programs will run predictably. The transportability of assembly language programs is not an issue here.

4.2.2 Object ... Implementation of an object level standard refers to transportability of the assembled or compiled source language. This may be viewed as a run-time architectural standard. HOL standards are implicit.

4.2.3 Box ... This is a hardware standard at the level of form, fit, and function. HOL and ISA standards are implicit.

4.2.4 LRU ... Similar to box level except that the implementation specifics are standard as well for logistics commonality. That is, computers at this level of standards conformance are spared identically. HOL, ISA, and box standards are implicit.

## 4.3 Weighting of Criteria

In general, the several criteria were not rated with regard to relative importance, although, as will be seen, certain criteria were weighted with respect to other criteria. The reason weighting was not attempted derives from the recognition that no weighting scheme can apply equally to all system applications. For example, hardware logistics would be much less important for space probe applications than it would for applications supported by recurring depot level maintenance. Differences among the four services in their support philosophies would also affect weighting. A weighting scheme of suitable generality would require considerably more study than could be undertaken within the time constraints of the workshop.

## 4.4 Interpretation of Tabular Information

The tables found later in this section summarize the issues attending the evaluation of the criteria against the six strategies. For most of the criteria, the evaluation is made at the four levels of standardization with the key to the level to which each evaluation applies given in the upper left hand corner of each table. The numbers accompanying the ratings (very good to very poor) refer to the notes following each table.

The evaluation ratings given (very good to very poor) must be used guardedly. Indeed, several of the panel stated that they would prefer the report not include these, for fear that if quoted, reproduced, or summarized out of the context of the accompanying premises and remarks (Notes 1 thru 67), the ratings could lead to inferences of doubtful correctness or value. _The tabular ratings are presented only as a means of providing a synoptic picture of the influences exerted by each of the strategies._

## 4.5 Discussion Specific to the Criteria of

Reliability and Maintainability

Software Logistics

Hardware Logistics

Military Personnel

Civilian Personnel

### 4.5.1 Amplification of Evaluation Criteria --

Reliability and Maintainability: These were evaluated relative to their effect on the availability of a computer to meet mission requirements and the ease of maintaining computers in the presence of failures. This criterion, then, measures the government's expectations of reliability under each of the acquisition strategies and standardization levels.

Software Logistics: This criterion was evaluated with respect to the effect on support software, to the exclusion of the transportability of applications software.

Hardware Logistics: This is within the context of the existing sparing and maintenance philosophies of the services.

Military Personnel: This relates to the impact on the technicians and system operators in the field. That is, it reflects the potential availability of the talent pool necessary to support the system as well as the level of training required to ensure the capability of maintaining and operating the system.

Civilian Personnel: This factor measures the availability of the necessary pool of talent within industry -- engineers, designers, and technicians -- able to respond to a particular acquisition strategy. Civilian contractors hired to provide services to the government were not considered to be an issue.

4.5.2 General --

1. Sub-Panel 1, to whom the above criteria were assigned, felt that since the laissez faire approach to acquisition does not have associated levels of standardization appropriate to it, and is, in any case, an unacceptable method, it would evaluate laissez faire without regard to levels of standardization, but use it merely as a basis of comparison.

2. The Accreditation I approach was evaluated for standardization only at the source level and at the object level. It really is valid only at the object level since the basis of the approach is an ISA standard and there is no attempt to standardize below the ISA.

4.5.3 Premises --

1. It was assumed by the sub-panel that standardization at the LRU level included standardization at the box and object levels. Likewise, standardization at the box level was assumed to imply standardization at the object level.

2. It was originally assumed by the sub-panel that source level standardization was only viable when high order language (HOL) direct executable machines were within the state of the art. After discussions with sub-Panel 3, it was accepted by sub-Panel 1 that source level standardization would mean that the contractor would deliver a computer and compiler and that HOL programs would run predictably on the machine and ISA or object level programs would not be transportable.

3. It was further assumed by the sub-panel that standardization at the source level means that multiple code generators would be required for target machines since ISA's would not be standard. Standardization at the object level implied CPU ISA standardization and no I/O standardization of any kind. Standardization at the LRU or box level implies I/O standardization and the

equivalent transportability of the HOL and object level software.

4. In terms of sparing philosophy, it was assumed by the sub-panel that when standardization was levied at the box level, the services would also replace and spare at the box level, thereby requiring personnel of lesser skill to maintain the equipment. This is in contrast to standardization at the LRU level where it was assumed that the services would also replace and spare at the LRU level.

5. It was assumed that major improvements in reliability would result from the ability of contractors to use more current technology in the computer design. It was also recognized that major improvements would also occur during a long production run with a fixed technology as a result of learning curve experience and reliability improvement programs; however, no attempt was made to further develop the advantages of one approach over the other.

4.5.4 Ratings Table -- See Figure 4-1.

4.5.5 Notes applicable to Rating Table --

1. RAM/Source and object/Accreditation I & II: Accreditation I & II do not inhibit technology infusion by the contractors; therefore it was determined that reliability and maintainability benefits would be increased.

2. Military personnel/object/Accreditation I & II: Under these strategies, and implementing ISA standardization, hardware commonality would be non-existent and it would be more difficult for the military to provide the personnel necessary to maintain and operate the systems.

3. Civilian personnel/object, box, LRU/Accreditation I & II: No comparison with laissez-faire was intended. The generally low rating is due primarily to the unavailability of specialized engineering/design personnel within the

FIGURE 4 - 1

| | |
|---|---|
| Source | Object |
| Box | LRU |

| | Reliability & Maintainability | | Software Logistics | | Hardware Logistics | | Military Personnel | | Civilian Personnel | |
|---|---|---|---|---|---|---|---|---|---|---|
| Laissez Faire | V. good | -- | V. poor | -- | V. poor | -- | V. poor | -- | average | -- |
| | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Accreditation I | V. good 1 | V. good 1 | Average | good | V. poor | V. poor | V. poor | V. poor 2 | good 21 | poor 3 |
| | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| Accreditation II | V. good 4 | V. good 1 | Average 4 | good | V. poor 4 | V. poor | V. poor 4 | V. poor 2 | good 4 | poor 3 |
| | V. good 5 | average | V. good 5 | V. good | average 6 | good 7 | V. good 8 | average 9 | poor 3 | poor 3 |
| Accreditation III | good 10 | good 10 | average | good | average | average | average 12 | average 12 | good 13 | average 13 |
| | good 10 | average 11 | V. good | V. good | average | good | V. good 12 | good 12 | good 13 | good 13 |
| Standardization I | good 10 | good 10 | average | good | good | good | V. good | V. good | good 13 | average 13 |
| | good 10 | average 11 | V. good | V. good | good | V. good | V. good | good | good 13 | good 13 |
| Standardization II | average 14 | average 14 | average | good 15 | good 16 | good 16 | V. good 17 | V. good 17 | V. good 18 | good 19 |
| | average 14 | poor 11 | V. good 15 | V. good 15 | good 16 | V. good 16 | V. good 20 | good 20 | V. good 20 | V. good 20 |

civilian community to be committed to the developments of specialized computers to meet military specified ISA's/HOL's.

4. Source/Accreditation II: Under the strategy of Accreditation II, technology used to build a computer is completely flexible with standardization at the source level. Reliability would be high, software logistics would be average because of the number of code generators that would be required for various target machines, hardware logistics would be extremely low due to the different technologies used by the multiple vendors, military personnel would be extremely low because of the training required to maintain multiple computers. Civilian personnel impact would be very favorable because of the improved productivity, especially if a high order language was selected by the military that was also in common use in commercial applications.

5. RAM & Software Logistics/box/Accreditation II: The high rating is a result of the fact that technology flexibility is completely available to the contractor. In addition, the box level of standardization favorably impacts software logistics since it implies both ISA and I/O standardization as well.

6. Hardware, logistics/box/Accreditation II: In this case, hardware logistics is affected by the additional cost and space required to support a maintenance and sparing philosophy required when there are multiple vendors for the same computer using different technologies. This approach also usually requires depot level repair as opposed to organization or intermediate level repair.

7. Hardware logistics/LRU/Accreditation II: The high benefi: is due to the fact that multiple manufacturers are producing standard LRU's. Therefore, the only risk is that of interchangeability of spares.

8. Hardware logistics/LRU/Accreditation II: The strategy of open accreditation would have most favorable benefit with box level standardization for military personnel because sparing would be at the box level and replacement would be likewise at the box level. Therefore, trouble shooting would involve only detecting a failed computer and replacing that computer with a spare.

9. Military personnel/LRU/Accreditation II: The impact in this area is primarily dependent upon the adequacy of the BIT. If BIT can be expected to exceed 99% isolation to one LRU, the benefit of this approach is definitely improved.

10. RAM/Source, object, box/Accreditation III & Standardization I: These generally high ratings are given because it is assumed that when more than one developer was involved, the competition will provide the government a better technical product using the latest technology and thereby improved reliability at the time of development.

11. RAM/LRU/Accreditation III, Standardization I, Standardization II: The lower ratings were assessed because constraining the developer to standardize at an LRU level would limit his flexibility in terms of utilizing the latest technology to achieve major reliability improvements.

12. Military personnel/Accreditation III: Under the assumption that standardization at the box levels implies sparing at the box level, the impact on military personnel would be the most favorable. LRU standardization would be somewhat less of a benefit because there would be a requirement for more skilled personnel coupled with the risk of non-interchangeability of LRU's between multiple vendors. The average benefit with standardization at the source or object level is due to the fact that multiple vendors would imply multiple logistics problems.

13. Civilian personnel/Accreditation III & Standardization I: Relative to a strategy involving N developers, the impact of civilian personnel is primarily in the development stage since it could in fact require N times as many engineers/designers, technicians, resulting in a general unwillingness of industry to participate.

14. RAM/Source, object, box/Standardization II: The strategy of one developer - one producer does not promote improved reliability through insertion of major technology advances through frequent competitions. However, this would be offset by a reliability improvement program that would allow improvements to be implemented during a long production cycle.

15. Software logistics/object, box, LRU/Standardization II: Standardization at the object, box or LRU level would in effect give the government a defacto software standard.

16. Hardware logistics/Standardization II: With a strategy of one developer - one producer it does not really make any difference what the level of standardization is because the hardware itself becomes a defacto standard at the LRU level. However, within a concept of standardization above the LRU level, the contractor could have the option of hardware modifications over the length of the production run which may cause differences at the LRU level. The significance of this impact would be dependent upon the configuration management controls that were imposed on the contractor by the government.

17. Military personnel/source & object/Standardization II: Standardizing at the source or object level is good for the military because the training and maintenance is the same for all personnel across all systems applications.

18. Civilian personnel/source/Standardization II: Although the impact is favorable, it was felt that industry would not participate unless it had the specific expertise required to develop a computer at a military specified HOL level of standardization.

19. Civilian personnel/object/Standardization II: Note 18 again applies except that it will be even more difficult for industry to find the expertise necessary to participate when it is also standardizing on a military specified ISA.

20. Military & Civilian personnel/box & LRU/Standardization II: It was considered that standardization at the lowest level, and using the acquisition strategy of one developer and one producer, was easiest for everyone concerned, military and civilian.

21. Civilian personnel/source/Accreditation I: A high rating is based on the assumption that a HOL adopted by the DoD would be compatible with a HOL used for commercial applications and supported by computer manufacturers in that regard.

4.5.6 Conclusions --

1. Except for reliability and maintainability, the more restrictive acquisition strategies (moving from laissez faire to standardization II) favorably improve the impact that the acquisition strategy has on the measures of comparison. Further, increasing the level of standardization (moving from source to LRU) also has a favorable effect on the criteria considered. That is, the more restrictive the acquisition strategy and the greater the level of standardization, the greater would be the benefits to the government.

2. RAM is the exception to the first conclusion. The reason is that the less restrictive the acquisition strategy, the greater the potential for contractors to be innovative in using the most current technology to achieve improved levels of reliability. The attendant ability to implement extensive BIT using newer technologies also improves maintainability.

## 4.6 Discussion Specific to the Criteria of

Funding Sources

Competition

Acquisition Cycle

### 4.6.1 Amplification of Evaluation Criteria --

Funding Sources: This refers to the willingness of industry to invest in computer development and qualification.

Competition: Competition was evaluated in terms of the ability of an acquisition strategy to promote a competitive environment.

Acquisition Cycle: This criterion is the expected effect of an acquisition strategy on the lengthening or shortening of the acquisition cycle of a weapons system (not of the computer).

### 4.6.2 General --

Although production quantity had initially been considered as a separate criterion, sub-Panel 2 discovered that quantity considerations were implicit in the discussions of the other criteria.

### 4.6.3 Premises --

1. DoD market stability (commitment) is critical to industry investment and to attracting competition (i.e., avoiding market fragmentation).

2. It is essential that both development and production be linked together to provide industrial incentive. A production allocation plan is critical to achieving multiple production suppliers, as in Accreditation III.

3. DoD microprocessor needs may promote different industry incentives and lead to other acquisition strategies.

4. Actual and perceived objectivity and fairness are critical to any DoD acquisition policy for industry acceptance (i.e., willingness to complete and invest).

5. In general, the attraction of industry investment was assumed to be good for the government. Thus, high ratings in this column of the rating table (Figure 4-2) translate into a tendency for industry to invest under the various standardization/investment strategies. Conversely, a low score should be interpreted as a tendency for industry not to invest, and one should expect government funding to dominate.

6. Sub-Panel 2 found the level of standardization (source et al) to be immaterial to the criteria of para 4.6. Accordingly, Figure 4-2 does not depict a "third dimension" to account for the four levels.

4.6.4 Ratings Table -- See Figure 4-2

4.6.5 Notes Applicable to Ratings Table --

22. Controversial findings: In the discussions with the entire Panel C membership, substantial disagreement emerged regarding the degree of industrial concentration which would actually take place. One school of thought held that industry will respond to the certain knowledge that its participation will be rewarded (as in Standardization II), while the other school believed that the greater the opportunity to make a sale (laissez faire as the extreme), the greater will be industry's willingness to risk investment.

23. Funding sources: The laissez faire and Accreditation II strategies tend to discourage investment because the market is too diffuse, lacks certainty [illegible] direct[illegible], and does not lend itself to accurate return on investment assessments. The two standardization strategies tend to draw the heaviest investment because the market is concentrated and defined, because of the implied [illegible] commitment, because risks are bounded, and because the business opportunity (return on investment) is more quantifiable.

NOTE: Standardization levels not significant to these criteria.

FIGURE 4 - 2

| | Source of Funding | Stimulate Competition | Acquisition Cycle |
|---|---|---|---|
| Laissez Faire | very poor 22,23 | very poor 22,24 | very poor 25 |
| Accreditation I | poor 23 | average 24 | good 25 |
| Accreditation II | very poor 22,23 | poor 24 | good 25 |
| Accreditation III | average 23 | good 24 | good 25 |
| Standardization I | good to very good 22, 23 | very good 22,24 | very good 25 |
| Standardization II | very good 22,23 | very good 22,24 | very good 25 |

Accreditation I and III were rated in between these boundaries, with Accreditation III having a slight edge over Accreditation I in market certainty and service commitment.

24. Competition: The laissez faire strategy was seen as being the least advantageous in fostering a broad competitive environment, primarily because it encourages concentration to a few (or perhaps one) firm willing or able to bear marketing and development burden across many different government program managers and system prime contractors. The barriers to entry are high and the "staying" costs are also high. This argument, too, drew some disagreement in the full Panel C deliberations and no resolution of differences was achieved.

Accreditation II was thought to experience the same competition pheonmenon, but not quite as severely as the laissez faire strategy. The two standardization strategies do not suffer from this in the initial competition, and hence were judged best on this factor. However, the issue of subsequent competition was not fully addressed.

25. Acquisition cycle time: This factor did not appear to be a significant driver in determining a computer acquisition strategy. Nevertheless, for completeness, the sub-panel did a relative ranking. The laissez faire strategy holds the most potential danger because each development is a "new game," and, as such, has the potential for impact on acquisition cycle. At the other pole, the two standardization strategies had the most favorable impact on this factor because products would be more or less instantly available once development was completed. The three accreditation strategies were viewed as only slightly less advantageous because some choice is involved and it was reasoned that this choice process would consume some time.

4.6.6 Conclusions --

1. Industrial investment is more likely to be drawn toward standardization strategies than to the more flexible strategies*.

2. Standardization strategies tend to promote a slightly more competitive environment than do accreditation strategies; laissez faire, least of all.

3. Standardization and accreditation strategies have about the same impact on system acquisition time, and both appear better than laissez faire. This factor does not seem to be a driver in determining a computer acquisition strategy.

4. The government members of sub-Panel 2 were asked to respond to the question: "What is the most important of these criteria with respect to their use in evaluating the various strategies?" Their unanimous response was the order of importance of Competition, Funding Sources, and Acquisition Cycle.

*Disagreement exists depending on the degree of industry concentration/ investment which will occur or is anticipated to occur.

## 4.7 <u>Discussion Specific to the Criteria of</u>

Life Cycle Cost

Product Availability

Capability of Achieving Technological Currency

### 4.7.1 Amplification of Evaluation Criteria --

Life Cycle Cost was divided into 5 categories:

- Hardware Development Cost -- The steady state <u>aggregate</u> cost to the DoD of the development of <u>all</u> embedded computer systems.
- Hardware Production Cost -- Cost to the DoD of embedded computers acquired in production quantities.
- Hardware Logistics Cost -- Cost of maintaining embedded computers.
- Software Development Cost -- Includes cost of tools, training, and code production.
- Software Logistics Cost -- Software maintenance.

Product Availability was divided into 2 categories:

- Assurance of Adequate Quantities -- The level of confidence that the DoD can procure the necessary quantities independent of the commercial climate, for an equitable price, for the expected production life of the product.
- Adequate Industry Participation -- Industry's willingness to invest, as a function of competition. Compare with para 4.6.1 (Funding Sources).

Technological Currency: This is a measure of the degree to which the latest available technology is captured in the computer acquired for any given weapon system.

4.7.2 General --

Faced with the forbidding task of evaluating 8 unique criteria, sub-Panel 3 decided to work first at the object level of standards and then perturb those results as appropriate to the source, box and LRU levels. As it developed, it was necessary to treat the box level hastily, and the LRU not at all as time ran out.

4.7.3 Premises --

1. Object level standardization was taken to mean that computer architectures would be restricted to those on a list such as proposed under 5000.5X. Currently accepted architectures were used in considering scenarios. At this and other levels of standards, transient effects caused by the institution of standards were disregarded. Steady state comparisons were sought, in spite of some strong feelings that nothing stands still long enough to achieve a steady state.

2. The average production cost for military computers is driven by many factors, but it was agreed by the sub-panel that the major discriminant between acquisition strategies is the length of productions runs; or, conversely, the degree to which the market is split up between different producers. The strategies also differ as to how effectively the production price can be competed and thereby driven down.

3. Hardware logistics cost is driven chiefly by the number of different computer types in service, each with its own parts list. Some cost penalty will be experienced when one type is produced by multiple sources, leading to problems with inter-changeability of parts. The sub-panel noted that the entire logistics problem is mitigated by higher levels of circuit integration. VLSI brings such a reduction in number of replaceable parts (with concurrent increase in reliability) that the contribuiton of hardware logistics to life cycle cost will diminish sharply in the years ahead.

4. In an unconstrained environment, each different architecture requires its own set of software tools, including assembler, compilers, simulator, and all elements of the environment. Because of limitations on development dollars, a complete software development environment can only be afforded for a few architectures. For other architectures, inadequate tools will mean high cost of application software. Architectural proliferation also increases training costs, and particularly increases the cost of all code which must be written in native language and assembled, which, at minimum, includes all I/O oriented code. When the number of architectures is constrained, the foregoing cost problems are controlled. Software costs are architecture dependent and are not affected by the manner or variety of implementation.

5. Assurance that production quantities will be available immunizes the government against industrial instability. It is driven entirely by the number of sources available. Sub-Panel 3 did not see this as a strong discriminant among acquisition strategies.

6. Adequate industry participation is a measure of the disadvantages of highly competitive strategies as viewed by industry. As the investment required to compete goes up and/or the chance of winning some profitable business goes down, it becomes less attractive for industry to get in the game. The sub-panel noted that in the case of each of the AN/UYK-43 and AN/UYK-44 competitions, only two companies perservered to the point of submitting proposals. The others dropped out for just the reasons cited. As the congress urges more strongly competitive strategies, this is a significant criterion to bear in mind.

7. Technological currency is a measure of the degree to which the

latest available technology is captured in the computer acquired for any given weapon system. Generally, it is high for computers developed specifically for a weapon system, less when the Program Manager has to procure through a central buy, which may not be a recent development. Sub-Panel 3 did not regard this criterion as a major one.

8. With regard to hardware development costs at the source level of standards, the sub-panel agreed that development costs would be increased to the extent new architectures were brought into play, but reduced to the extent that militarization of commercial machines would be permitted. The latter option becomes increasingly attractive as the level of circuit integration increases.

### 4.7.4 Ratings Table --

See Figures 4-3 and 4-4. Recall that notes applying to the object level of standards apply equally to the other levels unless noted otherwise. Where two ratings are separated by a solidus (/), the first applies to single source and the second to second source utilization.

### 4.7.5 Notes Applicable to Rating Table --

26. Hardware development cost/object level: This was seen as being generally proportional to the number of different computers being developed over a given period of time. The sub-panel also agreed that, since new architectures usually require some clean-up during the first implementation, standards which discourage proliferation of new architectures tend to lower development costs. Acquisition strategies that are predicated on contractor financing for development result in minimum cost to the DoD.

| Source | Object |
|---|---|
| Box | |

FIGURE 4 - 3

LIFE CYCLE COST FACTORS

| | Hardware Development | | Hardware Production | | Hardware Logistics | | Software Development | | Software logistics | |
|---|---|---|---|---|---|---|---|---|---|---|
| Laissez Faire | V. poor 61 | V. poor 26,27 | average 62 | average 33 | poor 62 | poor 38 | poor 63 | poor 42 | poor 63 | poor 44 |
| | poor to V. poor 66 | | average 66 | | poor 66 | | poor 66 | | V. poor 66 | |
| Accreditation I | poor 61 | poor 26,28 | average 62 | average 33 | average 62 | average 39 | good 62 | good 43 | good 63 | good 44 |
| | poor 66 | | average 66 | | average 66 | | good 66 | | good 66 | |
| Accreditation II | V. good | V. good 26,29 | average 62 | average 33 | average 62 | average 39 | good 63 | good 43 | good 63 | good 44 |
| | V. good 66 | | average 66 | | average 66 | | good 66 | | good 66 | |
| Accreditation III | V. good | average 26,30 | V. good 62 | V. good 34 | good 62 | good 40 | good 63 | good 43 | good 63 | good 44 |
| | average 66 | | V. good 66 | | good 66 | | good 66 | | good 66 | |
| Standardization I | average | average 26,30,32 | V. good 62 | V. good 35,37 | V. good 62 | V. good 41 | good 63 | good 43 | good 63 | good 44 |
| | average 66,67 | | V. good 66,67 | | V. good 66,67 | | good 66 | | good 66 | |
| Standardization II | good | good 26,31,32 | good 62 | good 36,37 | V. good 62 | V. good 41 | good 63 | good 43 | good 63 | good 44 |
| | good 66,67 | | good 66,67 | | V. good 66,67 | | good 66 | | good 66 | |

FIGURE 4 - 4

| Source | Object |
|---|---|
| Box | |

PRODUCT AVAILABILITY

| | Quantities | | Industry Participation | | Technological Currency | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Laissez Faire | average 62 | average 45 | V. good 64 | V. good 50 | V. good 65 | good 57 | | | | |
| | average 66 | | V. good 66 | | good 66 | | | | | |
| Accreditation I | average 62 | average 46 | average 64 | average 51 | V. good 65 | good 58 | | | | |
| | average 66 | | average 66 | | good 66 | | | | | |
| Accreditation II | good 62 | good 47 | V. poor 64 | V. poor 52 | good 65 | average 59 | | | | |
| | good 66 | | V. poor 66 | | average 66 | | | | | |
| Accreditation III | good 62 | good 48 | average 64 | average 53 | poor 65 | poor 60 | | | | |
| | good 66 | | average 66 | | poor 66 | | | | | |
| Standardization I | good 62 | average/ good 49 | poor/ average 64 | poor/ average 54,56 | poor 65 | poor 60 | | | | |
| | average/ good 66,67 | | poor 66 | | poor 66 | | | | | |
| Standardization II | good 62 | average/ good 49 | good 64 | good 55 | poor 65 | poor 60 | | | | |
| | average/ good 66,67 | | good 66 | | poor 66 | | | | | |

27. Hardware Development/object/laissez faire: This unconstrained approach leads to the largest number of different computer developments.

28. Hardware Development/object/Accreditation I: Here again, the tendency is to develop a new computer for each new platform or weapon system.

29. Hardware Development/object/Accreditation II: Development costs are incurred by the contractor, so DoD sees minimal cost.

30. Hardware Development/object/Accreditation III and Standardization I: Central buy strategies preclude new developments until they are clearly life cycle cost justified, keeping development costs low. Multiple developers cost more than single developers.

31. Hardware Development/object/Standardization II: Similar to note 30, but one developer costs less than multiple developers.

32. Hardware Development/object/Standardization I & II with second sourcing: An additional increment of development costs is required to develop and qualify a second source.

33. Hardware Production/object/laissez faire, Accreditation I, and Accreditation II: These strategies permit a larger number of types and therefore shorter runs than the central buy strategies.

34. Hardware Production/object/Accreditation III: Although this was ranked very good, the ranking relative to Standardization I is a function of market size. Central buy minimizes the number of different types, and multiple developers compete the production price. Breaking up the run among an even greater number of producers decreases the advantages of volume production but permits continuing competition for follow-on market share. The latter only predominates for very large total buys.

35. Hardware Production/object Standardization I: This has all the advantages of long runs and competition for production.

36. Hardware production/object/Standardization II: This, too, has all the advantages of long runs, but no competition after development.

37. Hardware production/object/Standardization I & II with second sourcing: A small increment of cost is added for this. This results from a reduction of production volume and the added costs of coordination and parts compatibility.

38. Hardware logistics/object/laissez faire: The maximum number of computer types generates the maximum logistics cost.

39. Hardware logistics/object/Accreditation I & II: Less proliferation of types than laissez faire, but more than central buy strategies.

40. Hardware logistics/object/Accreditation III: This has minimal proliferation of types, but includes some avoidable costs traceable to the difficulties of multiple sources.

41. Hardware logistics/object/Standardization I & II: These have minimum proliferation of types. With second sourcing, there is a small additional cost due to difficulties of multiple sources.

42. Software development/object/laissez faire: Architectural proliferation leads to high cost.

43. Software development/object/all but laissez faire: Costs are improved when architectural proliferation can be controlled.

44. Software logistics/object: Software maintenance can be viewed as "in-service development," so logistics costs run exactly parallel to development costs. Proliferation has an acute effect, because specialists cannot be afforded to handle the whole inventory.

45. Quantities/object/laissez faire: Only one source per computer.

46. Quantities/object/Accreditation I: Within the context of a single source per computer, this was rated average. However, the sub-panel noted that second sourcing is applicable to this category and would raise the

availability.

47. Quantities/object/Accreditation II: The feeling was that this strategy, if successful, would generate sufficient redundancy of suppliers to cushion any failures among them.

48. Quantities/object/Accreditation III: Rated good because multiple sources are available.

49. Quantities/object/Standardization I & II: These are only average with single sources. However, with second sourcing they share the advantages of Accreditation III.

50. Industry participation/object/laissez faire: By definition, this strategy permits the Program Manager to design his computer acquisition to get what he wants. It should be noted that this does not necessarily maximize competition.

51. Industry participation/object/Accreditation I: This is similar in business prospects to other single developer strategies, but is generally with lower production prospects.

52. Industry participation/object/Accreditation II: The supplier foots the whole development bill, and still has to enter a Chinese auction to get any business.

53. Industry participation/object/Accreditation III: The supplier must compete at least twice and contribute investment in development. This has a slight edge over Standardization I in that there is more than one winner of the last round.

54. Industry participation/object/Standardization I: Poorer than Accreditation III because there is only one winner of any profitable business.

55. Industry participation/object/Standardization II: This more conventional approach limits the investment required for a contractor to win the pot. This applies also to second sourcing.

56. Industry participation/object/Standardization I with second sourcing: This raises industry's willingness to participate because a loser can now get a second chance.

57. Technological currency/object/laissez faire: Good, because concurrent development is always an option.

58. Technological currency/object/Accreditation I: The Program Manager is still free to choose the latest technology.

59. Technological currency/object/Accreditation II: Rated as average because sometimes the Program Manager will have to take an option that is not the latest vintage.

60. Technological currency/object/Standardization I & II: A central buy holds off new developments until justified on a total life cycle cost basis.

61. Hardware development/source/laissez faire and Accreditation I: See premise 8. Sub-Panel 3 agreed that there should be a reduction in cost here relative to the object level, but were unable to agree on any change in the overall ranking of strategies for this criterion.

62. Hardware production, hardware logistics & quantity/source: These are identical to the issues at the object level.

63. Software development and software logistics/source: Admissibility of more architectures increases software development cost for all categories except laissez faire. Laissez faire remains the highest in cost, but the difference is no longer as significant.

64. Industry participation/source: Commercial VLSI computers can be inexpensively militarized, and this fact would increase participation where it was otherwise discouraged by heavy investment. The sub-panel was unable to agree on any change in rankings.

65. Technological currency/source: Militarization of the latest commercial models will improve currency for the laissez faire, Accreditation I, and Accreditation II strategies. These three had already been rated the highest at the object level.

66. Box level: The sub-panel observed that the central buy strategies, Accreditation III and the two Standardization strategies, really assume box level standardization, while the other strategies preclude it. Nonetheless, a matrix at this level was hurriedly prepared in the closing minutes of the session, and is presented without further notes of justification, save the next.

67. Hardware development, hardware production, hardware logistics, and quantities/box: Costs improve with second sourcing.

4.7.6 Conclusions --

1. Central buy strategies are best for hardware development, production and logistics costs. This is because central buy strategies do not permit the development of a new computer until it is fully justified on a life cycle cost basis versus continued use of its predecessor(s). These strategies therefore minimize the frequency of computer developments, maximize the production runs for each and minimize the number of types in the field.

2. Architectural standardization is the key procurement attribute for moderating software costs. By permitting the development of mature software production environments, it has a strong effect on moderating software development costs; and considering the more reasonable requirements for maintenance

personnel as well as the environmental benefits, it has a very strong effect on moderating software logistics costs.

3. Sub-Panel 3 expressed concern that the cleanness and feasibility of source level standardization not be overstated. True HOL compatibility, as defined in para 4.2, cannot be achieved without stringent controls on machine architecture (word length, number representation, arithmetic details, etc.) Lifetime maintenance of source level code is rare and hard to enforce. Assembly level code will continue to be needed -- as a minimum for I/O routines, interrupt handling, resource management, and diagnostics -- and these requirements will grow as code is written to control distributed systems. The debugging stage of software development continues to require knowledge of machine architecture. Thus, standardizing at the source level falls short of generating many of the advantages of ISA standardization.

In any case, the relative ratings at the source level are much the same as at the object level for the criteria considered by the sub-panel.

4. Multiple sources in production constitute a mixed blessing whose benefits grow with production volume. Dividing production volumes several ways increases costs, but competitive pressures reduce them. Commonality at the form, fit, function and interface level exacerbates hardware logistics costs; while commonality at the build-to-print level leads to extra non-recurring hardware costs to ensure full interchangeability, a set of problems with which industry has almost no successful experience. Advantages of assumed availability are of secondary importance.

5. As the Congress insists on more and more competitive procurement strategies, it should be remembered that these are likely to be less and less attractive to industry. In particular, N developers - 1 producer requires discouragingly heavy investment, and Accreditation II looks unreasonably

risky under most circumstances.

6. Although the less restrictive strategies abet the attainment of technological currency, it was not regarded as a criterion of major importance.

7. The sub-panel felt that all strategies other than the central buy methods really preclude box level standardization. Note that this view was not held by sub-Panel 1, the other sub-panel that found standardization level to be a factor. They felt that Accreditation II also applied to the box level.

8. As VLSI technologies become predominant in military computers, the criteria for selecting procurement strategies will change. Hardware development costs will generally increase, while hardware production and logistics costs will diminish sharply. Software considerations will not generally be affected. The militarization of commercially available computers, whose chip designs and fabrication masks are in place and whose software development environments have been paid for, will become increasingly attractive.

9. As sub-Panel 3 worked these issues, there was no tendency toward concensus for any one procurement strategy across the services.

## 4.8 General Conclusions

The following conclusions are in addition to those found in paras. 4.5.6, 4.6.6 and 4.7.6.

1. Accreditation relates to acquisition strategies rather than to an extent or level of standardization. Accreditation includes the process of certifying that a product meets a standard.

2. The success of an accreditation strategy will be driven by such factors as

   a. Hardware/software standardization

   b. Ability to satisfy DoD weapon system requirements

   c. Acceptance by DoD and industry

   d. Life cycle cost

   e. Availability of qualified personnel

Taken as a group, the two standardization practices appeared advantageous with respect to more evaluation factors than the group consisting of the three accreditation strategies. To determine the optimum strategy for a given weapon system would require a weighting of the criteria specific to that system.

3. Although a significant data base of issues relating to acquisition strategies for standardization was addressed and evaluated in a short period of time, more work is needed. Analyses conducted in depth with reference to hard data resources are also required. For example, the degree of industrial responsiveness and breadth of competition under each of the acquisition strategies deserves further quantitative analysis.

4. There has not been sufficient study to define and implement a single acquisition policy. Nevertheless, the potential exists for common embedded

computer acquisition strategies across all four services. If there were a single acquisition policy, it would either have to accommodate both central buy strategies and strategies employed for specific weapons systems, or compromise the advantages of both.

## 5. RECOMMENDATIONS

1. The panel recommends that the data base of acquisition/accreditation issues developed at Monterey II be used as the basis for further study, leading to the selection of computer acquisition strategies best able to achieve the benefits of standardization.

2. Studies of computer designs should be conducted to determine the impact of standardization level. These designs should include those capable of direct HOL execution. The level of detail should be sufficient to fully expose the implications to development, life cycle cost, and support of such designs.

## APPENDIX A
## PARTICIPANTS

| CHAIRPERSON | Co-CHAIRPERSON |
|---|---|
| Mr. Robert H. Dunn | Dr. Irv Hecker |
| ITT Avionics Division | Rolm Corporation |
| 100 Kingsland Road | 7700 Little River Tpke. |
| Clifton, N.J. 07014 | Annandale, Va. 22003 |
| (201) 284-2946 | (703) 750-0300 |

AS ORDERED BY SUB-PANEL:

### Sub-Panel 1

Mr. David A. Kolling, sub-Panel Chairperson
Sperry-Univac Defense Systems
P.O. Box 3525
Mail Station U2H21
St. Paul, Minn. 55165
(612) 456-4957/8

Mr. John M. Cole
USA CECOM
Attention: DRSEL-TCS-BK
Fort Monmouth, N.J. 07703
(201) 544-2759
AV 995-2759

Dr. Irv Hecker, Panel C co-Chairperson

LCDR William C. Hess
Naval Electronic Systems Command
Attention: ELEX 81411
Washington, D.C. 20360
(202) 692-8484/5
AV 222-8484/5

Lt. Col. Vance A. Mall
Ada Joint Program Office
OUSD (R&E)
Washington, D.C. 20301
(202) 694-5917
AV 224-5917

Capt. William Riski
HQ AFLC/LOEC
Wright-Patterson AFB, Ohio 45433
(513) 257-2054
AV 787-2054

Sub-Panel 2

Mr. Barry DeRoze, sub-Panel Chairperson
TRW/DSSG Mail Station R2 1410
One Space Park
Redondo Beach, Ca. 92714
(213) 535-4271

Mr. Robert E. Alger
Teledyne-Brown Engineering
300 Sparkman Drive
Mail Stop 202
Cummings Research Park
Huntsville, Alabama 35807
(205) 532-1257

Col. J. Frank Campbell
HQ, DARCOM
Attention: DRCDE-C
5001 Eisenhower Avenue
Alexandria, Va. 22333
(202) 274-9144
AV 284-9144

Mr. H. Mark Grove, Special guest of CRM
OUSD (R & E) ECR
2A318 Pentagon
Washington, D.C. 20301
(202) 695-0121
AV 225-0121

Mr. Richard J. Harrington
Attention: MAT08Y1
HQ, Naval Material Command
Navy Department
Washington, D.C. 20360
(202) 692-3966
AV 222-3966

Mr. Galen I. Ho
IBM/FSD
Mail Stop 101C573
Route 17C
Owego, New York 13827
(607) 751-3705

Lt. Col. James M. Riley
ASD/AXS
Wright-Patterson AFB, Ohio 45433
(513) 255-5945
AV 785-5945

1st Lt. Thomas M. Smith
HQ, United States Marine Corps (CCA-51)
Washington, D.C. 20380

Sub-Panel 3

Mr. Graham Jones, sub-Panel Chairman
IBM/FSD
6600 Rockledge Drive
Bethesda, Maryland 20034
(301) 493-1462

Mr. Philip S. Babel
ASD/EN
Wright-Patterson AFB
Ohio 45433
(513) 255-6008
AV 785-6008

Dr. A. Brinton Cooper
USAMSAA
Attention: DRXSY-CC
Aberdeen Proving Ground, Maryland 21005
(301) 278-5478
AV 283-5478

Mr. Jim Previte
RADC/ISCA
Griffiss AFB, New York 13441
(315) 330-3461
AV 487-3461

Mr. William Randolph Smith
Office of the Assistant
Secretary of the Navy (RE&S)
Washington, D.C. 20350
(202) 694-4691
AV 224-4691

Mr. Doug Stapp
ASD/ENASD
Wright-Patterson AFB
Ohio 45433
(513) 255-5181
AV 785-5181

Mr. James M. Tate
Vought Corporation
P.O. Box 225907
Mail Station 4-43300
Dallas, Texas 75265
(214) 266-2107

## APPENDIX B

## BIBLIOGRAPHY

1. B. Wise, Navy Embedded Computer Accreditation Program, prepared for Navy under Contract N00014-79-C-0722 by Georgia Institute of Technology, April 1980, DDC #AD A086490.

2. Navy Computer Accreditation Study, prepared for Navy under Contract N00014-79-C-0986 by IBM Federal Systems Division, Owego, May 1980. DDC #AD A088301.

3. Edgerton and Rudwick, Study of An Accreditation Policy for Navy Embedded Computers, prepared for Navy under Contract N00014-79-C-0862 by PRC Information Sciences Co., April 1980, DDC #AD A086063.

4. Navy Embedded Computer Accreditation Study, prepared for Navy under Contract N00014-79-C-0987 by General Electric Co., May 1980

5. H. Bennet Teates and Billy B. Wise, "Accreditation: A New Embedded - Computer Standardization Approach," Computer, Sept. 1980, IEEE Computer Society, pp 26-32.

6. MIL-STD-1750A, Airborne Computer Instruction Set Architecture

7. MIL-STD-1862, Nebula

8. DOD Instruction 5000.5X, Instruction Set Architecture Policy for Embedded Computers

# APPENDIX C

Presentation given by

Lt. Col. Vance Mall
Ada Joint Project Office

## COMPILER VALIDATION

| | | |
|---|---|---|
| PHASE I | — INITIAL CAPABILITY | OCT. 80 |
| PHASE II | — FULL CAPABILITY | OCT. 81 |
| PHASE III | — RESEARCH AND DEVELOPMENT | |

TO CONSIST OF

- ADA TEST PROGRAMS $\backsim$ 1000
- AUTOMATED TOOLS TO ASSIST VALIDATION
- IMPLEMENTER'S GUIDE

## ADA COMPILER VALIDATION

ADA COMPILERS TO BE USED ON DoD PROJECTS WILL BE REQUIRED TO PASS A VALIDATION TEST

VALIDATION PROCEDURE WILL BE STRUCTURED TO:

- CHECK PRESENCE OF REQUIRED FEATURES
- ABSENCE OF NON-STANDARD FEATURES

SOFTECH COMPETITIVELY SELECTED TO DEVELOP COMPILER VALIDATION CAPABILITY

- BASIC CAPABILITY - FALL 1980
- STATE OF THE ART CAPABILITY - FALL 1981

## ADA TEST PROGRAMS

- COMPILE TIME ACCEPTANCE OF LEGAL PROGRAMS
- COMPILE TIME REJECTION OF ILLEGAL PROGRAMS
- REJECTIONS OF ILLEGAL PROGRAMS — LINK/LOAD TIME
- CORRECT PROGRAMS TO BE EXECUTED — SELF CHECKING
- CHECK CAPACITY REQUIREMENTS
- CHECK FOR CORRECT WARNINGS
- CHECK OPERATION OF STANDARD PACKAGES

## APPENDIX D

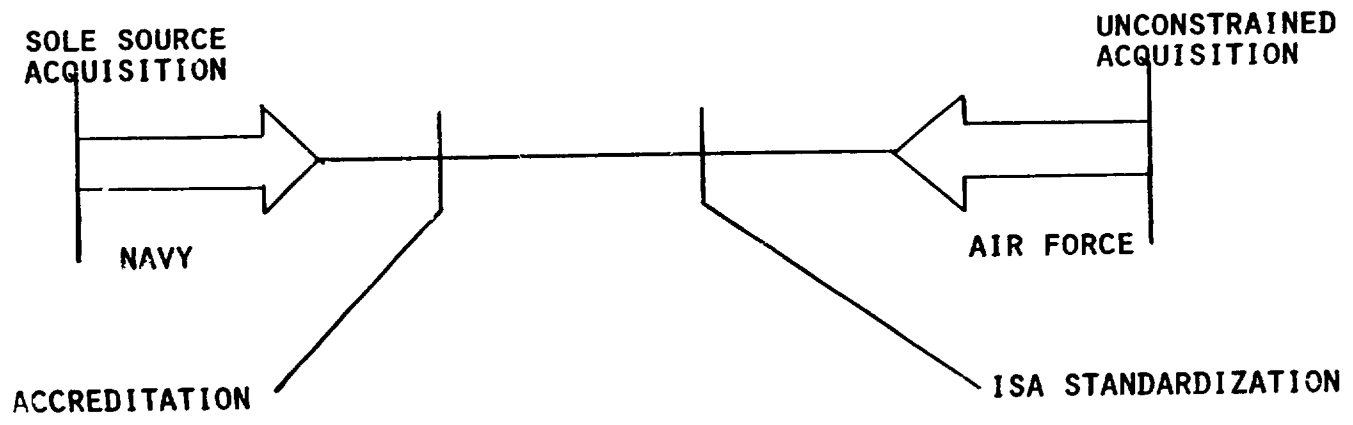
Presentation given by

Doug Stapp
ASD/ENASD

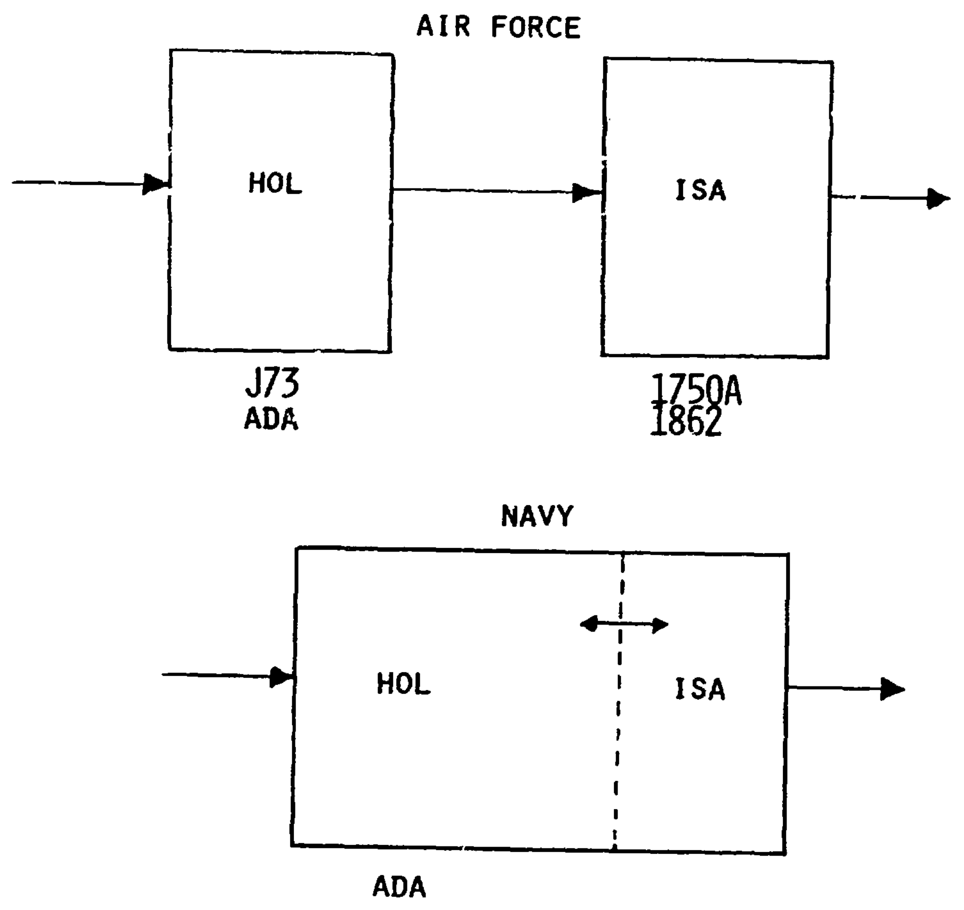
# AIR FORCE INSTRUCTION SET ARCHITECTURE STANDARDIZATION: MIL-STD-1750A

DOUGLAS STAPP
ASD/ENASD (SEAFAC)
22 JUN 1981

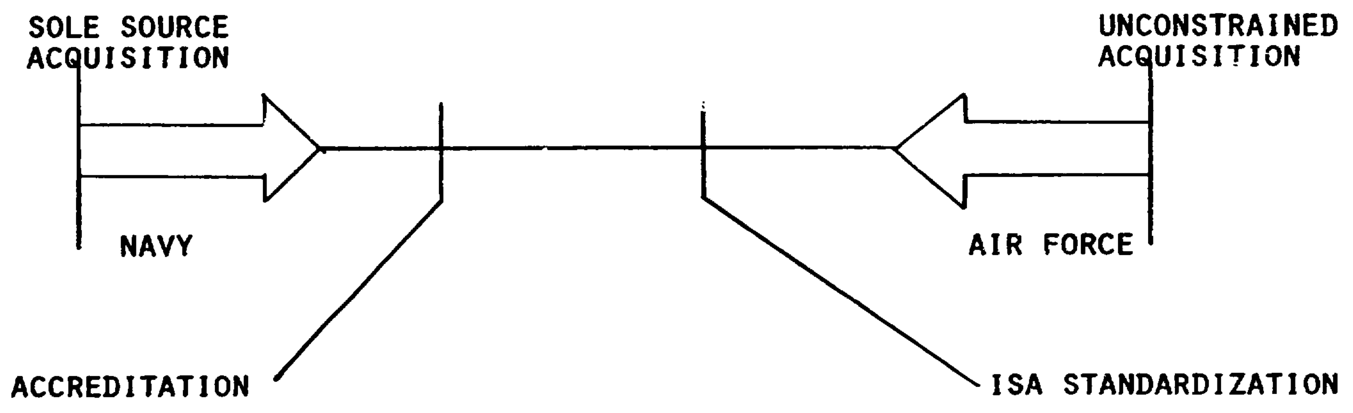
## PERSPECTIVE ON STANDARDIZATION



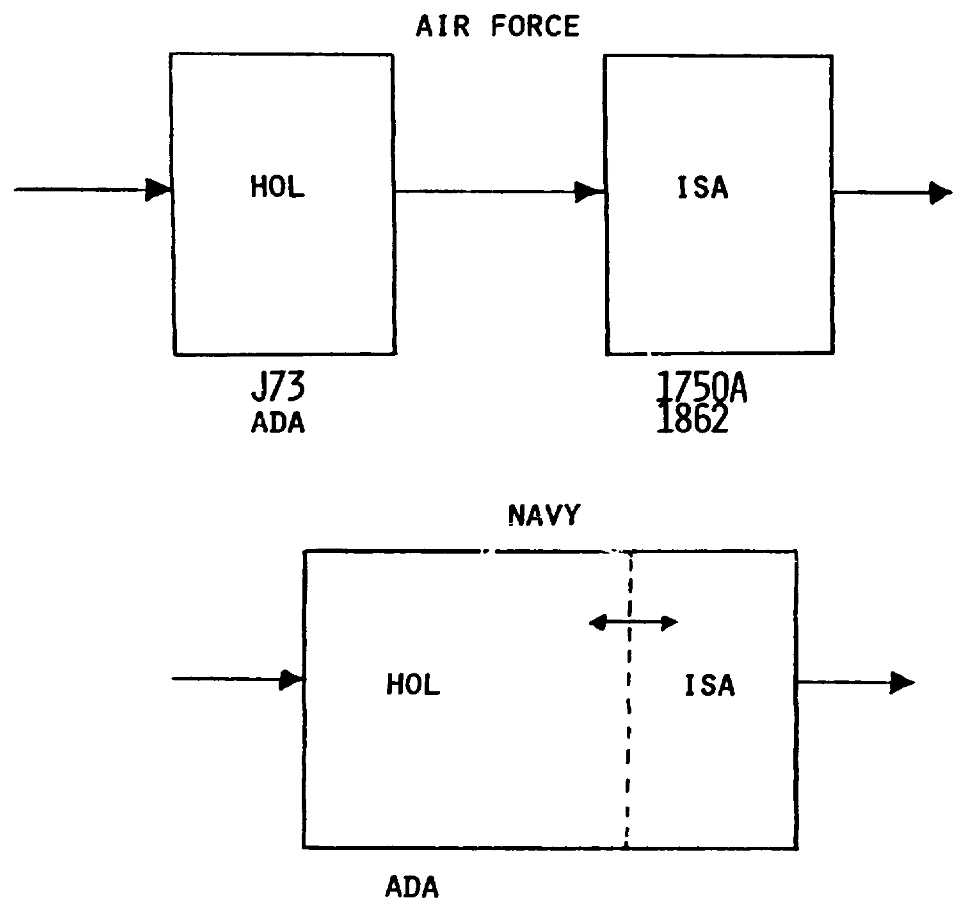
## GOAL OF STANDARDIZATION



## PERSPECTIVE ON STANDARDIZATION



## GOAL OF STANDARDIZATION



## IMPLEMENTATION ISSUES

- HARDWARE STANDARDIZATION
- INTERFACE
- POLICY FOR USE
- VERIFICATION
- CONTROL STRUCTURES

## HARDWARE STANDARDIZATION

| | ISA STD | ACCREDITATION |
|---|---|---|
| TECHNOLOGY | NOT ADDRESSED | EMULATION |
| FORM | NOT ADDRESSED | BOX SIZE |
| PERFORMANCE | 16-BIT OR 32-BIT CONTINUUM OF LEVELS | DISCRETE LEVELS |

## POLICY FOR USE

- MANDATORY FOR 16-BIT AVIONICS APPLICATIONS
- PREFERRED FOR ALL OTHER APPLICATIONS
- COULD BE REQUIRED FOR A SPECIFIC PROGRAM
- NO RESTRICTIONS ON SUPPLIERS (i.e. NO "QUALIFIED" SUPPLIERS)

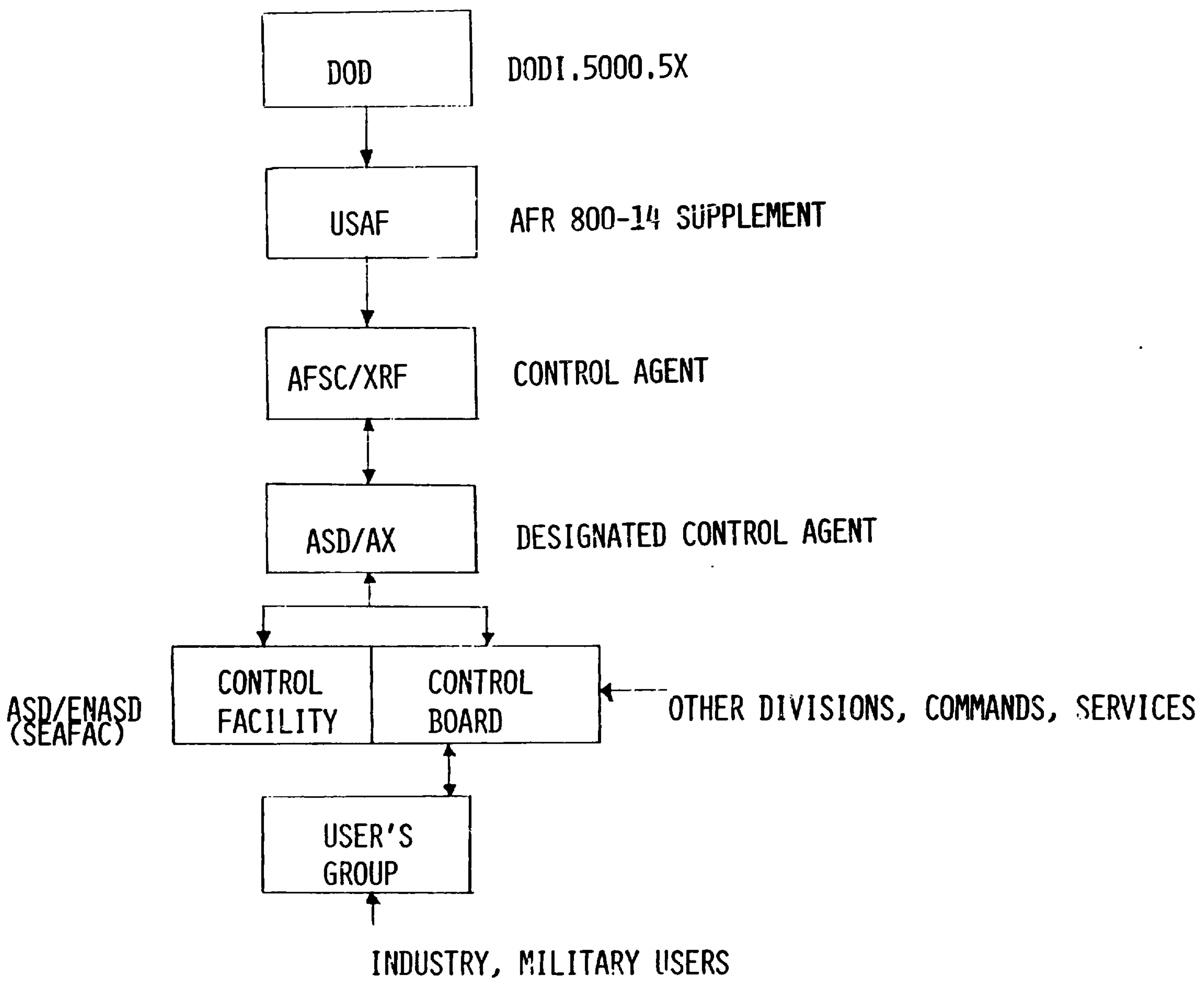
## INTERFACE

| | ISA STD | ACCREDITATION |
|---|---|---|
| PHYSICAL | NOT ADDRESSED | STANDARD |
| ELECTRICAL | NOT ADDRESSED | STANDARD |
| PROTOCOL - HIGH LEVEL | NOT ADDRESSED | ? |
| - LOW LEVEL | NOT ADDRESSED | STANDARD |

## VERIFICATION

- CHOICES
  - IMPLEMENTATION
  - VERIFICATION
    - HOW MANY TESTS
    - CONFIGURATION TO TEST
    - PASS/FAIL CRITERIA
- GUIDELINES
  - AT LEAST ONE VERIFICATION PER CONTRACT PER IMPLEMENTATION
  - NO VERIFICATION BY ARGUMENT OF SIMILARITY
  - NO RE-VERIFICATION REQUIRED
  - RE-TESTING MIGHT BE REQUIRED

## CONTROL STRUCTURE



## EVALUATION

✓ ● STIMULATE COMPETITION

✓ ● TECHNOLOGY INSERTION

● INCREASE FLEXIBILITY OF CHOICE

✓ ● SHORTEN AQUISITION CYCLE

✓ ● REDUCE COST OF SOFTWARE TOOLING AND MAINTENANCE

● LOGISTICS
  - HARDWARE
  - ✓ SOFTWARE

✓ ● INDUSTRY ACCEPTANCE

## APPENDIX E

Presentation given by

William Smith
Office of the Assistant Secretary of the Navy (R E & S)

## ACCREDITATION CONCEPT - GENESIS

ASN(RE&S) EMBEDDED COMPUTER REVIEW PANEL

PANEL RECOMMENDATIONS - OCT 1978 REPORT

1. STRENGTHEN/TIGHTEN MANAGEMENT OF COMPUTERS IN THE NAVY

2. ACCELERATE NEW EMBEDDED COMPUTER SYSTEM

3. REGARDING THE UYK-7

   A. DO NOT ADOPT THE RECOMMENDED UYK-7 IMPROVEMENT.

   B. INSTEAD, ACCELERATE THE DEVELOPMENT OF THE NEW NAVY EMBEDDED COMPUTER (NECS).

   C. IF NECS IS TOO LATE, DISTRIBUTE ANY OVERLOADED UYK-7 FUNCTIONS TO OTHER 7's OR 20's.

   D. IF C IS NOT FEASIBLE, HOLD A COMPETITION FOR AN EMULATED UYK-7/UYK-7 UPGRADE.

4. ADOPT A POLICY OF MOVING AWAY FROM TIGHT STANDARDIZATION

   A. TECHNOLOGY WILL YIELD THIS EVENTUALLY.

   B. EXPEND R/D TO ACCELERATE THIS MOVE.

NECRP REPORT

## DEFINITION

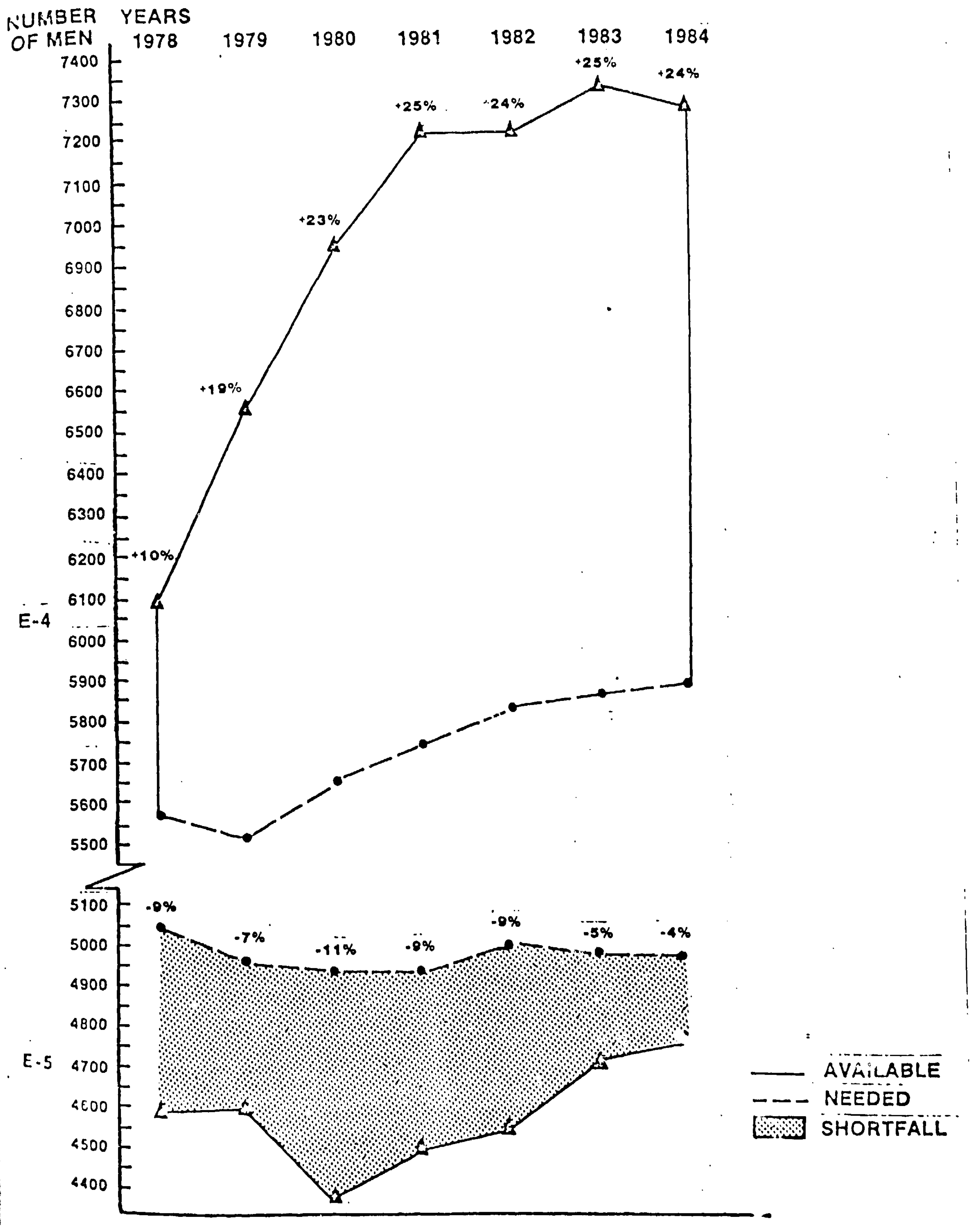
ACCREDITATION: A LIST OF ACCEPTABLE COMPUTERS IN A PERFORMANCE RANGE, AS OPPOSED TO ONE COMPUTER IN A PERFORMANCE RANGE, WHICH IS STANDARDIZATION.

## STANDARDIZATION

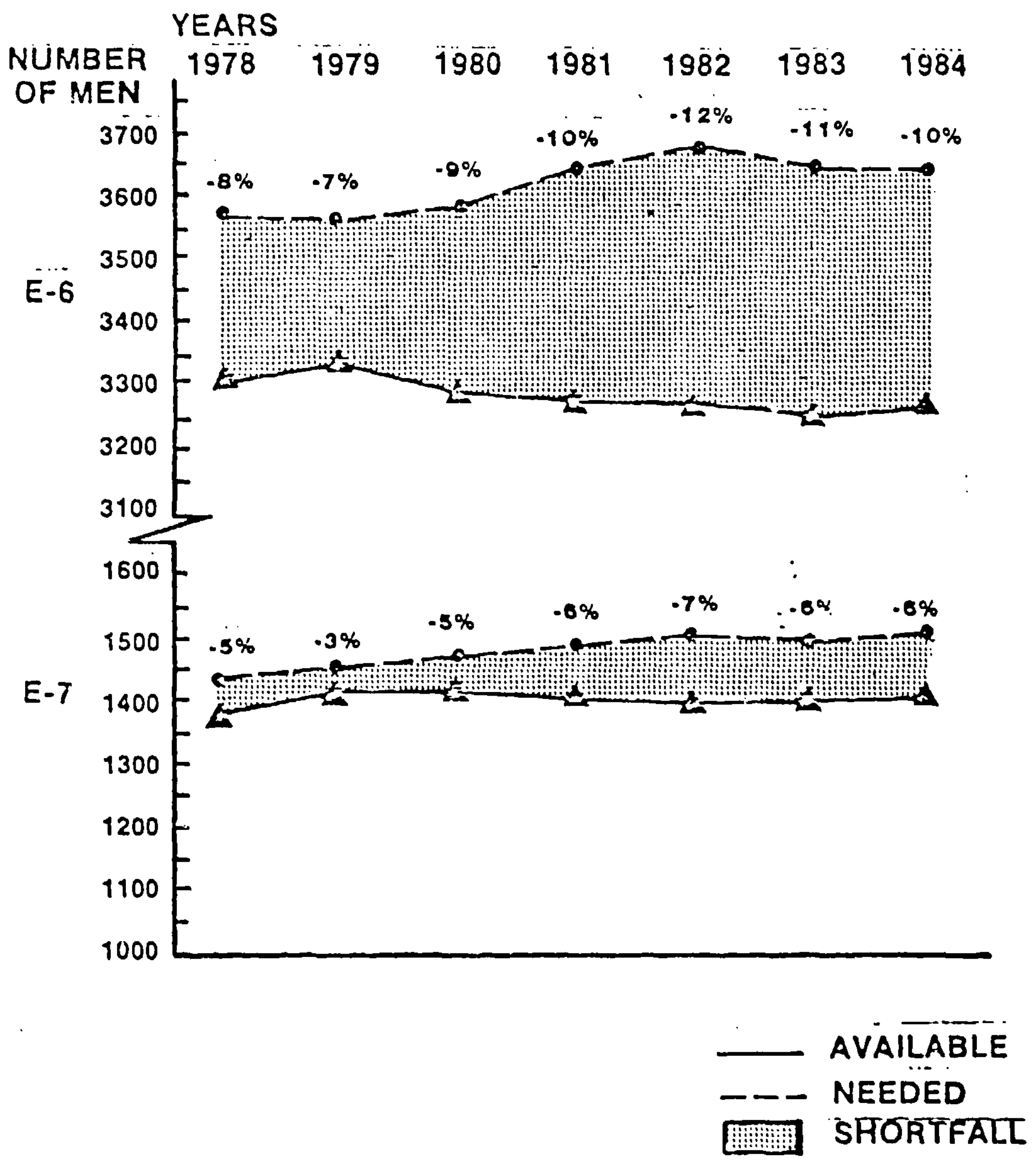
| | | |
|---|---|---|
| POLICY | - | 1 COMPUTER FOR A GIVEN PERFORMANCE RANGE |
| REASON | - | OPERABILITY ABOARD SHIP |
| OPERABILITY | - | RELIABILITY + MAINTENANCE AIDS + PEOPLE + PARTS |

NECRP REPORT

# ET (ELECTRONICS TECHNICIAN)



# ET (ELECTRONICS TECHNICIAN)



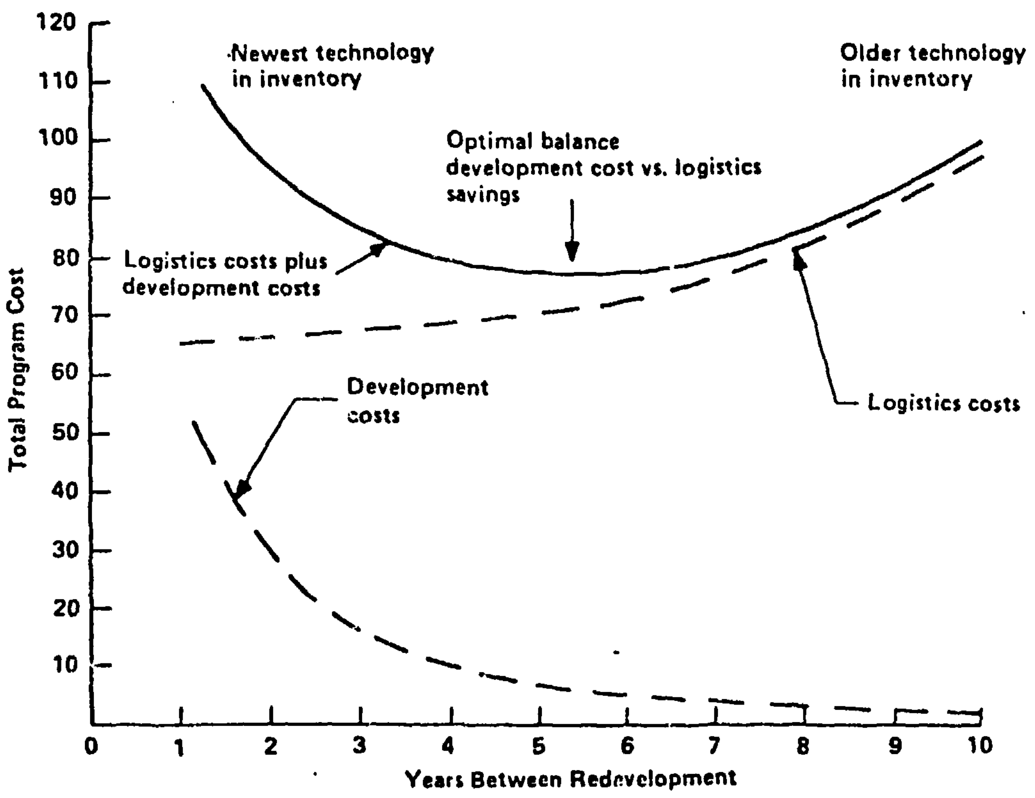
## ACCREDITATION STUDIES

| | | |
|---|---|---|
| OBJECTIVE | - | OBTAIN INDUSTRY CONCEPTS AND ANALYSES IN SUPPORT OF WORKABLE ACCREDITATION APPROACH |
| THEME | - | TO PERMIT THE NAVY TO OBTAIN THE BENEFITS OF NEW TECHNOLOGY AND OPEN COMPETITION AS FREQUENTLY AS POSSIBLE WHILE AT THE SAME TIME SATISFYING OPERATIONAL REQUIREMENTS, MILITARY LOGISTICS AND COST CONSTRAINTS |
| SOURCES | - | FOUR SEPARATE STUDIES AND VIEWPOINTS |

AS A MINIMUM THE STUDY SHALL CONSIDER THE FOLLOWING ISSUES

- REQUIREMENTS FOR OPERATIONAL AVAILABILITY. RELIABILITY, MAINTAINABILITY
- LOGISTICS SUPPORT, SPARES, PERSONNEL, TRAINING
- ENVIRONMENTAL REQUIREMENTS
- OPERATIONAL PERFORMANCE REQUIREMENTS
- QUALIFICATION AND TESTING
- PERTINENT TECHNOLOGY TRENDS
- ABILITY OF RELIABILITY/MAINTAINABILITY IMPROVEMENTS TO DIMINISH LOGISTICS CONCERNS
- APPROACHES TO LOGISTICS COMPATIBILITY - BOX LEVEL $F^3$, STANDARD MODULES, LEADER/FOLLOWER CONTRACTS, ETC.
- COMMERCIAL DEVELOPMENTS
- CRITERIA FOR ACCEPTANCE AND RETENTION OF COMPUTERS ON ACCREDITED LIST
- LIFE-CYCLE COST AS ACCREDITATION CRITERION
- FREQUENCY OF INTRODUCTION OF ACCREDITED COMPUTERS - AT REGULAR INTERVALS, AS REQUIREMENTS DEVELOP, ETC.
- PRACTICAL LIMITS ON NUMBER OF VENDORS AND/OR COMPUTER TYPES
- LIMITS OF APPLICATION OF ACCREDITATION CONTROLS TO DIGITAL PROCESSING DEVICES
- ACQUISITION/CONTRACTUAL STRATEGIES FOR OBTAINING ACCREDITATION CANDIDATES
- SOURCES OF FUNDING FOR DEVELOPMENT AND QUALIFICATION - GOVERNMENT OR INDUSTRY
- PROMOTION OF COMPETITIVE ENVIRONMENT
- TRANSITION FROM CURRENT NAVY PRACTICES TO ACCREDITATION -- APPROACH AND TIMING

| ISSUE | PRC | GIT | IBM | GE |
|---|---|---|---|---|
| Technology Trends | NA | C/P 5 - 25%/yr | C/P - 16%/yr<br>Rel 14%/yr | C/P - 15 to 20%/yr |
| Commercial Development | Use Best Avail. Comm. Tech. Still Need Design & Package for Mil. Qual. | NA | Commercial Technology Impeded in Mil. Computers by Small Volume & Stringent Req. | NA |
| Ao/Reliability | Stress MTTR; Simpler Maint. Graceful Degradation Systems Level Approach. | 4000 Hr. MTBF-Target.<br>Emphasize BIT. | 14%/yr Rel.<br>25 yr. to .95 Ao/2000 Hr. (Simplex)<br>12 yr. (Duplex),<br>5 yr. (8-Duplex) | Stress CSA Approach Distributed, Graceful Degredation. |
| Logistics/Training Personnel | Maintain/Spare at Module Level. | Maintain/Spare at Module Level. | May not be Computer Driven. Maintain/ Spare Modules. | NA |
| Form/Fit/Function | Box Level Specs Plus Standard ISA | Box Level Specs Plus Standard ISA $\rightarrow$ HOL. | Box Level Specs Plus Standard ISA $\rightarrow$ HOL | Box Level Specs Plus ISEM Plus Standard & New ISA's. $\rightarrow$ HOL |
| Environmental (Mil. Specs.) | Trim to Min. Required for Each Type of Application. | NA | Needed | Needed |
| Performance Types | 8 PT's = L and H of KOPS, I/O, Int Time | 15: Low, Med, High in Each of 5 Application Classes | 3: Micro, Mini, Midi | Micro, Mini, Super Mini, Array Proc. |
| Number of Accredited Sources | At Least 2 for Each PT. Suggests 3 or 4. | 2 or More Per PT. UL Not Determined. (Lack of Info.) | One Per PT Possibly Two. | Multiple. |
| Frequency of Intro. | When New Technology Results in Decrease in Total LCC of Old and New. | 4-7 yr. Min. LCC Recommend 5 yr. | 3-6 yr. Min. LCC Recommend 5 yr. | 5-7 yr. @ Technology<br>4-6 yr. @ LCC<br>As PM Requires. |



Total Program Cost
120
110
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
Years Between Redevelopment
Newest technology in inventory
Older technology in inventory
Optimal balance development cost vs. logistics savings
Logistics costs plus development costs
Development costs
Logistics costs

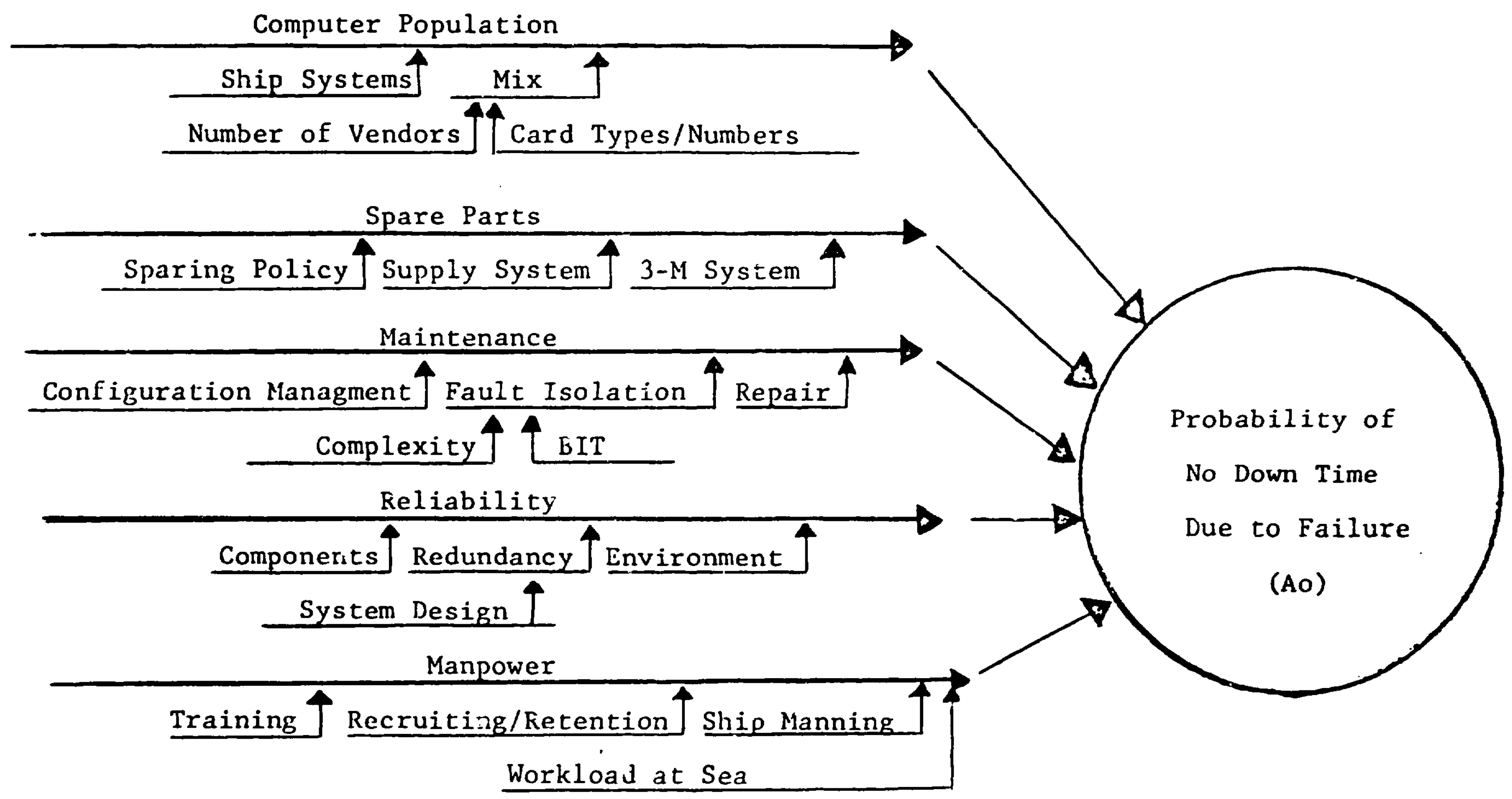
| ISSUE | PRC | GIT | IBM | GE |
|---|---|---|---|---|
| Selection Criteria | LCC | LCC | LCC | MOW: N-Tuple<br>LCC |
| Funding | Navy Fund FSED. | NA | Navy Fund FSED.<br>ROI Determines Level. | Navy Fund Entire Process. |
| Accreditation Administration | Single Policy Administrator To Dev. Procedures, Facilities, and to Conduct Dev. and Accred. | NA | Central Acq. Agency for Planning and Development. | NAVMAT Administrator with PMs for Special Requirements. |
| Transition & Timing | 2 yrs: System Estab.<br>2 yrs: Call for Candidates.<br>4 yrs: Units Accred. | 1980: UYK-43/44,2/PT<br>1985: ISA Accred.<br>1990: HOL Accred. | 1980: 43/44<br>5-year Period. | 43/44 → Accreditation |
| Acquisition Strategy | RFP for Candidates. Multiple FSED (No CD/AD). Accredit One or More. | NA | Concept Planning. RFPs: Dual Development. Accredit One (Two?). Multi-year Prod. Funding. | A-109 |
| Industry Dialogue | Education<br>Announce Intent<br>Open Support & Interest | Conferences<br>Publications<br>Industry Panels | Concept Planning<br>PR Review | Workshops<br>Steering Committees<br>Industry Associations |

## POLICY/IMPLEMENTATION EXTRACTIONS FROM STUDIES

- IMMEDIATELY TO FSED
- 2 TO 4 FSED FOR PERFORMANCE RANGE
- RE-DEVELOP EVERY "5" YEARS (WITH CONSTANT CHANGE RATES)
- $F^3$/ISA STANDARDIZATION (ISA $\longrightarrow$ HOL)
- 2 OR MORE PRODUCERS PER PERFORMANCE RANGE
- VISIBLE LCC, VENDOR ROI/PAYBACK & EFFECTIVENESS BENEFIT ANALYSIS AGAINST EXISTING ACCREDITED COMPUTERS
- INCLUDE MICRO & SIGNAL PROCESSOR PRs
- INCREASED NAVY PRODUCT MANAGEMENT
- VERTICAL LINK SYSTEM/PLATFORM/PROCESSOR & NAVY-WIDE LEVELS WITH LCC & MOW ANALYSES

## UNRESOLVED ISSUES

- ACCREDITED COMPUTER LINE
  - STILL HAVE TO RECOMPETE FOR EACH BUY
  - PIECEMEAL BUYS
  - NO MARKET CERTAINTY
  - SPECIFICITY OF CRITERIA FOR LIMITING THE N'TH + 1 OFFER
- LOGISTICS IMPACT
  - MULTIPLE SOURCES
  - MAINTENANCE DIFFICULTIES
  - TRAINING
  - SPARES



LOGISTICS FACTORS

## ACCREDITATION PREREQUISITES

- NO INCREASE IN LCC

  - AND -

- NO INCREASE IN MANPOWER NUMBERS OR QUALIFICATION/TRAINING REQUIREMENTS

  - AND -

- NO REDUCTION IN OPERATIONAL AVAILABILITY

  - AS COMPARED TO STANDARDIZATION -

## LOGISTICS

- LACK OF SPARE PARTS IS SINGLE LARGEST CONTRIBUTOR TO COMPUTER DOWN TIME AT SEA
- ACCREDITATION WILL REQUIRE INCREASED (OVER STANDARDIZATION) ON-BOARD SPARES TO MAINTAIN GIVEN LEVEL OF OPERATIONAL AVAILABILITY
- OPERATIONAL AVAILABILITY MUST BE MAINTAINED
- CONTINUED SHORTAGES OF HIGH GRADE, HIGHLY TRAINED COMPUTER/ELECTRONICS MAINTENANCE PERSONNEL PROJECTED THROUGH 1986 AND BEYOND
- ACCREDITATION WILL INCREASE MANPOWER/TRAINING REQUIREMENTS FOR SHIPBOARD COMPUTER MAINTENANCE
- NEW R&M TECHNOLOGY AND MILITARY PERSONNEL POLICIES WILL BE NEEDED TO OFFSET THIS BURDEN
- MORE EXTENSIVE USE OF STANDARD/ACCREDITED PROCESSORS IN PLACE OF NON-STANDARD PROCESSORS AND SPECIAL PURPOSE DIGITAL ELECTRONICS COULD SIGNIFICANTLY REDUCE OVERALL SYSTEM MAINTENANCE TRAINING AND SPARING REQUIREMENTS

## LIFE CYCLE COSTS

- ACCREDITATION WILL INCREASE NON-RECURRING ACQUISITION COSTS, E.G., MULTIPLE DEVELOPMENT, PRODUCTION LEARNING CURVES, ETC.
- ACCREDITATION WILL INCREASE MANPOWER TRAINING AND EQUIPMENT LOGISTICS COSTS
- ACCREDITATION MUST LOOK TO PRODUCTION COMPETITION COST BENEFITS TO OFFSET OTHER COST BURDENS
- VALUE (COST PERCENTAGE) OF MARKET COMPETITION NOT DETERMINISTIC, BUT FIGURE COULD BE "ESTABLISHED" FOR ACQUISITION DECISION PURPOSES
- "ESTABLISHED" MARGINAL VALUE OF COMPETITION CAN PROVIDE UPPER LIMIT CUT OFF POINT FOR DETERMINING NUMBER OF VENDORS TO ACCREDIT. LOW BID ON COMPUTED LCC FORMS BASELINE.

## SOFTWARE COMPATIBILITY

- SOFTWARE COMPATIBILITY AMONG ACCREDITED COMPUTERS IN A PERFORMANCE CLASS IS A NECESSITY
- NEAR TERM - USE THE PROVEN, ISA COMPATIBILITY MECHANISM WITH STANDARD HOLS
- LONG TERM - MOVE TO THE HOL LEVEL
  - DIRECT HOL MACHINE (ISA = HOL)?
  - IL MACHINE (ISA = IL) WITH STANDARD HOL?

## POSSIBLE ACCREDITATION PROGRAM OUTLINE

- ESTABLISH MODEL FOR COMPUTING/COMPARING LCC OF CANDIDATE ACCREDITED COMPUTERS
- DEFINE MINIMUM ESSENTIAL $F^3$ SPECS FOR ACCREDITATION, i.e., PACKAGE, ENVIRONMENTAL, I/O AND SOFTWARE INTERFACES, PERFORMANCE, R&M, ETC.
- AGGREGATE EMBEDDED COMPUTER REQUIREMENTS PER PERFORMANCE CLASS FOR 3-5 YEARS TO ACHIEVE ATTRACTIVE MARKET
- SOLICIT PROPOSALS FROM INDUSTRY FOR TWO OR MORE SUPPLIERS PER PERFORMANCE CLASS
- OBTAIN PRICE QUOTES FOR VARIOUS PRICE/PRODUCTION RATES AND QUANTITIES
- CONTRACT QUANTITIES/VENDORS TO MINIMIZE JOINT BUY LCC
- LIMIT JOINT-BUY LCC TO FIXED MARGIN OVER SINGLE LOWEST BID LCC. THIS WILL LIMIT NUMBER OF SUPPLIERS. SINGLE SUPPLIER IN PERFORMANCE CLASS MAY RESULT WHERE MARKET IS TOO SMALL TO SUPPORT MULTI-VENDOR, SPLIT-MARKET COMPETITION
- LIMIT COMBINED MAINTENANCE/TRAINING AND SPARING REQUIREMENTS OF NEW ACCREDITED COMPUTERS TO NOT GREATER THAN THAT OF THE EXISTING COMPUTER(S) TO BE DISPLACED. THIS WILL ALSO CAP NUMBER OF SUPPLIERS.
- CONTRACT FOR VENDOR PRODUCTION RATE/QUANTITY
- REINITIATE CYCLE AT 3-7 YEAR (5 YEAR?) INTERVAL
- PROHIBIT/DISCOURAGE OUT-OF-CYCLE ACCREDITATIONS FROM INDIVIDUAL SOURCES, EXCEPT PERHAPS FOR EXTRAORDINARY, UNFORSEEN TECHNOLOGICAL BREAK-THROUGH. MUST STILL SHOW OVERALL LCC WIN IN FACE OF MARKET DISRUPTION. TREAT SPECIAL REQUIREMENTS WITH WAIVERS.
- REVIEW/UPDATE ACCREDITATION CRITERIA PERIODICALLY
- KEEP INDUSTRY APPRISED

# REPORT OF THE PANEL ON ESTIMATING SOFTWARE COSTS

TAB D
Software Cost

PROCEEDINGS OF THE SOFTWARE WORKSHOP

JOINT LOGISTICS COMMANDERS

JOINT POLICY COORDINATING GROUP ON COMPUTER RESOURCE MANAGEMENT

MONTEREY, CALIFORNIA, 22-25 JUNE 1981

# REPORT OF THE PANEL ON ESTIMATING SOFTWARE COSTS

CHAIRMAN: R. DEAN HARTWICK
INTERCON SYSTEMS CORPORATION

CO-CHAIRMAN: ROBERT E. BERRI
AEROSPACE CORPORATION

TABLE OF CONTENTS

REPORT OF THE PANEL ON ESTIMATING SOFTWARE COSTS

PAGE

## 1. OBJECTIVE

Evaluate existing software cost estimating models and recommend to the Joint Logistics Commanders Joint Policy Coordinating Group on Computer Resource Management a Tri-Service approach to improve software cost estimating methodology.

## 2. SCOPE

The percentage of total embedded computer systems cost that is allocated to software is increasing. Software cost estimating methodology has never been satisfactory, and it is becoming more critical that a satisfactory methodology be developed. Such a methodology should:

- Improve the ability to forecast total software life cycle costs during the Concept Exploration phase.

- Predict software development and maintenance costs during the Demonstration and Validation phases and during the Full Scale Development phase.

- Allow a standard for evaluation of software components of proposals.

- Provide a means to update project schedule and resource estimates.

- Incorporate evolving software estimating technology and reflect advances of improved software engineering technology.

This panel was asked to review existing software estimating models, and assess each model's limitations and benefits, including where models are beneficial and where they are inadequate. Further, the panel was to establish for which life cycle phases software costs could be estimated, and assess the accuracy of the estimates by phase. The fundamentals of software cost estimating were to be reviewed, and an assessment made of the usefulness of parameters and the desirability of multi-variable algorithms.

The panel was also asked to determine software cost estimating needs for each life cycle phase. These needs were to be established through all phases, and were to be expressed in terms of the use to be made of the model, the output desired, the desired accuracy of the output, and the input required to drive the model. The means of using the methodology were to be reviewed, with analyses to include the constraints imposed by resources, and the probable background of users. The contractual aspects of using the methodology were to be studied and recommendations for implementing the methodology developed. Finally, a software cost estimating technology program needed to meet DoD needs was to be developed.

## 3. APPROACH

### 3.1 Preparation

The Panel had been provided references (Appendix C) to read in preparation for the meeting, prior to the Workshop. In addition, Panel members brought copies of reference material for use. These references are listed in Appendix B.

The first meeting of the Panel (see Appendix A for attendees) started with presentations on individuals' background, information on current cost models, industrial practices, and automated library practices. Copies of these presentations are provided in Appendix D.

### 3.2 Issues

Based upon the Panel references and the Panel presentations, the Panel established a list of issues which are tabulated in Table 1. The items in this table express concern in a number of different ways. The Panel concluded, for example, that most available cost models are deficient in some way. This conclusion is based on the considerable body of survey information already available, and it was concluded that additional surveys would not be productive. There also was concern that existing models do not meet all system life cycle needs, do not cover all software-related functions (e.g., software quality assurance) and require considerable expertise to use. Finally, there was a general concern about legal and contractual aspects of the use of models in the acquisition process.

### 3.3 Issue Consensus

The Panel collectively had a broad background and the members were able to present varying viewpoints on the subject, as itemized by the issues in Table 1. A list of items on which the entire Panel could concur was developed. This list is shown in Table 2. A significant conclusion is that DoD has common Software Cost Estimation (SCE) needs, the methodology should be directed toward embedded weapons systems methodology but that this SCE methodology was not necessarily implemented in one tool or model. It was suggested to the Panel that the DoD automated data processing community was considering selecting an existing model as a standard. The Panel concurred that no existing SCE model or technology would satisfy the needs of the embedded computer system community. Therefore, it concluded that, although such an existing methodology or model might be applicable to automated data processing needs, the Panel should restrict its attention toward an SCE methodology that is directed toward embedded weapons systems applications. Since the methodology would be at the DoD level, it also appears that a

Table 1. SCE Methodology Issues

|  | SOFTWARE COST ISSUES | SUB-PANEL |  |  |
|---|---|---|---|---|
|  |  | 1 | 2 | 3 |
| 1 | Definition of "ultimate" unit of pure software for model definition |  | X |  |
| 2 | The role of bottoms-up engineering-how do you get to it? |  | X |  |
| 3 | The WBS technology needed to support software estimating. |  | X |  |
| 4 | Should DoD data be separated from contract or data? | X |  |  |
| 5 | How are improved technology/productivity introduced in method? |  | X |  |
| 6 | How does method reflect standing resources to support software? | X | X |  |
| 7 | What can be reasonable expectations for accuracy? |  | X |  |
| 8 | How can methodology be adjusted for user? | X |  | X |
| 9 | Will proprietary models or data be a problem? |  |  | X |
| 10 | Is another survey needed (or a survey of surveys)? |  | X |  |
| 11 | How do you check the reasonableness of ECPs? | X |  | X |
| 12 | What do we need cost models for? | X |  |  |
| 13 | What is a plan for getting better models? |  | X |  |
| 14 | What software-related data are needed for models? | X | X | X |
| 15 | What metric data are needed to support technology? | X | X | X |
| 16 | For what system life cycle phase do you need a model? | X |  |  |
| 17 | Are models needed for software-related functions (e.g., QA)? | X |  |  |
| 18 | What do you do with model data once you have it? | X |  | X |
| 19 | What is ultimate expectation of this technology? |  | X |  |
| 20 | What are parameters/benefits, by life cycle phase, of model? | X |  |  |
| 21 | How do you account for wasted effort in Support model? |  | X |  |
| 22 | Do we need both manual and automated models? | X | X |  |
| 23 | Must models be transferrable? Do we need multiple models? | X | X |  |
| 24 | How dependent are models on application or sensitivity of parameters? |  | X |  |
| 25 | What are required provisions for adopting a standard model? | X | X |  |
| 26 | How should feed back be used to assess/tune model? | X |  | X |
| 27 | What kind of estimating regulatory requirements should DoD use? | X |  | X |

Table 2. Panel Census Items

| | CONSENSUS ITEMS |
|---|---|
| 1 | DoD should have a standard SCE methodology. |
| 2 | This standard SCE methodology does not exist now. |
| 3 | Data collection for SCE should be prescribed by DoD. |
| 4 | SCE methodology should address all phases of the S/W life cycle. |
| 5 | SCE methodology should be readily understandable by all anticipated users. |
| 6 | DoD SCE methodology should provide for firmware too. |
| 7 | The DoD SCE methodology should be directed toward embedded systems. |
| 8 | JLC should designate a central organization to control and evolve SCE methodology in light of technical upgrade. |
| 9 | Stat-of-the-art of SCE models is not sufficient. SCE technology should be upgraded (data required, assumptions, methodology). |
| 10 | Assumptions and adaptation parameters must be tailored by all users (for models and methods). |
| 11 | Model (methodology) should be used for: (a) subset of system cost model for contract feasibility; (b) budget projection; (c) POM resource estimation; (d) source qualification and selection; (e) industrial planning and bidding; (f) quantify impact of technology upgrade; (g) predict impact of requirements changes; (h) predict impact of ECPs; (i) monitor performance and estimate cost of completion; and (j) scope impact of budget modifications. |
| 12 | MIL-STD-881 is deficient and should be improved for SCE needs. |
| 13 | Embedded systems developments require unique SCE methodology. |
| 14 | DoD should have a common, tri-service software acquisition method. |
| 15 | DoD should not impose the use of an exclusive model now. |
| 16 | DoD should encourage the use of models now in acquisition practices. |

central organization would be required to control the process and improve it as technology and acquisition regulations change.

The Panel failed to concur on only one point: should cost data reporting be tied to present financial reporting. This could have legal implications about which the Panel could not achieve a consensus without considerable analysis.

## 3.4 Issue Resolution

The issues and consensus items (Tables 1 and 2) were divided into the three principal groups indicated. Three Sub-Panels were formed, and items assigned to the three Sub-Panels for development of appropriate conclusions. The Sub-Panels addressed: (1) definition of the SCE methodology requirements for all aspects of the system life cycle and the expected use of the methodology; (2) evaluation of both the existing methodologies and the Panel's derived requirements, and recommendations for research and development of the required methodology; and (3) legal and contractual implications associated with using a DoD - level SCE methodology. The Sub-Panel findings are provided in Section 4. Members of the three subpanels were:

| Sub-Panel 1 Life Cycle Requirements | Sub-Panel 2 Model Technology | Sub-Panel 3 Contractual Considerations |
|---|---|---|
| Robert Earnest | Roger Bate | Marilyn Fujii |
| Ronald Leask | Deane Bergstrom | Clell Gladson |
| Herman Oberkrom | Joseph Duquette | Richard Latshaw |
| Richard Page | James McCall | George Robertson |
| David Thornell | Robert Paulsen | Newnam Thompson |
| David Usechak | | George Trever |

The panel co-chairmen, Dean Hartwick and Robert Berri, coordinated work among the Sub-Panels. For the remainder of the workshop, the Panel primarily worked within the subpanel sessions, with periodic meetings to prepare the presentations to the entire workshop.

## 4. DISCUSSION

Many different software cost estimating models have been developed and used by DoD and industry, with varying results. Within DoD, SCE models have most commonly been used to develop independent cost estimates for budgetary purposes. More recently, they have been used to evaluate sources and support software developer selection.

Most DoD contractors use software cost estimation models during proposal activities. In some cases these models are proprietary and have been developed solely by the contractor organization based on its past development experiences. However, because of the use by government organizations of one of the commonly available cost models and because the development of an accurate cost model is difficult, most contractors tend to use one or more of the common models. The use of software cost estimation models by contractors is almost always a supplementary activity to the traditional bottoms-up engineering estimate generated by the technical contributors to the proposal. The bottoms-up engineering estimate can be very accurate if the development is based on similar past experiences of the technical personnel. The disadvantages of this technique for JLC use is its questionable applicability to new or unique developments and the requirement that a significant amount of preliminary design effort is required to effectively make a bottoms-up engineering estimate. This detail of information is not generally available nor is the time or resources available to generate it early in the acquisition life cycle when estimates are required.

The organizations, both government and industry, that are making effective use of the models have certain common characteristics in how they are using the models. These characteristics are:

- The models are used within a cost estimation and evaluation methodology which tightly controls the input to and use of the models. These methodologies cause a certain discipline to be enforced and result in consistent, repeatable, and documented use of the models.

- The models have been calibrated and tuned to the specific application environment in which they are used.

- The models are used to perform risk analysis and identify sensitivities and deficiencies.

- The models only supplement common sense and the estimates of knowledgeable and experienced personnel.

- The methodologies incorporate a concept of evolutionary development and refinement of the cost estimation models.

In Section 4.1, the SCE models in common use by DoD and industry are summarized according to modeling parameters, applicability to different life cycle phases, and how closely the models estimate actual efforts. A series of conclusions are drawn in Section 4.1, but an overall assessment is that no existing SCE model or methodology can even remotely be considered now as a basis for a JLC embedded computer software cost estimating standard.

One of the characteristic shortcomings attributed to SCE models is that they have all been developed to satisfy an

Table 3. Summary of SCE Model Surveys

| | Models | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Assessment Technique | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SURVEYS | AEROSPACE | VCS | DoD MICKO ESTIMATING | DOTY | FARR & ZAGORSKI | PRICE-S | SLIM/PUTNAM | TECOLOTE | TRW/WOLVERTON | ARON/IBM | STRUCTURED PROG SERIES/IBM | MYERS/IBM | MALONE/IBM | GRC DISAGGREGATED | GRC AGGREGATED | WALSTON-FELIX | GTE SYLVANIA | SCHWARTZ-CSC | HANSON-SAMSO | SDC | HAHN-SONE/MITRE | GUEL-OKUMOTO | KUSTANOWITZ | RUBEY | SCHNEIDER | CHEN | ESD SOFTWARE WORKSHOP | NADC | SEL | MODEL FEATURES | COST FACTORS | ACCURACY ASSESSMENT | LIFE CYCLE PAHSE COVERAGE |
| THIBODEAU (GRC) | X | X | X | X | X | X | X | X | X | | | | | | | | | | | | | | | | | | | | | ✓ | ✓ | ✓ | ✓ |
| BIZATMAN (SDC) | X | X | X | | | X | X | X | | X | X | X | X | X | X | | X | X | X | X | X | | | | | | | | | ✓ | ✓ | | ✓ |
| QUANTITATIVE SOFTWARE MODEL SURVEY (RADC) | X | | | X | X | | X | X | X | | | | | | X | | | | | X | | X | X | X | X | X | X | | | ✓ | ✓ | | |
| JAMES (AFAL) | X | | | | | | | X | X | | | | X | | | | | | | | | | | | | | X | X | | | ✓ | ✓ | |
| JUNK (GE) | X | X | | X | | X | X | X | X | | | | | | X | X | | | | X | | | | | X | | | | | ✓ | ✓ | | |
| COOK (NASA) | | | | X | | X | X | X | X | | | | | | X | X | | | | | | | | | | | | | X | | ✓ | ✓ | |

existing software development environment. In Section 4.2, the requirements that SCE methodology (including the use of SCE models) must satisfy to adequately estimate DoD embedded computer software life cycle costs are discussed. Three different needs are formulated - using the SCE methodology to prepare long range budget estimates; supporting software acquisition; and to support software maintenance and enhancement after deployment. Additionally, the necessity of the SCE methodology to react to improved software engineering technology and to satisfy operational needs are considered.

All SCE models gain their credibility and responsiveness as a result of the data used in their development. A shortcoming of most models is that a coherent and well-defined data base has not been consistently collected or maintained. In Section 4.3, this problem is discussed, and an approach that the JLC can follow to correct these problems is proposed. This approach consists of establishing a single Government agency to serve as an SCE data repository, and assigning it the task of defining data collection techniques and procedures. Also it is concluded that MIL-STD 881A should be modified to better reflect the complexity of computer software.

A most important consideration in the use of SCE methodology is the manner in which it should be used. In Section 4.4, the ramifications of using SCE model results are considered. Among the conclusions drawn there are that SCE methodology should not be used to determine fees and progress payments. However, SCE methodology can be used to support source selection and to aid in Government technical performance measurement.

A final area of discussion, Section 4.5, relates to the question of the evolution of SCE methodology. That is, given the present state-of-art and the obvious needs of the embedded computer software community, what steps can be taken by JLC to encourage technology advancement. It is concluded that procedures and guidelines should be established to support DoD in using existing SCE methodology over the next three years, while JLC should sponsor a technology improvement program to achieve a common methodology in the next three to seven years.

## 4.1 State-of-the-Art Of Software Cost Estimation Technology

Several extensive surveys of cost models have been conducted under DoD sponsorship to assess the utility of current software cost estimation models. Table 3 identifies six surveys and the models evaluated during those surveys. Also indicated in Table 3 are the assessment techniques utilized in conducting the evaluation. These assessment techniques include evaluating the features of the model, what cost factors are taken into account in the model, what life cycle phases are covered by the model, and how accurately the model predicted the costs of actual software developments.

These surveys represent evaluation of the models from the following application viewpoints:

- Management information systems
- Command and control
- Data base management systems - commercial
- Command, control, communication and intelligence
- Flight dynamics
- Real-time applications
- Space, aircraft, avionics
- Weapon systems
- Scientific and engineering
- Logistics and maintenance

In each case a structured consistent evaluation process was utilized. To illustrate some of the findings:

- Table 4 identifies what cost factors are accounted for in each of the cost models surveyed. Note that no one model considers all of the factors that intuitively have an impact on the cost of a software development project.

- Table 5 identifies what cost elements and what life cycle phases are covered by each of the cost models surveyed. Table 6 represents the same evaluation from another survey. In this case a weight or score was assigned. Note that the phase coverage of the models is limited. Either the model does not cover the phase or does not cover it in the level of detail required.

- Table 7 provides an error evaluation of several of the models across three very distinct application areas: a commercial data base management system, the Air Force Data System Design Center, and NASA/Goddard flight dynamic systems. Note that the models, in general, demonstrated poor accuracy in estimating actual costs and also their accuracy varied greatly from one application environment to another.

- Table 8 describes the results of an accuracy assessment within one development environment for one application. The findings here indicated that a very simple model, the SEL model, developed strictly from local data, was as effective as more complex and expensive models.

TABLE 4. COST FACTORS CONSIDERED IN COST ESTIMATION TECHNIQUE

| COST FACTORS | DOTY | IBM | GRC | PUTNAM | SDC | TRW | PRICE-S | BOEING |
|---|---|---|---|---|---|---|---|---|
| DEVELOPMENT EXPERTISE | | | | | | | | |
| PERSONNEL EXPERIENCE AND QUALIFICATIONS | | X | | I | X | | I | X |
| DESIGN PERSONNEL PARTICIPATION IN DEVELOPMENT | | X | | | X | | | |
| EXPERIENCE WITH COMPUTER SYSTEM | | X | | | | | I | |
| EXPERIENCE WITH PROGRAMMING LANGUAGE | | X | X | | X | | I | |
| EXPERIENCE WITH APPLICATION | | X | | | X | | I | X |
| CUSTOMER ENVIRONMENT | | | | | | | | |
| CUSTOMER INTERFACE COMPLEXITY | | X | | | | | | X |
| USER PARTICIPATION IN REQUIREMENTS DEFINITION | | X | | | | | | |
| REQUIREMENTS STABILITY | X | X | X | | X | | I | |
| CUSTOMER EXPERIENCE WITH APPLICATION | | X | | | | | | X |
| SECURITY REQUIREMENTS | | X | | | | X | | |
| DOCUMENTATION REQUIREMENTS | | X | | | X | | I | X |
| MULTIPLE OPERATIONAL SITES | | | | | | | | |
| MULTIPLE CONTRACTORS | | | | | | | | |
| FORMALIZED VV&C | | | | | | | I | |
| TRAINING OF OPERATING AGENCY | | | | | | | | |

| COST FACTORS | DOTY | IBM | GRC | PUTNAM | SDC | TRW | PRICE-S | BOEING |
|---|---|---|---|---|---|---|---|---|
| DEVELOPMENT ENVIRONMENT | | | | | | | | |
| NUMBER OF PERSONNEL | | | X | | | | | X |
| DURATION OF PERSONAL ASSIGNMENTS | | | | | X | | | |
| CONCURRENT HARDWARE/SOFTWARE DEVELOPMENT | X | X | | | X | | I | |
| ACCESS TO DEVELOPMENT COMPUTER | X | X | | X | | | | |
| USE OF MODERN PROGRAMMING PRACTICES | | X | | X | | | I | |
| DETAIL OF REQUIREMENTS DEFINITION | X | | | | X | | | |
| DEVELOPMENT VIA INTERACTIVE OR BATCH | X | | X | X | X | | | X |
| DEVELOPMENT AT OPERATIONAL SITE | X | | | | X | | I | |
| OPERATIONAL SYSTEM DIFFERENT FROM DEVELOPMENT SYSTEM | X | | | | X | | | |
| DEVELOPMENT AT MORE THAN ONE LOCATION | X | | | | X | | | |
| CAPABILITY OF TOOLS AND AIDS | | | | X | | | | X |
| SUPPORT PERSONNEL AVAILABILITY | | | | | | | | |
| RELIABILITY OF DEVELOPMENT SYSTEM | | | | | | | | |
| MANAGEMENT TECHNIQUES | | | | | | | | |
| STANDARDS AND PROCEDURES | | | | | | | | |
| CONFIGURATION MANAGEMENT | | | | | | | I | |
| IMPLEMENTATION LANGUAGE HOL/MOL | | | | X | | X | | X |

X = MODEL DIRECTLY CONSIDERS FACTOR
I = MODEL INDIRECTLY CONSIDERS FACTOR

Reference:
Junk, et al, "Survey of Software Cost Estimating Techniques," GE/MDSO CIS 010, May 1978

Table 4. Cost Factors Considered in Cost Estimation Technique (Con't.)

| COST FACTORS | DOTY | IBM | GRC | PUTNAM | SDC | TRW | PRICE-S | BOEING |
|---|---|---|---|---|---|---|---|---|
| <u>TECHNICAL</u> | | | | | | | | |
| NUMBER OF DELIVERED INSTRUCTIONS | X | X | X | X | | X | X | X |
| NUMBER OF PROGRAM MODULES | | | | X[1] | | | X | |
| APPLICATION CATEGORIES | X | X | | | X | | | |
| INSTRUCTION MIX | | X | | | I | X | X | X |
| DATA BASE SIZE | | X | | X[1] | X | | I | |
| COMPLEXITY OF CODE | | X | X | | X | X | | |
| OUTPUT/DISPLAY REQUIREMENTS | X | | | X[1] | X | | | |
| RESOURCE CONSTRAINTS | X | X | X | | X | X | X | |
| TECHNOLOGY LEVELS | | | | X | | | X | |
| MODIFICATION OF EXISTING SOFTWARE | | | X | P | | X | X | X |
| FIRST SOFTWARE DEVELOPED ON CPU | X | | | | X | | I | |
| QUALITY REQUIREMENTS | | | | | | X | | |
| <u>SCHEDULE</u> | | | | | | | | |
| ELAPSED TIME | | | | X | | | X | |
| MILESTONE PLACEMENT | | | | | | | X | |

X = MODEL DIRECTLY CONSIDERS FACTOR

I = MODEL INDIRECTLY CONSIDERS FACTOR

P = MODEL PARTIALLY CONSIDERS FACTORS

[1] SECONDARY RELATIONSHIP

TABLE 5. EVALUATION OF COST
ESTIMATION MODELS AGAINST SELECTION CRITERIA

| SURVEYED MODELS | LARGE REAL TIME COMMAND/CONTROL | REQUIREMENT DEFINITION PHASE | DEFINE | DESIGN | CONTRUCT | TEST | INTEGRATE | TRANSITION PHASE | OP & MAINT PHASE | CONVERSION | COMPUTER | DOCUMENTATION | MANAGEMENT | SYSTEM ENGINEERING | CONFIGURATION CONTROL | GENERIC TYPE | APPLICABLE MODELS | REQUIRED COSTS | COMPLEMENT RY MODELS | SELECTED MODELS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1. DoD, MICRO LEVEL | | | P | P | P | P | P | | | | | S | | | | AF | | | | |
| 2. DoD, MACRO LEVEL | | | P | P | P | P | P | P | P | | | | | | | PE | | | | |
| 3. ARON-IBM | X | | S | P | P | P | S | | | | S | | S | | | PP | ✓ | ✓ | | |
| 4. RADC, S.P. SERIES | X | | S | P | P | P | S | | | | S | | S | | | PP | ✓ | ✓ | ✓ | |
| 5. MYERS-IBM | X | | S | S | P | S | S | | | | | | | | | PP | ✓ | ✓ | | |
| 6. MALONE-IBM | X | [illegible] | P | P | P | P | S | | | | S | | S | | | RE | ✓ | | | |
| 7. GRC-FEFINED DISAGGRE-GATED | X | | P | P | P | P | P | | | | S | S | S | | | RE | ✓ | ✓ | ✓ | ✓ |
| 8. GRC-AGGREGATED | X | | S | P | P | P | S | | | | S | | S | | | PP | ✓ | ✓ | | |
| 9. TELCOLOTE | X | P | P | P | P | P | | | | | P | | | | | RE | ✓ | | | |
| 10. DOTY/PUTNAM-ESD | X | | P | P | P | P | S | S | S | | | S | S | | | RE* | ✓ | ✓ | ✓ | ✓ |
| 11. GTE SYLVANIA | | | P | P | P | P | P | | | | S | S | | | | PP | | | | |
| 12. SCHWARTZ-CSC | | | S | S | P | P | S | | | | | | | | | AF | | | | |
| 13. HANSEN-SAMSO | X | P | P | P | P | P | P | | | | | S | S | | | RE | ✓ | | | |
| 14. PRICE | X | | P | P | S | S | S | | | | | S | S | S | S | PP | ✓ | ✓ | ✓ | ✓ |
| 15. RCA-INTERIM STANDARD | | S | S | P | P | S | S | | | | | | | | S | PP | | | | |
| 16. BOEING COMPUTER SERVICES | X | S | S | S | S | S | S | | | | S | | | | | PP | ✓ | | | |
| 17. SDC COST ESTIMATION METHOD | X | P | P | P | P | P | P | | | | | S | | | | PP | ✓ | ✓ | ✓ | ✓ |
| 18. HAHN & STONE-MITRE | X | | | | | | | | | P | | | | | | AF** | ✓ | | ✓ | ✓ |

*COMBINATION OF REGRESSION EQUATION AND PUTNAM MODELS

**COMBINATION OF PROGRAMMER PRODUCTIVITY AND ARITHMETIC FORMULA MODELS

***A SINGLE PRIMARY COST ELEMENT IS ALLOCATED BY EXTRAPOLATION TO THE LIFE CYCLE ACTIVITIES

Reference: Bratman, "Analysis of Cost Estimation Models," SDC-TM-L-6132, 16 June 1978.

**LEGEND**

| | |
|---|---|
| P | PRIMARY COST ELEMENTS |
| S | SECONDARY COST ELEMENTS |
| PP | AGGREGATED COST ELEMENTS |
| SS | AGGREGATED COST ELEMENTS |
| AF | ARITHMETIC FORMULA MODEL |
| PE | PUTNAM'S EQUATION MODEL |
| PP | PROGRAMMER PRODUCTIVITY MODEL |
| RE | REGRESSION EQUATION MODEL |

TABLE 6. SUMMARY OF MODEL COMPLIANCE

WITH AIR FORCE ESTIMATING REQUIREMENTS

| ESTIMATING SITUATION | AEROSPACE | BOEING | DOD MICRO | DOTY | FARR & ZAGORSKI | PRICE S | SLIM | TECOLOTE | WOLVERTON |
|---|---|---|---|---|---|---|---|---|---|
| 1. CONCEPTUAL | | | | | | | | | |
| Scope | 2 | 4 | 4 | 1 | 1 | 3 | 5 | 4 | 4 |
| Detail | 5 | 5 | 5 | 4 | 4 | 5 | 5 | 5 | 5 |
| 2. CONCEPTUAL | | | | | | | | | |
| Scope | 2 | 4 | 4 | 1 | 1 | 3 | 5 | 4 | 4 |
| Detail | 4 | 4 | 4 | 5 | 5 | 5 | 4 | 4 | 4 |
| 3. CONCEPTUAL | | | | | | | | | |
| Scope | 3 | 4 | 4 | 2 | 2 | 4 | 5 | 4 | 4 |
| Detail | 2 | 4 | 3 | 2 | 2 | 4 | 3 | 3 | 4 |
| 4. VALIDATION | | | | | | | | | |
| Scope | 3 | 3 | 4 | 2 | 2 | 4 | 5 | 3 | 4 |
| Detail | 1 | 3 | 2 | 1 | 1 | 3 | 2 | 1 | 4 |
| 5. FULL SCALE DEVELOPMENT | | | | | | | | | |
| Scope | 3 | 3 | 4 | 3 | 3 | 4 | 5 | 3 | 4 |
| Detail | 1 | 3 | 2 | 1 | 1 | 3 | 2 | 1 | 4 |

NOTE: Numbers indicate degree to which the model satisfies the particular estimating requirements. 5 indicates nominal satisfaction of the requirement.

Reference: Thibodeau, "An Evaluation of Software Cost Estimating Models", RADC TR-, February, 1981.

TABLE 7. SUMMARY OF MODEL ESTIMATING PERFORMANCE

| MODEL TYPE | RMS ERROR* MEAN PROJECT SIZE DATA SET COMMERCIAL | DSDC | SEL |
|---|---|---|---|
| REGRESSION | | | |
| A AEROSPACE | 1.35 | 2.11 | 0.605 |
| B DOTY | | 1.05 | |
| C FARR & ZAGORSKI | | 16.9 | |
| D TECOLOTE | | 4.36 | |
| E (Recalibrated) | 0.643 | 0.933 | 0.309 |
| HEURISTIC | | | |
| F BCS | | 0.787 | |
| G DoD MICRO | | 1.26 | |
| H PRICE-S | 0.383 | 1.44 | 0.297 |
| I TRW/WOLVERTON | | 0.927 | |
| PHENOMENOLOGICAL | | | |
| J SLIM | 0.246 | 0.216 | 0.865 |

$$\text{*RMS ERROR} = \left[ \frac{1}{N} \cdot \sum_{i=1}^{N} (ACT_i - EST_i)^2 \right]^{1/2}$$

Reference: Thibodeau, "An Evaluation of Software Cost Estimating Models," RADC TR-, February 1981.

TABLE 8. AVERAGE % ERROR FINDINGS

| MODELS | AVE % ERROR* |
|---|---|
| DOTY | 80 |
| WALSTON-FELIX | 16 |
| TECOLOTE | 19 |
| GRC AGG | 112 |
| SLIM | 18 |
| PRICE-S | 20 |
| SEC | 16 |

* Applied to 8 actual projects.

Reference: Cook, "An Appraisal of Selected Models for Software Systems", NASA X-582-81-1, December 1980.

● Figure 1 illustrates a comparative evaluation of the effort-time trade-off between three models. The Putnam model and PRICE-S model are the only models which recognize this trade-off.

The conclusions drawn from these surveys are as follows:

Conclusion 1: The statistical confidence with which one can use the current cost estimation models is quite low. The demonstrated accuracy of current cost estimation models is insufficient for JLC application.

Conclusion 2: The current cost estimation models' performance varies greatly from one development environment to another and from one application to another.

Conclusion 3: The current cost estimation models do not typically cover all of the life cycle phases of a software product in the required level of detail.

Conclusion 4: The current cost estimation models do not typically cover all of the cost elements in the required level of detail nor are they related to the Work Breakdown Structure of the development.

Conclusion 5: Complicated models have not proven to perform better than very simple models.

Conclusion 6: The burden of an accurate estimate is on the user. The current models require full understanding of their characteristic behavior to a particular environment even though this behavior cannot be logically related within the model to specific characteristics of the development environment. The user is also required to fully understand the definition of the input parameters even though user documentation for the models is typically poor. In order to minimize estimation error the user must do extensive calibration and tuning of the models.

Conclusion 7: Current cost estimation models are better able to satisfy needs of JLC early in acquisition life cycle.

Conclusion 8: No one current cost estimation model satisfies all of the JLC cost estimation needs.

Conclusion 9: Reliable historical data for model development or validation is almost non-existent.

The following technology needs exist based on this assessment of the present state-of-the-art of software cost estimation models:

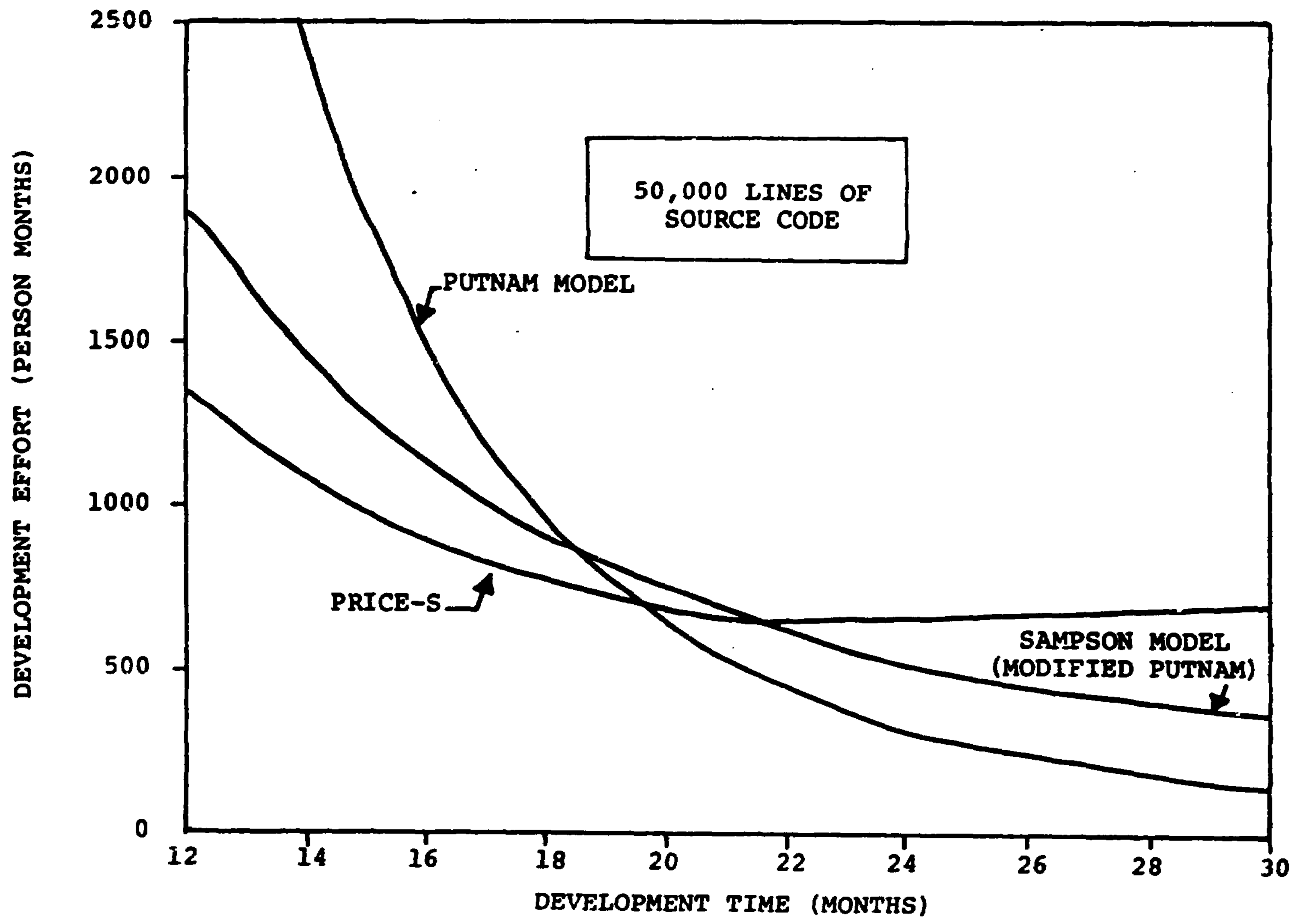


Figure 1. Effort-Time Tradeoff

Reference: Junk, et al, "Survey of Software Cost Estimating Techniques," GE/MDSO 78 CIS 010, May 1978

Conclusion 10: A software cost estimation (SCE) methodology (including definition of SCE models) should be developed which covers the life phases completely by providing the required cost estimation information to the user based on data available at that time in the acquisition life cycle. They should be adaptable to development environment, application, and software development methodology, and be capable of being automated, with tools to support decision analysis.

Conclusion 11: Basic research should be conducted into techniques, determining parameters that characterize the software development environment, and the influence of application environment upon cost model accuracy.

Conclusion 12: While no more surveys appear to be required, additional evaluation of why the current models perform differently in different application and development environments would provide additional insight.

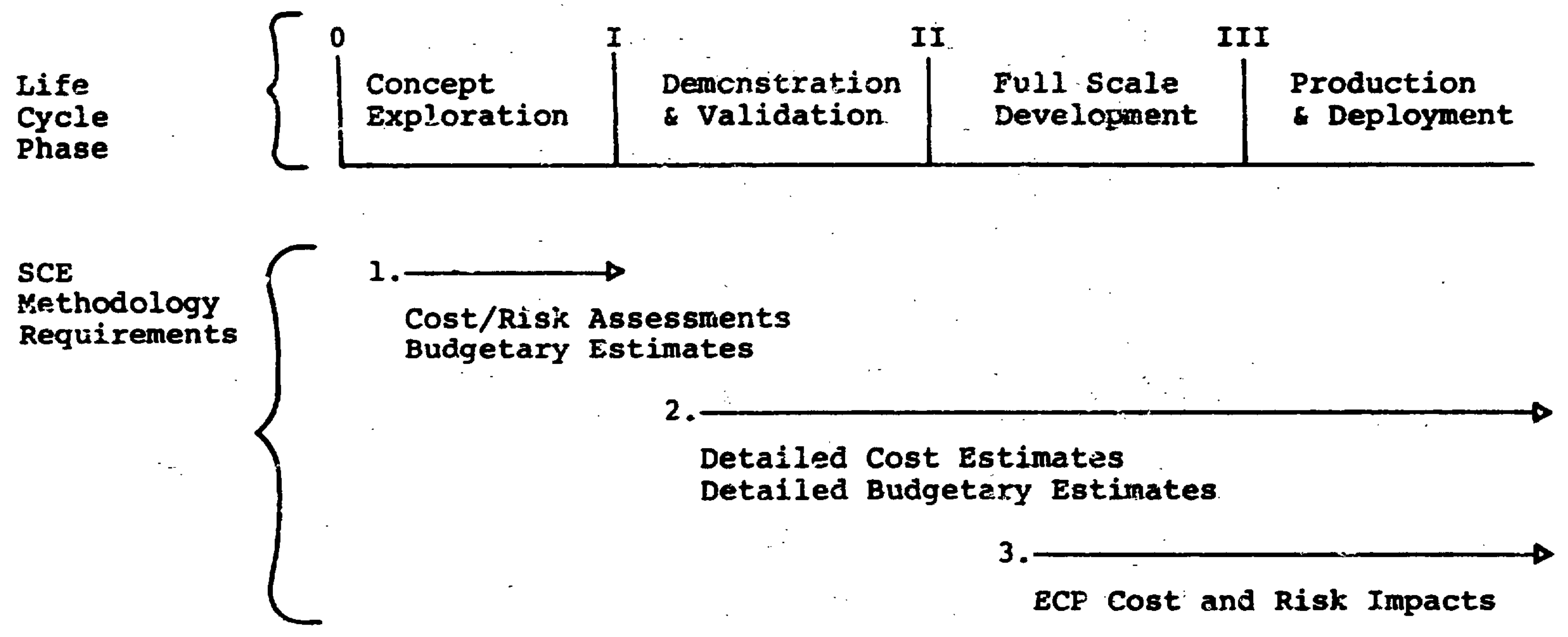
## 4.2 DoD Software Estimating Requirements

### 4.2.1 Life Cycle Considerations

Software cost estimates are typically of two classes: software lifecycle costs for a specific system, and cost and schedule for a specific change to the software of the specific system. Prior to developing recommendations for an overall SCE methodology, the issue of what SCEs are required at various phases in the system life cycle (Figure 2) in order to support budget actions and management program planning must be addressed. Within the various phases of the software system life cycle certain types of information are needed to provide costing estimates. These estimates can then be used for system life cycle cost estimating and budgeting.

An SCE methodology can be used in Concept Exploration to perform cost/risk assessments. This is the first opportunity to examine longer range costs of software, given a reasonable, though imprecise system concept together with gross functional allocations of hardware/software. The SCE methodology, used with a historical data base of cost information. can be used to formulate rough estimates of system cost and potential risk in development. A model is needed to generate the above output during the early phases of the system life cycle. Different types of data are required for this early process. First a need or operational requirement should have been stated which as a minimum should contain system performance testing requirements, and support philosophy (i.e. maintenance and manning). Secondly, the type of (hardware and software) technology anticipated for use on the program must be included. Third, historical system data derived from past similar systems, technologies, and methodologies should be included to form the

Figure 2. System Acquisition Life Cycle



initial cost estimating base. Figure 3 is a graphic representation of this. At the end of any of the software system life cycle phases and before the start of the next phase, more refined cost estimate data are required in order to allow for appropriate decisions.

In the Demonstration and Validation phase, SCE methodology can be used to assist the Government/Contractor in estimating the cost of a particular system design on either a near-term development basis or a longer term life cycle cost basis. Variations of the design can also be tested for cost sensitivity and allocations of functions can be shifted from across the hardware/software boundary to reach an acceptable system cost advantage or cost/benefit tradeoff. As a new technology or design approach is considered, SCE methodology can assess the cost advantage of a new "strawman" design in comparison with earlier technology. The types of input and output data required for this detailed estimation are shown in Figure 4.

As a system follows the acquisition cycle as given in Figure 2, a more detailed cost estimate is needed very early in the Full-Scale Development phase. As the decision is reached to build a particular item in the Full Scale Development phase, more accurate assessments of cost/performance can be made through hardware/software allocations and tradeoff analyses. The basic design is unfolding and the results of analyses become more precise and firm. The hardware/software design is more visible, and historical data being acquired during the course of the contract are more exact. However, the same methodology shown by Figure 4 can be used, merely using the better input to improve estimates. The SCE methodology can expose preferred design approaches as various parts of the system are more finely tuned. At this point, the most cost-effective approach should appear in terms of development potential.

During the production phase, SCE methodology can be used to assess all cost impacts due to the many late changes that must be incorporated into the production design. Incorporating multiple software changes into the production version of the system may become increasingly complex because of the necessity to integrate modified CPCIs into a finished software package. Here, SCE methodology assists the Government/Contractor in assessing the overall cost impact of late, and possibly voluminous, changes to production versions of system software. During software maintenance, especially for large-scale ground based systems deployed in widely dispersed geographic locations, extensive support resources can be required to maintain system software. SCE methodology should be applied to determine the long term cost of support services. Because of the critical requirement for weapon system readiness, adequate staffing with skilled personnel implies a costly continuing training and retraining activity to be carried on throughout the system life cycle.

Figure 3. SCE Parameters during Concept Exploration Phase

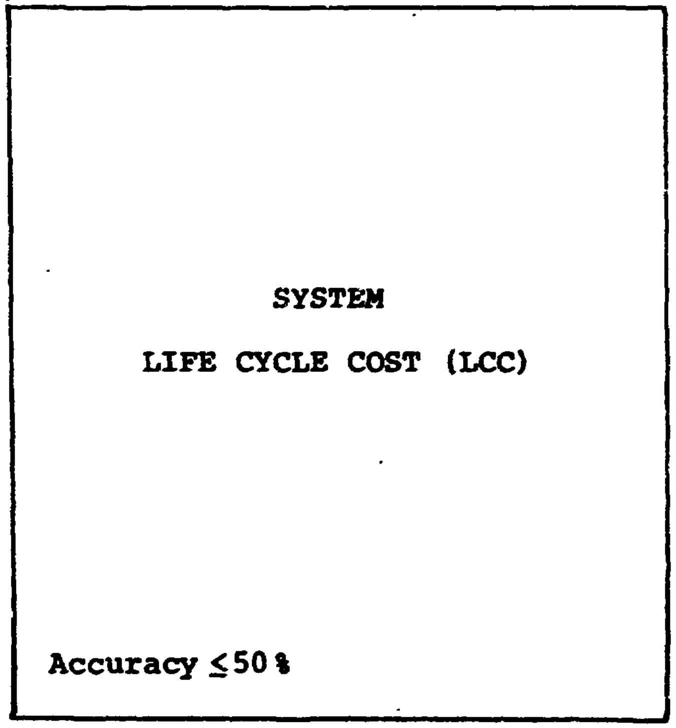
**INPUT DATA**

System Requirements
- performance
- testing

Support Philosophy
- maintenance
- manning

Technology
- hardware
- software

Historical System Data Base
- Similar Systems
- Similar Technologies
- Similar Methodologies



**OUTPUT DATA**

Long Range Budget Projections
- System R and D Prototypes
- Production Estimates
- Maintenance Estimates

Figure 4. SCE Parameters During Demonstration and Validation Phase

INPUT DATA

Government/Contractor

System Requirements
- Performance
- Testing
- IV&V

Documentation Requirements
- MIL-STD-
- Commercial

Schedule Requirements

Support Philosophy
- Government
- Contractor
- Manning

Cost Estimates
- System LCC
- Hardware
- Software
- Actual Vs Predicted

Technology
- Hardware
- Software

Historical Data
- Similar System
- Similar Technologies
- Similar Methodologies

SYSTEM

LIFE CYCLE COST

Accuracy $\leq 25\%$

OUTPUT DATA

Long Range Budget Projections
- System R and D - Prototypes
- Production Estimates
- Maintenance Estimates

Manage Support Activities
- Trade-Off Analysis
- Risk Analysis
- Resource Mix
- Impact Assessment
  - Schedule
  - Personnel

Development Cost Estimates
- Software
- Hardware
- Support
- Maintenance
- Documentation
- Facilities
- Manning

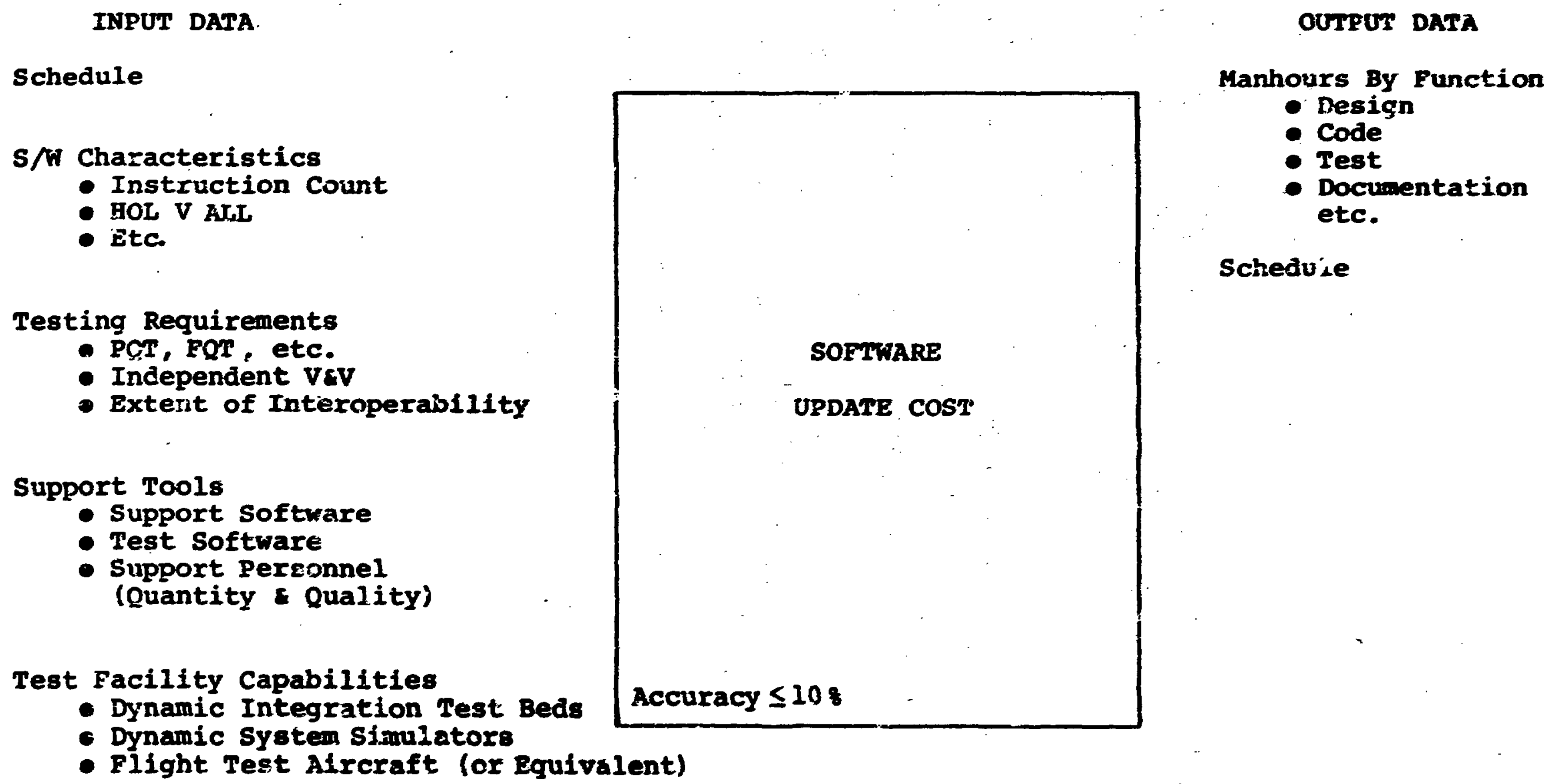
As soon as configuration updates are required to baselined software, there is a need for management to accurately estimate the cost and schedule associated with each particular update as well as the impact that the update may have on the software life cycle cost. The methodology that is needed to fulfill that requirement is shown in Figure 5. The methodology would have as input data uniquely associated with the software update in question. The specific characterics of the software would be quantifiable in detail, as would the testing requirements, the capabilities of the testing facilities and tools, as well as the qualifications of the personnel involved in the update process. As DoD systems become increasingly integrated with each other, the methodology must account for the influence of systems interoperability requirements as well as mandated schedules for the completion of the update. The output should be an update cost and a schedule which is apportioned into the functions or phases as shown. Separate local user adaptation of the methodology should be authorized to allow the system program management personnel to develop cost and schedule estimates for elements or subsets of the total system update for internal planning and management purposes.

Conclusion 13: Based upon an evaluation of the intended use of the SCE, and the available input data during the system life cycle, it is determined that three classes of software cost estimating methodology are needed. SCE methodology is needed during the Concept Exploration phase when input data are limited to historical life cycle data on comparable systems. The second level of SCE methodology is an enhancement/refinement of the first in that it incorporates estimates based upon specific design characteristics of the system as they evolve during the development process. The primary output is an estimated system/software life cycle cost. The last level of SCE methodology uses actual historical data on the specific system and is used to estimate cost and schedule associated with specific software updates. All of the software cost estimating methodologies should be usable at the location of the organization having management responsibility for the software.

## 4.2.2 DoD Goals

A major element of the DoD SCE goals is establishing a reasonable, representative, and standardized SCE methodology. This is not to say that DoD should select a specific model of the current state-of-the-art and declare it a standard. The Panel concluded that there are many dangers in doing this. The Panel believes that DoD would be better advised to specify the general procedure for estimating software costs (i.e., major activities model selection, model documentation, estimate documentation and management actions required to use the results of any software cost estimating effort). The establishment of this estimating methodology should be in concert with the data collection goals

Figure 5. SCE Parameters During Production and Development Phase



and should make use of the data collected to "fine-tune" current models and develop new models. The model/methodology development should possess the following attributes.

- Open discipline - A procedural envelope could be defined in which rationale for specific model selection and exercise is provided. This allows flexibility in estimating while providing a disciplined approach to the development of a software cost estimate.

- The use of multiple models - It should be pointed out that the term methodology is meant to imply the approach that is taken to employ specific SCE models. This methodology should allow for the employment of a number of models as required by life cycle phase or for application. This attribute will allow for a best-fit situation and yield the most accurate results.

- Reproducible - Given the same situation and the same data, two independent activities should be able to produce the same estimate of cost. This attribute provides for accuracy assessments and auditability in the software cost estimating exercise. In providing such an attribute, comparative trade-off studies are enhanced, source evaluation is expedited and continuity in program monitoring activities is achieved.

- Living Methodology - Whatever methodology is chosen, it must constantly be updated to reflect the current state of software technology. This attribute is achieved though institutionalizing methodology, and through DoD instructions, manuals, regulations and standards. These should be constantly updated to reflect the results of current research in the SCE activity.

Another DoD SCE goal should be the encouragement and support of software cost estimating research. A comprehensive DoD SCE technology development plan should be developed, identifying SCE research requirements into the future. The DOD centralized group should be charged with the management of this plan and be provided with sufficient resources to accomplish the objectives of the plan. Clearly the SCE technology research should address itself to near term, interim, and long term objectives, all of which support the data collection and model development activities.

There are certain general characteristics which SCE models should have in order to be useful and cost-effective in the variety of applications expected of them. Some of these are listed below. Special mention should be made of the general finding of surveys of present SCE models that they are sensitive to the development environment. It is clear that, for management of development programs, models that contain adjustable parameters to reflect different environments are appropriate, but there is a need for models that compute the "should-cost" amount for the best possible environment in order that the government shall have a standard against which to measure development environments of vendors.

Before a model is admitted to general use, there should be adequate testing against an appropriate set of different software projects in order to adequately describe its effective application domain and the expected accuracy of the model in that domain. Desirable characteristics include:

- Automated execution.
- Transportable for all commonly used computers. Written in HOL.
- Model algorithms should be thoroughly documented and available to all users.
- Outputs should be flexible and tailorable to the several applications.
- The total set of models should cover the whole software life cycle, although individual models may be specific to certain phases of the life cycle or certain applications of the SCE.
- Models should be able to deal effectively with missing data.
- Models should be conservative of use of resources for data loading and computer time.

A wide variety of applications of SCE models exist, and a detailed description of the output required for each application is impractical in this paper. Several different outputs which should be considered as candidates are given below. Further study should be undertaken to develop a more specific description. Each of the items listed below is an output of at least one SCE model which exists now demonstrating that the list is not unreasonable.

- Total manpower effort by phase and by effort type.
- Reasonable development time.
- Amount of documentation.
- Staffing profile.
- Computer costs.

- Cost-schedule trade-off factors.
- Sensitivity of output to input variations.
- Expected rate of software deficiency repor s.
- Milestone occurence times.
- Risk profiles.

SCE models must be designed so that the necessary input can be reasonably expected to be available when the model is applied. Most existing models require that the number of lines of code be input. Where models are to be used before coding, these models are not very helpful because a difficult manual estimation must be accomplished before applying the model. Models which use more readily available information are urgently needed. For example, models which use the size and complexity of a language for the system to estimate development effort would be very useful. It should be noted that some applications, e.g. maintenance resource requirements, may have completed code metrics available as input to a SCE model. Tools are needed to analyze code and determine useful metrics such as number of lines of code, complexity measures, number of operators, number of operands etc., which can be used to estimate maintenance costs or to measure software quality.

<u>Conclusion 14:</u> Model requirements should be developed in the areas of input and output parameters, and refined to correlate specific requirements with each anticipated area of model application. Research should include establishing accuracy requirements in each instance, and determining if the required accuracy is attainable.

## 4.3 Software Cost Estimation Metrics and Data Collection

In just about every piece of literature or special report on SCE, a common conclusion exists: the data or data base for SCE is in bad shape in that the data are non-homogenous. Therefore, it is difficult to achieve such desirable ends as model validation, financial performance, technical performance, data base development, and advanced SCE modeling. In addition, the ability to try other predictive metrics is missing, thus code size is usually the predominent predictor. Another point that is clear is that the clarity of underlying definitions and precision or integrity of the reported data is poor. Clearly, there isn't a shortage of data. However, for the above reasons it is generally unuseable. The Panel consensus is that data collection should be a major element in the DoD SCE goals. This data collection effort should be well defined, properly controlled, manageable and produce the highest quality data possible.

To establish data collection and methodology for DoD, an implementation approach must be established. This implementation approach will take different forms as the SCE technology improves. Some of the general requirements that should be present in any implementation approach are:

- Data collection and cost estimating methodology should be well defined and specified in appropriate regulatory form, i.e., in DoDs, MIL-STDs, DoD-STDs, DIDs, etc.

- A centralized group should be provided to control SCE methodology application and development.

- Weapon system acquisition Program Management Directions (PMDs) should specify the requirement for software data collection. In addition, the PMD should allow or provide for the required resource to do that data collection.

- Flexible accuracy standards should be developed so that the results of SCE activity can be properly evaluated in view of the current software LCC phase. For each phase of the software life cycle, a separate accuracy standard should be established with regard to the results of the software cost estimate for that phase. These accuracy standards should have an increasingly finer granularity as the program moves through its life cycle.

Software metrics have been produced that are used to measure various qualities of software. The quality of software referred to does not imply "good" or "bad" software. Instead, software qualities are those attributes of software by which we determine software reliability, portability, maintainability, etc. Work has been done to define the attributes which comprise the respective software qualities. A few automated tools have been developed to make the data collection task more precise, manageable, and less time consuming. It appears feasible to extend the software quality metrics role to include measures that will concentrate on attributes which are the "cost drivers" in system life cycle phases. In certain instances the attributes that make up a certain quality are directly applicable to the cost elements to be modeled. If a system is complex (new and/or difficult) then the cost associated with the system over its life cycle in all phases should be higher than those associated with less complex implementations. Algorithmic complexity, structural complexity, and performance requirements are just beginning to be understood and measured. Thus, these measures offer a way to quantify the cost aspects of complexity as an input to the modeling situation. Many other software metrics may serve as candidates to obtain cost parameters.

Once a suitable (beginning) set of software metrics for cost estimation is derived, data must be collected, stored in a centralized location, and analyses conducted that will result in the next generation of more precise, accurate, and viable software cost estimating methodologies. This gives rise to implications for both management and technology. A uniform (standarized) data collection instrument must be designed that will enable data collection in a consistent manner. This approach is mandatory to avoid problems arising over which data to collect, when to collect them, and how to maintain the data in machine readable format for subsequent storage and analysis.

A candidate Data Item Description (DID) has been prepared by the Air Force Systems Command Electronic Systems Division (AFSC/ESD) which specifies magnetic tape formats for reporting financial and technical data on system acquisition programs under 800 series regulations. The DID specifies requirements for a Software Cost Performance Report (S/W CPR) to be used in conjunction with a companion document for an expanded work breakdown structure (WBS). The S/W CPR is an automated system which produces magnetic tapes in accordance with the DID format instructions.

Management policy is needed to enforce the use of the data collection vehicle and sufficient resources allocated. A repository is needed to focus the data collection activities. The repository could also be chartered to perform the analytic functions on the resulting data base. Open access for technologists, researchers, management personnel should be guaranteed for industry and government alike. Data could be "sanitized" to remove the onus of competitive advantage and possible revelation of corporate cost estimating strategies. A very positive benefit is obtained by allowing a view across the data base and more quickly obtaining a critical mass of data for cost modeling and methodology development. One possible choice for such a repository is the Data and Analysis Center for Software (DACS) located at the Rome Air Development Center. DACS is presently responsible for collecting and disseminating software data and could be called upon to perform an expanded role for cost estimation. In Government fiscal year 1983, the DACS will be operated under the aegis of the Defense Technology Information Center system of Information Analysis Centers (IACs) for the Defense Logistics Agency.

The ultimate unit of software selected for measurement and data collection will vary depending on the life cycle phase of interest and the particular cost estimating methodology and models employed. Thus, the unit of software at any point in time should be considered as a variable. During the conceptual phase of the life cycle a less detailed view of software is needed than during the full scale development phase where functional allocation has been made to computer program configuration items (CPCI's) and lower. Models used during early phases from concept through preliminary design will utilize software units where scale factors are more important than implementation details. Later in the life cycle, cost models may be employed for cost performance and tracking. The granularity of software units and cost elements will correspond. Technology for metrics and data collection will need to account for the varying granularity of software components.

A work breakdown structure (WBS) to support metric and data collection is needed. The current version of MIL-STD 881A specifies a WBS for system acquisition with elements at the first, second, and third levels. Additional levels of detail in the WBS are needed if effective cost modeling is to be obtained. Software related WBS elements must account for operational computer software, computer programs that support the acquisition of operational software, and software that support the services, facilities, and data required to acquire the defense system. A draft military specification has been prepared by ESD which expands the current WBS to the fourth, fifth, and sixth levels. This WBS will be applied by ESD on a pilot basis in the near future and should be monitored for suitability and performance vis-a-vis cost estimation. Consideration should also be given to a companion document prepared for ESD which establishes reporting requirements needed to characterize computer software developed during system acquisition. This document should require that software data be collected on structure (i.e. functionality), schedules, factors indicating size, degree of difficulty, numbers of development personnel, development changes from initial plans, and deviations from accepted practices. These data could then be used to monitor software acquisition in conjunction with the reported software cost data. The cost data will explicitly identify cost problems, and the analysis of the other data should provide reasons for cost problems. In view of the poor performance of existing software cost estimation models and methodology, it is clear that a better understanding of the software life cycle is needed in order to establish new insights into the process for cost estimation purposes. Technology improvements are needed for cost modeling, but these improvements will not be realized unless there is a corresponding improvement in software metrics and data collection. These tasks are the foundation by which more precise, accurate, and timely cost estimation can be performed.

The above discussion supports the following conclusions:

Conclusion 15: Additional research in the area of software metrics is needed to define software attributes which are the cost drivers over the life cycle.

Conclusion 16: Data collection activities must be established using the metrics in an organized and standarized manner.

Conclusion 17: Data collection should be automated and outputs provided in machine readable format.

Conclusion 18: A central repository should be designated for storing and analyzing software cost estimation data.

Conclusion 19: MIL-STD-881A should be modified to permit a work breakdown structure that supports data collection and analysis.

## 4.4 Contractual Implications of Applying (SCE) to Contractor Performance

### 4.4.1 General Contractual Considerations

The Government uses various tools to ascertain the contractor's performance in fulfulling the terms of a contract. Information derived from these tools might or might not be combined with SCE methodology to derive an additional view of contract performance. Three areas have been identified where the SCE methodology may play a significant, contractual role. These areas are: technical performance measurement (TPM), progress payments, and fee determination.

In general, potential contractual problems could exist if specific SCE methods were imposed by the Government. If the Government imposed a type or types of models for use by contractors, there would be immediate problems with validating or calibrating the methodology for the contractor's environment. This would lead to misinterpretations, inaccuracies, and potential legal problems when, or if, the SCE methodology would be applied to a performance measurement situation. Hence, it would be in the mutual interest of the Government and industry to constrain contractual application of SCE methodologies to a set of narrow guidelines.

If the Government desires the use of SCE methodology, then the contract should only specify that the contractor will define the particular SCE methodology, that the particular SCE methodology will be analytical in structure, and that both parties will mutually agree to the minimum input and output parameters. Provisions should also be made to assure that the contractor's SCE methodology could sustain modest refinement (i.e, change to factors or structure), but prohibit major or radical method changes without a two party agreement. The use of the SCE methodology by the contractor should be oriented more as a tool for the contractor to communicate to his managers and developers in addition to the Government. Thus, the SCE methodology would best function if integrated into the contractor's management system.

The first area where SCE methodologies could be applied to the performance measurement process is the technical performance measurement program, as defined in MIL-STD-499A. Software cost estimation methodologies would be a reasonable adjunct to the evaluation of technical parameters, particularly when risk assessment (costs and schedule) is needed. The addition of a SCE methodology would provide a consistent, timely and cost-effective means of including management factors with technical performance factors. It would also assure a quantitative feedback so that reasonable priorities and tasks could be formed to address identified problem areas. The SCE methodology would aid in communicating to all parties the scope of a problem.

> Conclusion 20: The Government needs to integrate the SCE methodology requirement into its technical performance measurement system. Since the application of the SCE methodology is strictly informational, no contractual problems would result. The contractor, however, must be given the latitude to select, modify, or develop the evaluation methodology that is best suited for the particular procurement environment.

Progress payments represent another area where SCE methodologies could have a significant contractual impact. A software model (or models) could be used to compare predictions with actual cost/schedule performance independently of any cost tracking system (e.g., C/SCSC or C/SSR). The results of such a comparison could give management the basis to reevaluate contract performance. However it appears inappropriate for the Government to use such results in a progress payment determination. SCE methods presently have not demonstrated sufficient accuracy to serve as criteria for payment or nonpayment. Also, cost estimates tend to be real-time and do not reflect established baselines. It is more appropriate, then, to expect the software cost estimation techniques to be predictive of future performance and not applicable to evaluating past performance.

> Conclusion 21: The SCE methodologies should not be applied to [illegible]ogress payment determinations. Such methodologies are relatively inaccurate and only predictive in nature.

In establishing a fee payment structure, SCE methodologies could possibly be applied. For instance, the requirement for stating and meeting cost and schedule objectives could be tied to a fee payment. This would give an incentive for the contractor to avoid misleading or misrepresenting management and technical factors that make up a forecast. However, as was the case for progress payments, one recognizes the inaccuracies associated with pres[illegible]t d[illegible] [illegible]CE methodologies are only predictive in nature. Hence, [illegible] woul[illegible] [illegible]em inappropriate for such methods to be applied or connected with fee determinations. The present, appropriate approach is to use metric methodologies which rely on measures of past events.

> Conclusion [illegible] SCE methodology should not be employed in the fee [illegible]rmination process as it is predictive, and cannot adequately support evaluations of predetermined baselines and criteria.

### 4.4.2 Program Management Considerations

An accurate picture of projected costs and schedules is essential for effective program management. Every program manager would prefer to have projections that are based on the best and most current data available. Such information also can serve as a valuable contractor tool for assessing the need for reallocation of resources. The desire for the best available information must be balanced against the cost to obtain such information. It is apparent that the need for updated cost estimates will vary with the size of the software development effort and the level of risk associated with the development. It is unreasonable to require a small, straight forward project to bear the cost burden of extensive data collection and SCE computation. However, every program involving software development should gather software cost estimating data. Data should be collected at the end of each phase of the development life cycle.

A definite requirement exists for regularly scheduled updates to cost and schedule estimates. The development contractor should be obligated to provide the necessary data, and funding should be provided for this purpose. Since critical program decisions often are associated with results from formal program reviews, these program review points constitute logical points for Government review and recalculation of SCEs.

For programs that can be identified as involving a high degree of risk, due either to technical complexity or uncertainty in original costs, contractor recalculation of cost/schedule estimates should be required in conjunction with each formal program review. This additional point of comparison should assist the Government in the monitoring of high risk efforts.

In addition to the scheduled recalculation of SCEs in conjunction with formal program reviews, recalculations should be required whenever major changes or deviations occur in a program. Examples of such changes include modifications to the program scope, budget reductions or changes in the development environment. When such changes are proposed by the contractor, change proposals should be accompanied by cost/schedule recalculation using the contractor's SCE methodology. In all cases, the contractor should be obligated to provide necessary input data and the Government should recalculate costs/schedules.

The balance between the need for current estimates and the cost to obtain such estimates will change as collection methods become more efficient and, thus, less costly. The best opportunity to reduce collection costs involves increased use of automation in the collection effort.

<u>Conclusion 23:</u> SCE data should be collected at each formal program review point. These data can be gathered by

automated or other "no cost" means. Successful application depends on cost efficient (automated) collection methods.

## 4.5 Software Cost Estimation Technology Development Plan

The first step in establishing an effective DoD technology plan is to clearly state DoD goals. A DoD standard and guidebook is in order to insure a standard methodology for collecting and categorizing data for the system and software data base. The DoD standard would also insure commonality for the data base item descriptions and terminology. The DoD standdard would be used by both Government and industry to insure proper identification of data for the data base by system type, development methodology, complexity , schedule duration, etc. It is also anticipated that a guidebook will be required to describe how to access and use the data base and how to utilize the cost models developed from the data base. The guidebook should contain procedures for using the various models of software costs and should indicate how these models may be tailored for a particular application.

The embracing of SCE methodologies will be an evolutionary process. Policy should be in conformance with and serving to implement an overall SCE technology plan. This policy should be initiated by JLC with a policy statement that addresses the following:

- Intent and goals with respect to SCE technology
- RFP consideration
- Source selection
- Data acquisition
- Contractor performance evaluation
- Use by life cycle phase
- Model technology development
- Regulatory policy requirements from DoD
- Protection of proprietary information
- Prime/subcontractor interrelationship
- Scope of programs affected

As the use of SCE methodology matures, additional guidance can be prepared and disseminated. This could be in the form of a DoD directive, guide books, central data repositories, etc.

The Panel's assessment of current SCE technology reveals an absolute need for improved methodology to accurately predict the software resources required over the complete software life cycle. As advances in software engineering are made, these advantages should be reflected and incorporated in the improved SCE methodologies. The SCE technology also should be useful to all of the large numbers of potential users, each of whom has a diversity of application for this technology. An undertaking to collect statistically valid historical data is the next logical step in developing better SCE models. This effort should begin as soon as possible and should be accomplished as a requirement of all software related contracts and activities. It is desirable that a central DoD agency be identified to provide the proper incentive, direction, and control for this effort.

A number of activities are proposed as a plan to achieve both an interim capability and the long term development of a complete SCE methodology. A phased development plan is considered appropriate to achieve near-term and long term goals.

In the near term (next three years), DoD should concentrate on implementing procedures to best use SCE methodology as it now exists, and establish a data base suitable to develop advanced SCE methodology. Factors to be reviewed include:

- A guide to applying existing SCE methodology.

- A DoD policy for use of SCE methodology.

- Establishment of a DoD control center for technology and ssociated working groups to encourage research in SCE technology in the form of alternate metrics and advanced simulation models.

- Definition of the metrics and data required for the collection procedure that has provisions for changing collection requirements.

- Implementation of data collection procedures that have provisions for changing collection requirements.

- Reevaluation of the goals and objectives for SCE technology.

On a broader front, DoD should concentrate on gaining an accepted, standard SCE methodology for general application. This could be achieved in three to seven years. Goals at this level should include:

- Development of improved methodology for software resource estimates (e.g. develop SCE model inputs for program size estimation based program requirements, languages, or program design languages).

- Validation of existing models based on the data base acquired in the near term.

- Establishment of prototypes of advanced SCE methodology.

- Implementation of improved SCE methodology to gap the existing versus the complete methodology.

- Establishment of guidelines/procedures for the application of the complete SCE methodology.

- Development of complete SCE methodology.

Conclusion 24: The JLC should issue a policy that implements an SCE direction. This policy should direct that SCE methodology be adopted that makes existing SCE technology usable by program managers. It should also provide for a technology upgrade to SCE methodology over the next seven years.

## 5. RECOMMENDATIONS

The Panel has developed four recommendations for the Joint Logistics Commanders as a means to best utilize existing software cost estimating technology in improving software cost estimates, and a program to follow to gain better and more accurate cost estimates in the future.

### Recommendation 1: Use of Existing SCE Models

The Panel believes that no existing SCE model is sufficient to adapt as an embedded computer system standard. The Panel recommends that the JLC not adapt any existing SCE model as a standard. (Refer to conclusions 1, 2, 3, 4, 5, 6, 8, and 9).

### Recommendation 2: Application of Existing SCE Methodology

The Panel believes that a judicious use of SCE models and methodology can improve the acquisition and management of software. There is no accepted methodology or guide to using current SCE models or methodology in a generally acceptable manner. The Panel recommends that a guidebook be developed that can be used by program offices to orderly qualify models and methodologies to develop better software cost estimates throughout the entire software life cycle. (Refer to conclusions 7, 13, 19, 20, 21, 22, 23, and 24).

### Recommendation 3: Improving SCE Methodology

The Panel believes that an SCE methodology can be developed that would result in SCE models that could be generally applied by both the Government and industry to estimate embedded computer system software costs well. The Panel recommends that JLC sponsor a program to implement an improved SCE methodology. (Refer to conclusions 10, 11, 12, 13, 14, 15, and 24).

Recommendation 4: Establishing an SCE Data Base

The Panel believes that an improved software cost estimating methodology must be supported by the gathering and maintenance of an accurate, complete, and coherent data base. The Panel recommends that JLC appoint an existing Government agency as an SCE data base repository and empower this agency to develop data collection standards. (Refer to conclusions 15, 16, 17, 18, 19, and 23).

## APPENDIX A - PARTICIPANTS

ESTIMATING SOFTWARE COSTS

PANEL D

CHAIRPERSON:

Mr. R. Dean Hartwick
Intercon Systems Corporation
11306 East 183rd Street
Cerritos, California 90701
(213) 924-7770

CO-CHAIRPERSON:

Mr. Robert E. Berri
Aerospace Corporation
6216 Mosley Avenue
Los Angeles, California 90056
(213) 648-7477

ARMY MEMBERS

Mr. David Usechak
PMO SOTAS
Fort Monmouth, New Jersey 07703
(201) 544-5171
AV 996-5171

NAVY MEMBERS

Mr. George W. Robertson
FCDSSA, San Diego
200 Catalina Blvd
Attn: Code 82F
San Diego, California 92147

Mr. Clell W. Gladson
Naval Oceans Systems Center (NOSC)
Attn: Code 9133
San Diego, California 92152
(714) 255-7615
AV 933-7615

Mr. Ronald B. Leask
Naval Underwater Systems Center
Attn: Code 3251
Fort Trumbull
New London, Connecticut 06320
(203) 447-4366
AV 636-4366

AIR FORCE MEMBERS

Major Joseph A. Duquette
HQ ESD/ACCI
Hanscom AFB, Massachusetts 01731
(617) 861-3844
AV 478-3844

Major George W. Trever
AFCMD/KRR
Kirtland AFB, New Mexico 87117
(505) 844-0859
AV 244-0859

Mr. David V. Thornell
OO-ALC/MME
Hill AFB, Ogden, Utah 84406
(801) 777-7355
AV 458-7355

Mr. Robert M. Earnest, Jr.
AFALD/PTEC
Wright-Patterson AFB, Ohio 45433
(513) 255-4991
AV 785-4991

## ESTIMATING SOFTWARE COSTS

## PANEL D - (CONT'D)

Mr. Dean Bergstrom
RADC/ISIE
Rome Air Development Center
Griffiss AFB, New York 13441
(315) 330-3827
AV 587-3827

## PANEL INVITEES (NON-DOD)

Dr. Roger R. Bate
Texas Instruments, Incorporated
P.O. Box 405
Mial Stop 3407
Lewisville, Texas 75067
(214) 462-4790

Mr. Richard L. Page
SAI Corporation
P.O. Box A-81126
2801 Camino Del Rio, South
San Diego, Califonria 92108
(714) 293-7500/225-2043
AV 933-2043

Mr. James A McCall
General Electric Space Division
1277 Orleans Drive
Sunnyvale, California 94086
(408) 734-4980

Mr. Newman E. Thompson
System Development Corporation
Monmouth Mall
Route 35
Eatontown, New Jersey 07724
(201) 542-8305

Mr. Richard J. Latshaw
Teledyne-Brown Engineering
11300 Rockville Pike
Rockville, Maryland 20852
(301) 881-2090

Ms Marilyn Fujii
Logicon, Inc.
P.O. Box 471
San Pedro, California 90733
(213) 831-0611

Mr. Herman F. W. Oberkrom
System Group of TRW
1344 Woodman Drive
Dayton, Ohio 45433
(513) 254-8664

Mr. Robert H. Paulsen
Hughes Aircraft Corporation
P.O. Box 3310, Bldg. 618
Mail Stop E215
Fullerton, California 92364
(714) 732-2947

## APPENDIX B - BIBLOGRAPHY

## BIBLIOGRAPHY

The following documents were used by the panel:

1. Aron, J., "Estimating Resources For Large Programming Systems", NATO Conference on Software Engineering Techniques, Mason Charter, NY, 1969.
2. "Avionics Software Support Cost Model Statement of Work", Avionics Laboratory, Wright-Patterson AFB, 1980.
3. Basili, V., and Zelkowitz, M., "Analyzing Medium Scale Software Developments," Third International Conference On Software Engineering, Atlanta, GA, May 1978.
4. Box, G., and Pallensen, L., "Software Budgeting Model", Mathematics Research Center, University of Wisconsin, Madison, 1977.
5. Byrne, W. E., "Draft Military Specification: Software Prepared for ESD, MITRE Corp, Bedford, MA, February 1981.
6. Byrne, W. E., "Draft Military Specification: Software Acquisition Explanatory Data for ESD, MITRE Corp, Bedford, MA, February 1981.
7. Daly, E., "Management of Software Development", IEEE Transactions of Software Engineering, May 1977.
8. Doty Associates, Inc., "Software Cost Estimates Study", Vol. 1, RADC RT 77-220, June 1977.
9. Dumas, R. L., "Draft S/W Cost Performance Report (S/W CPR) Data Item Description (DID)", Prepared for ESD, MITR Corp, Bedford, MA, February 1981.
10. Duquette, J., "ESD-Standard Parametric Software Dost Estimating Model (SPSCEM)-Version 0 User's Guide", ESD, HAFB, MA, March 1981.
11. Duquette, J., "Software Cost Estimating Workshop Notes", ESD, HAFB, MA, March 1981.
12. Gilb, T., "Software Metrics", Cambridge, Winthrop Publishers, 1977.
13. Halstead, M., "Elements of Software Science", NY Elsevier, North Holland, 1977.
14. James, Thomas G. Jr., "Software Cost Estimating Methodology", AFAL-TR-77-66, WPAFB, OH August 1977.
15. Jones, V. E., et al, "Final Report of the Software Acqusition and Development Working Group", Prepared for CSD-C3-I, Washington, D.C., July 1980.
16. Junk, W. S., et al, "Survey of Software Cost Estimating Techniques", GE Info. System Program, Sunnyvale, CA, May 1978.
17. Lasher, William, "Software Cost Evaluation and Estimation: A Government Source Selection Case Study", 1979. Prepared for Air Force Space and Missile Systems Organization. Includes two supporting documents: (a) "Instructions for Completing Cost Proposal Supporting Data" and (b) "Software Cost Estimation and Evaluation Handbook."
18. Norden, P., "Usefull Tools For Project Management", M. K. Starr (ed) Penguin Books, Inc., Baltimore, MD, 1980.

## BIBLIOGRAPHY (CONT.)

19. Putnam, L., "A general Empirical Solution To The Macro Software Sizing and Estimating Problem", IEEE Transactions On Software Engineering Vol. 4, 1978.
20. Putnam., L., and Wolverton, R., "Quantitive Management: Software Cost Estimating", IEEE, NY, 1977.
21. Putnam, L. H., "Example of an Early Sizing Cost and Schedule Estimate for a Hypothetical Application Software System", ISP, GE Co.
22. Putnam, L. H., "Measurement Data to Support Sizing Estimating and Control of the Software Life Cycle", IEEE Compcon '78, 1978.
23. Putnam, L. H., and Wolverton, R. W., Tutorial on Quantitative Management: Software Cost Estimating, IEEE Catalog, NO EHO 129-7, 1977.
24. "Predictive Software Cost Model Study", the Hughes Aircraft Company Support Systems, Canoga Park, CA, 1980.
25. Rubey, R., "Quantitive Aspects of Software Validation", IEEE Transactions On Software Engineering Vol. 1, No. 2, 1975.
26. Schneider, V., "Prediction of Software Effort and Project duration: Four New Formulas", SIGPLAN Notices 13, Number 6, 1978.
27. Shapiro, O., "Final Report: FY80 Software Acquisition Value Estimator (SAVE) Task", Prepared for ESD, MITRE Corp, Bedfored, MA, December 1980.
28. Spare Systems Cost Analysis Group, Standardization Subgroup, "Standard Work Breakdown Structure for Space Systems", no date.
29. Stephenson, W., "An Analysis of the Resources used in the SAFEGUARD Software Development", Proceedings, 2nd International Conference on Software Engineering, 1976, p. 312-321.
30. Thibodeau, R., "An Evaluation of Software Cost Estimating Models", Unpublished report for Rome Air Development Center, February 1981.
31. Walston, C., and Felix, C., "A Method of Programming Measurement and Estimation", IBM Systems Journal Vol. 16, No. 1, 1977.
32. Wolverton, R., "The Cost of Developing Large Scale Software", IEEE Transactions on Computers Vol. 23, No. 6, 1974.
33. Zelkowitz, M., Shaw, A., and Gannon, J., "Principles of Software Engineering and Design", Englewood Cliffs, NJ, Prentice Hall, 1979.

## APPENDIX C - LETTER TO PANEL

INTERCON

**INTERCON SYSTEMS CORPORATION**

14 May 1981

To: Members of the JLC Panel on Estimating Software Costs

Thank you for agreeing to serve on the Estimating Software Costs Panel of the Joint Logistics Commander Joint Policy Coordinating Group on Computer Resource Management (JPCG-CRM). Bob Berri and I look forward to participating with you. As you all know, the workshop will take place in Monterey, from 22 June through 25 June. The tentative agenda shows registration on 22 June between 1300 and 1500, followed by a general work-shop session until 1715. At that time, we will meet as a panel for a get acquainted and a get organized session. Thereafter, we will mostly meet as a panel until presenting our results at a general work-shop session at 1500, 25 June.

Attached is a copy of suggestions for the panel that was given to us by JPCG-CRM, which effectively serves as a charter for our panel. Our expected results might very well be broken down to another level of detail, as follows:

1. Evaluation of pertinence of existing models.
2. Evaluation of limitation of existing models.
3. Determination of desired estimation model output by software life cycle phase.
4. Determination of minimum parameters to serve as input for estimation models.
5. Determination of acceptable interfaces for exchanging information between DOD/industry for bidding, contracting, and monitoring purposes.
6. Recommendation of R & D direction to improve technology.

Most of you will be somewhat familiar with the various estimation models now published. A bibliography is attached for those of you who wish to do some homework before the work-shop. In addition to

CORPORATE OFFICES
11306 EAST 183RD. STREET
CERRITOS, CALIFORNIA 90701
(213) 924-7770

INTERCON

this, we solicit all of you to bring any of your own estimating procedures, present or future needs, results (good and bad) from past efforts, thoughts for borrowing from other technologies, et cetera to share with the panel.

Our panel will have approximately 18 people on it. (Attached is a list of the members and their affiliations for your interest). As a result, we will likely want to split our effort into three or four narrower issues, with subpanels addressing themselves to these issues. Each of you might give this same thought, and we'll try to decide on the subpanels at our get organized meeting. Some suggestions for subpanel topics that have been advanced are:

- Evaluation of existing model strengths and limitations.
- Recommendations for R & D technology directions.
- Definition of use of estimation model in software life cycle activities.
- Implications of estimation model use in contractual relations.

Our panel has been assigned a topic that has tradionally received a lot of attention, occupied many pages of literature, and has generally fallen into the category of black magic. It will be a real challenge to see if our three days of effort can advance the state of art. Bob and I believe that we can, given the collective talent and experience that is being brought together. We look forward to interesting sessions with you all in Monterey. If you have any questions, please call either Bob or me, or the workship coordinator, 2Lt. Ed Petersime, Hq. AFLC/LOEC at (513) 257-2054 or AV 787-2054.

Very truly yours,

R. Dean Hartwick

RDH:lb

| | | |
|---|---|---|
| cc: | Ms. Antonia Schuman | Maj. Larry Fry |
| | Mr. Dick Maher | CDR Ronald Ohlander |
| | Mr. Robert Dunn | LCDR John Barnes |
| | Mr. Robert Berri | Dr. Matt Fisher |
| | Mr. Jack Munson | LTC Casper Klucas |

## PANEL D

## SUGGESTIONS FOR A MONTEREY PANEL ON ESTIMATING SOFTWARE COSTS

1. Issue: Given the increasing percentage of total system cost that must be devoted to software, it is becoming more critical that viable procedures be developed for accurately predicting the cost of software. Is this possible using the Software Life Cycle Cost Models currently available?

2. Questions:

-- How and where are current Software Life Cycle Cost Models of benefit?

-- For which phases of system development can Software costs be accurately estimated? (Why?)

-- For which phases of system development can Software costs not be accurately estimated? (Why?)

-- Is the use of program size or lines of code an accurate guage for estimating costs?

3. Expected Results:

-- An evaluation of Software Life Cycle Cost Models identifying deficiencies or strengths of each.

-- Recommendation on how to improve Software cost estimating.

## Estimating Software Costs

### Panel D

**Chairman:** MR. DEAN HARTWICK
INTERCON SYSTEMS CORPORATION
11306 E. 183rd Street
Cerritos, California 90701
(213) 924-7770

**Co-Chairman:** MR. ROBERT BERRI
Aeorspace Corporation
A-1/4022
P.O. Box 92957
Los Angeles, California 90009
(213) 643-1966

### Army Members

DR. GERALD ANDERSON
Commander
USA ERADCOM
Attn: DRDEL-CT-A
2800 Powder Mill Road
Adelphi, Maryland 20783
(202) 394-3585
AV 290-3585

MR. DAVID USECHAK
PMO SOTAS
Fort Monmouth, New Jersey 07703
(201) 544-5171

### Navy Members

MR. GEORGE ROBERTSON
FCDSSA, San Diego
Attn: Code 82F
San Diego, California 92147
(714) 225-2622
AV 933-2622

MR. RON LEASK
NUSC
Attn: Code 3251
Fort Trumbull
New London, Connecticut 06320

MR. CLELL W. GLADSON
NOSC
Attn: Code 9133
San Diego, California 92152
(714) 225-7615

## Air Force Members

CAPT JOE DUQUETTE
ESD/ACC
Hanscom AFB, Massachusetts 01731

MR. DAVID THRONELL
OO-ALC/MME
Hill AFB, Utah 84406
AV 458-7355

MR. STAN BROWN
AFCMD/KR
Kirtland AFB, New Mexico 87117
AV 244-3614

MR. ROBERT EARNEST
AFALD/PTEC
Wright-Patterson AFB, Ohio 45433
AV 785-4991

## Panel Invitees (Non-DOD)

DR. ROGER R. BATE
Texas Instrument, Incorporated
P.O. Box 405
Mail Stop 3407
Lewisville, Texas 75067
(214) 462-4790

MR. H. F. OBERKROM
System Group of TRW
1344 Woodman Drive
Dayton, Ohio 45432
(513) 254-8664

MR. DICK PAGE
SAI Corporation
P.O. Box A-81126
2801 Camino Del Rio
San Diego, California 92138

MR. JIM MCCALL
GE Space Division
1277 Orleans Drive
Sunnyvale, California 94086
(408) 734-4980

MR. RICHARD LATSHAW
Teledyne-Brown Engineering
11300 Rockville Pike
Rockville, Maryland 20852

MR. N. THOMPSON
System Development Corporation
Monmouth Mall
Route 35
Eatontown, New Jersey 07724
(201) 542-8305

MS. MARILYN FUJII
Logicon Incorporated
P.O. Box 471
San Pedro, California 90733
(213) 831-0611

## Bibliography

1. Aron, J., "Estimating resources for large programming systems," NATO Conference on Software Engineering Techniques, Mason Charter, NY, 1969.
2. Basili, V., and Zelkowitz, M., "Analyzing medium scale software developments," Third International Conference on Software Engineering, Atlantic, GA, May, 1978.
3. Box, G., and Pallensen, L., "Software Budgeting Model," Mathematics Research Center, University of Wisconsin, Madison, 1977.
4. Daly, E., "Management of Software Development," IEEE Transactions on Software Engineering, May, 1977.
5. Doty Associates, Inc., "Software cost estimates study," Vol.1, RADC RT 77-220, June, 1977.
6. Gilb, T., "Software Metrics," Cambridge, Winthrop Publishers, 1977.
7. Halstead, M., "Elements of Software Science," NY Elsevier, North Holland, 1977.
8. Norden, P., "Useful Tools For Project Management," M. K. Starr (Ed) Penguin Books, Inc., Baltimore, MD, 1970.
9. Putnam, L., "A general empirical solution to the macro software sizing and estimating problem," IEEE Transactions on Software Engineering Vol. 4, 1978.
10. Putnam, L., and Wolverton, R., "Quantitative Management: Software Cost Estimating," IEEE, NY, 1977.
11. Rubey, R., "Quantitative aspects of software validation," IEEE Transactions on Software Engineering Vol. 1, No. 2, 1975.
12. Schneider, V., "Prediction of software effort and project duration: four new formulas," SIGPLAN Notices 13, Number 6, 1978.
13. Stephenson, W., "An Analysis of the Resources used in the SAFEGUARD Software Development," Proceedings, 2nd International Conference on Software Engineering, 1976, p. 312-321.
14. Walston, C., and Felix, C., "A method of programming measurement and estimation," IBM Systems Journal Vol. 16, No. 1, 1977.
15. Wolverton, R., "The cost of developing large scale software," IEEE Transactions on Computers Vol. 23, No. 6, 1974.
16. Zelkowitz, M., Shaw, A., and Gannon, J., "Principles of Software Engineering and Design," Englewood Cliffs, NJ, Prentice Hall, 1979.

## APPENDIX D - PANEL PRESENTATIONS

PRESENTATION MADE BY

ROBERT BERRI
AEROSPACE CORPORATION

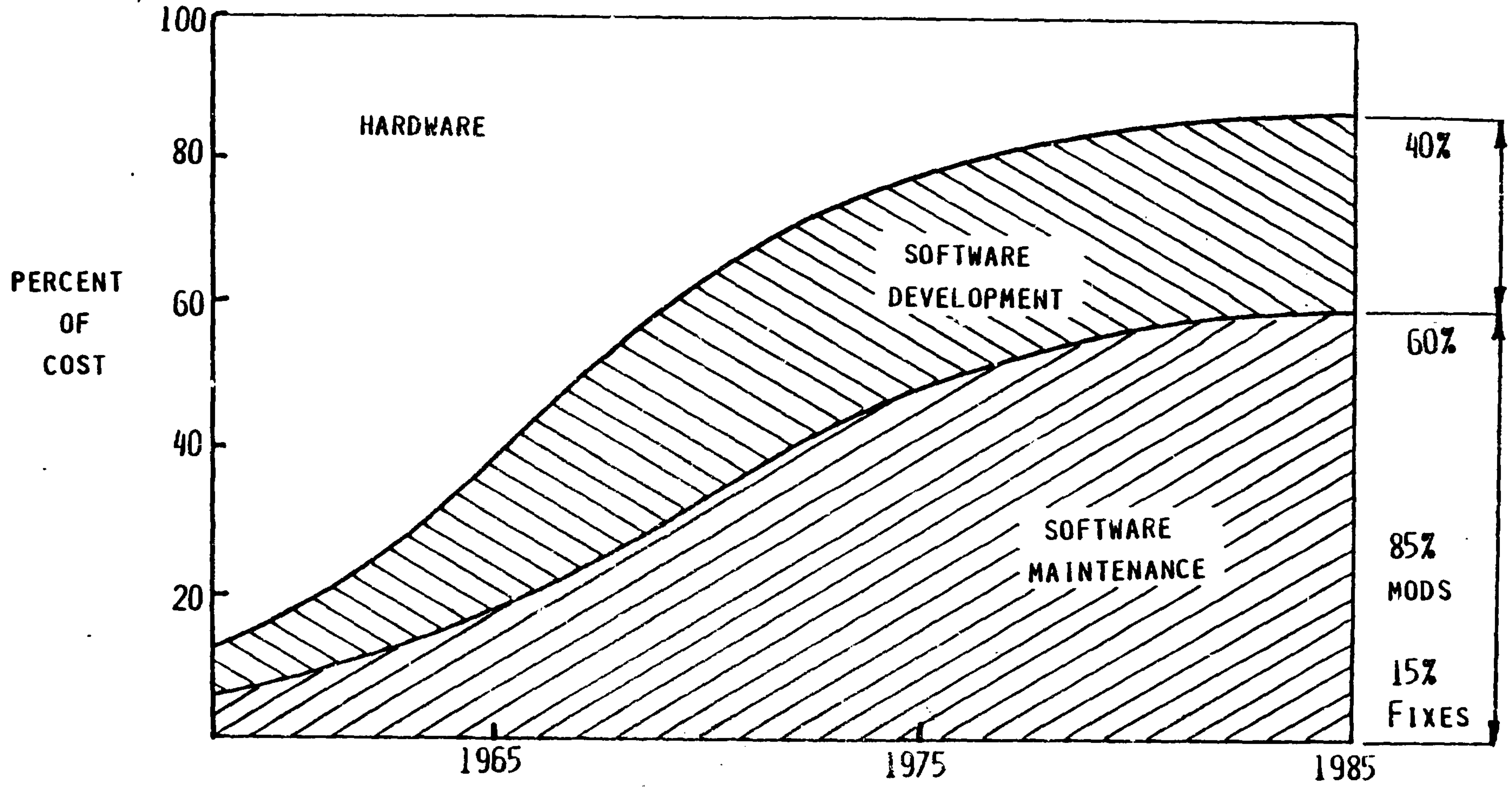
TO

PANEL ON ESTIMATING SOFTWARE COSTS

## QUESTIONS FOR PANEL D

- ARE GURUS ALWAYS RIGHT?
- IS "WHEN" AS IMPORTANT AS "HOW MUCH"?
- HOW MANY COST MODELS IS ENOUGH?
- ARE RULES OF THUMB DANGEROUS?
- CAN YOU ESTIMATE WITHOUT A DEVELOPMENT MODEL?
- "WHAT DO YOU MEAN BY THAT"?
- IS MIL-STD-881 ROTTEN?
- IS FIRMWARE SOFTWARE?
- ARE WE "EMBEDDED"?
- WHAT IS SOFTWARE "MAINTENANCE"?
- WHAT DOCUMENTS ARE RIGHT?

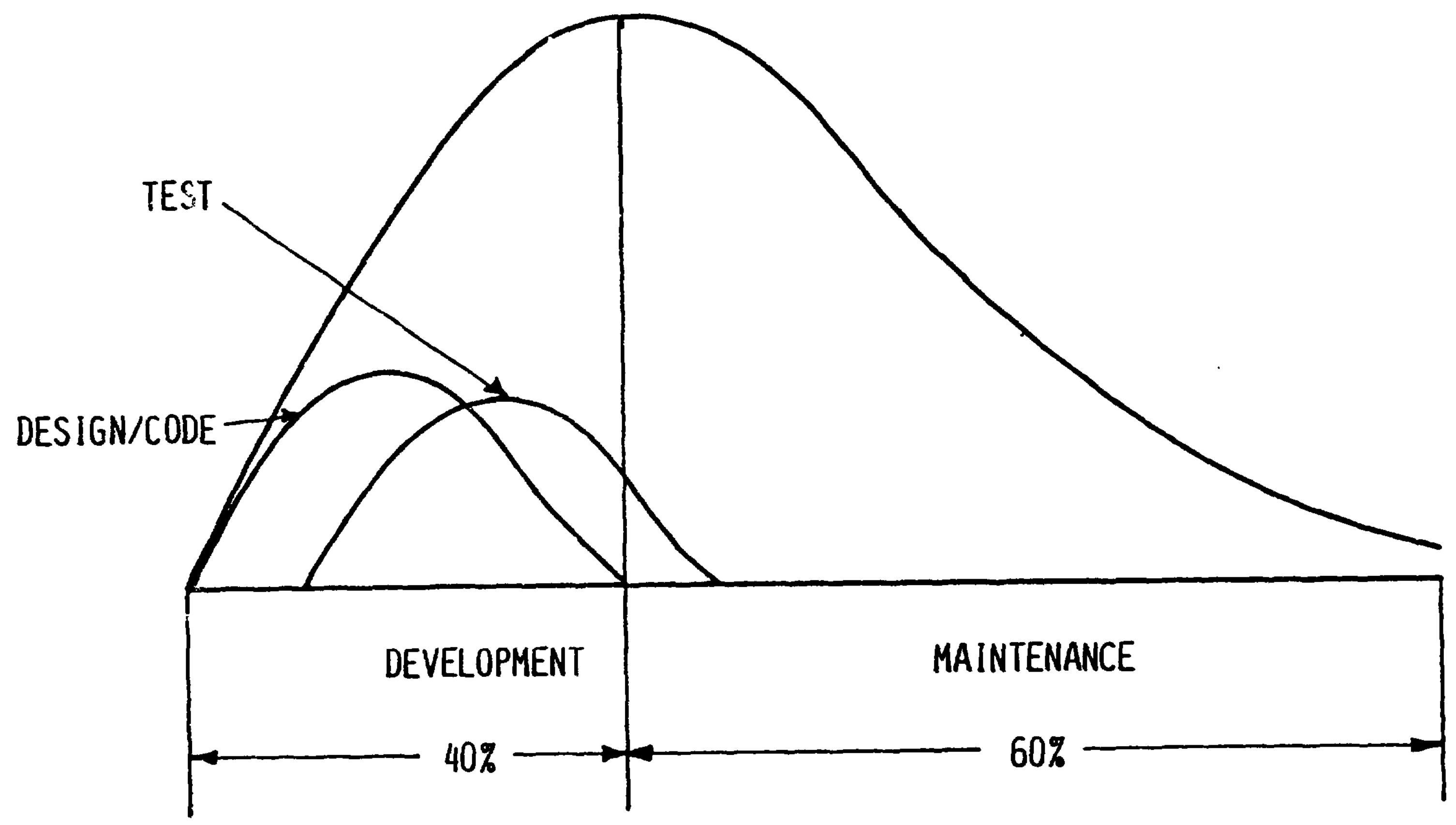
EMBEDDED COMPUTER SYSTEMS COST TREND



NOTES: (1) DATA SYSTEM ONLY
(2) EXAMPLES OF FREE HARDWARE EXIST

COST DISTRIBUTION

PUTNAM INDICATES A NORDEN-RAYLEIGH



## SOFTWARE COSTING MODELS

IBM/WALSTON AND FELIX

JENSEN

PRICE-S (RCA)

RADC/DOTY ASSOCIATES (OLD DATA, OLD METHODOLOGY)

SPACE DIVISION

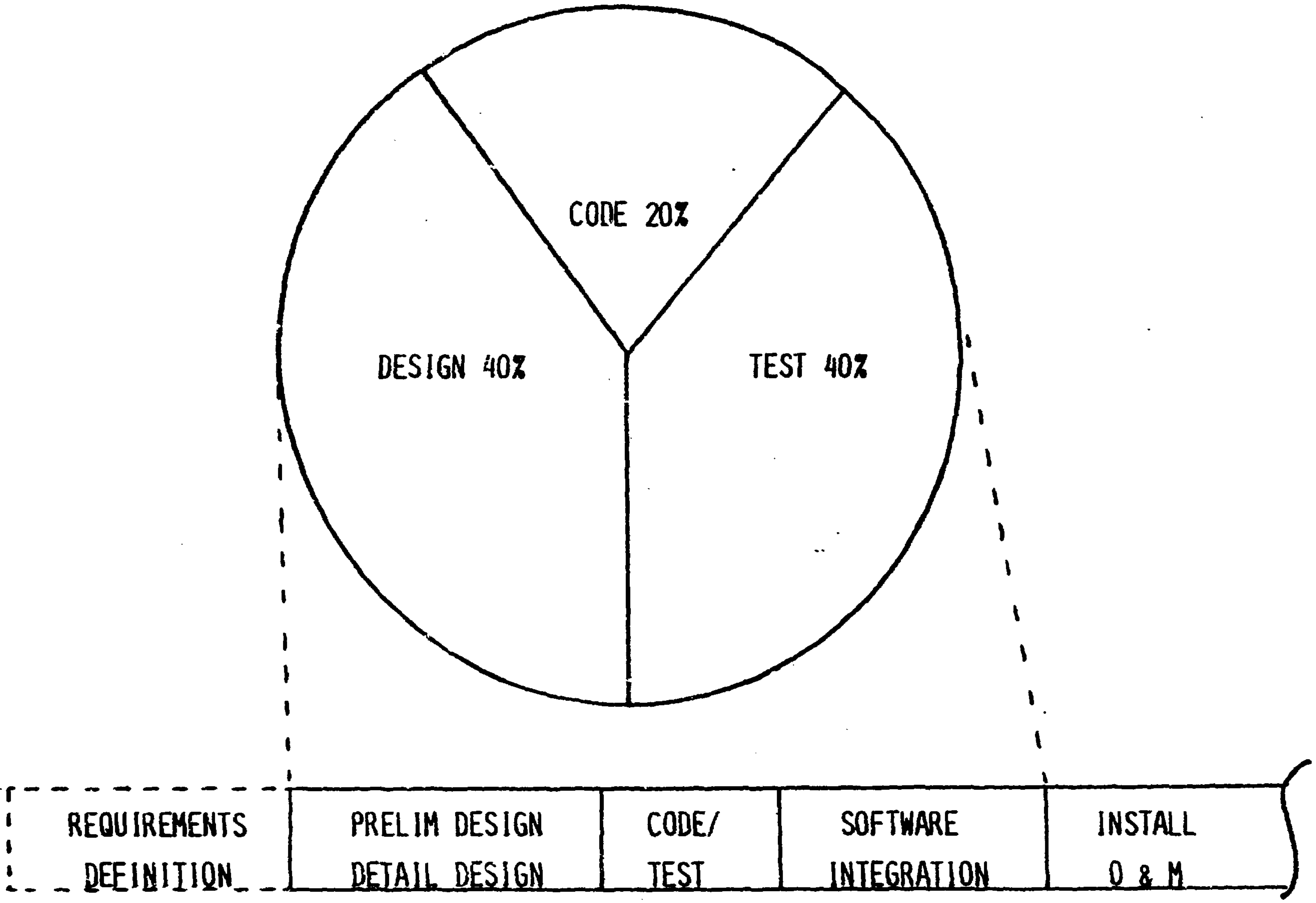
SLIM/L PUTNAM

SOFTWARE SCIENCE/HALSTEAD AND SCHNEIDER

TRW/WOLVERTON

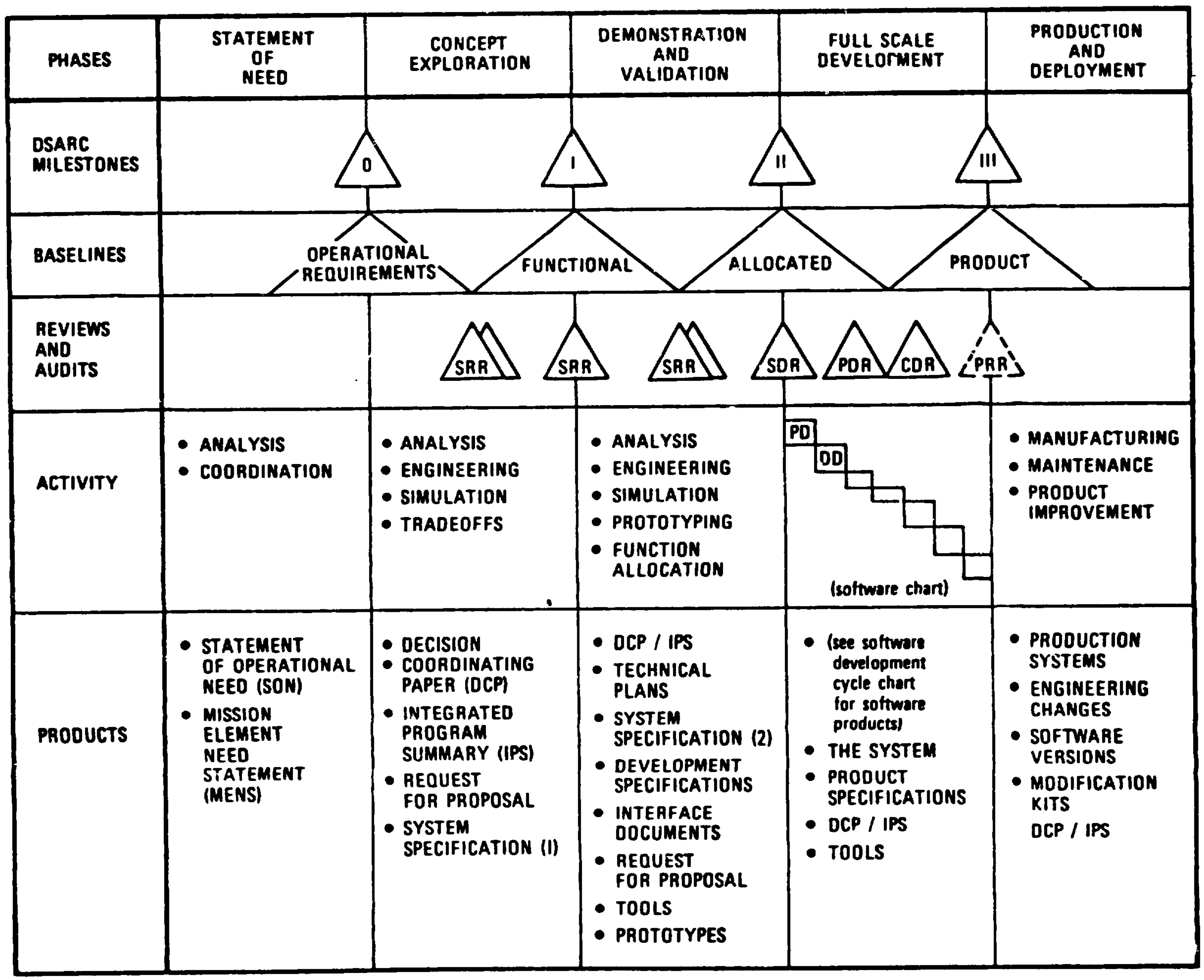
DSARC (BOEING)

CLASSICAL SOFTWARE DEVELOPMENT COST BREAKDOWN



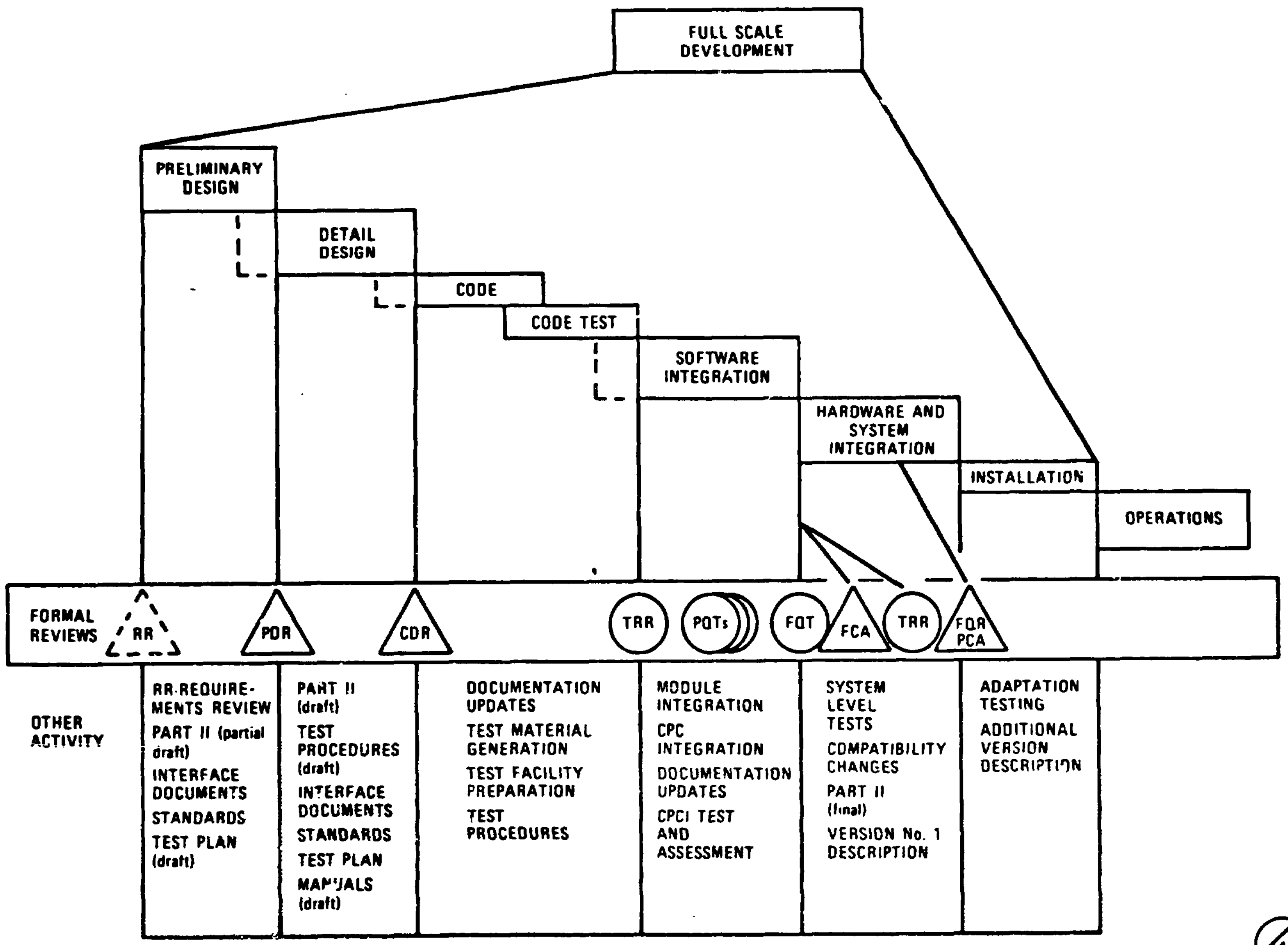
A6120

## System Life Cycle

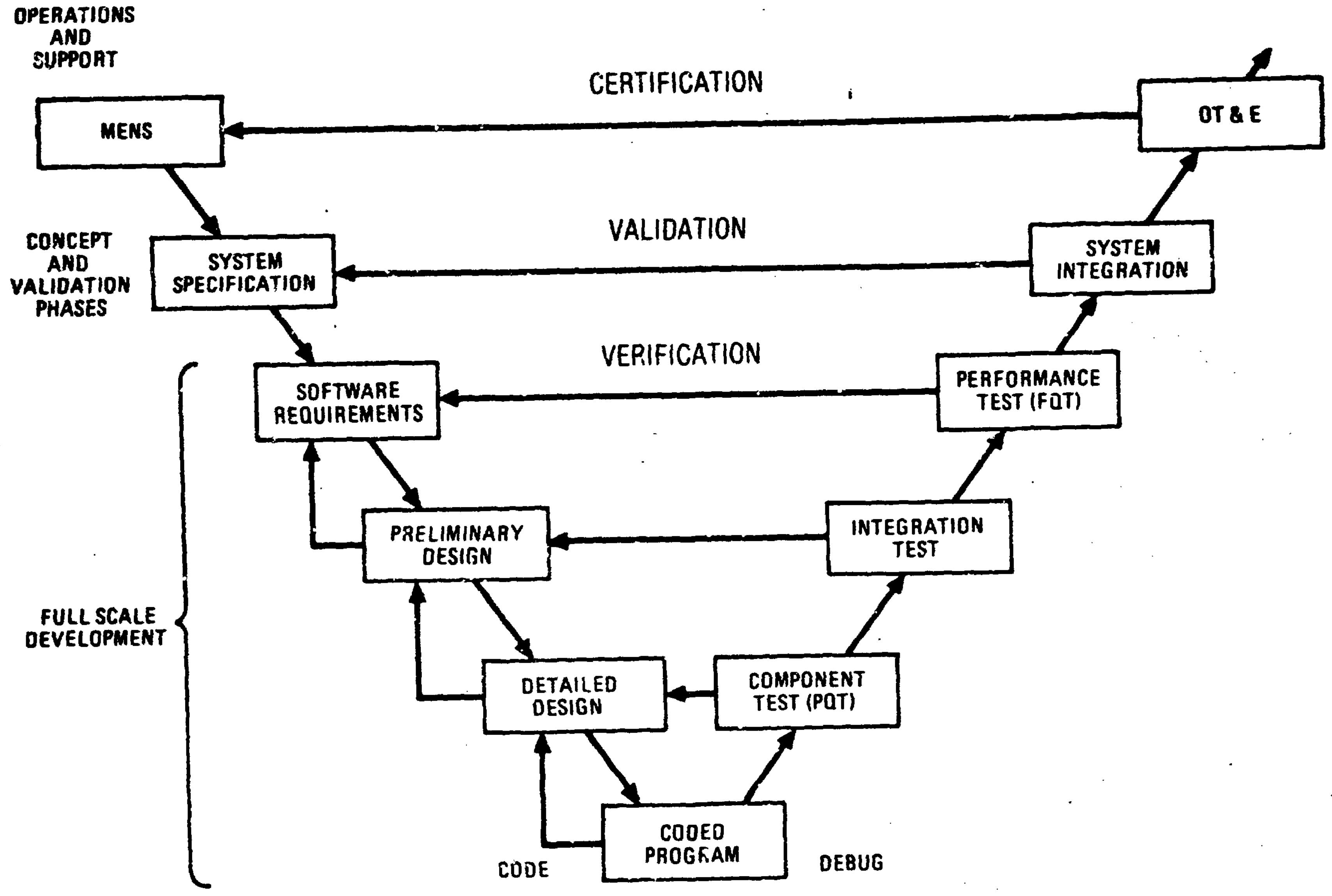


| PHASES | STATEMENT OF NEED | CONCEPT EXPLORATION | DEMONSTRATION AND VALIDATION | FULL SCALE DEVELOPMENT | PRODUCTION AND DEPLOYMENT |
|---|---|---|---|---|---|
| DSARC MILESTONES | | 0 | I | II | III |
| BASELINES | | OPERATIONAL REQUIREMENTS | FUNCTIONAL | ALLOCATED | PRODUCT |
| REVIEWS AND AUDITS | | SRR | SRR, SRR | SDR, PDR, CDR | PRR |
| ACTIVITY | • ANALYSIS<br>• COORDINATION | • ANALYSIS<br>• ENGINEERING<br>• SIMULATION<br>• TRADEOFFS | • ANALYSIS<br>• ENGINEERING<br>• SIMULATION<br>• PROTOTYPING<br>• FUNCTION ALLOCATION | PD<br>DD<br>(software chart) | • MANUFACTURING<br>• MAINTENANCE<br>• PRODUCT IMPROVEMENT |
| PRODUCTS | • STATEMENT OF OPERATIONAL NEED (SON)<br>• MISSION ELEMENT NEED STATEMENT (MENS) | • DECISION<br>• COORDINATING PAPER (DCP)<br>• INTEGRATED PROGRAM SUMMARY (IPS)<br>• REQUEST FOR PROPOSAL<br>• SYSTEM SPECIFICATION (I) | • DCP / IPS<br>• TECHNICAL PLANS<br>• SYSTEM SPECIFICATION (2)<br>• DEVELOPMENT SPECIFICATIONS<br>• INTERFACE DOCUMENTS<br>• REQUEST FOR PROPOSAL<br>• TOOLS<br>• PROTOTYPES | • (see software development cycle chart for software products)<br>• THE SYSTEM<br>• PRODUCT SPECIFICATIONS<br>• DCP / IPS<br>• TOOLS | • PRODUCTION SYSTEMS<br>• ENGINEERING CHANGES<br>• SOFTWARE VERSIONS<br>• MODIFICATION KITS<br>DCP / IPS |

## Software Development Cycle



## Verification - Validation Certification



# STANDARD WORK BREAKDOWN STRUCTURE FOR SPACE SYSTEMS

BY

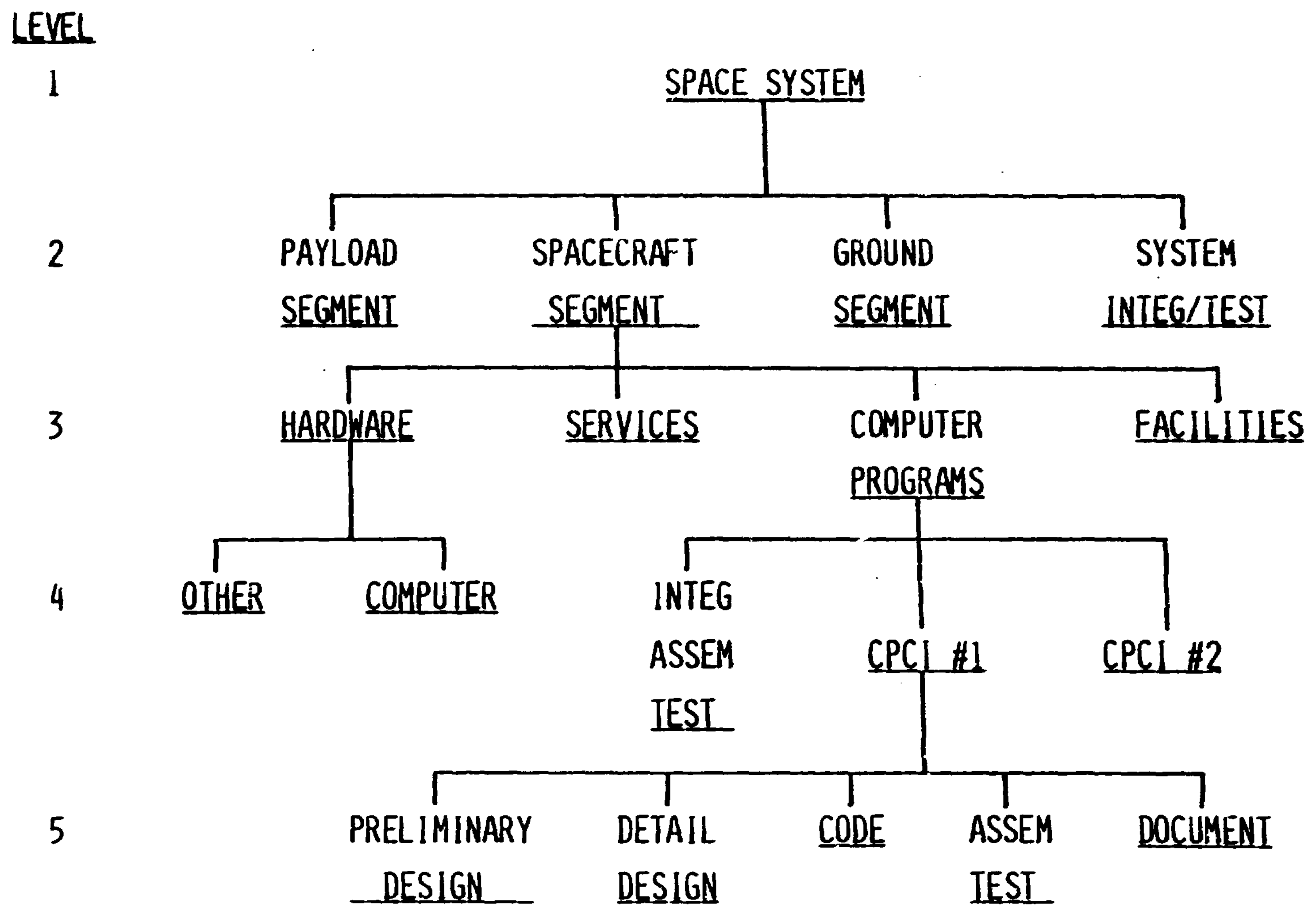
SPACE SYSTEMS COST ANALYSIS GROUP

STANDARDIZATION SUBGROUP

SSCAG

WORK BREAKDOWN STRUCTURE

(REFERENCE: PROPOSED APPENDIX F TO MIL-STD-881A)

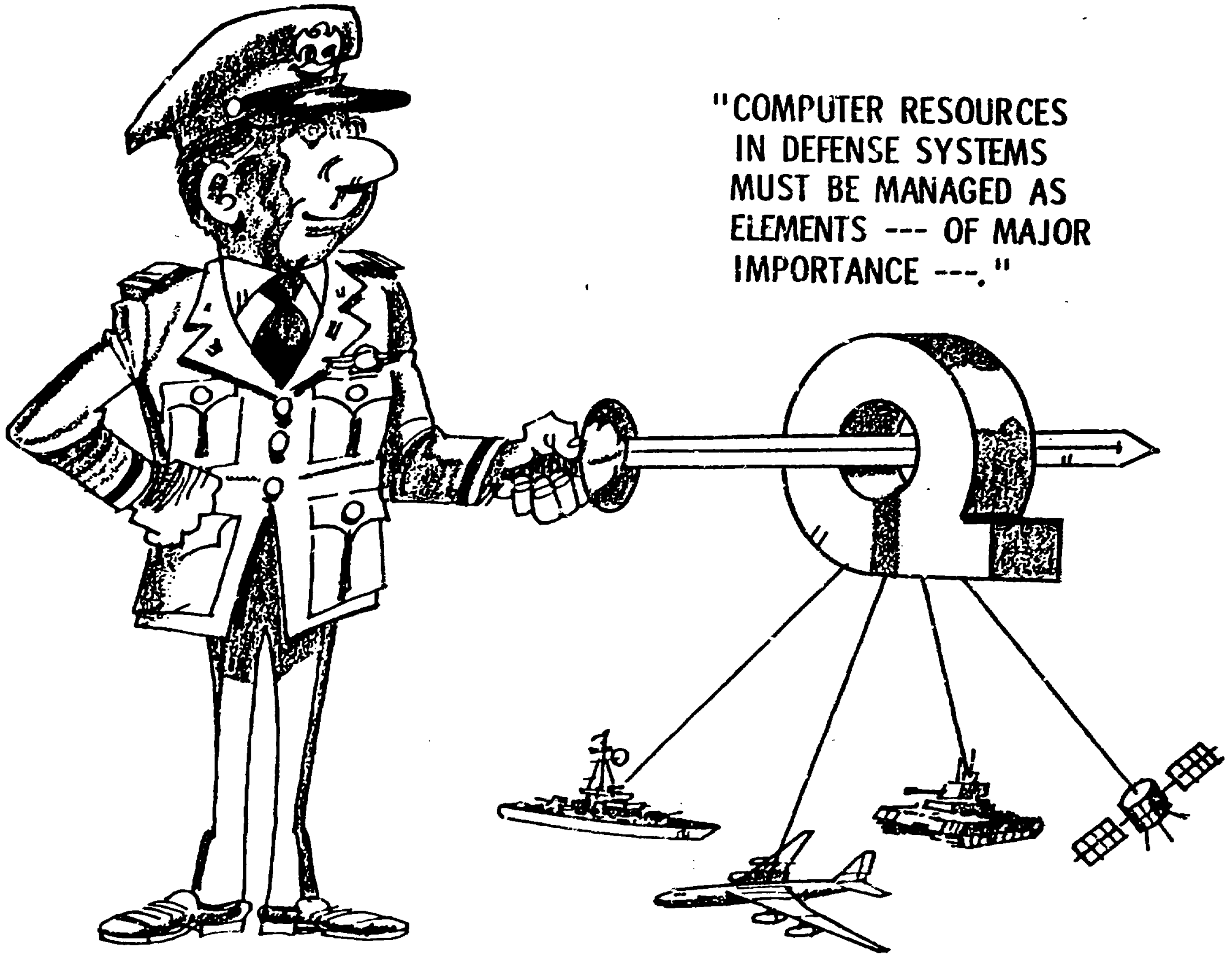


K 7823

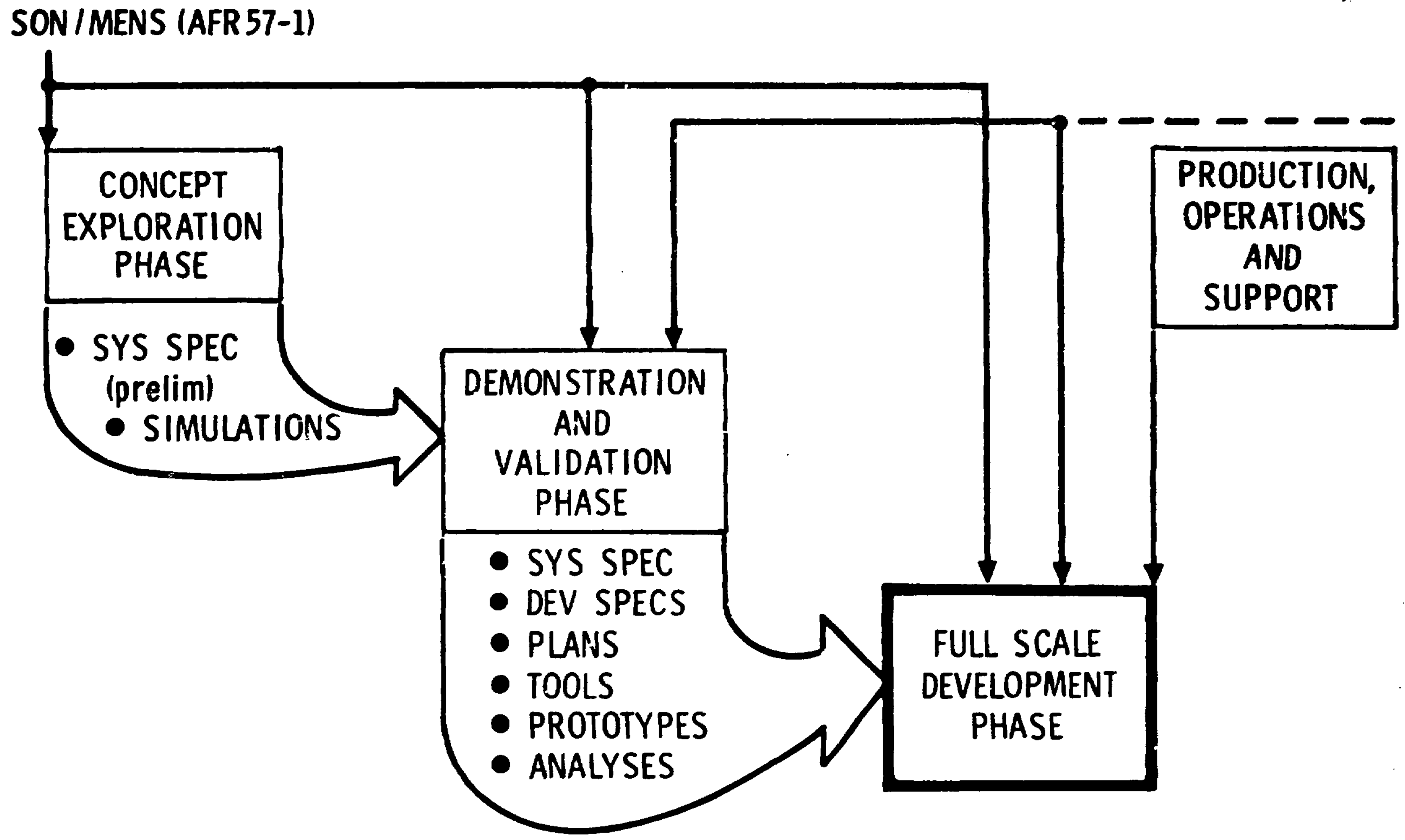
## DoD Says

(REFERENCE: DoDD 5000.29, 1976)

"COMPUTER RESOURCES IN DEFENSE SYSTEMS MUST BE MANAGED AS ELEMENTS --- OF MAJOR IMPORTANCE ---."



## Requirements


SON / MENS (AFR 57-1)
CONCEPT EXPLORATION PHASE
● SYS SPEC (prelim)
● SIMULATIONS
DEMONSTRATION AND VALIDATION PHASE
● SYS SPEC
● DEV SPECS
● PLANS
● TOOLS
● PROTOTYPES
● ANALYSES
FULL SCALE DEVELOPMENT PHASE
PRODUCTION, OPERATIONS AND SUPPORT

A-1045

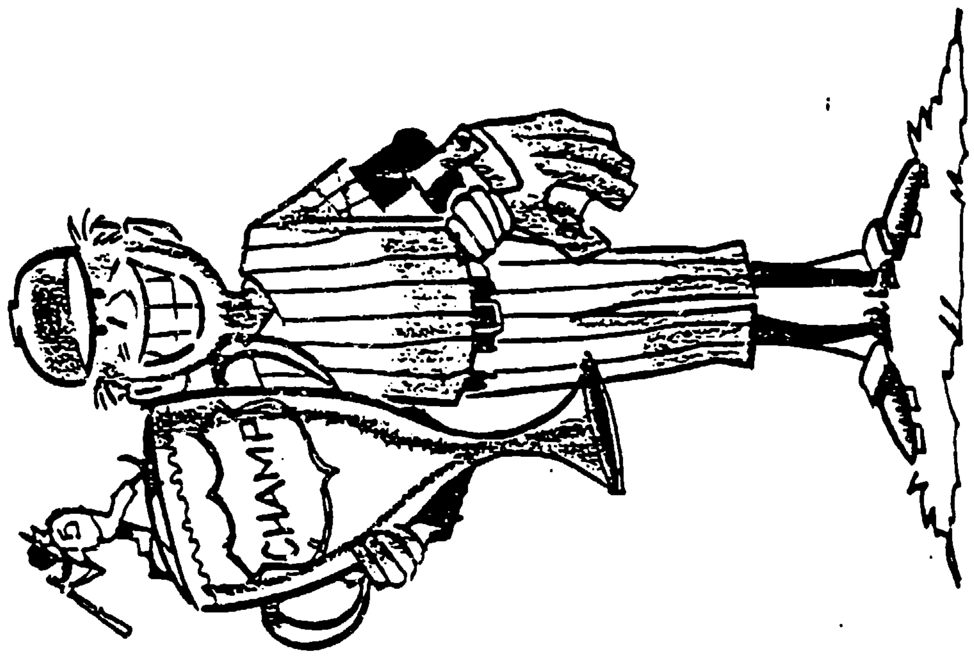
## DOD Standard 7935.1-S Comparison

| REFERENCE | | EQUIVALENT |
|---|---|---|
| ● SYS SPEC | | FUNCTIONAL DESCRIPTION |
| ● DEV SPEC | | SYSTEM/SUBSYSTEM SPECIFICATION<br>DATA REQUIREMENTS DOCUMENT |
| ● PROD SPEC | ✓ | PROGRAM SPECIFICATION<br>DATA BASE SPECIFICATION |
| ● ICD | | --- |
| ● T PLAN<br>● T PROC | | TEST PLAN |
| ● T RPT | | TEST ANALYSIS REPORT |
| ● U/M | | USERS MANUAL |
| ● P/M | | PROGRAM MAINTENANCE MANUAL |
| --- | ✓ | COMPUTER OPERATION MANUAL |

## SEC NAV INST 3560.1 Comparison

| REFERENCE | EQUIVALENT |
|---|---|
| ● SYS SPEC | TACTICAL OPERATIONAL REQUIREMENT<br>SYSTEM OPERATIONAL SPECIFICATION<br>SYSTEM OPERATIONAL DESIGN |
| ● DEV SPEC | PROGRAM PERFORMANCE SPECIFICATION<br>FUNCTION OPERATIONAL SPECIFICATION<br>FUNCTION OPERATIONAL DESIGN |
| ● PROD SPEC | PROGRAM DESIGN SPECIFICATION<br>PROGRAM DESCRIPTION DOCUMENT<br>✓ DATA BASE DESIGN |
| ● ICD | INTERFACE DESIGN SPECIFICATION |
| ● T PLAN | TEST PLAN<br>TEST SPECIFICATION |
| ● T PROC<br>● T RPT | TEST PROCEDURES AND REPORTS |
| ● U/M | ✓ COMMAND AND STAFF MANUAL<br>SYSTEM OPERATOR'S MANUAL |
| ● P/M | PROGRAM DESIGN MANUAL<br>✓ OPERATOR'S MANUAL |

K4295



PRESENTATION MADE BY

JIM McCALL
GENERAL ELECTRIC

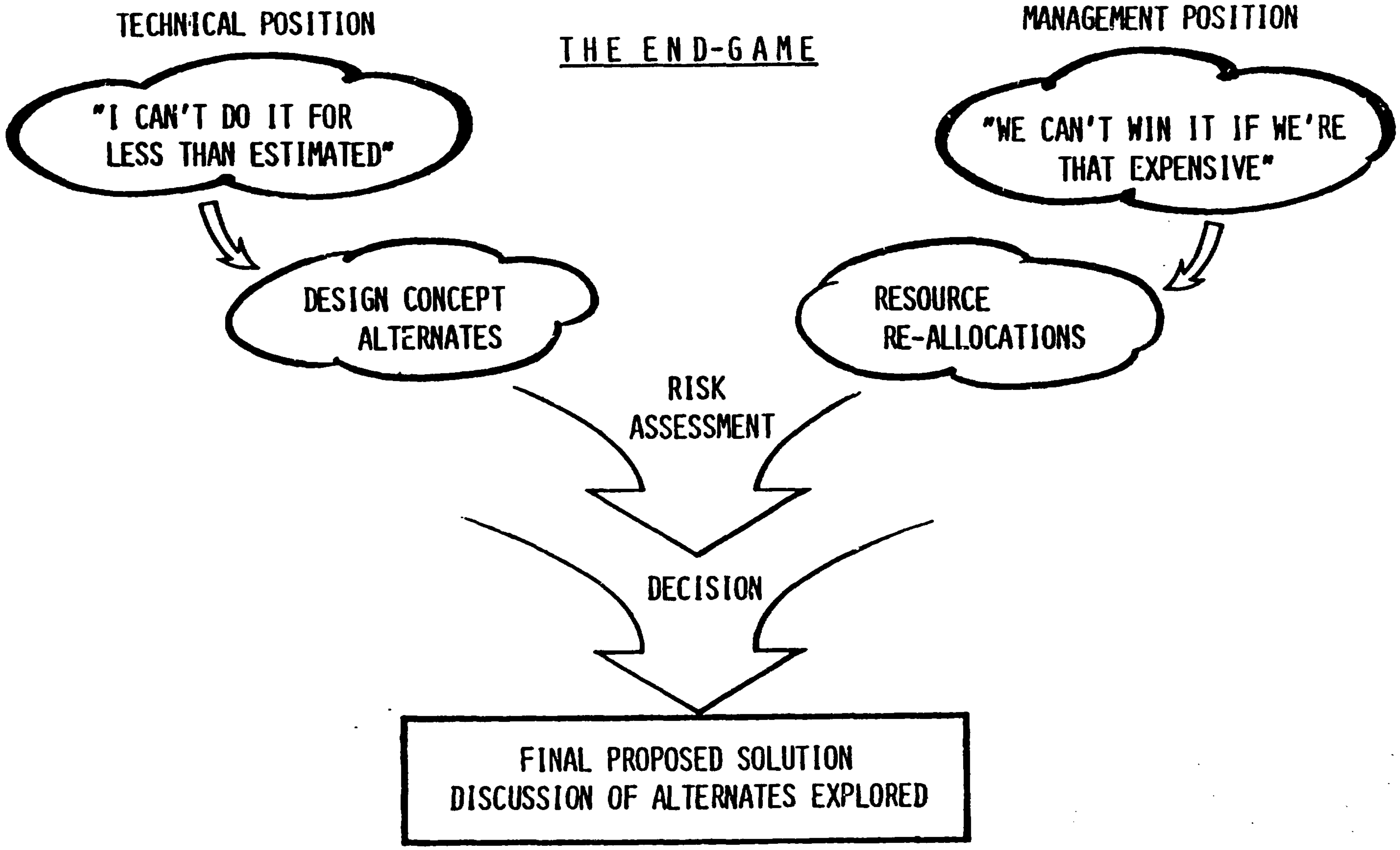
TO

PANEL ON ESTIMATING SOFTWARE COSTS

## SOFTWARE COST ESTIMATION

MODELS
METHODOLOGY
TOOLS

JIM McCALL


TECHNICAL POSITION
THE END-GAME
MANAGEMENT POSITION
"I CAN'T DO IT FOR LESS THAN ESTIMATED"
"WE CAN'T WIN IT IF WE'RE THAT EXPENSIVE"
DESIGN CONCEPT ALTERNATES
RESOURCE RE-ALLOCATIONS
RISK ASSESSMENT
DECISION
FINAL PROPOSED SOLUTION
DISCUSSION OF ALTERNATES EXPLORED

4130-Q

GENERAL ELECTRIC

# TRADITIONAL SSEB ENVIRONMENT

space division

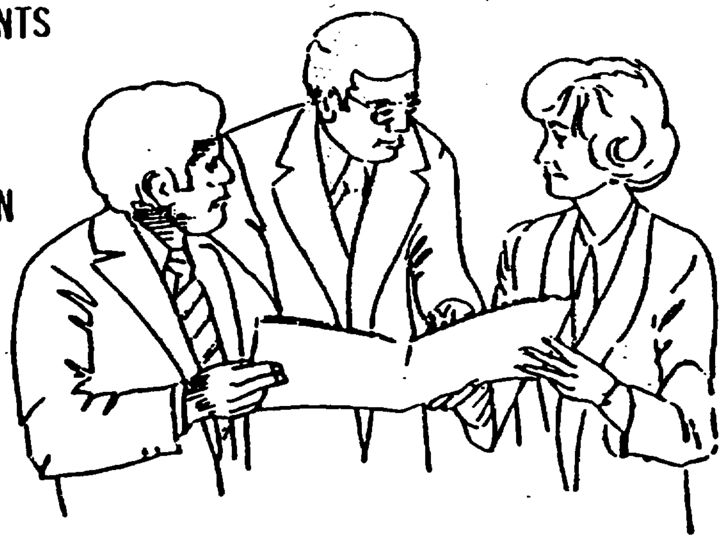
6181

## MANAGEMENT PANEL

- MIL-STDS
- MANAGEMENT APPROACH
- EVALUATION CRITERIA

- DEFICIENCIES
- EVALUATION SCORE

## TECHNICAL PANEL



- REQUIREMENTS
- TECHNICAL APPROACH
- EVALUATION CRITERIA

- DEFICIENCIES
- EVALUATION SCORE

## COST PANEL

- COST PROPOSAL
- DD 633
- BUDGET
- CERTIFICATIONS
- AUDIT

4138-AC

D-20

GENERAL ELECTRIC

## MAJOR FACTORS INFLUENCING SOFTWARE COST

space division

6181

ALTHOUGH IT'S DIFFICULT TO ISOLATE AND QUANTITATIVELY MEASURE THEIR IMPACT, MANY FACTORS DIRECTLY AFFECT COST

FACTORS UNIQUE TO SOFTWARE

- NUMBER OF INSTRUCTIONS
- TYPE OF SOFTWARE (APPLICATION)
- SIZE OF DATA BASE
- PROGRAMMING LANGUAGE
- TOOLS AND AIDS
- DEVELOPMENT METHODOLOGY
- HARDWARE CONSTRAINTS

FACTORS COMMON TO HARDWARE & SOFTWARE

- DEVELOPMENT SCHEDULE
- COMPLEXITY AND NOVELTY
- QUALITY CONSIDERATIONS
- PERSONNEL CAPABILITIES
- CUSTOMER ENVIRONMENT
- REQUIREMENTS INSTABILITY
- CONCURRENT HARDWARE/SOFTWARE DEVELOPEMENT
- SECURITY
- DOCUMENTATION

THESE FACTORS ARE NEVER CONSTANT FROM ONE PROJECT TO THE NEXT

## FREQUENTLY USED COST ESTIMATING METHODS

| METHOD | BENEFITS | LIMITATIONS |
|---|---|---|
| SIMILAR EXPERIENCE<br>ESTIMATE BASED ON SIMILARITY OF CURRENT EFFORT TO ONE PREVIOUSLY COMPLETED. | PREVIOUS EXPERIENCE IS A GOOD INDICATOR OF FUTURE PERFORMANCE.<br>EASY TO VISUALIZE AND RELATE TO. | REQUIRES A PREVIOUS PROJECT OF SIMILAR SIZE, SCHEDULE, APPLICATION AND DEVELOPMENT ENVIRONMENT. |
| STATISTICAL (DELPHI)<br>GENERATES A WEIGHTED AVERAGE COST AND ALSO PROVIDES OPTIMISTIC/ PESSIMISTIC FIGURES. | CAN BE APPLIED IN SITUATIONS WHERE THERE IS MINIMAL DATA AVAILABLE OR NO PREVIOUS EXPERIENCE TO USE AS A BASIS FOR ESTIMATING. | A WIDE VARIATION IN ESTIMATES FROM INDIVIDUAL TO INDIVIDUAL<br>MUST BE CAREFUL NOT TO INAPPROPRIATELY BIAS RESULTS. |
| CONSTRAINT<br>FIXED RESOURCE WITH WORK LEVEL AS THE VARIABLE. | APPLICABLE TO SMALL PROJECTS. | NOT GENERALLY USEFUL IN NORMAL CONTRACT SITUATION. |
| UNIT-OF-WORK<br>PROJECT IS SUBDIVIDED INTO PIECES WHICH CAN BE DONE BY A SINGLE INDIVIDUAL OVER A SPECIFIED TIME INTERVAL. | BASED ON DETAILED PLANNING AND WORK FLOW. | REQUIRES EXCESSIVE SUB-DIVISION FOR ALL BUT THE SMALLEST PROJECT.<br>MAY OVERLOOK "SYSTEM" LEVEL ACTIVITIES IF NOT DONE TOP-DOWN. |
| MATHEMATICAL METHODS & MODELS<br>GENERALLY BASED ON COST ESTIMATING RELATIONSHIPS (CER's) TYPICALLY EMPLOYING PROGRAMMER PRODUCTIVITY AND/OR SYSTEM SIZE AS THE MAJOR INPUTS. | PROVIDES AN OBJECTIVE REPRESENTATION OF PAST PERFORMANCE. | OFTEN DIFFICULT TO ACCURATELY ESTIMATE THE INPUT PARAMETERS.<br>DEGREE OF REPRESENTATIVENESS. |

## COST ESTIMATION TECHNIQUES SURVEYED

6128

| TECHNIQUE | APPROACH |
|---|---|
| ● RADC/DOTY ASSOCIATES | ● NONLINEAR REGRESSION |
| ● GENERAL RESEARCH CORP. | ● NONLINEAR REGRESSION |
| ● IBM/WALSTON | ● PRODUCTIVITY |
| ● PUTNAM | ● PHENOMENOLOGICAL MODELS |
| ● RCA PRICE-S | ● PHENOMENOLOGICAL MODELS |
| ● TRW/WOLVERTON | ● COST-PER-INSTRUCTION |
| ● SYSTEM DEVELOPMENT CORP. | ● LINEAR REGRESSION |
| ● BOEING COMPUTER SERVICES | ● PRODUCTIVITY |

$\left.\begin{array}{l}\text{USAF/ESD}\\ \text{USAF/SAMSO}\\ \text{TECOLOTE}\end{array}\right\}$ MINIMUM DATA AVAILABLE

## STRUCTURED EVALUATION

6128

EACH TECHNIQUE EVALUATED ACCORDING TO:

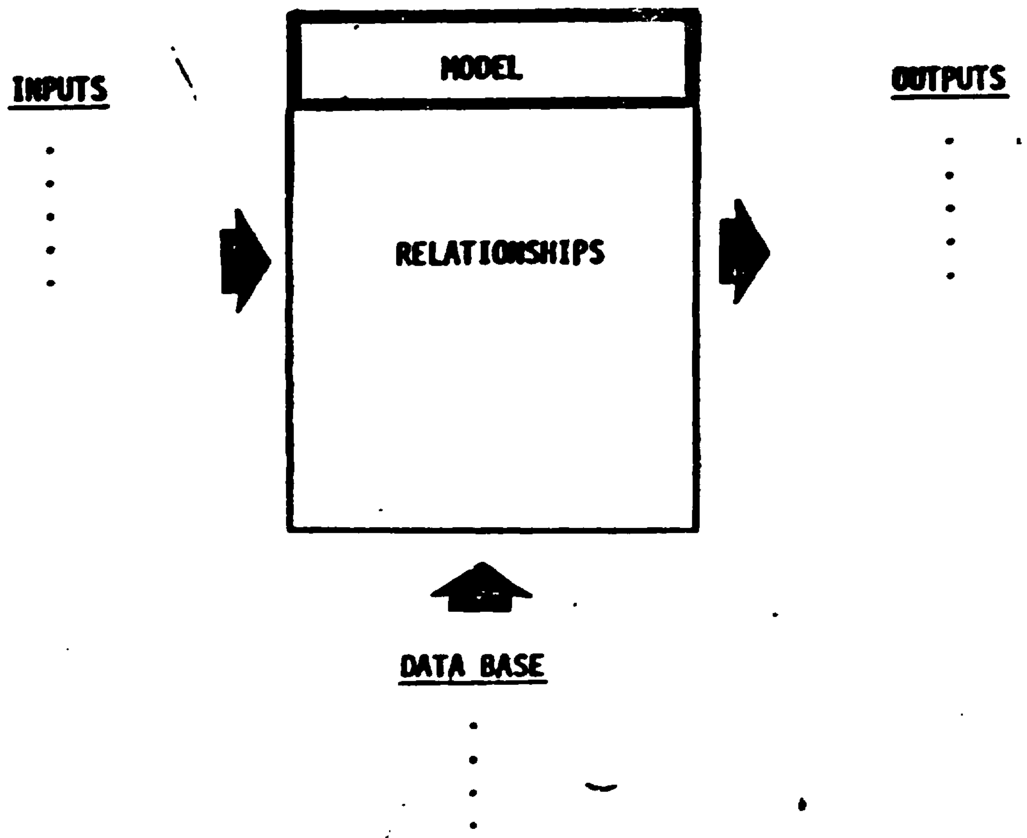
- NINE DECISION PROCESS FACTORS
- SIX USER PROCESS FACTORS
- EIGHT MODEL CHARACTERISTICS
- 46 COST DRIVERS
  - DEVELOPMENT EXPERTISE
  - CUSTOMER ENVIRONMENT
  - DEVELOPMENT ENVIRONMENT
  - SYSTEM TECHNICAL CHARACTERISTICS

GENERAL ELECTRIC

## DETAILED EVALUATION APPROACH

space division

92



- BACKGROUND
- DESCRIPTION OF TECHNIQUE
- NARRATIVE EVALUATION
- REFERENCES
- EVALUATION FACTOR ASSESSMENT

GENERAL ELECTRIC

## COST FACTORS CONSIDERED IN COST ESTIMATION TECHNIQUES

space division

| COST FACTORS | DOTY | IBM | GRC | PUTNAM | SDC | TRW | PRICE-S | BOEING |
|---|---|---|---|---|---|---|---|---|
| DEVELOPMENT EXPERTISE | | | | | | | | |
| PERSONNEL EXPERIENCE AND QUALIFICATIONS | | X | | I | I | | I | X |
| DESIGN PERSONNEL PARTICIPATION IN DEVELOPMENT | | X | | | X | | | |
| EXPERIENCE WITH COMPUTER SYSTEM | | X | | | | | I | |
| EXPERIENCE WITH PROGRAMMING LANGUAGE | | X | X | | X | | I | |
| EXPERIENCE WITH APPLICATION | | X | | | X | | I | X |
| CUSTOMER ENVIRONMENT | | | | | | | | |
| CUSTOMER INTERFACE COMPLEXITY | | X | | | | | | X |
| USER PARTICIPATION IN REQUIREMENTS DEFINITION | | X | | | | | | |
| REQUIREMENTS STABILITY | X | X | X | | X | | I | |
| CUSTOMER EXPERIENCE WITH APPLICATION | | X | | | | | | X |
| SECURITY REQUIREMENTS | | X | | | | X | | |
| DOCUMENTATION REQUIREMENTS | | X | | | X | | I | X |
| MULTIPLE OPERATIONAL SITES | | | | | | | | |
| MULTIPLE CONTRACTORS | | | | | | | | |
| FORMALIZED V&C | | | | | | | I | |
| TRAINING OF OPERATING AGENCY | | | | | | | | |

X = MODEL DIRECTLY CONSIDERS FACTOR

I = MODEL INDIRECTLY CONSIDERS FACTOR

GENERAL ELECTRIC

# COST FACTORS CONSIDERED IN COST ESTIMATION TECHNIQUES (CONTINUED)

space division

)2

| COST FACTORS | DOTY | IBM | GRC | PUTNAM | SDC | TRW | PRICE-S | BOEING |
|---|---|---|---|---|---|---|---|---|
| DEVELOPMENT ENVIRONMENT | | | | | | | | |
| NUMBER OF PERSONNEL | | | X | | | | | X |
| DURATION OF PERSONAL ASSIGNMENTS | | | | | X | | | |
| CONCURRENT HARDWARE/SOFTWARE DEVELOPMENT | X | X | | | X | | I | |
| ACCESS TO DEVELOPMENT COMPUTER | X | X | | X | | | | |
| USE OF MODERN PROGRAMMING PRACTICES | | X | | X | | | I | |
| DETAIL OF REQUIREMENTS DEFINITION | X | | | | X | | | |
| DEVELOPMENT VIA INTERACTIVE OR BATCH | X | | X | X | X | | | X |
| DEVELOPMENT AT OPERATIONAL SITE | X | | | | X | | I | |
| OPERATIONAL SYSTEM DIFFERENT FROM DEVELOPMENT SYSTEM | X | | | | X | | | |
| DEVELOPMENT AT MORE THAN ONE LOCATION | X | | | | X | | | |
| CAPABILITY OF TOOLS AND AIDS | | | | X | | | | X |
| SUPPORT PERSONNEL AVAILABILITY | | | | | | | | |
| AVAILABILITY OF DEVELOPMENT SYSTEM | | | | | | | | |
| MANAGEMENT TECHNIQUES | | | | | | | | |
| STANDARDS AND PROCEDURES | | | | | | | | |
| CONFIGURATION MANAGEMENT | | | | | | | I | |
| IMPLEMENTATION LANGUAGE HOL/MOL | | | | X | | X | | X |

X = MODEL DIRECTLY CONSIDERS FACTOR

I = MODEL INDIRECTLY CONSIDERS FACTOR

GENERAL ELECTRIC

## COST FACTORS CONSIDERED IN COST ESTIMATION TECHNIQUES (CONTINUED)

space division

5192

| COST FACTORS | DOTY | IBM | GRC | PUTNAM | SDC | TRW | PRICE-S | BOEING |
|---|---|---|---|---|---|---|---|---|
| TECHNICAL | | | | | | | | |
| NUMBER OF DELIVERED INSTRUCTIONS | X | X | X | X | | X | X | X |
| NUMBER OF PROGRAM MODULES | | | | X[1] | | | X | |
| APPLICATION CATEGORIES | X | X | | | X | | | |
| INSTRUCTION MIX | | X | | | I | X | X | X |
| DATA BASE SIZE | | X | | X[1] | X | | I | |
| COMPLEXITY OF CODE | | X | X | | X | X | | |
| OUTPUT/DISPLAY REQUIREMENTS | X | | | X | X | | | |
| RESOURCE CONSTRAINTS | X | X | X | | X | X | X | |
| TECHNOLOGY LEVELS | | | | X | | | X | |
| MODIFICATION OF EXISTING SOFTWARE | | | X | P | | X | X | X |
| FIRST SOFTWARE DEVELOPED ON CPU | X | | | | X | | I | |
| QUALITY REQUIREMENTS | | | | | | X | | |
| SCHEDULE | | | | | | | | |
| ELAPSED TIME | | | | X | | | X | |
| MILESTONE PLACEMENT | | | | | | | X | |

X = MODEL DIRECTLY CONSIDERS FACTOR

I = MODEL INDIRECTLY CONSIDERS FACTOR

P = MODEL PARTIALLY CONSIDERS FACTORS

[1] SECONDARY RELATIONSHIP

ISP

6157

## SDC MODEL

INPUTS

- APPLICATION DESCRIPTORS (LINES OF CODE NOT USED)
- ENVIRONMENT DESCRIPTORS

SYSTEM DEVELOPMENT CORPORATION

- $PM = C_0 + \sum_{i=1}^{n} C_i X_i$
- $CH = K_0 + \sum_{i=1}^{n} K_i X_i$
- $S = D_0 + \sum_{i=1}^{n} D_i X_i$
- GUIDELINES FOR PLANNING FACTORS
- PRODUCTIVITY FACTORS

4097E

OUTPUTS

- COST IN PERSON-MONTHS (PM)
- ELAPSED DEVELOPMENT TIME (S)
- COMPUTER HOURS FOR DEVELOPMENT (CH)

DATA BASE

SDC DATA (169 DEVELOPMENTS)

# SDC MODEL

6157

## Cost Factor Descriptions

| | |
|---|---|
| $X_1$ | - VAGUENESS OF DESIGN REQUIREMENTS DEFINITION. CODED: DIRECT TRANSLATION OF (SIMPLE) MANUAL TASKS TO AUTOMATIC FUNCTIONS - 0; ONLY A FEW NEW FUNCTIONS TO BE AUTOMATED, WITH REQUIREMENTS RELATIVELY CLEAR AND PRECISE = 1; MANY NEW FUNCTIONS TO BE AUTOMATED, REQUIREMENTS RELATIVELY PRECISE = 2; MANY ILL-DEFINED AND UNSTRUCTURED FUNCTIONS TO BE AUTOMATED = 3. |
| $X_2$ | - INNOVATION REQUIRED. CODED: YES = 1; NO = 0. |
| $X_3$ | - LACK OF KNOWLEDGE OF OPERATIONAL REQUIREMENTS. CODED: IN GREAT DETAIL = 0; IN BROAD OUTLINE = 1; ONLY VAGUELY = 2. |
| $X_4$ | - NUMBER OF ORGANIZATIONAL USERS. CODED: MINIMUM VALUE OF 1. |
| $X_5$ | - NUMBER OF ADP CENTERS. CODED: MINIMUM VALUE OF 1. |
| $X_6$ | - COMPLEXITY OF PROGRAM SYSTEM INTERFACE. CODED: MORE THAN 50% OF DESIGN EFFORT DEVOTED TO DATA TRANSFER PROBLEMS TO OR FROM THE PROGRAM DATA POINT = 2; BETWEEN 10% AND 50% EFFORT TO DATA TRANSFER PROBLEMS = 1; LESS THAN 10% = 0. |
| $X_{11}$ | - STABILITY OF DESIGN. CODED: INITIAL DESIGN CARRIED THROUGH WITHOUT CHANGE = 0; FEW CHANGES TO INITIAL PROGRAM DESIGN = 1; FREQUENT CHANGES TO PROGRAM DESIGN = 2; INITIAL PROGRAM DESIGN ALMOST COMPLETELY REVISED = 3. |
| $X_{17}$ | - NUMBER OF CONDITIONAL BRANCHES. CODED IN THOUSANDS. |
| $X_{18}$ | - NUMBER OF WORDS IN THE DATA BASE. CODED IN THOUSANDS. |
| $X_{19}$ | - NUMBER OF CLASSES OF ITEMS IN THE DATA BASE. CODED IN RAW SCORE. |
| $X_{25}$ | - PERCENT CLERICAL INSTRUCTIONS. CODED IN PERCENT. |
| $X_{26}$ | - PERCENT MATHEMATICAL INSTRUCTIONS. CODED IN PERCENT. |
| $X_{30}$ | - PERCENT INFORMATION STORAGE AND RETRIEVAL FUNCTIONS. CODED IN PERCENT. |
| $X_{35}$ | - PERCENT GENERATION FUNCTIONS. CODED IN PERCENT. |
| $X_{37}$ | - FREQUENCY OF OPERATION. CODED: NOT APPLICABLE = 0; LESS THAN 1/MONTH, MORE THAN 1/MONTH AND LESS THAN 1/WEEK = 2; MORE THAN 1/WEEK AND LESS THAN 1/DAY = 3; DAILY = 4; UTILITY OR ON LINE (INCLUDING COMPILERS) = 5. |
| $X_{40}$ | STRINGENT TIMING REQUIREMENTS. CODED: SCHEDULE RESTRICTIONS IMPOSED = 1; NONE IMPOSED = 0. |
| $X_{41}$ | - NUMBER OF SUBPROGRAMS. CODED IN RAW SCORE. |
| $X_{42}$ | - PROGRAMMING LANGUAGE. CODED: HOL = 1; POL = 0. |
| $X_{46}$ | - EXTERNAL DOCUMENTATION. CODED: NUMBER OF PAGES WRITTEN FOR, OR DISTRIBUTED TO, CUSTOMERS |

| | |
|---|---|
| $X_{47.1}$ | - TOTAL NUMBER OF EXTERNAL DOCUMENT TYPES WRITTEN. |
| $X_{47.3}$ | - TOTAL NUMBER OF INTERNAL DOCUMENT TYPES WRITTEN. |
| $X_{48.1}$ | - BUSINESS } CODED: AS MUTUALLY EXCLUSIVE BINARY VARIABLES; I.E., PROGRAMS CLASSIFIED AS BUSINESS APPLICATION = 1; REMAINING APPLICATIONS = 0. |
| $X_{48.3}$ | - UTILITY } |
| $X_{48.5}$ | - STAND ALONE. CODED: YES = 1; NO = 0. |
| $X_{51}$ | - FIRST PROGRAM ON COMPUTER: CODED: YES = 1; NO = 0. |
| $X_{52}$ | - AVERAGE TURNAROUND TIME. CODED: LESS THAN 2 HOURS = 0; 2 TO 11 HOURS = 1; 12 TO 24 HOURS = 2; GREATER THAN 24 HOURS = 3. |
| $X_{53}$ | - ADP COMPONENTS DEVELOPED CONCURRENTLY. CODED: COMPONENTS DEVELOPED CONCURRENTLY = 1; COMPONENTS NOT DEVELOPED CONCURRENTLY = 0. |
| $X_{54}$ | - SPECIAL DISPLAY EQUIPMENT: CODED: SPECIAL DISPLAY EQUIPMENT USED = 1; NOT USED = 0. |
| $X_{55}$ | - CORE CAPACITY. |
| $X_{56}$ | - RANDOM ACCESS DEVICE USED. CODED: USE OF SUCH STORAGE = 1; SUCH STORAGE NOT USED = 0. |
| $X_{57}$ | - NUMBER OF BITS PER WORD. |
| $X_{61}$ | - PERCENT SENIOR PROGRAMMERS. CODED: THE RATIO OF THE TOTAL NUMBER OF SENIOR OR SYSTEMS PROGRAMMERS TO THE TOTAL NUMBER OF PROGRAMMERS ASSIGNED TO THE PROJECT. |
| $X_{62}$ | - AVERAGE PROGRAMMER EXPERIENCE WITH LANGUAGE. CODED IN MONTHS. |
| $X_{63}$ | - AVERAGE PROGRAMMER EXPERIENCE WITH APPLICATION. CODED IN MONTHS. |
| $X_{64}$ | - PERCENT PROGRAMMERS PARTICIPATING IN PROGRAM DESIGN. |
| $X_{65}$ | - PERSONNEL CONTINUITY. CODED: NUMBER OF PERSONNEL WORKING FOR THE DURATION OF THE PROJECT, DIVIDED BY THE MAXIMUM NUMBER ASSIGNED AT ANY ONE TIME. |
| $X_{71}$ | - PROGRAM DEVELOPED AT SITE OTHER THAN THE OPERATIONAL INSTALLATION. CODED: YES = 1; NO = 0. |
| $X_{72}$ | - DIFFERENT COMPUTERS FOR PROGRAMMING AND OPERATION. CODED: YES = 1; NO = 0. |
| $X_{74}$ | - NUMBER OF LOCATIONS FOR PROGRAM DATA POINT DEVELOPMENT. |
| $X_{75}$ | - NUMBER OF MAN TRIPS. |
| $X_{76}$ | - PROGRAM DATA POINT DEVELOPED BY MILITARY ORGANIZATIONS. CODED: YES = 1; NO = 0. |

ISP

6157

# IBM/WALSTON AND FELIX MODEL

**INPUTS**

- LINES OF DELIVERED CODE (L) (000)
- TWENTY-NINE ENVIRONMENTAL FACTORS USED TO COMPUTE PRODUCTIVITY INDEX (I)

**IBM - WALSTON AND FELIX**

**EFFORT**

$E = 5.2\ L^{0.91}$

$I = \sum_{i=1}^{29} W_i X_i$

WHERE:

$W_i$ = FACTOR WEIGHT

$X_i$ = FACTOR VALUE (+1, 0, -1)

**PAGES OF DOCUMENTATION**

$D = 49\ L^{1.01}$

**DURATION**

$M = 4.1\ L^{0.36}$, $M = 2.47\ E^{0.35}$

**AVERAGE STAFF SIZE**

$S = 0.54\ E^{0.6}$

**COMPUTER COST**

$C = 1.84\ L^{0.96}$, $C = 1.1\ E^{0.81}$

4097B

**OUTPUTS**

- PERSON-MONTHS OF DEVELOPMENT (E)
- PAGES OF DOCUMENTATION (D)
- DURATION OF DEVELOPMENT (M)
- AVERAGE STAFF SIZE (S)
- COMPUTER COSTS (C)

DETAILED REPORTS FROM IBM FEDERAL SYSTEMS DIVISION ON THE SOFTWARE DEVELOPMENT ENVIRONMENT, THE PRODUCT, RESOURCES, AND SCHEDULE FOR 60 COMPLETED SOFTWARE DEVELOPMENT PROJECTS. SIZE RANGE FROM 4,000 TO 467,000 LINES OF CODE AND FROM 12 PM TO 11,578 PM. WIDE VARIETY OF APPLICATIONS REFLECTED IN DATA.

ISP

6157

# IBM/WALSTON AND FELIX MODEL

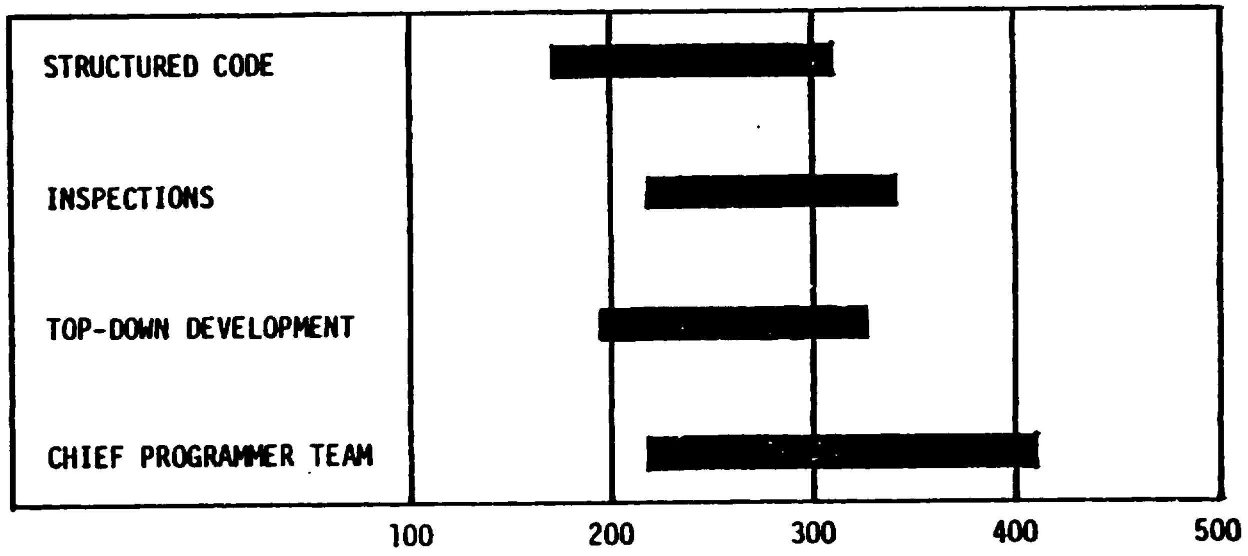
Factors Showing High Correlation with Productivity

| FACTOR | RESPONSE GROUP MEAN PRODUCTIVITY (DSL/PM) | | |
|---|---|---|---|
| STRUCTURED PROGRAMMING | 0 - 33% 169 | 34 - 66% — | >66% 301 |
| DESIGN AND CODE INSPECTIONS | 0 - 33% 220 | 34 - 66% 300 | >66% 339 |
| TOP DOWN DEVELOPMENT | 0 - 33% 196 | 34 - 66% 237 | >66% 321 |
| CHIEF PROGRAMMER TEAM USAGE | 0 - 33% 219 | 34 - 66% — | >66% 408 |
| OVERALL COMPLEXITY OF CODE DEVELOPED | <AVERAGE 314 | | >AVERAGE 185 |
| COMPLEXITY OF APPLICATION PROCESSING | <AVERAGE 349 | AVERAGE 345 | >AVERAGE 168 |
| COMPLEXITY OF PROGRAM FLOW | <AVERAGE 289 | AVERAGE 299 | >AVERAGE 209 |
| OVERALL CONSTRAINTS ON PROGRAM DESIGN | MINIMAL 293 | AVERAGE 286 | SEVERE 166 |
| PROGRAM DESIGN CONSTRAINTS ON MAIN STORAGE | MINIMAL 391 | AVERAGE 277 | SEVERE 193 |
| PROGRAM DESIGN CONSTRAINTS ON TIMING | MINIMAL 303 | AVERAGE 317 | SEVERE 171 |
| CODE FOR REAL-TIME OR INTERACTIVE OPERATION, OR EXECUTING UNDER SEVERE TIMING CONSTRAINT | <10% 279 | 10 - 40% 337 | >40% 203 |
| PERCENTAGE OF CODE FOR DELIVERY | 0 - 90% 159 | 91 - 99% 327 | 100% 265 |
| CODE CLASSIFIED AS NONMATHEMATICAL APPLICATION AND I/O FORMATTING PROGRAMS | 0 - 33% 188 | 34 - 66% 311 | 67 - 100% 267 |
| NUMBER OF CLASSES OF ITEMS IN THE DATA BASE PER 1000 LINES OF CODE | 0 - 15 334 | 16 - 80 243 | >80 193 |
| NUMBER OF PAGES OF DELIVERED DOCUMENTATION PER 1000 LINES OF DELIVERED CODE | 0 - 32 320 | 33 - 88 252 | >88 195 |

| FACTOR | RESPONSE GROUP MEAN PRODUCTIVITY (DSL/PM) | | |
|---|---|---|---|
| CUSTOMER INTERFACE COMPLEXITY | <NORMAL 500 | NORMAL 295 | >NORMAL 124 |
| USER PARTICIPATION IN THE DEFINITION OF REQUIREMENTS | NONE 491 | SOME 267 | MUCH 205 |
| CUSTOMER ORIGINATED PROGRAM DESIGN CHANGES | FEW 297 | | MANY 196 |
| CUSTOMER EXPERIENCE WITH THE APPLICATION AREA OF THE PROJECT | NONE 318 | SOME 340 | MUCH 206 |
| OVERALL PERSONNEL EXPERIENCE AND QUALIFICATIONS | LOW 132 | AVERAGE 257 | HIGH 410 |
| PERCENTAGE OF PROGRAMMERS DOING DEVELOPMENT WHO PARTICIPATED IN DESIGN OF FUNCTIONAL SPECIFICATIONS | <25% 153 | 25 - 50% 242 | >50% 391 |
| PREVIOUS EXPERIENCE WITH OPERATIONAL COMPUTER | MINIMAL 146 | AVERAGE 270 | EXTENSIVE 312 |
| PREVIOUS EXPERIENCE WITH PROGRAMMING LANGUAGES | MINIMAL 122 | AVERAGE 225 | EXTENSIVE 385 |
| PREVIOUS EXPERIENCE WITH APPLICATION OF SIMILAR OR GREATER SIZE AND COMPLEXITY | MINIMAL 146 | AVERAGE 221 | EXTENSIVE 410 |
| RATIO OF AVERAGE STAFF SIZE TO DURATION (PEOPLE/MONTH) | <0.5 305 | 0.5-0.9 310 | >0.9 173 |
| HARDWARE UNDER CONCURRENT DEVELOPMENT | NO 297 | | YES 177 |
| DEVELOPMENT COMPUTER ACCESS, OPEN UNDER SPECIAL REQUEST | 0% 226 | 1 - 25% 274 | >25% 357 |
| DEVELOPMENT COMPUTER ACCESS, CLOSED | 0 - 10% 303 | 11 - 85% 251 | >85% 170 |
| CLASSIFIED SECURITY ENVIRONMENT FOR COMPUTER AND 25% OF PROGRAMS AND DATA | NO 289 | | YES 155 |

## EFFECT OF STRUCTURED PROGRAMMING PRACTICES ON PRODUCTIVITY★

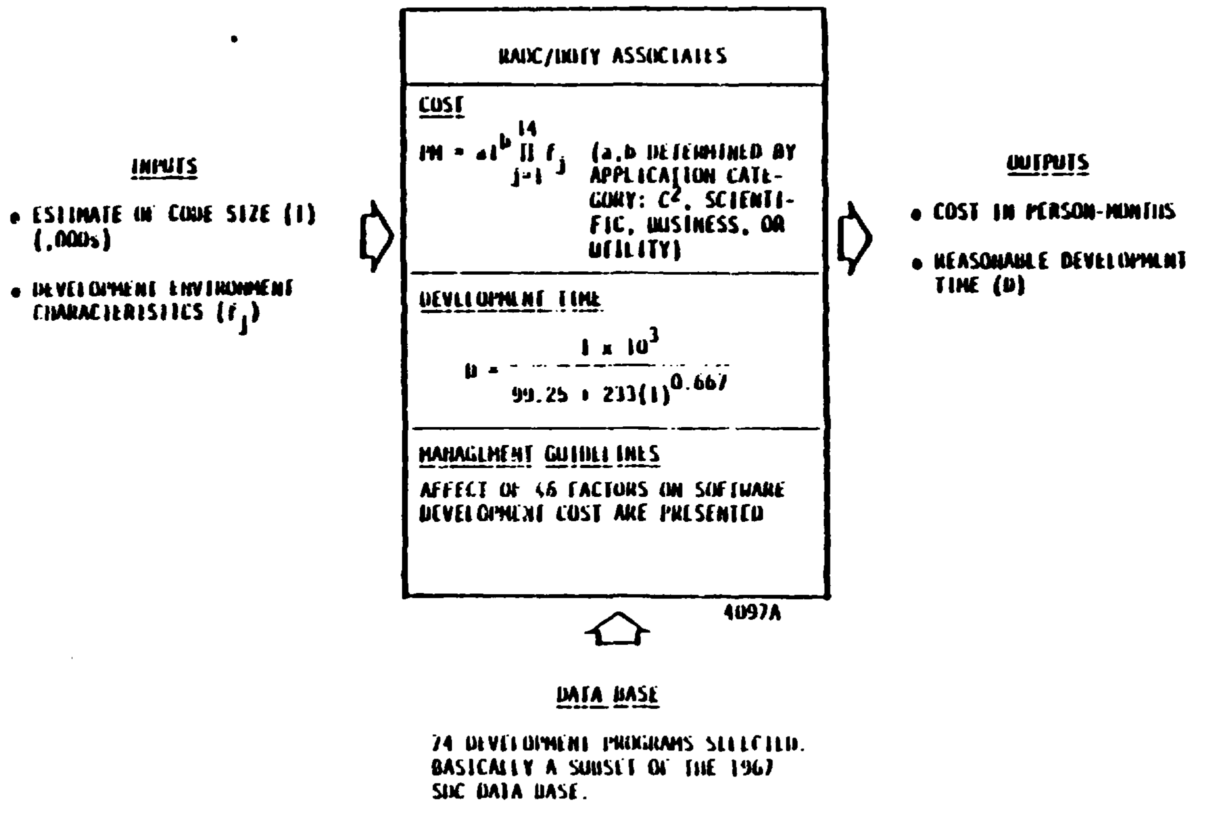
6157



DELIVERED SOURCE INSTRUCTIONS/PERSON-MONTH
(INCLUDING COMMENTS)

★WALSTON AND FELIX, IBM, 1977

## RADC/DOTY ASSOCIATES COST ESTIMATING RELATIONSHIPS



<u>INPUTS</u>

- ESTIMATE OF CODE SIZE (I) (,000s)
- DEVELOPMENT ENVIRONMENT CHARACTERISTICS ($f_j$)

RADC/DOTY ASSOCIATES

<u>COST</u>

$PM = aI^b \prod_{j=1}^{14} f_j$ (a,b DETERMINED BY APPLICATION CATEGORY: $C^2$, SCIENTIFIC, BUSINESS, OR UTILITY)

<u>DEVELOPMENT TIME</u>

$$D = \frac{1 \times 10^3}{99.25 + 233(I)^{0.667}}$$

<u>MANAGEMENT GUIDELINES</u>

AFFECT OF 46 FACTORS ON SOFTWARE DEVELOPMENT COST ARE PRESENTED

4097A

<u>OUTPUTS</u>

- COST IN PERSON-MONTHS
- REASONABLE DEVELOPMENT TIME (D)

<u>DATA BASE</u>

74 DEVELOPMENT PROGRAMS SELECTED. BASICALLY A SUBSET OF THE 1967 SDC DATA BASE.

# RADC/DOTY ASSOCIATES MODEL

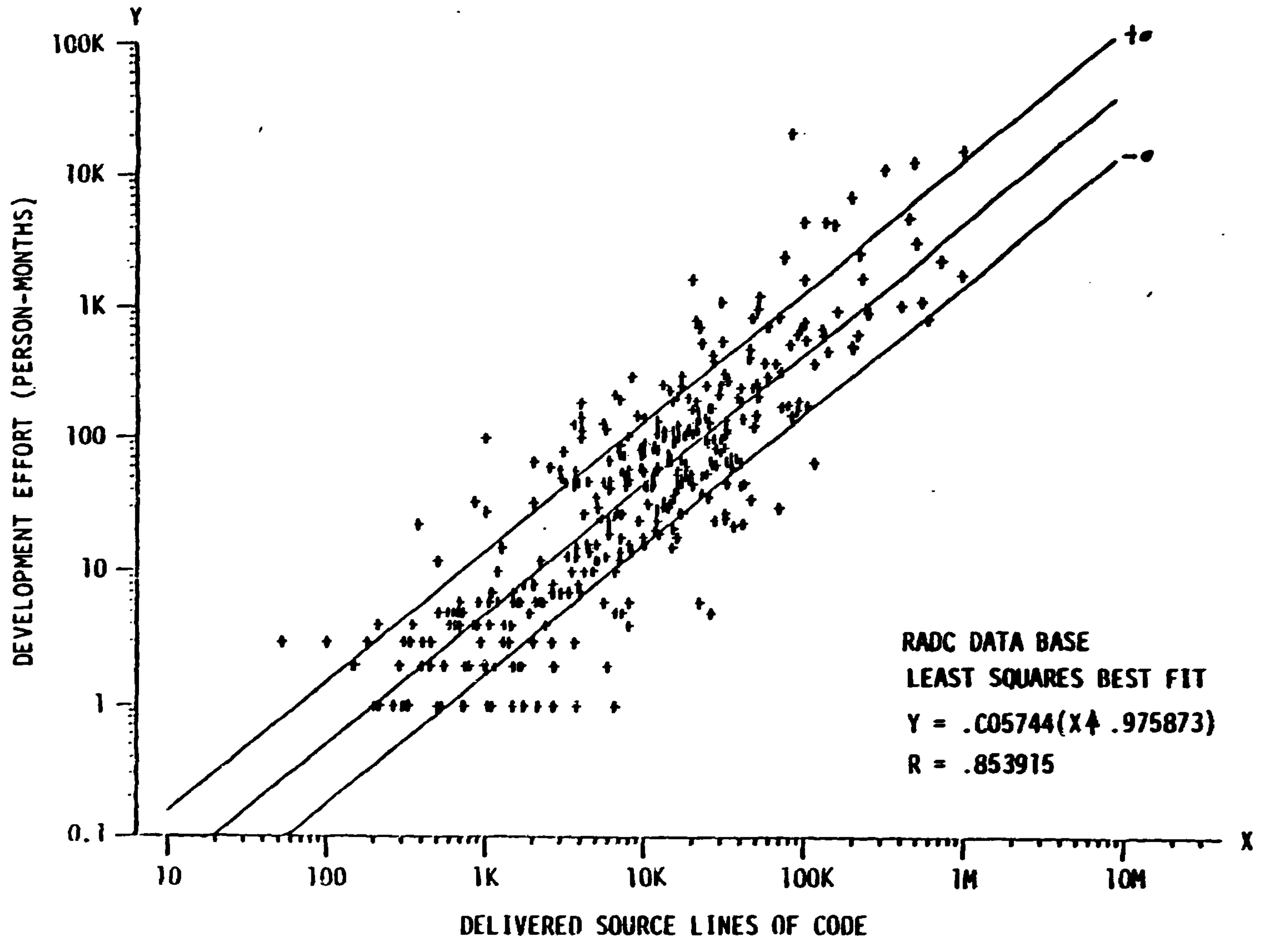
Software Development Resource Estimating
Algorithms Reflecting Development Environment

| ESTIMATOR FOR PM: / FACTOR | | APPLICATIONS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ALL | | COMMAND & CONTROL | | SCIENTIFIC | | BUSINESS | | UTILITY | |
| | | $2.060\ I^{1.047} \prod_{j=1}^{j=14} f_j$ | | $0.501\ I^{1.263} \prod_{j=1}^{j=14} f_j$ | | $2.011\ I^{1.019} \prod_{j=1}^{j=14} f_j$ | | $3.742\ I^{0.781} \prod_{j=1}^{j=14} f_j$ | | $1.744\ I^{0.811} \prod_{j=1}^{j=14} f_j$ | |
| | | YES | NO | YES | NO | YES | NO | YES | NO | YES | NO |
| SPECIAL DISPLAY | $f_1$ | 1.11 | 1.00 | 1.11 | 1.00 | 1.11 | 1.00 | 1.43 | 1.00 | 1.00 | 1.00 |
| DETAILED DEFINITION OF OPERATIONAL REQUIREMENTS | $f_2$ | 1.00 | 1.11 | 1.00 | 1.54 | 1.00 | 2.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| CHANGE TO OPERATIONAL REQUIREMENTS | $f_3$ | 1.05 | 1.00 | 1.05 | 1.00 | 1.05 | 1.00 | 1.05 | 1.00 | 1.05 | 1.00 |
| REAL TIME OPERATION | $f_4$ | 1.33 | 1.00 | 1.33 | 1.00 | 1.67 | 1.00 | 1.00 | 1.00 | 1.43 | 1.00 |
| CPU MEMORY CONSTRAINT | $f_5$ | 1.43 | 1.00 | 1.25 | 1.00 | 1.25 | 1.00 | 1.00 | 1.00 | 1.18 | 1.00 |
| CPU TIME CONSTRAINT | $f_6$ | 1.33 | 1.00 | 1.51 | 1.00 | 1.67 | 1.00 | 1.00 | 1.00 | 2.32 | 1.00 |
| FIRST S/W DEVELOPED ON CPU | $f_7$ | 1.92 | 1.00 | 1.92 | 1.00 | 1.92 | 1.00 | 1.92 | 1.00 | 1.92 | 1.00 |
| CONCURRENT DEVELOPMENT OF ADP H/W | $f_8$ | 1.82 | 1.00 | 1.67 | 1.00 | 2.22 | 1.00 | 1.33 | 1.00 | 1.25 | 1.00 |
| TIME SHARE, VIS-A-VIS BATCH PROCESSING IN DEVELOPMENT | $f_9$ | 0.83 | 1.00 | 0.83 | 1.00 | 0.83 | 1.00 | 0.83 | 1.00 | 0.83 | 1.00 |
| DEVELOPER USING COMPUTER AT ANOTHER FACILITY | $f_{10}$ | 1.43 | 1.00 | 1.43 | 1.00 | 1.43 | 1.00 | 1.43 | 1.00 | 1.43 | 1.00 |
| DEVELOPMENT AT OPERATIONAL SITE | $f_{11}$ | 1.39 | 1.00 | 1.39 | 1.00 | 1.39 | 1.00 | 1.39 | 1.00 | 1.39 | 1.00 |
| DEVELOPMENT COMPUTER DIFFERENT THAN TARGET COMPUTER | $f_{12}$ | 1.25 | 1.00 | 2.22 | 1.00 | 1.11 | 1.00 | 1.00 | 1.00 | 1.43 | 1.00 |
| DEVELOPMENT AT MORE THAN ONE SITE | $f_{13}$ | 1.25 | 1.00 | 1.25 | 1.00 | 1.75 | 1.00 | 1.25 | 1.00 | 1.21 | 1.00 |
| PROGRAMMER ACCESS TO COMPUTER | $f_{14}$ | LIM: 1.00 / UNLIM: 0.90 | | 1.00 / 1.00 | | 1.00 / 0.67 | | 1.00 / 0.90 | | 1.00 / 0.67 | |

I in thousands of lines

# TYPICAL DEVELOPMENT EFFORT

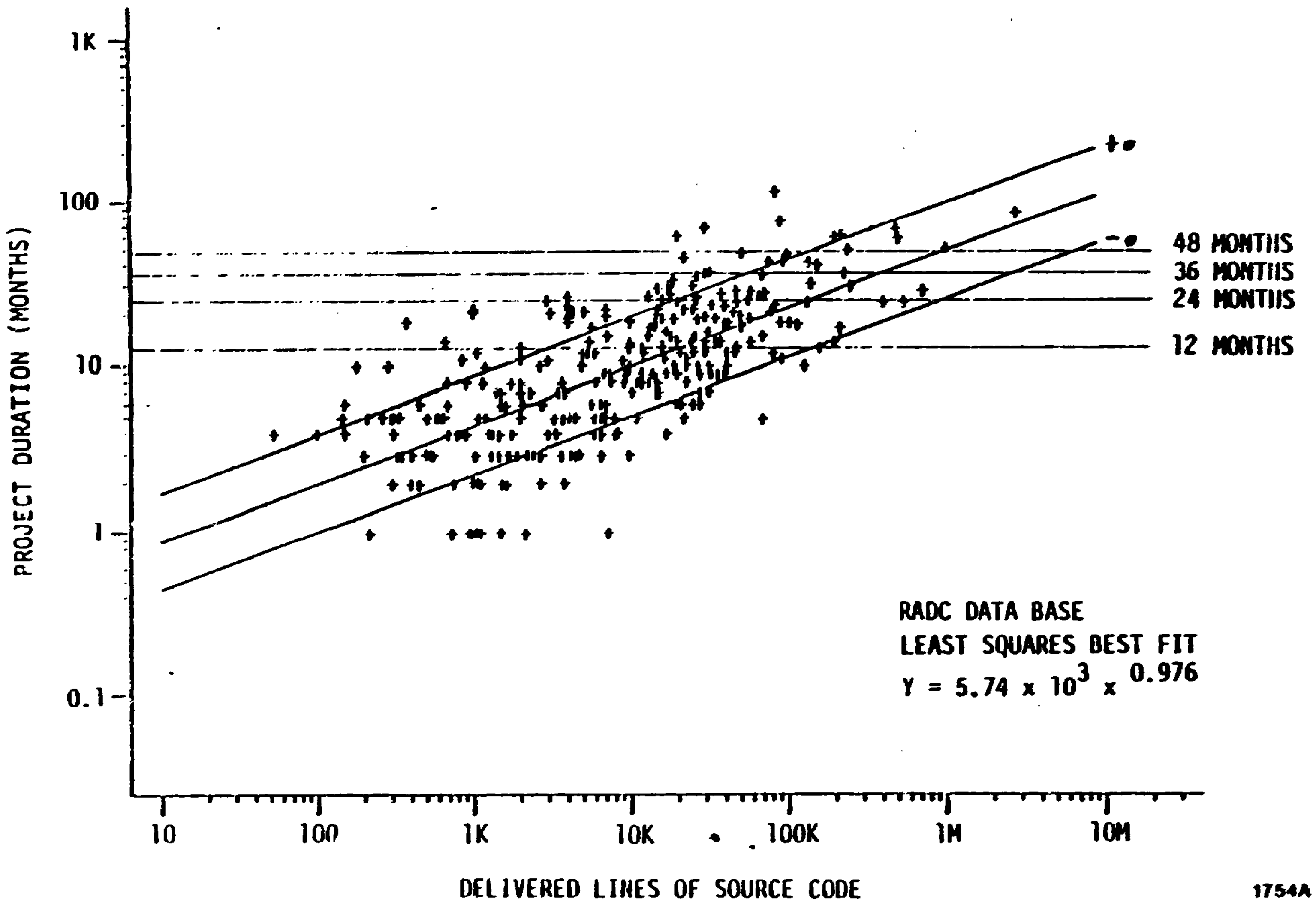
6157



17548

6157

# TYPICAL PROJECT DURATION



# PUTNAM MODEL

**INPUTS**

- LINES OF SOURCE CODE ($S_s$)
- TECHNOLOGY CONSTANT ($C_K$)
- DEVELOPMENT TIME ($t_d$)

PUTNAM MODEL

EXPENDITURE RATE

$$Y' = \frac{K}{t_d^2} t e^{-t^2/2t_d^2}$$

WHERE:

$Y'$ = EXPENDITURE RATE

$K$ = TOTAL LIFE CYCLE EFFORT

$t_d$ = DEVELOPMENT TIME

$t$ = TIME

SYSTEM SIZE

$$S_s = C_K K^{1/3} t_d^{4/3}$$

$$S_s = 2.49 \cdot \overline{PR} \cdot K/6$$

WHERE:

$S_s$ = DELIVERED SOURCE INSTRUCTIONS

$C_K$ = TECHNOLOGY CONSTANT

$\overline{PR}$ = AVERAGE PRODUCTIVITY RATE

$$f_i(K,t_d) = \alpha_i X_1 + \beta_i X_2 + \gamma_i X_3$$

WHERE:

$X_1$ = NUMBER OF FILES

$X_2$ = NUMBER OF REPORTS

$X_3$ = NUMBER OF APPLICATION PROGRAMS

4097D

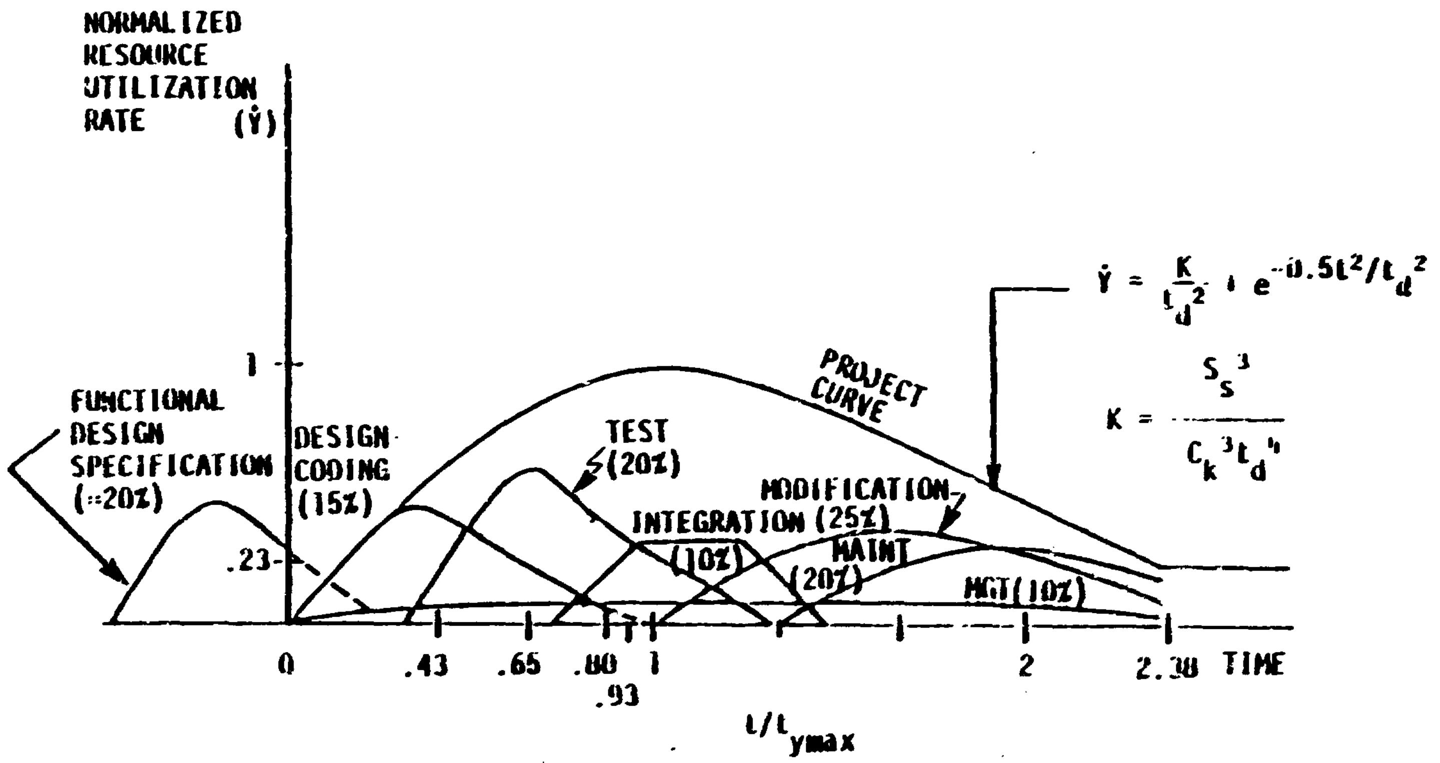
**OUTPUTS**

- RESOURCE EXPENDITURE PROFILE (Y')
- TOTAL LIFE CYCLE EFFORT (Y)
- DEVELOPMENT EFFORT (≈0.4Y)
- COST-SCHEDULE TRADEOFFS

FORTY SOFTWARE DEVELOPMENT PROJECTS AT U.S. ARMY COMPUTER SYSTEMS COMMAND. VERIFIED AGAINST DATA FROM 150 OTHER SOFTWARE DEVELOPMENTS.

# BASIC PUTNAM MODEL LIFE CYCLE CURVE

6157



## Inputs

- LINES OF SOURCE CODE, $S_s$
- DEVELOPMENT TIME, td
- TECHNOLOGY CONSTANT, $C_k$

## Outputs

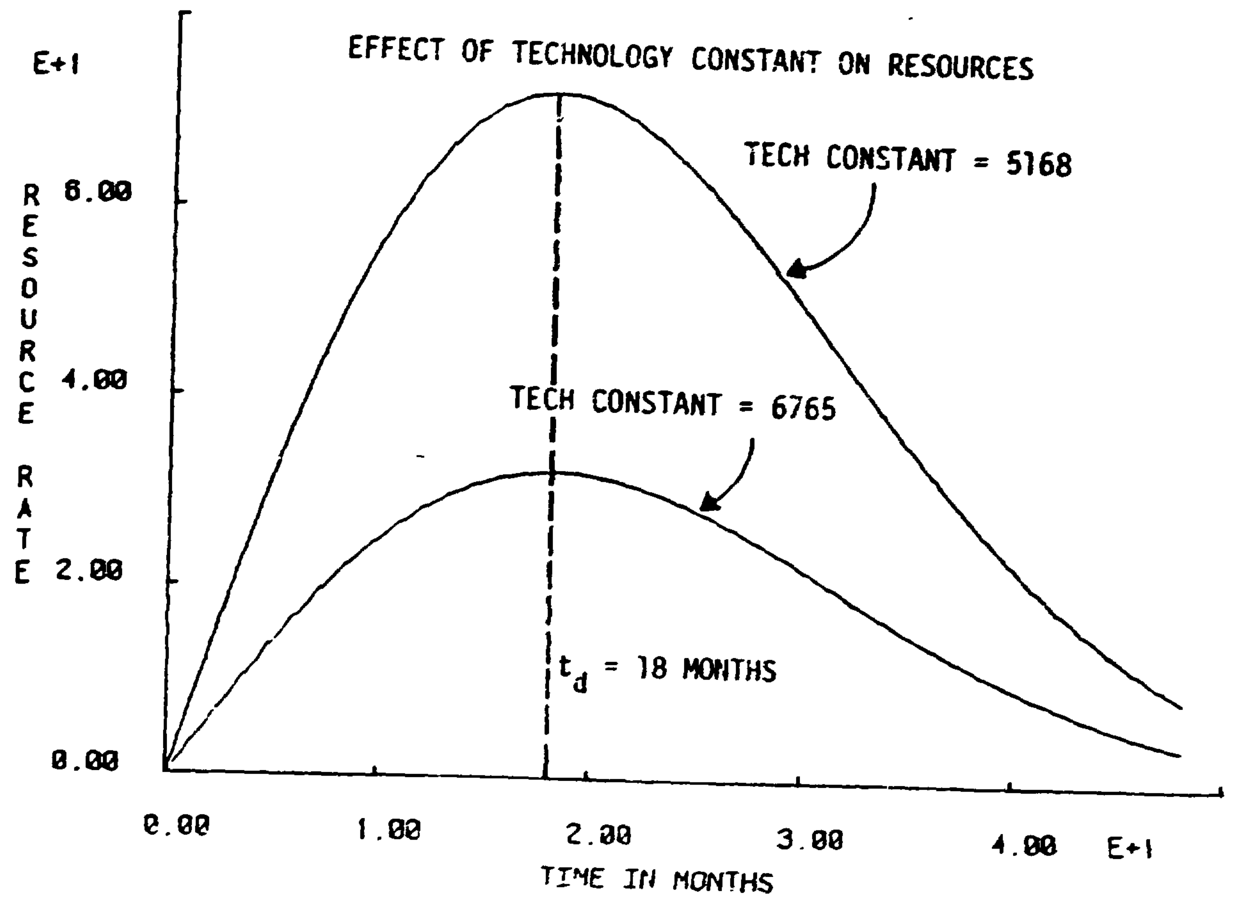
- DISTRIBUTION OF EFFORT, $\dot{Y}$
- TOTAL LIFE CYCLE EFFORT, K

(SUBSTANTIALLY MORE INPUTS AND OUTPUTS AVAILABLE IN THE COMPUTERIZED VERSION, SLIM)

ISP

6157

## PUTNAM MODEL SENSITIVITY



## KNOWLEDGE GAINED - PUTNAM MODEL

### *Calibration*

- CALIBRATION IS BASICALLY A PROCESS OF DETERMINING TECHNOLOGY CONSTANT AND ORGANIZATION'S CONSTRAINTS

### *Input Data*

- USE OF HIGH ORDER LANGUAGE STATEMENT COUNT IS PREFERABLE TO THE PRICE-S APPROACH
- TECHNOLOGY CONSTANT IS ABSTRACT AND NOT EASILY ESTABLISHED

### *Model Characteristics*

- ATTEMPTS TO MODEL "OPTIMUM" RESOURCE ALLOCATION
- MILESTONE PLACEMENT NOT CHARACTERISTICS OF WHAT WE SEE IN OUR DATA
- RESOURCE ALLOCATION NOT CHARACTERISTIC OF CONTRACTUAL ENVIRONMENT
- EXTREMELY SENSITIVE TO DEVELOPMENT TIME

### *Output Data*

- RESULTS MUST BE MANUALLY ALLOCATED TO APPROPRIATE SEGMENTS OF THE WORK
- COMPUTERIZED VERSION (SLIM) PROVIDES DATA IN A FORMAT VERY SIMILAR TO THAT PROVIDED BY PRICE-S

## RCA PRICE-S SOFTWARE COST MODEL

6157

### *Inputs*

- MACHINE LEVEL INSTRUCTION COUNT
- APPLICATION DESCRIPTION
  - INSTRUCTION MIX
  - PERCENT NEW DESIGN
  - PERCENT NEW CODE
- SYSTEM CONFIGURATION
- SCHEDULE
- ENVIRONMENTAL FACTORS
- RUN/REPORT OPTIONS

### *PRICE-S*

PROPRIETARY MODEL OF HOW EXPERIENCED MANAGERS AND COST ESTIMATORS PREPARE SOFTWARE DEVELOPMENT COST ESTIMATES. CERs PROVIDED TO CONVERT PARAMETRIC DATA TO COST. NOT BASED ON CERs DETERMINED FROM REGRESSION ANALYSIS OF HISTORICAL DATA.

NORMAL MODE

CALCULATES EXPECTED COST FROM INPUTS SUPPLIED

ECIRP MODE

CALIBRATES MODEL CONSTANTS USING SIZE AND COST DATA FROM PREVIOUS DEVELOPMENT

DESIGN-TO-COST MODE

CALCULATES SIZE GIVEN TARGET COST

INTEGRATION MODE

CALCULATES COST OF AN INDEPENDENT TEST AND INTEGRATION ACTIVITY OFTEN REQUIRED IN LARGE DEVELOPMENTS.

OPTIONS

SENSITIVITY ANALYSIS, MONTHLY EXPENDITURE PROFILE, SCHEDULE PENALTY ASSESSMENT.

4097G

### *Outputs*

- COST BY PHASE AND BY COST ELEMENT

  PHASES

  DESIGN
  IMPLEMENTATION
  TEST AND INTEGRATION

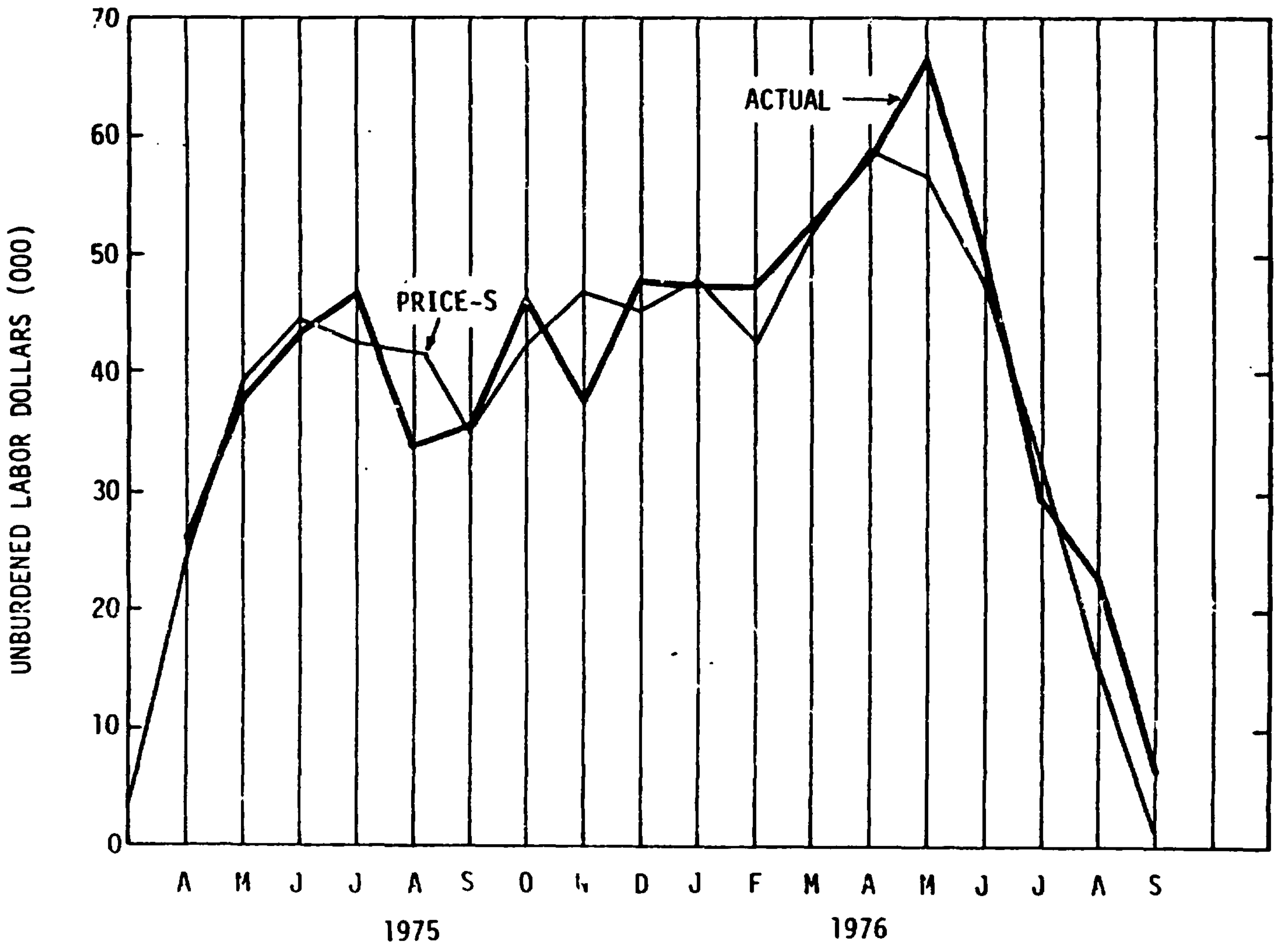
  COST ELEMENTS

  SYSTEM ENGINEERING
  PROGRAMMING
  CONFIGURATION CONTROL
  DOCUMENTATION
  PROGRAM MANAGEMENT
- MONTHLY COST PROFILE
- SCHEDULE (WITH PHASE OVERLAP)
- RESULTS OF SENSITIVITY ANALYSIS AND SCHEDULE IMPACT ANALYSIS

ISP

6157

## CALIBRATING PRICE-S



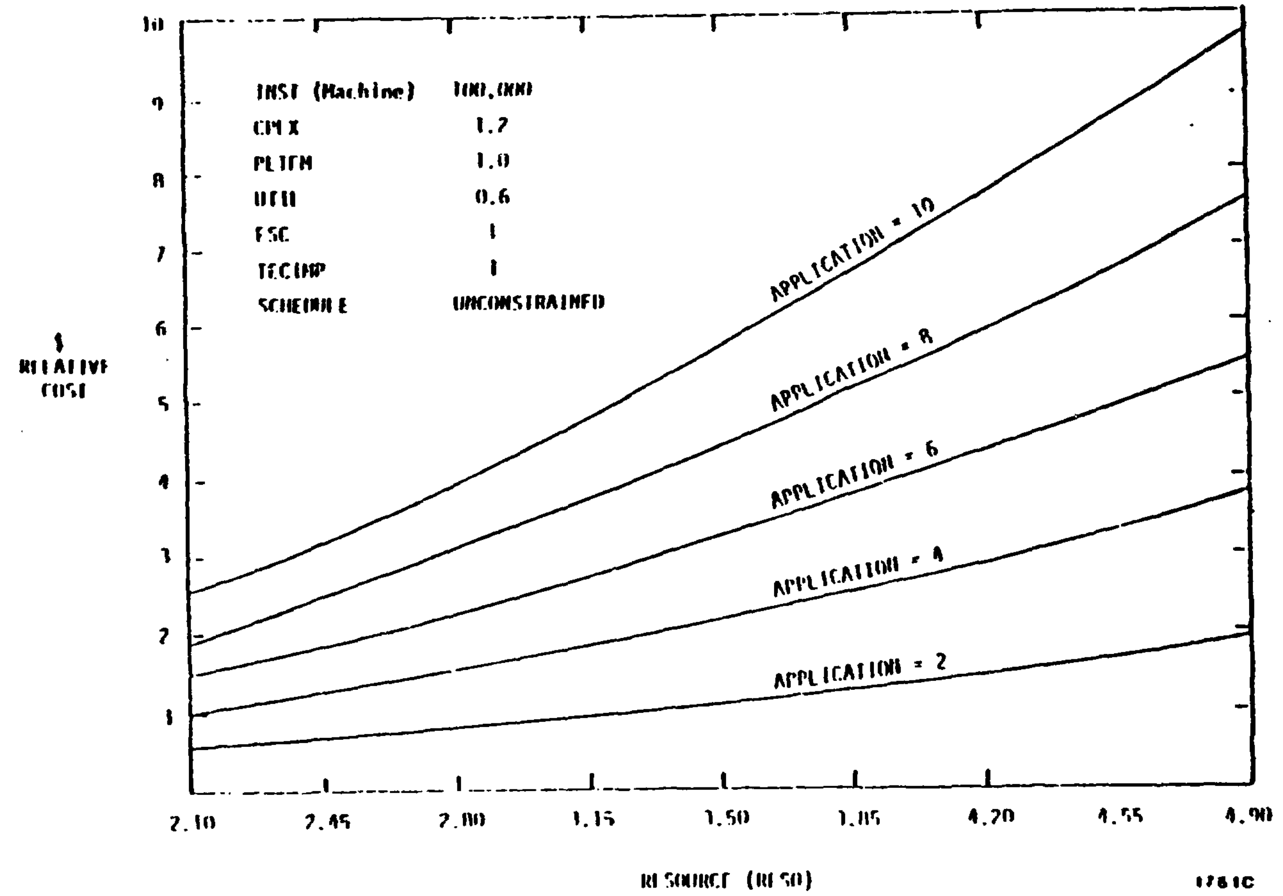
1240G

ISP

6157

## PRICE-S SENSITIVITY

EFFECT OF RESOURCE AND APPLICATION ON COST



APPLICATION SUMMARIZES THE MIX OF INSTRUCTIONS. VALUES TOWARDS THE LOWER END ARE PREDOMINANTLY MATH AND STRING MANIPULATION WHILE TOWARD THE HIGHER END REPRESENT EMPHASIS ON SYSTEMS, REAL-TIME COMMAND AND CONTROL, AND INTER-ACTIVE OPERATIONS.

RESOURCE RELATES TO AN ORGANIZATIONS WAY OF DOING BUSINESS i.e., SKILL LEVEL, EXPERIENCE, PRODUCTIVITY, EFFICIENCY, LABOR RATES (FACTORS WHICH DON'T CHANGE RAPIDLY).

ISP

6157

## KNOWLEDGE GAINED - PRICE-S MODEL

### Calibration

- SEVERAL LEVELS OF CALIBRATION ARE POSSIBLE (AND USEFUL) DEPENDING ON THE THE QUANTITY AND QUALITY OF HISTORICAL DATA. OVER CALIBRATION (TUNING) IS POSSIBLE AND MUST BE AVOIDED.

### Input Data

- THE USE OF MACHINE LEVEL INSTRUCTION COUNT INSTEAD OF HIGH ORDER LANGUAGE STATEMENT COUNT CREATES PROBLEMS IN DETERMINING A VALUE TO USE FOR THE SIZING INPUT
- THERE ARE SEVERAL ABSTRACT INPUT VARIABLES, e.g., RESOURCE, COMPLEXITY, INSTRUCTION MIX, APPLICATION, AND PLATFORM, WHICH ARE DIFFICULT TO ESTIMATE

### Model Characteristics

- MODEL ASSUMES OVERLAPPING DEVELOPMENT PHASES WHICH AREN'T APPROPRIATE IN MANY SITUATIONS
- RESULTS ARE AFFECTED BY THE WAY THE MODEL IS APPLIED

### Output Data

- DISTRIBUTION OF RESOURCES IS ALMOST ALWAYS BIMODAL
- USUALLY NECESSARY TO CONVERT OUTPUT DATA TO DIFFERENT FORM

## GRC MODEL

**INPUTS**

- LINES OF OBJECT INSTRUCTIONS

**GENERAL RESEARCH CORPORATION**

DISAGGREGATED MODE

ANALYSIS AND DESIGN:

$$\ln Y = -1.17 + 1.03 \ln X_1 + 1.73 X_2$$

CODING:

$$\ln Y = -2.3 + 1.13 \ln X_1 + 2.12 X_2$$

INTEGRATION & TEST:

$$\ln Y = -0.86 + 0.934 \ln X_1 + 1.64 X_2$$

WHERE:

$Y$ = PERSON-YEARS

$X_1$ = 1.7 x DELIVERED NUMBER OF OBJECT INSTRUCTIONS (ADJUSTED TO 32-BIT WORD LENGTH)

$X_2$ = 1 IF DEVELOPMENT IS HARDWARE CONSTRAINED
0 OTHERWISE

AGGREGATED MODEL

A PROCEDURE USING SEGMENTS OF COST ESTIMATING EFFORTS FOUND IN THE LITERATURE (SEE NARRATIVE FOR DESCRIPTION)

**OUTPUTS**

- PERSON-YEARS FOR DESIGN, CODING, AND TEST AND INTEGRATION PHASES

4097C

DATA BASE

SEVEN PROJECTS FROM PARMIS. ALSO USED TO SOME DEGREE HISTORICAL DATA FROM GRC, ADREP, IBM, AND SDC

ISP

# GRC MODEL

6157

Productivity

| SOFTWARE TYPE | PRODUCTIVITY (INSTRUCTIONS/ PERSON-MONTH) |
|---|---|
| MATHEMATICAL OPERATIONS | 166 |
| REPORT GENERATION | 125 |
| LOGIC OPERATIONS | 83 |
| SIGNAL PROCESSING OR DATA REDUCTION | 50 |
| REAL-TIME OR EXECUTIVE FUNCTIONS | 25 |

Language Productivity

| | LOW | AVERAGE | HIGH |
|---|---|---|---|
| FORTRAN | 3.3 | 4.5 | 5.7 |
| COBOL | 4.5 | 5.8 | 7.2 |
| JOVIAL | 4.0 | 5.7 | 9.8 |
| MOL* | 6.6 | 8.1 | 9.7 |

*INSTRUCTIONS/PERSON-DAY

Productivity Table

| DIFFICULTY \ DURATION | 6-12 MONTHS | 12-24 MONTHS | MORE THAN 24 MONTHS |
|---|---|---|---|
| EASY (VERY FEW INTERACTIONS) | 20 | 500 | 10,000 |
| MEDIUM (SOME INTERACTIONS) | 10 | 250 | 5,000 |
| HARD (MANY INTERACTIONS) | 5 | 125 | 1,500 |
| UNITS | INSTRUCTIONS PER PERSON-DAY | INSTRUCTIONS PER PERSON-MONTH | INSTRUCTIONS PER PERSON-YEAR |

## BOEING COMPUTER SERVICES MODEL

INPUTS

- NUMBER OF STATEMENTS
- STATEMENT MIX
- EXPENDITURE DISTRIBUTION
- ENVIRONMENTAL FACTORS

BOEING COMPUTER SERVICES

BASIC ALGORITHM

| | |
|---|---|
| MATHEMATICAL | 6 PM/1000 STMTS |
| REPORT | 8 PM/1000 STMTS |
| LOGIC | 12 PM/1000 STMTS |
| SIGNAL | 20 PM/1000 STMTS |
| REAL TIME | 40 PM/1000 STMTS |

EXPENDITURE DISTRIBUTION

| | | |
|---|---|---|
| REQUIREMENTS | 5% | (TYPICAL VALUES) |
| DESIGN | 25% | |
| CODE | 10% | |
| CHECKOUT | 25% | |
| INTEG & TEST | 25% | |
| SYSTEM TEST | 10% | |

ADJUSTMENT FACTORS

MULTIPLIERS WHICH ARE APPLIED TO INDIVIDUAL TASK EFFORTS TO ACCOUNT FOR SPECIAL ENVIRONMENTAL FACTORS

COMPUTER TIME

COMPUTER RESOURCE UNITS/PERSON-MONTH BASED UPON TYPE OF SOFTWARE

409711

DATA BASE - UNKNOWN

OUTPUTS

- TOTAL DEVELOPMENT EFFORT BY PHASE (PERSON-MONTHS)
- COMPUTER RESOURCE UNITS (COMPUTER TIME)

# BOEING COMPUTER SERVICES MODEL

Labor Estimate Adjustment Factors

| | REQUIREMENTS DEFINITION | DESIGN AND SPECIFICATION | CODE PREPARATION | CODE CHECKOUT | INTEGRATION AND TEST | SYSTEM TEST |
|---|---|---|---|---|---|---|
| 1. REIMPLEMENTATION OF EXISTING SOFTWARE | 0.2 | 0.2 | 0.8 | | 0.8 | |
| 2. FOLLOW-ON CONTRACT WITH CURREN, CUSTOMER | 0.7 | | | | | 0.9 |
| 3. NUMBER OF PROGRAMMERS (INTERPOLATE BETWEEN VALUES IF NEEDED: | | | | | | |
| 1 TO 2 | 0.2 | 0.5 | 0.8 | | 0.2 | |
| 6 TO 10 | 1.0 | 1.0 | 1.0 | | 1.0 | 1.0 |
| MORE THAN 20 | 6.0 | 3.3 | 1.2 | | 3.0 | 3.0 |
| 4. HIGHER-ORDER LANGUAGE (SEASONED COMPILER) | | 0.3 | 0.3 | 0.2 | | |
| 5. MACROLANGUAGE | | | | | | |
| - IN CODING | | 0.9 | 0.9 | 0.9 | | |
| - FORMS FOR DOCUMENT | | 0.8 | | | 0.8 | |
| 6. ON-LINE CODE/DATA ENTRY | | | 0.9 | 0.9 | | |
| 7. ON-LINE DEBUGGING | | | | 0.6 | | |
| 8. POOR (OR NO) DEBUG TOOLS EXCEPT DUMPS | | | | 1.4 | 1.4 | |
| 9. PROGRAMMING EXPERIENCE WITH ENGINEERING/TECHNICAL DISCIPLINE OF APPLICATION: | | | | | | |
| ENTRY-LEVEL | 2.0 | 3.0 | 1.5 | | | 1.5 |
| MODERATE | 1.0 | 1.0 | 1.0 | | | 1.0 |
| HIGH | 0.6 | 0.5 | 0.8 | | | 0.7 |

NOTE: IN MATRIX POSITIONS HAVING NO ENTRY, THE ASSUMED MULTIPLIER IS 1.0.

ISP

6157

## TRW/WOLVERTON MODEL

**INPUTS**

- NUMBER OF OBJECT INSTRUCTIONS
- FUNCTIONS
  - CONTROL
  - INPUT/OUTPUT
  - PRE/POST PROCESSOR
  - ALGORITHM
  - DATA MANAGEMENT
  - TIME CRITICAL
- NEWNESS
- DIFFICULTY
- LANGUAGE

TRW/WOLVERTON

- CATEGORIZE MODULES BY FUNCTION
- ESTIMATE MODULE SIZE
- DETERMINE LIFE CYCLE PHASES TO USE
- DETERMINE COST ELEMENTS
- USE COST PER INSTRUCTION FOR EACH FUNCTIONAL TYPE
- SPREAD COST BY ELEMENT OVER LIFE CYCLE

409/F

TRW PROPRIETARY DATA

**OUTPUTS**

DEVELOPMENT COST SPREAD OVER A 25 X 7 MATRIX -- COST ELEMENT BY LIFE CYCLE PHASE

ISP

6157

# TRW/WOLVERTON MODEL

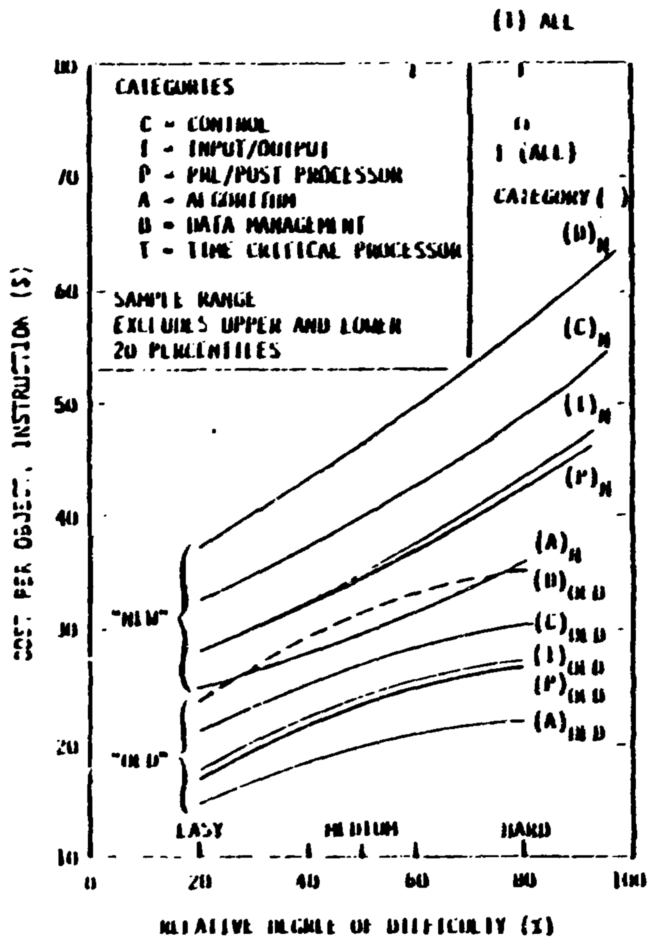
Activities as a Function of Software Development Phase

| ACTIVITY | | DEVELOPMENT PHASE | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | A | B | C | D | E | F | G | H |
| | | PERFORMANCE AND DESIGN REQUIREMENTS | IMPLEMENTATION CONCEPT AND TEST PLAN | INTERFACE AND DATA REQUIREMENTS SPECIFICATION | DETAILED DESIGN SPECIFICATION | CODING AND AUDITING | SYSTEM TESTING | CERTIFICATION AND ACCEPTANCE | OPERATIONS AND MAINTENANCE |
| MANAGEMENT | 1 | PROGRAM MANAGEMENT | PROGRAM MANAGEMENT | PROGRAM MANAGEMENT | PROGRAM MANAGEMENT | PROGRAM MANAGEMENT | PROGRAM MANAGEMENT | PROGRAM MANAGEMENT | PROGRAM MANAGEMENT |
| | 2 | PROGRAM CONTROL | PROGRAM CONTROL | PROGRAM CONTROL | PROGRAM CONTROL | PROGRAM CONTROL | PROGRAM CONTROL | PROGRAM CONTROL | PROGRAM CONTROL |
| REVIEWS | 3 | | PRELIMINARY DESIGN REVIEW (PDR) | INTERFACE DESIGN REVIEW (IDR) | CRITICAL DESIGN REVIEW (CDR) | SYSTEM TEST REVIEW (STR) | | ACCEPTANCE TEST REVIEW (ATR) | OPERATIONAL CRITIQUES |
| DOCUMENTS | 4 | DOCUMENT AND EDIT | DOCUMENT AND EDIT | DOCUMENT AND EDIT | DOCUMENT AND EDIT | DOCUMENT AND EDIT | | DOCUMENT AND EDIT | DOCUMENT AND EDIT |
| | 5 | REPRODUCTION | REPRODUCTION | REPRODUCTION | REPRODUCTION | REPRODUCTION | | REPRODUCTION | REPRODUCTION |
| REQUIREMENTS AND SPECIFICATIONS | 6 | REQUIREMENTS DEFINITION | PDR MATERIAL | EVENT GENERATION INTERFACE | PRODUCT CONFIG DETAILED TECH DESCRIPTION (PART II) (WITHOUT LISTINGS) | TECHNICAL DESCRIPTION UPDATE | PRODUCT CONFIG DETAILED TECH DESCRIPT UPDATE (PART II) (WITH LISTINGS) | REQUIREMENTS CERTIFICATION | SOFTWARE PROBLEM REPORTS (SPR) |
| | 7 | REQUIREMENTS ALLOCATION | PERFORMANCE AND DESIGN REQUIREMENTS (PART I) | COMMAND DEFINITION I/F | | TRAINING DOCUMENTATION | | FINAL DOCUMENTATION UPDATE (PART II) | |
| | 8 | | | TELEMETRY DEFINITION I/F | | | INTERFACE SPECIFICATIONS UPDATE | | |
| | 9 | | | OPERATIONAL ENVIRONMENT I/F | | | | | |
| DESIGN | 10 | TRADE STUDIES | TRADE STUDIES | TRADE STUDIES | TRADE STUDIES | | | | |
| | 11 | INTERFACE REQUIREMENTS | FUNCTIONAL DEFINITION | DATA DEFINITIONS | ALGORITHM DESIGN | ALGORITHM UPDATE | PROGRAM UPDATE | | |
| | 12 | HUMAN INTERACTION | STORAGE AND TIMING ALLOCATION | | PROGRAM DESIGN | | | | |
| | 13 | STANDARDS AND CONVENTIONS | DATA BASE DEFINITION | DATA BASE DESIGN | | DATA BASE UPDATE | DATA BASE UPDATE | | |
| | 14 | | SOFTWARE OVERVIEW (PRELIMINARY) | | SOFTWARE OVERVIEW UPDATE | | | | |
| CODING | 15 | | | | PROTOTYPE CODING | OPERATIONAL CODING | | | |
| TESTING, CONFIGURATION CONTROL AND QRA | 16 | | PRODUCT AND CONFIGURATION CONTROL | PRODUCT AND CONFIGURATION CONTROL | PRODUCT AND CONFIGURATION CONTROL | PRODUCT AND CONFIGURATION CONTROL | PRODUCT AND CONFIGURATION CONTROL | PRODUCT AND CONFIGURATION CONTROL | PRODUCT AND CONFIGURATION CONTROL |
| | 17 | | DATA BASE CONTROL | DATA BASE CONTROL | DATA BASE CONTROL | DATA BASE CONTROL | DATA BASE CONTROL | DATA BASE CONTROL | DATA BASE CONTROL |
| | 18 | TEST REQUIREMENTS | TEST PLANS | INTERFACES | TEST PROCEDURES | DEVELOPMENT TEST PLANNING | SYSTEM TEST PLANNING | ACCEPTANCE AND TEST PLANNING | INTEGRATION TESTING |
| | 19 | | | | | DEVELOPMENT TEST | SOFTWARE SYSTEM TEST | ACCEPTANCE DEMONSTRATION | SPR CLOSURE |
| | 20 | | | | | | HARDWARE/SOFTWARE SYSTEM TESTING | | |
| | 21 | ACCEPTANCE TEST REQUIREMENTS | QUALITY AND RELIABILITY ASSURANCE PLANS | QA AND RA MONITORING | QA AND RA MONITORING | Q AND RA MONITORING | Q AND RA MONITORING | Q AND RA MONITORING | Q AND RA MONITORING |
| | 22 | | | | | TEST SUPPORT | TEST SUPPORT | TEST SUPPORT | TEST SUPPORT |
| | 23 | | | | | | | | |
| OPERATIONS | 24 | | OPERATIONAL CONCEPT | OPERATIONAL TIMELINE | OPERATIONAL CONCEPT UPDATE | USER'S MANUAL (PRELIMINARY) | OPERATIONAL TIMELINE UPDATE | USER'S MANUAL UPDATE | INTEGRATION SUPPORT |
| | 25 | | TRAINING PLAN | | TRAINING PLAN UPDATE | TRAINING | TRAINING | TRAINING AND REHEARSAL | TRAINING AND REHEARSAL |

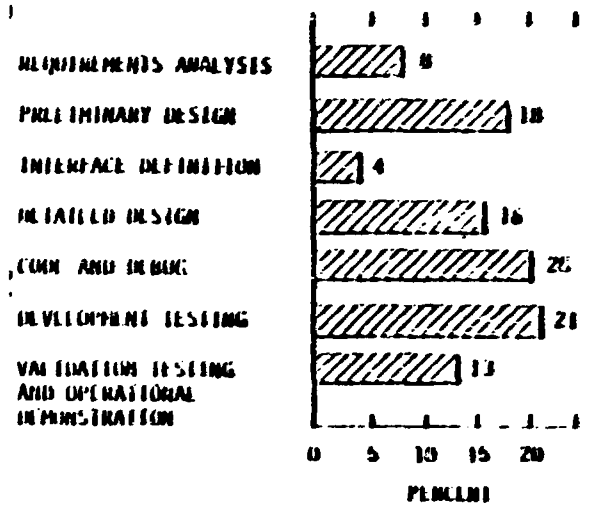
ISP

6157

# TRW/WOLVERTON MODEL



Cost Per Object Instruction vs.
Relative Degree of Difficulty



Typical Allocation of Resources in
Custom Software Development and Test

## COST OF RE-USING SOFTWARE (TRW)

6157

COST OF REVISING SOFTWARE RAPIDLY RESEARCHES THE POINT OF DIMINISHING RETURN

| DESCRIPTION | % OF NOMINAL DEVELOPMENT COST | | | |
|---|---|---|---|---|
| | PRELIM. DESIGN | DETAIL DESIGN | CODE AND UNIT TEST | INTEG. AND TEST |
| MINIMAL INTERNAL CHANGES, CLEAN INTERFACES | 35 | 30 | 20 | 45 |
| SMALL NO. OF INTERFACE AND INTERNAL CHANGES | 70 | 60 | 60 | 85 |
| MODERATE NO. OF INTERFACE AND INTERNAL CHANGES | 90 | 80 | 90 | 95 |
| REDESIGN AND RECODE | 100 | 100 | 100 | 100 |

ISP

## USAF/ESD MODEL

6157

INPUTS

- LINES OF SOURCE INSTRUCTIONS
- APPLICATION TYPE
- FAMILIARITY
- QUALITY
- HARDWARE CONSTRAINTS

USAF/ESD COST FACTORS

BASIC RELATIONSHIP IS LINEAR WITH RESPECT TO THE NUMBER OF DELIVERED SOURCE INSTRUCTIONS, MODIFIED BY OTHER FACTORS AS FOLLOWS:

$ 6-$12/SOURCE INSTRUCTION FOR HOL
$12-$24/SOURCE INSTRUCTION FOR MOL
$30-$60/SOURCE INSTRUCTION FOR REAL TIME

- IF OS, MULTIPLY BY 2.5
- IF UNFAMILIAR WITH APPLICATION, MULTIPLY BY 1.5 - 2.0
- IF MAN-RATED, TEST COSTS ~40% OF TOTAL, OTHERWISE, TEST COSTS ~15% OF TOTAL
- DOCUMENTATION~10% OF TOTAL
- HARDWARE CONSTRAINTS - ASYMPTOTIC AS APPROACH FULL CAPACITY
- SCHEDULE COMPRESSION - COST GOES UP BY SAME PERCENT AS SCHEDULE COMPACTED
- OTHER SUBJECTIVE FACTORS INCLUDE COMPLEXITY, DEVELOPMENT METHODS, PERSONNEL, STABILITY OF REQUIREMENTS, SIZE OF DATA BASE, MANAGEMENT

40971

OUTPUTS

- TOTAL COSTS

DATA BASE

NO REPORTS IN LITERATURE OF THIS TECHNIQUE BEING FORMALLY APPLIED.
IT WAS BASED ON EXPERIENCES OF PERSONNEL AT WORKSHOP.

ISP

6157

## TECOLOTE MODEL

**INPUTS**

- LINES OF MACHINE LANGUAGE INSTRUCTIONS (DELIVERED OR OPERATING)

TECOLOTE RESEARCH, INC.

$$PM = \begin{cases} 2.43\ (D)^{1.18} \\ \text{OR} \\ 2.52\ (O)^{1.24} \end{cases}$$

WHERE:

D = DELIVERED MACHINE LANGUAGE INSTRUCTIONS (000s)

O = OPERATING MACHINE LANGUAGE INSTRUCTIONS (000s)

D INCLUDES SOFTWARE DELIVERED SUCH AS SIMULATORS. TEST PACKAGES, DIAGNOSTICS WHICH ARE NOT PART OF OPERATIONAL SOFTWARE, O

(NOTE: OTHER CERs ESTABLISHED WHICH WERE BASED ON TACTICAL MISSION-RELATED INPUT VARIABLES ARE NOT PRESENTED SINCE THEY ARE NOT GENERALLY APPLICABLE)

4097K

**OUTPUTS**

- TOTAL PERSON-MONTHS

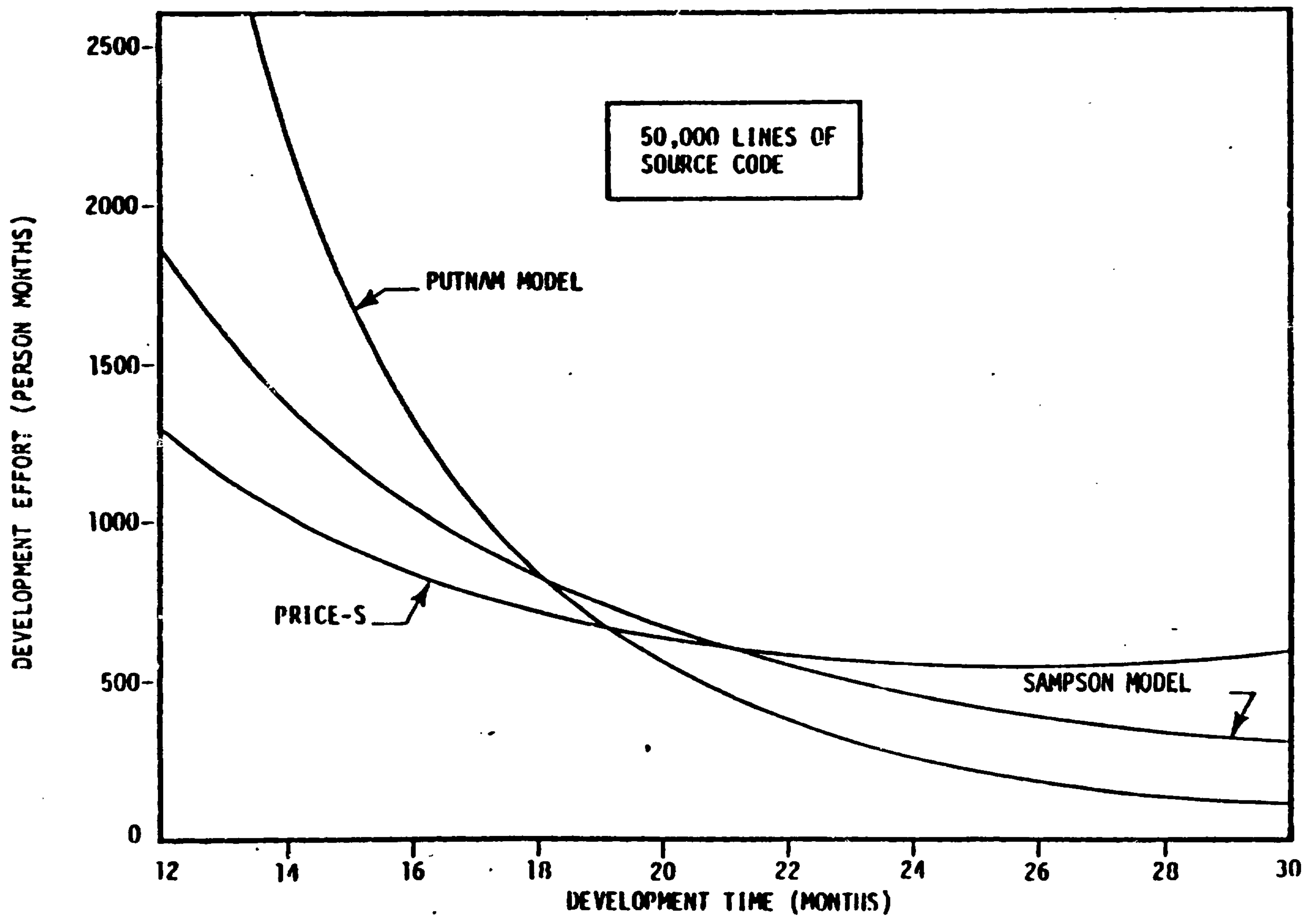
DATA BASE

FIVE DATA POINTS RELATED TO NAVY TACTICAL SYSTEMS WERE USED TO ESTABLISH CERs. WORK DONE FOR CHIEF, NAVAL OPERATIONS.

GENERAL
ELECTRIC

## EFFORT-TIME TRADEOFF

space division

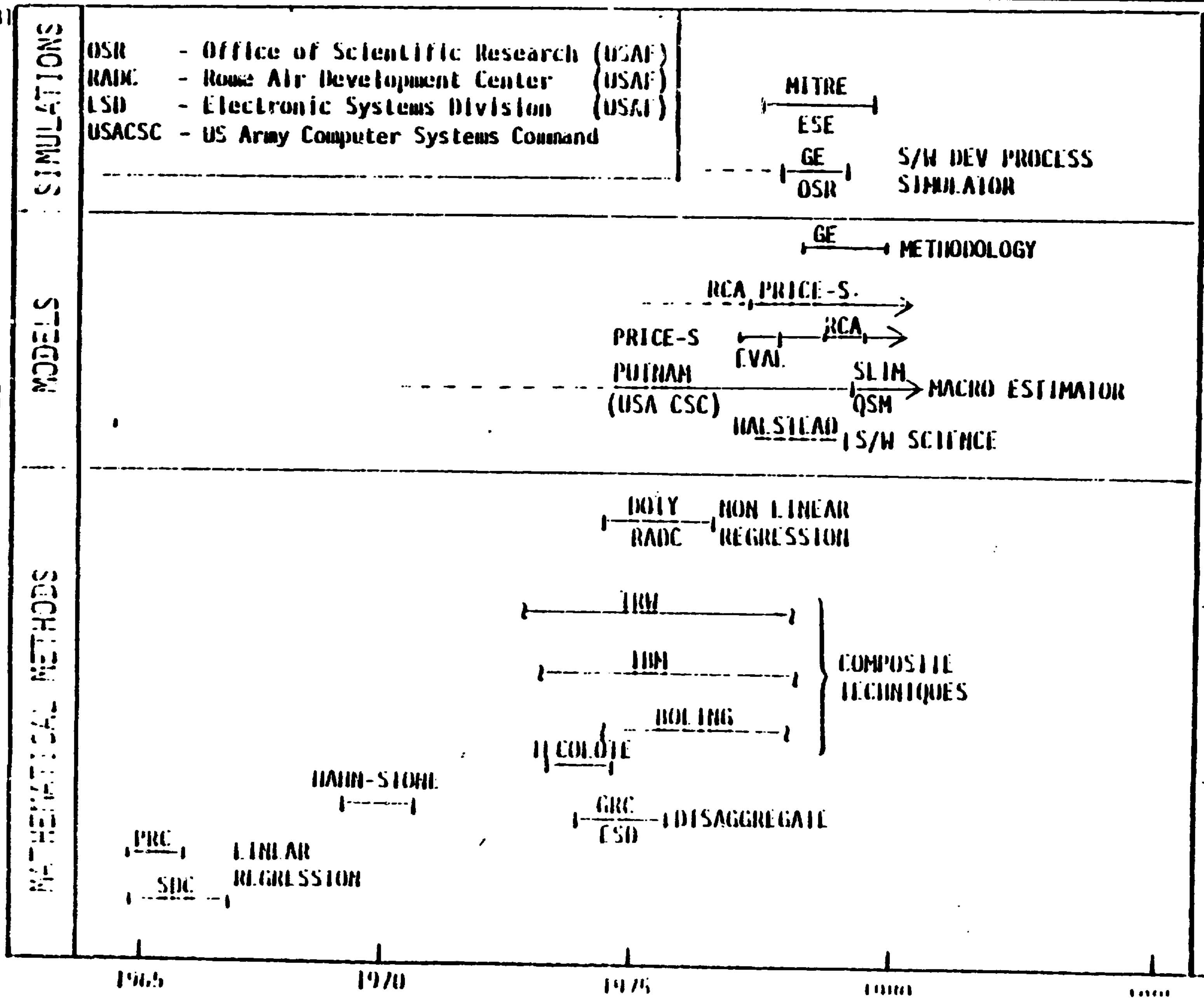


GENERAL ELECTRIC

# SOFTWARE COST ESTIMATION HISTORY

space division

OSR - Office of Scientific Research (USAF)
RADC - Rome Air Development Center (USAF)
ESD - Electronic Systems Division (USAF)
USACSC - US Army Computer Systems Command



TRENDS

DECREASING
Dependence on the number of projects supplying data

INCREASING
Input and Output Granularity and More Tailoring to the Application

INCREASING
Consideration of Environmental Factors as Cost Drivers

INCREASING
Complexity and Sophistication of Techniques

INCREASING
Use of Techniques in the Management Decision Making Process

ISP

6157

## CHARACTERISTICS OF CURRENT COST TECHNIQUES

### *Limited Visibility*

MODELS TEND TO BE "BLACK BOXES" WITH FINAL PROJECT COST AND/OR SCHEDULE AS THE OUTPUT. LIMITED INFORMATION ABOUT DYNAMIC ASPECTS OF PROJECT. HARD TO TAILOR TO SPECIFIC APPLICATION.

### *Weak Understanding*

MODELS TEND TO BE "MACROSCOPIC" AND DO NOT CONTRIBUTE TO UNDERSTANDING OF IMPACT OR INTERRELATIONSHIP OF FACTORS

### *Minimum Decision Analysis*

MODELS PROVIDE A MINIMUM OF ASSISTANCE IN DETERMINING HOW MANAGEMENT AND DEVELOPMENT DECISIONS WILL AFFECT PROJECT COST AND SCHEDULE

GENERAL ELECTRIC

## IMPORTANT CHACTERISTICS

space division

6181

*A Cost Estimation Methodology Must*

- ACCOUNTS FOR THE SIGNIFICANT FACTORS WHICH EFFECT THE DEVELOPMENT RESOURCES
- FIT SPECIFIC APPLICATION CHARACTERISTICS AND AN ORGANIZATION'S WAY OF DOING BUSINESS
- UTILIZES INPUT DATA WHICH IS NORMALLY AVAILABLE OR CAN BE REASONABLY ESTIMATED DURING THE PERIOD IN WHICH IT IS USED
- NOT OVERBURDEN THE SYSTEM WITH USELESS DETAIL OR PAPERWORK
- PROVIDE SAFEGUARDS (CHECKS AND BALANCES) AGAINST ERRONEOUS ESTIMATION
- PERMIT TRADEOFF ANALYSIS
- BE USABLE AT SEVERAL POINTS THROUGHOUT THE LIFE CYCLE

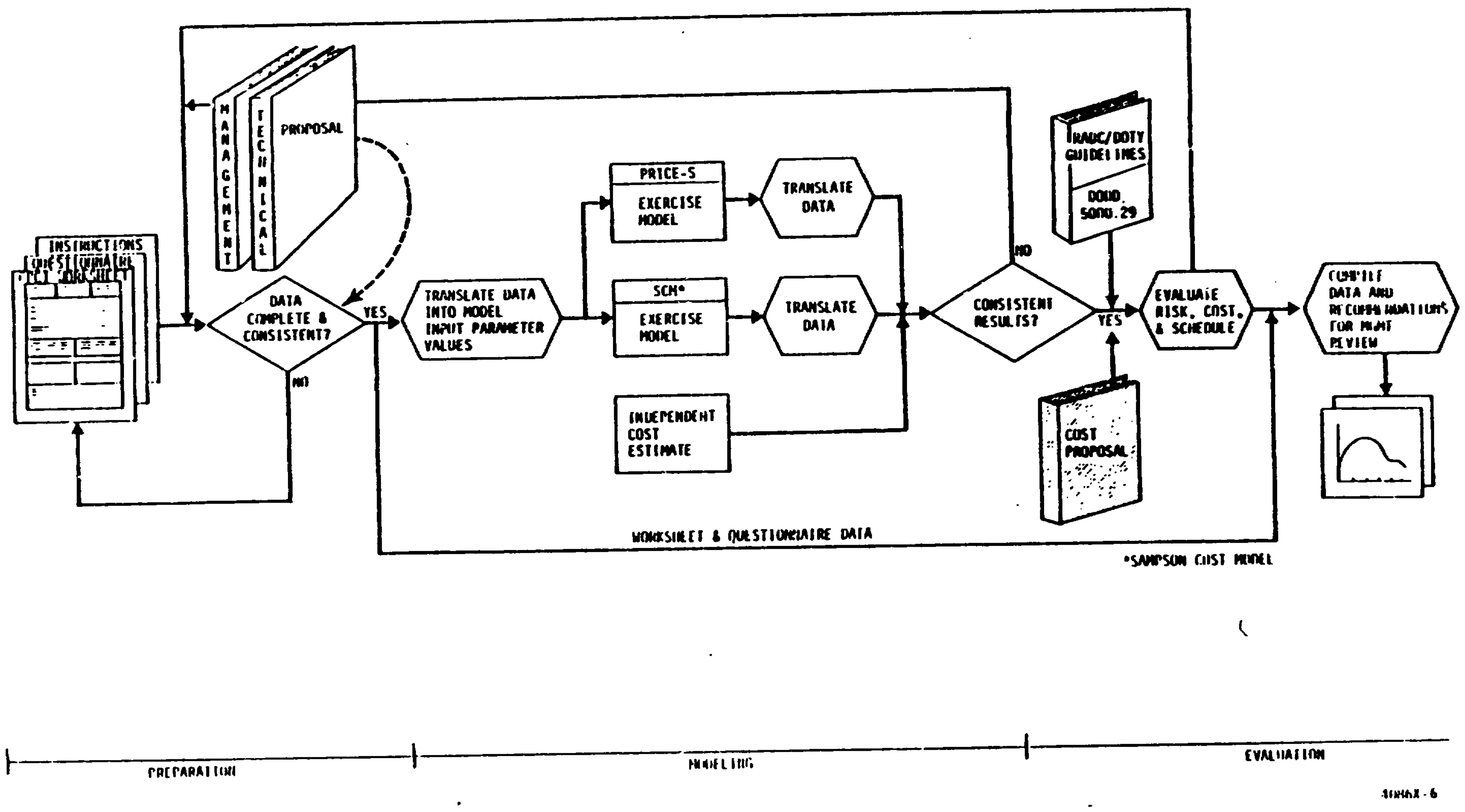


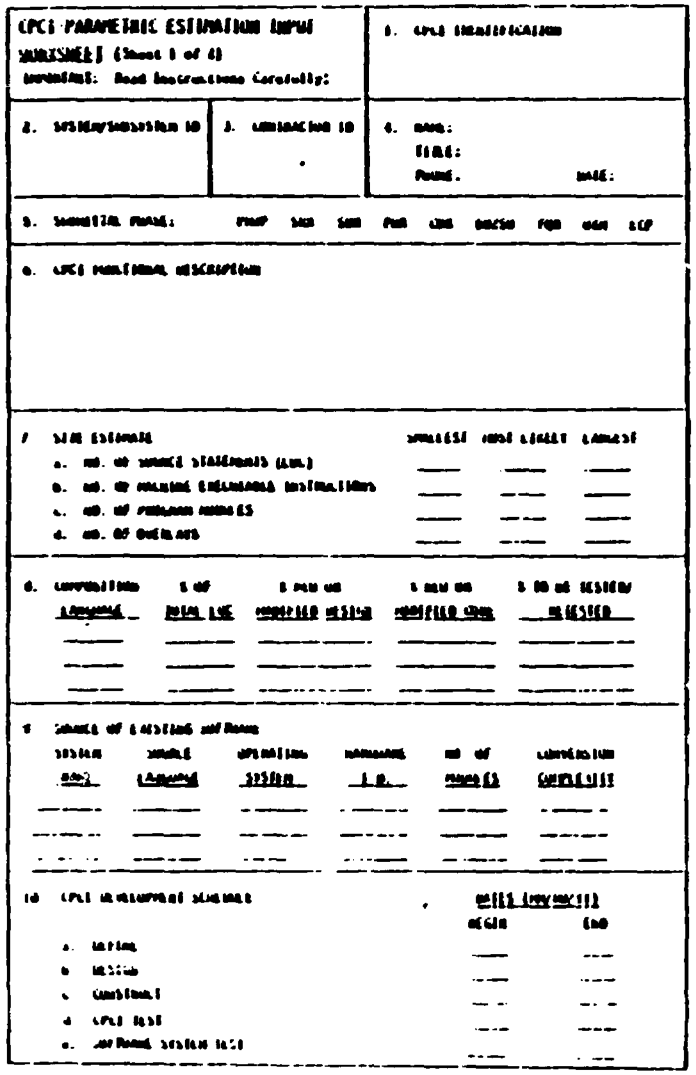
Figure 2.1-1  Software Cost Estimation and Evaluation Methodology

GENERAL ELECTRIC

# CPCI WORKSHEET

space division

6181



CPCI PARAMETRIC ESTIMATION INPUT WORKSHEET (Sheet 1 of 4)
IMPORTANT: Read Instructions Carefully!

1. CPCI IDENTIFICATION

2. SYSTEM/SUBSYSTEM ID

3. CONTRACTOR ID

4. NAME:
TITLE:
PHONE: DATE:

5. SUBMITTAL PHASE: PROP SRR SDR PDR CDR [illegible] FQR [illegible] ECP

6. CPCI FUNCTIONAL DESCRIPTION

7. SIZE ESTIMATE

| | SMALLEST | MOST LIKELY | LARGEST |
|---|---|---|---|
| a. NO. OF SOURCE STATEMENTS (LOC) | | | |
| b. NO. OF MACHINE EXECUTABLE INSTRUCTIONS | | | |
| c. NO. OF PROGRAM MODULES | | | |
| d. NO. OF OVERLAYS | | | |

8.

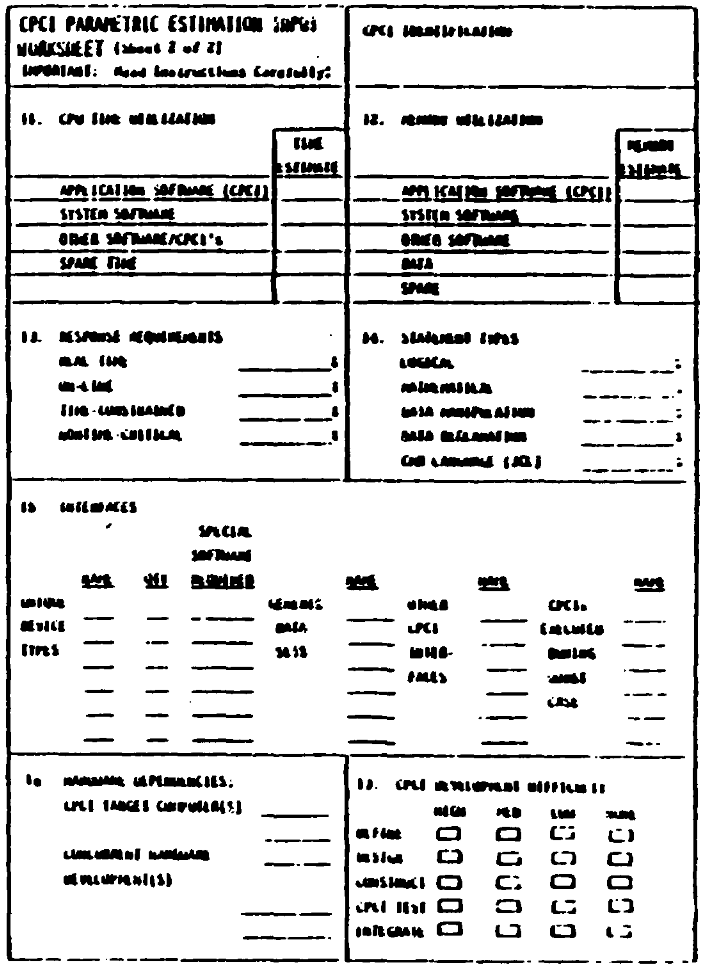
| COMPILATION LANGUAGE | % OF TOTAL LOC | % NEW OR MODIFIED DESIGN | % NEW OR MODIFIED CODE | % TO BE RETESTED |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

9. SOURCE OF EXISTING SOFTWARE

| SYSTEM NAME | SOURCE LANGUAGE | OPERATING SYSTEM | HARDWARE I.D. | NO. OF MODULES | CONVERSION COMPLEXITY |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |

10. CPCI DEVELOPMENT SCHEDULE

| | DATES (MO/YR) BEGIN | END |
|---|---|---|
| a. DEFINE | | |
| b. DESIGN | | |
| c. CONSTRUCT | | |
| d. CPCI TEST | | |
| e. SOFTWARE SYSTEM TEST | | |



CPCI PARAMETRIC ESTIMATION INPUT WORKSHEET (Sheet 2 of 2)
IMPORTANT: Read Instructions Carefully!

CPCI IDENTIFICATION

11. CPU TIME UTILIZATION

| | TIME ESTIMATE |
|---|---|
| APPLICATION SOFTWARE (CPCI) | |
| SYSTEM SOFTWARE | |
| OTHER SOFTWARE/CPCI's | |
| SPARE TIME | |
| | |

12. MEMORY UTILIZATION

| | MEMORY ESTIMATE |
|---|---|
| APPLICATION SOFTWARE (CPCI) | |
| SYSTEM SOFTWARE | |
| OTHER SOFTWARE | |
| DATA | |
| SPARE | |

13. RESPONSE REQUIREMENTS

| | |
|---|---|
| REAL TIME | % |
| ON-LINE | % |
| TIME-CONSTRAINED | % |
| NONTIME-CRITICAL | % |

14. STATEMENT TYPES

| | |
|---|---|
| LOGICAL | % |
| MATHEMATICAL | % |
| DATA MANIPULATION | % |
| DATA DECLARATION | % |
| JOB CONTROL LANGUAGE (JCL) | % |

15. INTERFACES

| | NAME | QTY | SPECIAL SOFTWARE REQUIRED | | NAME | | NAME | | NAME |
|---|---|---|---|---|---|---|---|---|---|
| UNIQUE | | | | GENERAL | | OTHER | | CPCIs | |
| DEVICE | | | | DATA | | CPCI | | EXECUTED | |
| TYPES | | | | BASES | | INTER- | | DURING | |
| | | | | | | FACES | | SAME | |
| | | | | | | | | CASE | |
| | | | | | | | | | |
| | | | | | | | | | |

16. HARDWARE DEPENDENCIES:
CPCI TARGET COMPUTER(S) ______
______
CONCURRENT HARDWARE DEVELOPMENT(S) ______
______
______

17. CPCI DEVELOPMENT DIFFICULTY

| | HIGH | MED | LOW | NONE |
|---|---|---|---|---|
| DEFINE | ☐ | ☐ | ☐ | ☐ |
| DESIGN | ☐ | ☐ | ☐ | ☐ |
| CONSTRUCT | ☐ | ☐ | ☐ | ☐ |
| CPCI TEST | ☐ | ☐ | ☐ | ☐ |
| INTEGRATE | ☐ | ☐ | ☐ | ☐ |

D-61

# AUTOMATED COST ESTIMATION SYSTEM

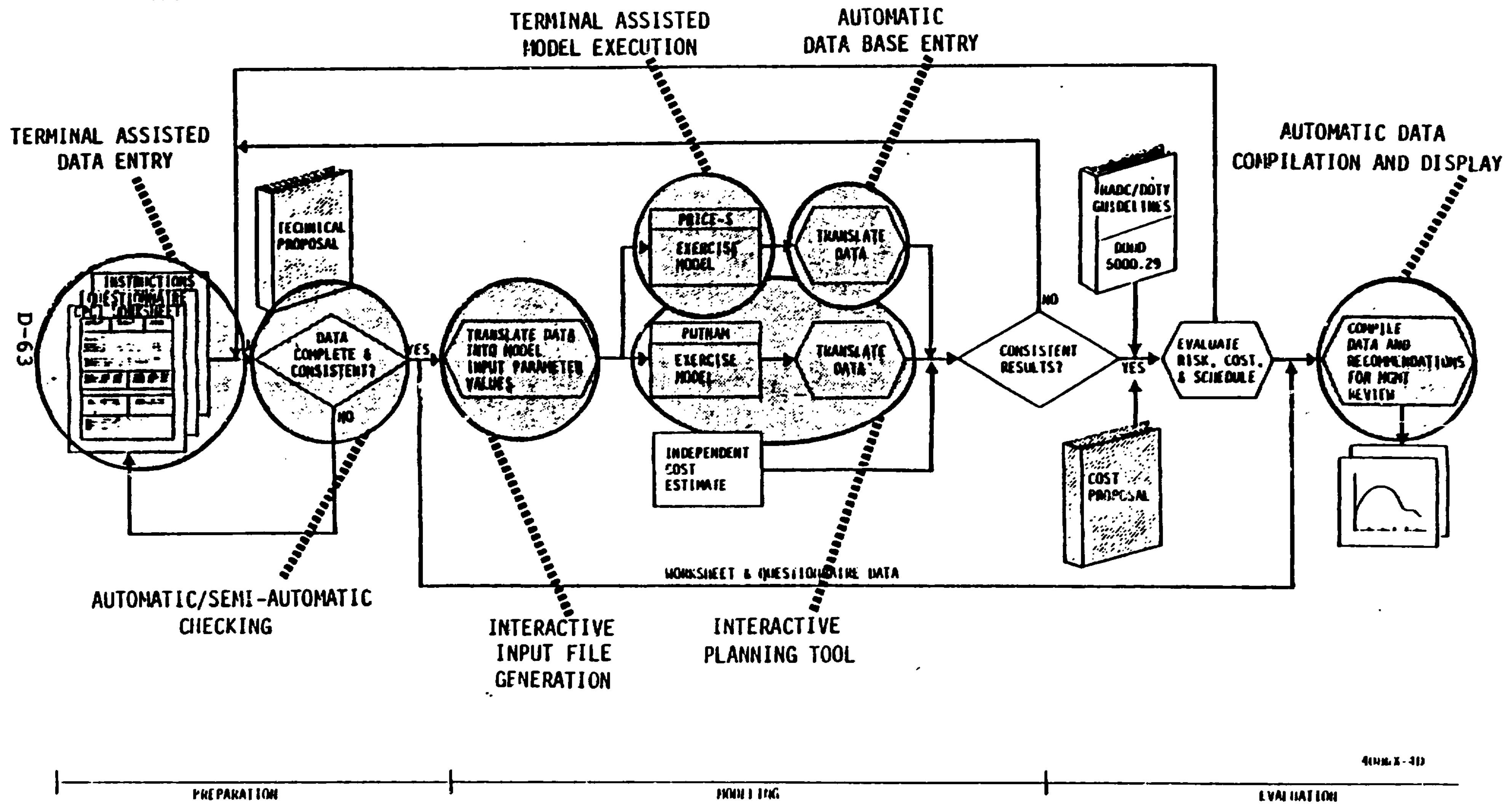
ACES

GENERAL ELECTRIC

# AUTOMATING THE METHODOLOGY

space division

6178



TERMINAL ASSISTED MODEL EXECUTION
AUTOMATIC DATA BASE ENTRY
TERMINAL ASSISTED DATA ENTRY
AUTOMATIC DATA COMPILATION AND DISPLAY
INSTRUCTIONS
QUESTIONNAIRE
WORKSHEET
TECHNICAL PROPOSAL
DATA COMPLETE & CONSISTENT?
YES
NO
TRANSLATE DATA INTO MODEL INPUT PARAMETER VALUES
PRICE-S
EXERCISE MODEL
TRANSLATE DATA
PUTNAM
EXERCISE MODEL
TRANSLATE DATA
INDEPENDENT COST ESTIMATE
CONSISTENT RESULTS?
NO
YES
NADC/DOTY GUIDELINES
DOD 5000.29
COST PROPOSAL
EVALUATE RISK, COST, & SCHEDULE
COMPILE DATA AND RECOMMENDATIONS FOR MGMT REVIEW
WORKSHEET & QUESTIONNAIRE DATA
AUTOMATIC/SEMI-AUTOMATIC CHECKING
INTERACTIVE INPUT FILE GENERATION
INTERACTIVE PLANNING TOOL
PREPARATION
MODELING
EVALUATION

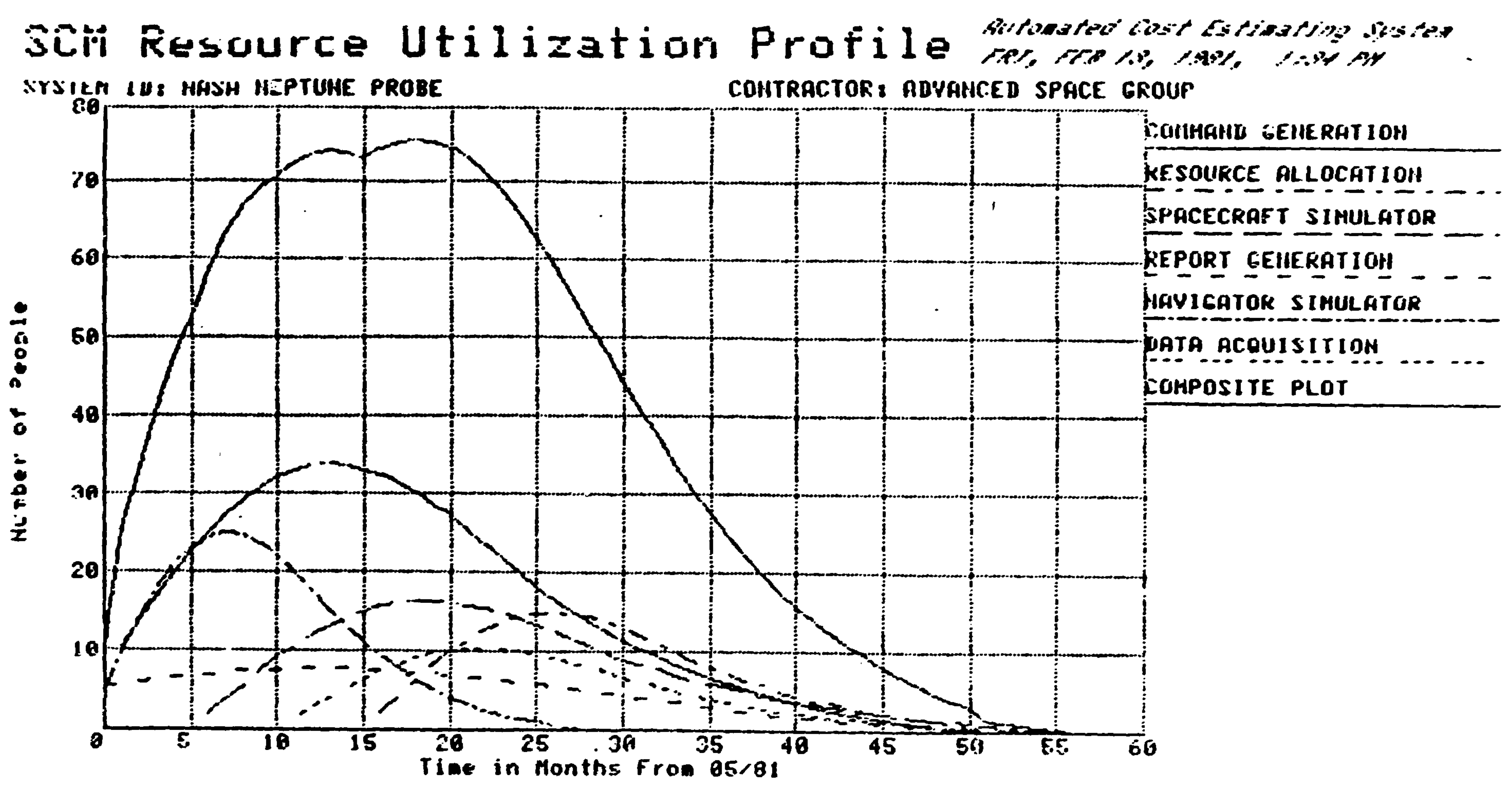


Figure 3.4.8.4-2 Resource Utilization Profile

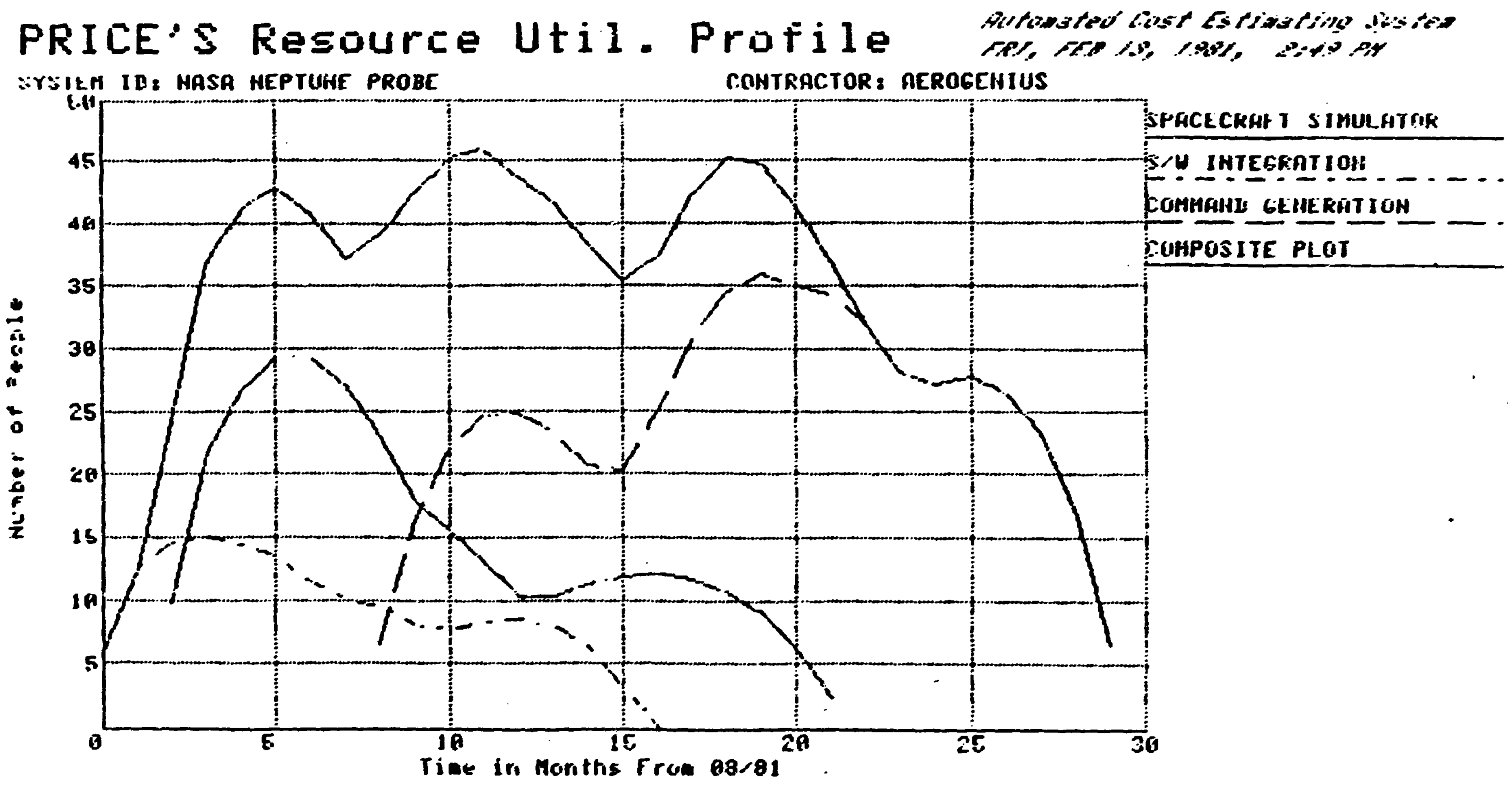


Figure 3.4.8.4-7 PRICE-S Resource Utilization

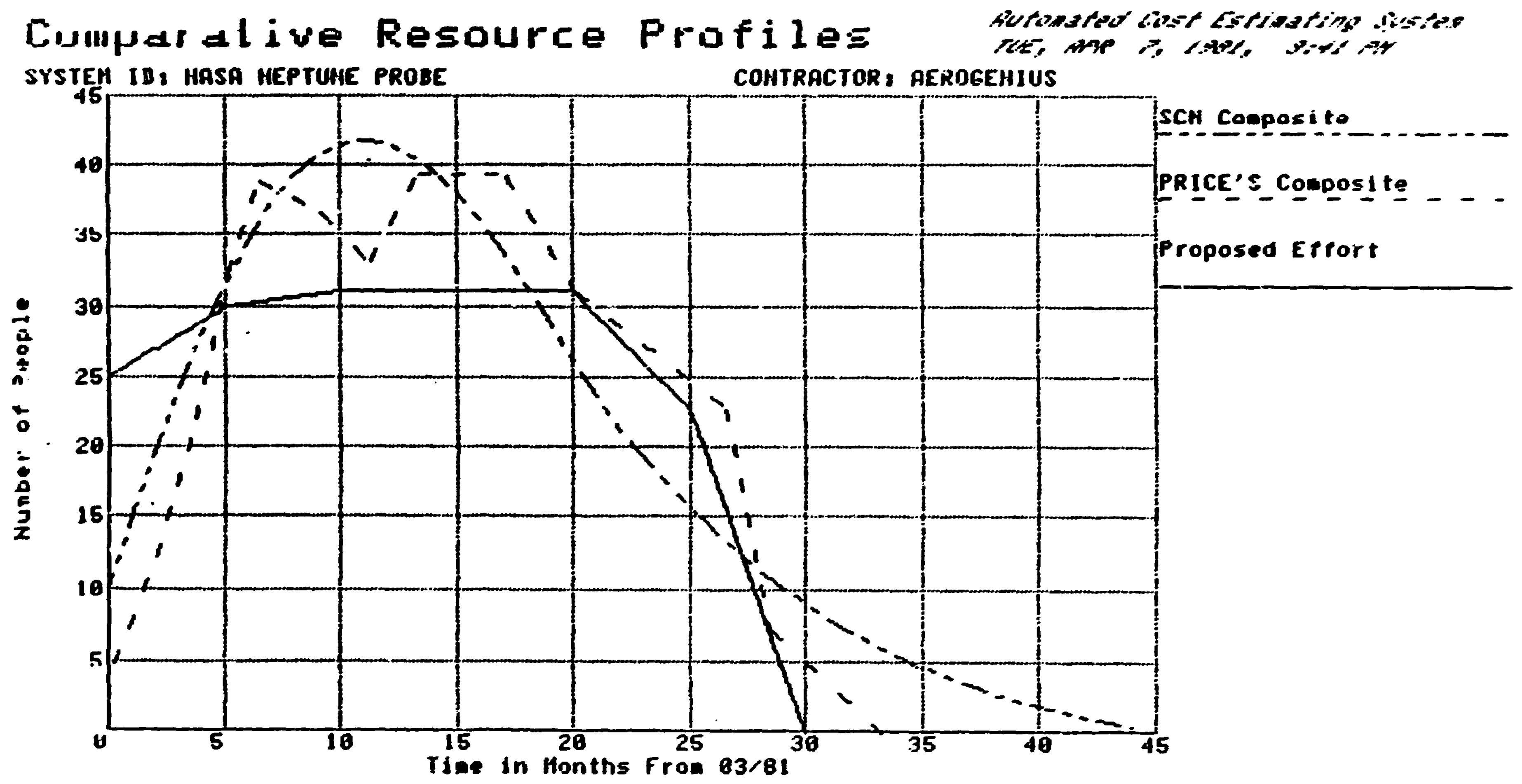


Figure 2.2.2-6 Comparative Resource Profiles

6128

## SUMMARY

- THE PROCESS WILL WORK AND PROVIDE VALUABLE INFORMATION
- KEEP COMPLETE RECORDS, DOCUMENT ASSUMPTIONS AND DECISIONS (WHAT? WHERE? WHEN? WHAT EXTENT?)
- EXPECT SOME INCONSISTENCIES BUT DON'T FORCE THE ESTIMATES TO CONVERGE WITHOUT SUFFICIENT JUSTIFICATION
- MAINTAIN PROCESS INTEGRITY. CONSISTENCY IS THE KEY BUT DON'T BE AFRAID TO EXPERIMENT
- CONTINUE TO RE-ESTIMATE THROUGHOUT THE DEVELOPMENT PHASE AS BETTER INFORMATION BECOMES AVAILABLE
- DOCUMENT "LESSONS LEARNED" SO THAT THE PROCESS CAN BE REFINED AND IMPROVED

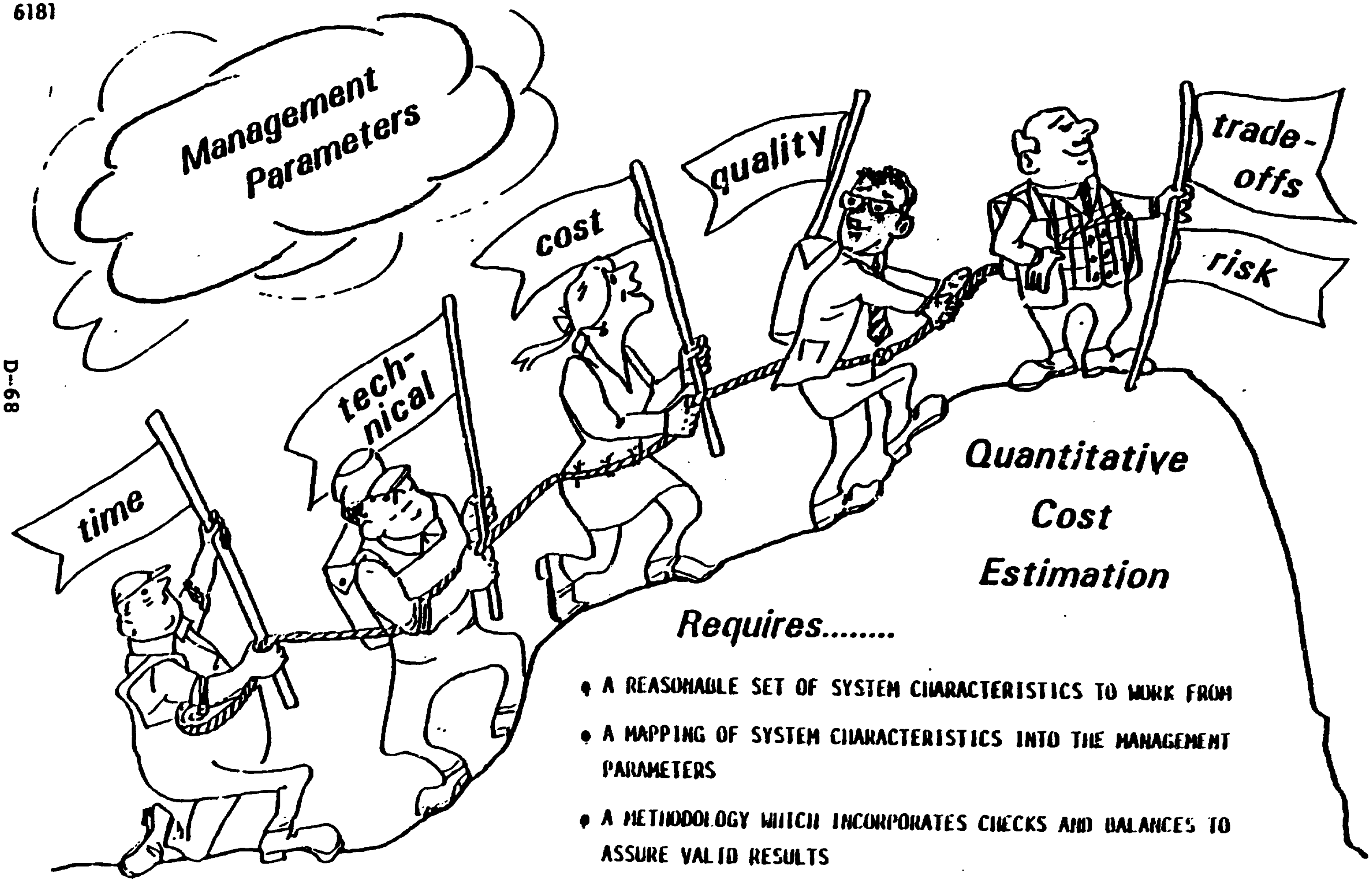
THE METHODOLOGY IS A TECHNIQUE - NOT A SOLUTION. IT CANNOT SUBSTITUTE FOR SOUND MANAGEMENT AND TECHNICAL JUDGMENT.

GENERAL ELECTRIC

# GIVING MANAGEMENT THE RIGHT INFORMATION

space division

6181



Quantitative Cost Estimation

Requires........

- A REASONABLE SET OF SYSTEM CHARACTERISTICS TO WORK FROM
- A MAPPING OF SYSTEM CHARACTERISTICS INTO THE MANAGEMENT PARAMETERS
- A METHODOLOGY WHICH INCORPORATES CHECKS AND BALANCES TO ASSURE VALID RESULTS

4138C-1

PRESENTATION MADE BY

ROGER BATE
TEXAS INSTRUMENTS

TO

PANEL ON ESTIMATING SOFTWARE COSTS

## TEXAS INSTRUMENTS EQUIPMENT GROUP BUDGETARY COST ESTIMATING MODEL

BACKGROUND

- MODEL DEVELOPED IN 1977 FOR BUDGETARY ESTIMATION
- MODEL DEVELOPMENT STRONGLY INFLUENCED BY WOLVERTON (TRW)
- MODEL WOULD BE CLASSIFIED AS PARAMETRIC AND HEURISTIC BY THIBODEAU
- MODEL IS USED AS PART OF A COST ESTIMATING METHODOLOGY

[illegible] (1-395

## CPU RESOURCE UTILIZATION FACTOR

STEP 1. DETERMINE PERCENT UTILIZATION OF CPU MEMORY.1

STEP 2. DETERMINE PERCENT UTILIZATION OF CPU TIME.2

STEP 3. SELECT THE LARGER OF STEP 1 AND STEP 2.

STEP 4. USE CPU RESOURCE UTILIZATION CURVE TO OBTAIN A FACTOR (F1) EQUAL TO THE ORDINATE CORRESPONDING TO THE PERCENT UTILIZATION FROM STEP 3.

STEP 5. DIVIDE PROJECT PRODUCTIVITY (P1) BY THE FACTOR (F1) OBTAINED FROM THE SYSTEM COMPLEXITY INDEX PROFILE AND THE SYSTEM PRODUCTIVITY MATRIX. THE NEW PRODUCTIVITY (P1:=F1XP1) WILL BE EFFECTIVE PROJECT PRODUCTIVITY.

NOTES: 1. IF CPU MEMORY IS DIVIDED, USE MOST CRITICAL CASE.

2. IF MULTIPROCESSOR, USE MOST CRITICAL CASE.

RRB: 06/22/81: 01-946

## SOFTWARE COST ESTIMATING PROCEDURES ARE NECESSARY FOR:

- PROVIDING PRE-PROPOSAL GROSS BALLPARK ESTIMATES
- DETAILED PROPOSAL ESTIMATES
- DETAILED ESTIMATES TO SUPPORT NEGOTIATIONS
- ESTABLISHING PROJECT BUDGETS

RRB: [illegible]

## SOFTWARE COST ESTIMATING

PROBLEM DESCRIPTION

- SOFTWARE COSTS ARE METHODOLOGY DEPENDENT
- SOFTWARE DEVELOPMENT MODEL IS POORLY UNDERSTOOD OR DEFINED
- HISTORICAL DATA IS ABSENT OR POORLY STRUCTURED

[illegible]

SOFTWARE COST ESTIMATING

THE APPROACH: DEVELOP A MODEL

- DEFINE SOFTWARE DEVELOPMENT MODEL BY LIFE CYCLE AND WBS
- STRUCTURE DATA WITH A STANDARD UNIT OF PRODUCTIVITY
- IDENTIFY AND APPLY COST INFLUENCES
- IMPOSE MEDIAN AND EXTREME PRODUCTIVITY BOUNDS
- DEVELOP ALTERNATIVE PROCEDURES
- DEVELOP REASONABLENESS CHECKS
- DEVELOP FLOW/TABLE PROCEDURE SYSTEM
- VALIDATE

[illegible]

## SOFTWARE COST ESTIMATING

## A STANDARD PRODUCTIVITY

## MEASURE

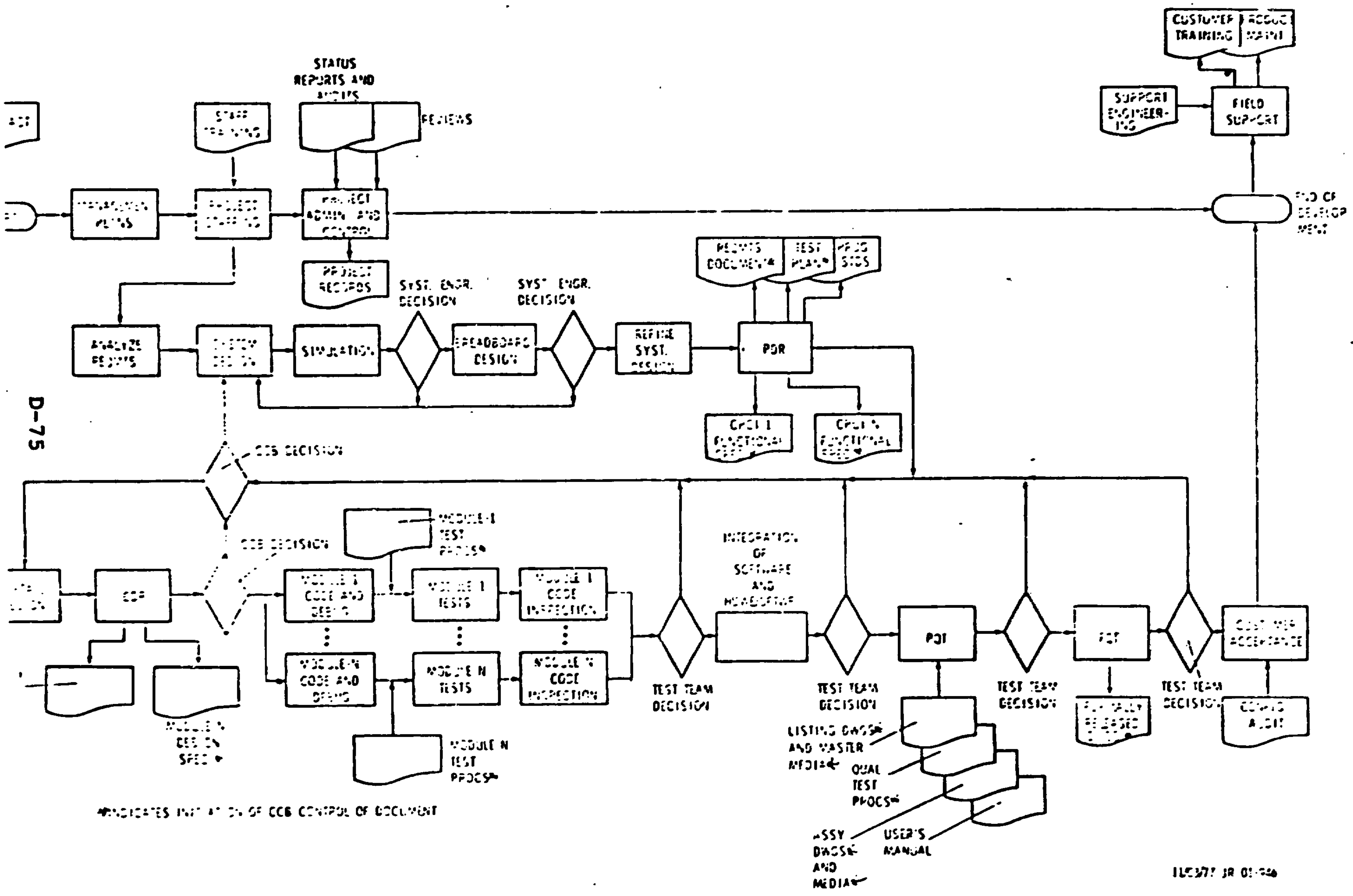
- SHOULD BE EASILY DETERMINED FROM PROJECT DATA
- SHOULD RELATE TO TOTAL SOFTWARE DEVELOPMENT EFFORT
- SHOULD BE PORTABLE AMONG DIFFERENT HARDWARE PROCESSORS
- DEFINITION OF PRODUCTIVITY:

  P = MACHINE INSTRUCTIONS / MAN-HOUR

RRB: 0[illegible]2 [illegible]: [illegible]-946

Figure C-1.

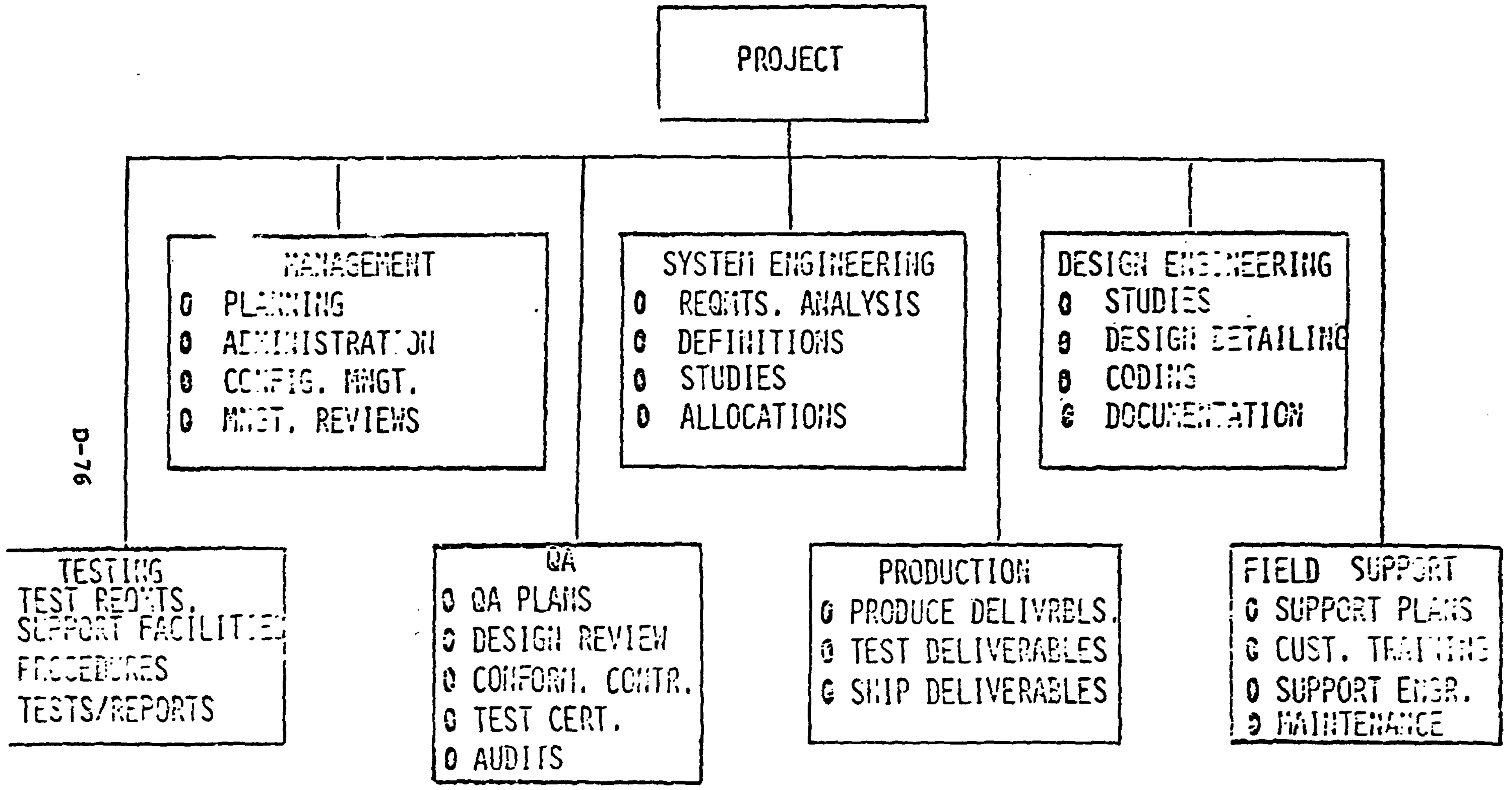
A DYNAMIC MODEL FOR SOFTWARE DEVELOPMENT



*INDICATES INITIATION OF CCB CONTROL OF DOCUMENT

SOFTWARE COST ESTIMATING

WBS ACTIVITIES



6/3/77:JR:01-945

A DEFINITION OF

A MACHINE INSTRUCTION

- DEFINITION

  A MACHINE INSTRUCTION IS A CODE THAT CAUSES A PROCESSOR TO PERFORM A SINGLE OPERATION.

- ILLUSTRATIVE EXAMPLES

  - LOAD REGISTER
  - MOVE
  - ADD
  - OR
  - BRANCH
  - ETC.

PRELIMINARY CONVERSION FACTORS

1 FORTRAN STATEMENT = 5 MACHINE INSTRUCTIONS

1 PASCAL STATEMENT = 5 MACHINE INSTRUCTIONS

RRB: 06/22/81: 01-946

## SOFTWARE COST ESTIMATING

## COST INFLUENCES

- O COMPLEXITY
  - O SIZE
  - O MODULARITY
  - O DIFFICULTY
  - O NOVELTY
  - O METHODOLOGY
  - O PROCESSOR DESIGN MATURITY
- O CPU RESOURCE UTILIZATION
- O ADDED COSTS
- O RISKS - POSSIBLE ADDED COST FACTORS

RRB: 06/22/81: 01-946

SOFTWARE COST ESTIMATION

THE TECHNIQUE

- PROVIDES FOUR DIFFERENT METHODS TO ESTIMATE SOFTWARE COST
  - TOP-DOWN WITH NO COMPARISON (BUDGETARY)
  - TOP-DOWN WITH APPLICATION OF SOFTWARE COST DATA FROM PREVIOUS PROJECT
  - BOTTOM-UP (MODULE COMPARISON) ESTIMATES
  - BOTTOM-UP ESTIMATE VIA ENGR. COST ESTIMATES
- INTEGRATED WITH A REFINED SOFTWARE DEVELOPMENT METHODOLOGY
- FACILITATES MODEL UPGRADING AS EXPERIENCE IS ACCUMULATED

[illegible] 3/22/81: [illegible]-946

SOFTWARE COST ESTIMATING

METHOD VERSUS INTENDED APPLICATION

| METHOD | APPLICATION |
|---|---|
| BUDGETARY | PRE-PROPOSAL STAGE OR WHEN A GROSS BALLPARK ESTIMATE IS SUFFICIENT. |
| SYSTEM COMPARISON | PRE-PROPOSAL OR WHEN A MORE "REFINED" BALLPARK ESTIMATE IS REQUIRED. METHOD REQUIRES AN EXISTING SYSTEM FROM WHICH COMPARISONS CAN BE MADE. |
| MODULE COMPARISON | PROPOSAL, NEGOTIATIONS, OR WHEN PROJECT BUDGETS ARE ESTABLISHED. REQUIRES EXISTING MODULES FROM WHICH COMPARISONS CAN BE MADE. |
| BOTTOM-UP (DETAILED ENGINEERING ESTIMATES) | PROPOSAL, NEGOTIATIONS, OR WHEN PROJECT BUDGETS ARE ESTABLISHED. REPRESENTS METHOD WITH GREATEST ACCURACY IF ESTIMATES MADE AGAINST REQUIRED WBS. |

## SOFTWARE COST ESTIMATING

### COST INFLUENCES

- COMPLEXITY
  - o SIZE
  - o MODULARITY
  - o DIFFICULTY
  - o NOVELTY
  - o METHODOLOGY
  - o PROCESSOR DESIGN MATURITY
- o CPU RESOURCE UTILIZATION
- o ADDED COSTS
- o RISKS - POSSIBLE ADDED COST FACTORS

DRB: 06/22/81: 01-946

## SYSTEM COMPLEXITY PROFILE

I. SYSTEM TYPE

| | |
|---|---|
| SINGLE PROCESSOR, STD. PERIPHERALS | 1 |
| SINGLE PROCESSOR, EXTENDED PERIPHERALS | 2 |
| MULTI-PROCESSOR, STD. PERIPHERALS | 3 |
| MULTI-PROCESSOR, COMPLEX PERIPHERALS | 4 |
| DISTRIBUTED MUTLIPROCESSOR SYSTEMS | 5 |

II. SYSTEM SIZE

| | |
|---|---|
| 1-10 MODULES (200-2K INSTR.) | 1 |
| 11-40 MODULES (2K-4K INSTR.) | 2 |
| 41-80 MODULES (8K-10K INSTR.) | 3 |
| 81-150 MODULES (10K-30K INSTR.) | 4 |
| >150 MODULES (>30K INSTR.) | 5 |

III. MISSION CRITICALITY

| | |
|---|---|
| STATIONARY, DATA PROCESSING | 1 |
| MOBILE, REAL-TIME | 2 |
| AIR, UNMANNED, REAL-TIME | 3 |
| AIR, MANNED, REAL-TIME | 4 |
| SPACE, REAL TIME | 5 |

IV. SYSTEM NOVELTY

| | |
|---|---|
| MINOR REDESIGN OF A FAMILIAR SYS. | 1 |
| MODERATE REDESIGN OF A FAMILIAR SYS. | 2 |
| MAJOR REDESIGN OR NEW FAMILIAR SYS. | 3 |
| MODERATELY NOVEL DESIGN | 4 |
| REVOLUTIONARY | 5 |

V. METHODOLOGY (SEE METHODOLOGY QUESTIONNAIRE)

| | |
|---|---|
| 0-2 NO ANSWERS | 1 |
| 3-4 NO ANSWERS | 2 |
| 5-6 NO ANSWERS | 3 |
| 7-8 NO ANSWERS | 4 |
| 9-10 NO ANSWERS | 5 |

RRB: 06/22/81: 01-946

SYSTEM COMPLEXITY PROFILE

(CONTINUED)

VI. <u>PROCESSOR DESIGN MATURITY</u>

| | |
|---|---|
| NO CHANGES | 1 |
| FEW CHANGES | 2 |
| MODERATE CHANGES | 3 |
| UNSTABLE/MANY CHANGES | 4 |
| CONCURRENT PROCESSOR DESIGN | 5 |

RRB: 06/22/81: 01-946

## A METHODOLOGY QUESTIONNAIRE

- IS THE SOFTWARE OFF THE CRITICAL SCHEDULE PATH?
- IS THE OPERATING SYSTEM AVAILABLE?
- IS THE HARDWARE PROCESSOR AVAILABLE?
- IS THE DESIGN ALLOCATED TO MODULES?
- IS TOP-DOWN DESIGN PLANNED?
- IS TOP-DOWN INTEGRATION PLANNED?
- ARE PROJECT NOTEBOOKS PLANNED?
- ARE CODE INSPECTIONS PLANNED?
- ARE SOFTWARE TEST PLANS FORMULATED?
- ARE AUTOMATED TEST TOOLS PLANNED?

RRB: 06/22/81: 01-946

## SYSTEM PRODUCTIVITY MATRIX*

| INDEX | CAT. I | II | III | IV | V | VI | COMPOSITE (P) |
|---|---|---|---|---|---|---|---|
| 1 | 3.40 | 1.73 | 1.73 | .84 | 1.18 | 1.18 | 11.900 |
| 2 | 2.85 | 1.45 | 1.45 | .71 | .99 | .99 | 4.170 |
| 3 | 2.40 | 1.21 | 1.21 | .61 | .83 | .83 | 1.480 |
| 4 | 2.01 | 1.02 | 1.02 | .52 | .70 | .70 | 0.530 |
| 5 | 1.69 | .85 | .85 | .44 | .59 | .59 | 0.187 |
| IMPACT RATIO** | 2.00 | 2.04 | 2.04 | 1.91 | 2.00 | 2.00 | 63.63 |

* SELECT A PRODUCTIVITY WEIGHT FOR EACH CATEGORY (C) WITH THE COMPLEXITY INDEX FOR THAT CATEGORY. THE PRODUCT OF THE SIX WEIGHTS GIVES THE COMPOSITE PRODUCTIVITY ESTIMATE.

EXAMPLE:

| CATEGORY | INDEX | WEIGHT |
|---|---|---|
| I | 1 | 3.40 |
| II | 2 | 1.45 |
| III | 4 | 1.02 |
| IV | 3 | .61 |
| V | 2 | .99 |
| VI | 5 | .59 |
| | COMPOSITE | 1.79 (PRODUCT) |

** THE IMPACT RATIO REPRESENTING THE MAXIMUM POSSIBLE COST RATIO FOR THAT FACTOR.

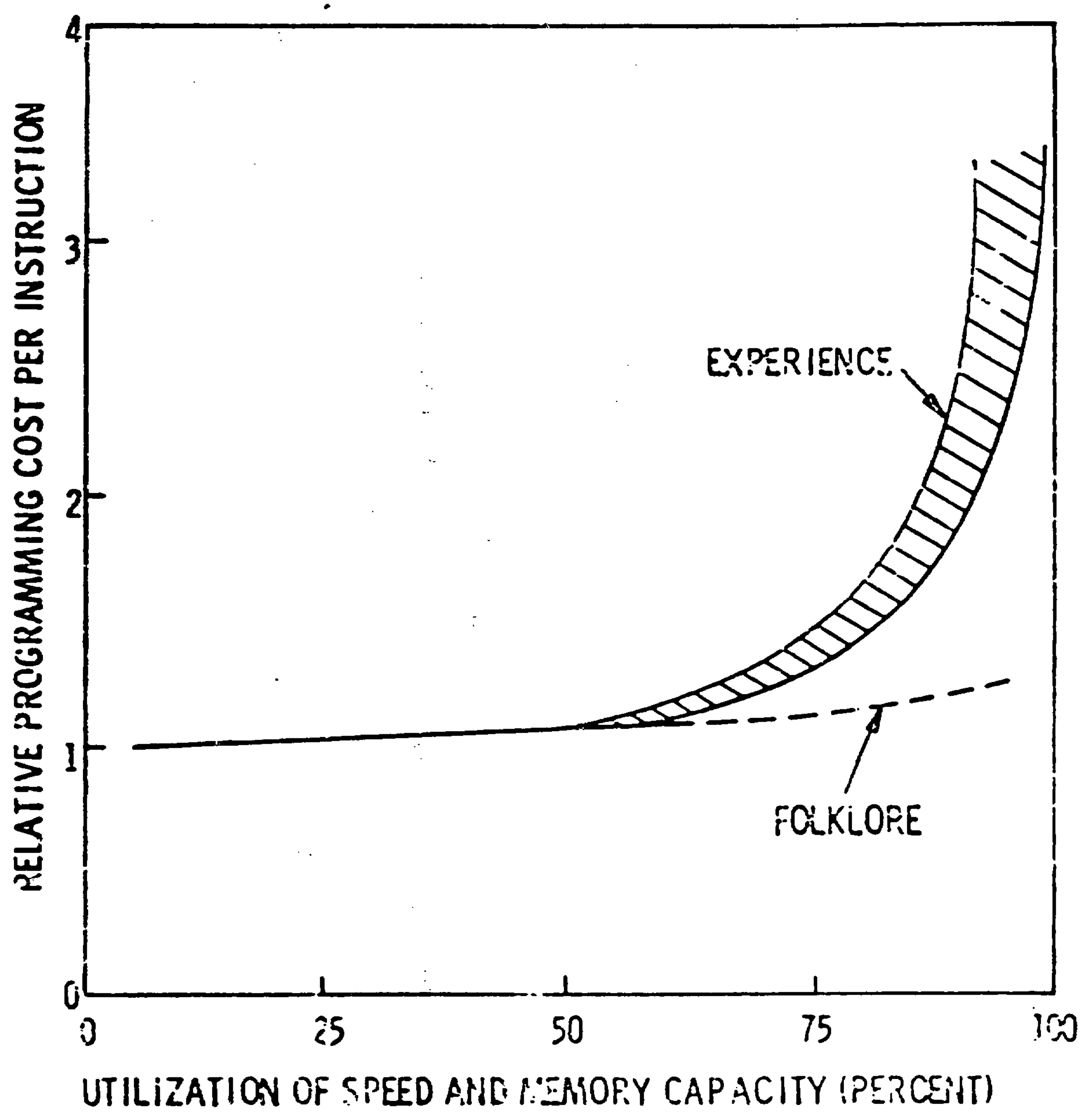
MOD. 06/22/81; 01-006

## SOFTWARE COST ESTIMATING

### COST INFLUENCES

- o COMPLEXITY
  - o SIZE
  - o MODULARITY
  - o DIFFICULTY
  - o NOVELTY
  - o METHODOLOGY
  - o PROCESSOR DESIGN MATURITY
- o CPU RESOURCE UTILIZATION
- o ADDED COSTS
- o RISKS - POSSIBLE ADDED COST FACTORS

RRB: 06/22/81 01-946



2-4-77 DJ 01-946

# BUDGETARY ESTIMATE PROCEDURE

STEP 1. DETERMINE REQUIREMENTS

STEP 2. ALLOCATE TASKS/SUBTASKS

STEP 3. COMPLETE HIGH LEVEL DESIGN

STEP 4.* ALLOCATE HARDWARE/SOFTWARE TASKS

STEP 5. PARTITION DESIGN TO MODULE LEVEL

STEP 6.* ESTIMATE NUMBER OF MACHINE INSTR. FOR MODULES (OR PER MODEL)

STEP 7. SUMMARIZE TOTAL MACHINE INSTR. (N1)

STEP 8. CALCULATE PRODUCTIVITY FROM THE SYSTEM COMPLEXITY INDEX PROFILE. COMPARE WITH AN ASSESSMENT OF SYSTEM DIFFICULTY AS "VERY HARD" (P1 = 0.5 INSTR./M-HR.), "NORMAL" (P1 = .77 INSTR./M-HR.), OR "VERY EASY" (P1 = 6 INSTR./M-HR.)

STEP 9. ESTIMATE TOTAL SOFTWARE LABOR AS:
L1 = N1/(175 x P1) IN M-MONTHS

STEP 10. ESTIMATE LABOR COSTS AS:
L1 x R1, WHERE R1 IS AN APPROPRIATE RUNNING RATE FOR LABOR

* STEP MAY BE OMITTED WHERE KNOWLEDGE OF SYSTEM SIZE IN MACHINE INSTRUCTIONS EXISTS OR WHERE THIS QUANTITY IS ESTIMATED DIRECTLY.

PAGE 1 OF 2

RRB: 06/22/81: 01-946

## BUDGETARY ESTIMATE PROCEDURE (CONT.)

STEP 11. ESTIMATE COMPUTER SUPPORT COSTS AS S1 = COMPUTER FACTOR x L1 x R1 (S1 IN $).

STEP 12. ESTIMATE DOCUMENTATION COST AS: N1/5 x D1, WHERE D1 IS $/PAGES.

STEP 13. ESTIMATE TRAVEL T1 IN $.

STEP 14. COMPUTE SOFTWARE COST AS: C1 = L1 x R1 + S1 + D1 x N1/5 + T1.

## SOFTWARE COST ESTIMATING

## COST INFLUENCES

- COMPLEXITY
  - SIZE
  - MODULARITY
  - DIFFICULTY
  - NOVELTY
  - METHODOLOGY
  - PROCESSOR DESIGN MATURITY
- CPU RESOURCE UTILIZATION
- ADDED COSTS
- RISKS - POSSIBLE ADDED COST FACTORS

RRB: 06/22/81: 01-946

## ADDED COSTS ESTIMATE

STEP 1. IDENTIFY ALL ADDED COST ITEMS, E.G.:

- SUB-CONTRACTED ITEMS
- LEASED COMPUTER SUPPORT SYSTEMS
- SPECIAL FACILITY REQUIREMENTS
- LEASED COMMUNICATIONS
- CAPITAL SOFTWARE DEVELOPMENT SUPPORT SYS.
- OTHER COSTS NOT COVERED ELSEWHERE

STEP 2. ESTIMATE EACH ADDED COST ITEM IN $.

STEP 3. SUMMARIZE ADDED COSTS, CA IN $.

STEP 4. UPDATE SOFTWARE COST ESTIMATE, C3 = C2 + CA

RRB: 06/22/81: 01-946

## SOFTWARE COST ESTIMATING

## COST INFLUENCES

- COMPLEXITY
  - SIZE
  - MODULARITY
  - DIFFICULTY
  - NOVELTY
  - METHODOLOGY
  - PROCESSOR DESIGN MATURITY
- CPU RESOURCE UTILIZATION
- ADDED COSTS
- RISKS - POSSIBLE ADDED COST FACTORS

RRB: 06/22/81: 01-946

## RISKS - POSSIBLE ADDED COST FACTORS

- OMITTED OR INCORRECT REQUIREMENTS
- SYSTEM RELIABILITY DEMO REQUIRED
- CRITICAL PROGRAMS TO BE CFE
- CUSTOMER IMPOSED QA PROVISIONS
- CRITICAL DATA BASES TO BE CFE
- TEST DATA TO BE CFE
- CRITICAL COMPONENTS TO BE DEVELOPED BY YOU AT CUSTOMER FACILITY
- KEY PERSONNEL TO BE CFE
- SPATIALLY SEPARATED DEVELOPMENT FACILITY
- UNUSUALLY INEXPERIENCED PERSONNEL
- APPLICATION IS HIGHLY CLASSIFIED
- HARDWARE PROCESSOR CHANGE IN MID-PROJECT
- FIXED PRICE, INCENTIVE OR PENALTY CONTRACT

## SOFTWARE COST ESTIMATE

## REASONABLENESS CHECKS

- A BUDGETARY ESTIMATE MAY BE USED AS A REASONABLENESS CHECK AGAINST ANOTHER ESTIMATION METHOD.

- A BUDGETARY ESTIMATE MAY BE COMPARED FOR REASONABLENESS AGAINST PREVIOUS EXPERIENCE WITH SOFTWARE DEVELOPMENT PROGRAMS FOR SIMILAR SYSTEMS.

A SOFTWARE DEVELOPMENT

EFFORT MODEL*

| DEVELOPMENT PHASE | IND** AVG | TRW** (SCIENTIFIC) | DADS (FORECAST) | PROPOSED MODEL |
|---|---|---|---|---|
| ANALYSIS AND DESIGN AND DOCUMENTATION | 52% | 48% | 48% | 50 |
| CODING | 16% | 24% | 21% | 20 |
| INTEGRATION AND TESTING | 32% | 28% | 31% | 30% |

* EXCLUDING PROPOSAL AND FIELD SUPPORT.

** TRW SYSTEMS DIVISION, "THE COST OF DEVELOPING LARGE-SCALE SOFTWARE," RAY WOLVERTON, IEEET VOL. C-23, No. 6, JUNE 1974.

## MODEL IMPLEMENTATION

THE MODEL IS CURRENTLY IMPLEMENTED ON

- THE TI-59 HAND-HELD PROGRAMMABLE CALCULATOR
  - USES ALL OF MEMORY
  - USER INTERFACE NOT PARTICULARLY FRIENDLY
- THE TI 990/10 AND 990/12 MINICOMPUTER
  - BETTER HUMAN INTERFACE

## REPORT OF THE PANEL ON SOFTWARE REUSABILITY

PROCEEDINGS OF THE SOFTWARE WORKSHOP

JOINT LOGISTICS COMMANDERS

JOINT POLICY COORDINATING GROUP ON COMPUTER RESOURCE MANAGEMENT

MONTEREY, CALIFORNIA, 22-25 JUNE 1981

# REPORT OF THE PANEL ON SOFTWARE REUSABILITY

PANEL E

CHAIRPERSON: PAUL A. MAURO
HUGHES AIRCRAFT COMPANY

CO-CHAIRPERSON: RICHARD J. MORREALE
COMPUTER SCIENCE CORPORATION

TABLE OF CONTENTS

## REPORT OF THE PANEL ON SOFTWARE REUSABILITY

## 1.0 OBJECTIVE

The Software Reusability Panel was convened to evaluate whether reusability represents a potentially valuable concept to reduce cost and elapsed time to develop embedded computer system software. Further, if there was a consensus on the viability of this issue, then what barriers must be overcome and how does the program manager and/or software manager make reusability a reality?

## 2.0 SCOPE

A growing concern exists in DOD and in industry over the lack of reapplication of software and the difficulty in adapting software to even very similar systems. By contrast, standardization in the hardware arena has already yielded DOD gains in various aspects of life cycle costs. In the commercial world, reuse of software has been widespread and profitable for many companies. These lessons need to be drawn upon by the DOD software community to achieve parallel gains in this critical area.

Two other dimensions are discernible as stimuli toward achievement of reusability. First, software applications are growing at a tremendous rate in both size and complexity. This growth rate will far outstrip the capacity of our universities to turn out software engineers. A way must be found to produce more software with a small increase in staff. An obvious solution would be to find ways to reapply existing software rather than to create new software for each new application. Secondly, the technology trends towards microcomputers and very large scale integration are going to push for complex embedded software implementing specific functions in these small hardware elements. When this occurs, software must be reusable in order for the hardware to be extendable into many different applications. Conversely, if the software does not become reusable, the gains in life cycle cost and risk reduction in DOD embedded systems acheived via hardware standardization will be eroded.

Three specific types of reusability were surfaced and addressed by the panel:

1) Reusing functional software systems across multiple configurations of the same basic system. An example of this is the variation which occurs between different platforms supported by the Naval Tactical Data System Software.

2) Reuse of generic software components for different applications. An example of this would be for the Air Force to reapply radar tracking algorithms from one radar system to another.

3) Reuse of prototype and development software during the evolution of a system through its life cycle. Software must provide its appropriate contribution in support of DOD's Preplanned Product Improvement Initiative (PppI).

A new acronym was established by the panel to facilitate discussion and reporting: RUS is used herein to denote Reusable Software. The term software package used herein refers to a separately compiled and configured unit(s) of computer program source code.

## 3.0 APPROACH

The panel initially addressed the very basic issues of reusability: definition, areas of greatest reusability, areas of least reusability, and expected payoff. The following definition was formed by the panel:

> Reusable software is existing software, including specification, design, code, and/or documentation, which can be employed or adapted, in part or total, into a new end use.

It was pointed out by members of the panel that this definition is in conflict with the DAR definition of software since the latter does not include specification, design or documentation as part of software. However, for purposes of the panel deliberations, this distinction was overlooked. The above definition does highlight one specific approach of the panel: That is to consider reuse of the specification, and of the design, as well as the code itself.

The panel's input in greatest reusability and least reusability reflected the diversity of background of the group. Although there was no general consensus, observations of attributes for greatest reusability were: standard interface, structure, and style of software; modularization to very small elements; reapplication in a family of systems; reapplication by the same developer; and applications with a very specific single requirement (e.g., a compiler).

Least reusable software was characterized as the converse of the above as well as software which interfaces to sensors (including radar, transducers); software which is highly machine-dependent; and software which has a specific narrow mission. It was observed that embedded computer systems generally share these latter attributes.

The payoff discussion similarly reflected the diversity of the group. Those with a DOD embedded applications orientation declared a more modest expected payoff (25% cost reduction) than those from a business systems background (up to 66%) where reuse has been a reality for several years. There was clear consensus that reuse would improve the project schedule and reduce the risk of new system developments. Further benefit would be realized throughout the life cycle, both development and maintenance, and would apply in many different areas: documentation, testing, training, reduced personnel skill requirements and rapid prototyping.

Subsequent to these early deliberations, the panel was subdivided into four Subpanels:

SUBPANEL 1: (Section 4.1) How to design and build reusable software. What are the techniques, tools and significant considerations in constructing a software system, including specifications, documentation and code, such that this software will be reusable? The orientation of this group lays the groundwork for what software developers should consider in order to achieve reuse in their future developments.

SUBPANEL 2: (Section 4.2) Managing a "ported" development. What are the unique aspects of managing a project where at least 50% of the software is reused instead of developing all new software? The group considered the source of the software as either within one's own company or from an outside agency.

SUBPANEL 3: (Section 4.3) How to employ existing software and what can be learned from past experience. There have been projects which have successfully adopted existing software. What were the lessons (both good and bad) from these efforts? Also, what steps must be accomplished to use existing software?

SUBPANEL 4: (Section 4.4) Implications on DOD policy and acquisition practice and strategy. The group considered new policy requirements, impact on existing standards, and how to provide incentives to government program managers and contractors to reuse existing software. Also, what role must DOD play to make software accessible to potential reusers?

## 4.0 DISCUSSION

This section contains the reports from the four subpanels introduced above. In general, it was agreed that a much greater degree of reusability in embedded systems is achievable. Currently, existing software cannot be expected to see much reuse but in the future, new software can be built for reuse. However, very significant changes are required in development methodology, support tools, acquisition practices, and last, and most significant, in the attitudes of software managers and developers. The gains in overcoming "private" programming have only been achieved with great effort and patience. Reuse of software on a large scale will require even greater effort and patience. This is further dramatized by problems of moving software between contractors. Software acquisition managers will have to recognize that there are practical limitations to the amount of reuse which can be expected to be achieved in a new application. New applications always have some unique requirements, hence new code. They can help achieve reuse by carefully questioning each new requirement to verify it is in fact a requirement, not just a "nice to have" feature. And, finally, by including design reuse under this topic, even "new" code can benefit from the past.

### 4.1 Implementing Reusable Software

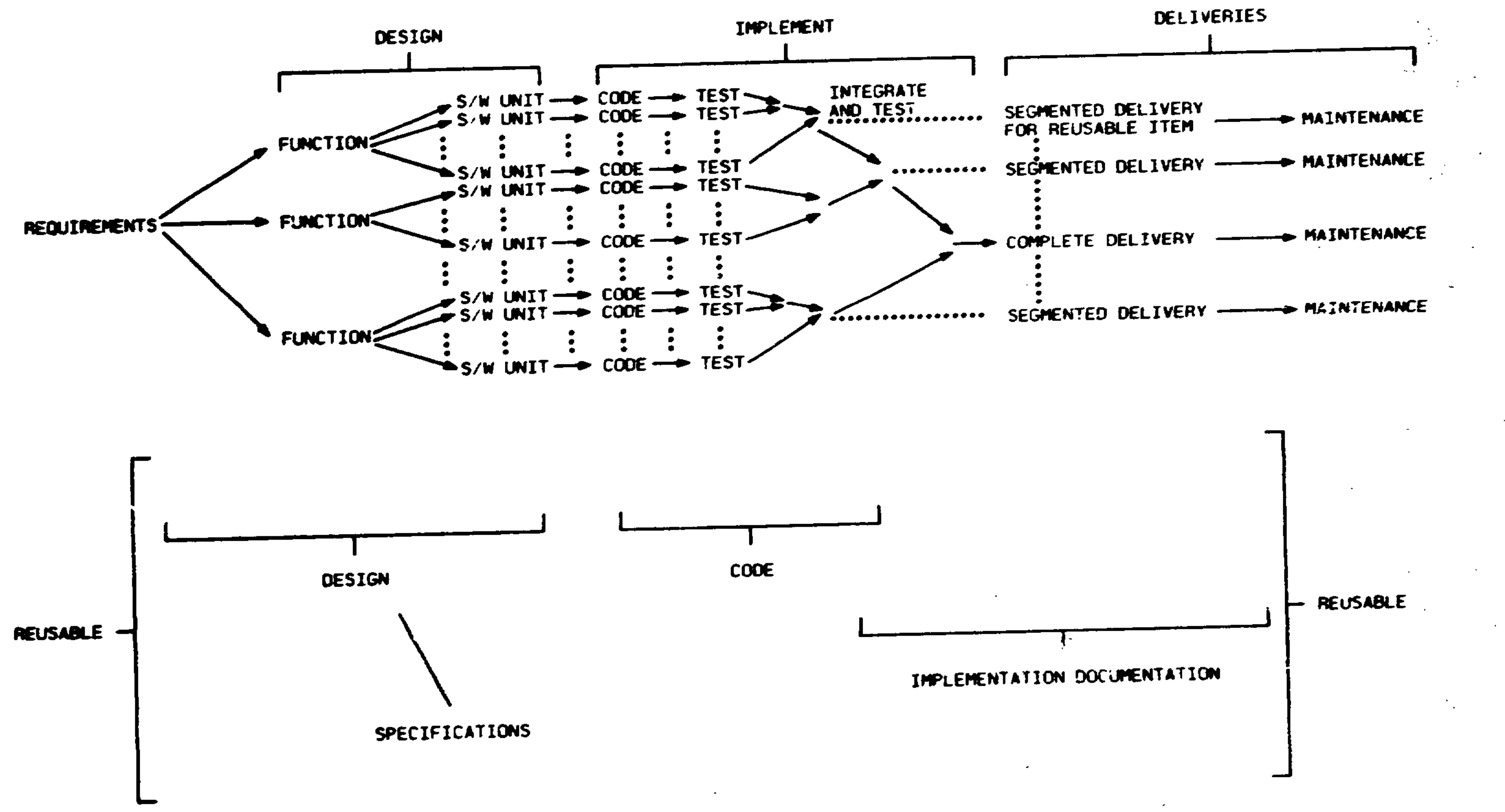
A typical software development cycle begins with the mapping of requirements to software functions, then performing functional design of software, and, in a top down manner, providing a successively more detailed design of software functions, modules and packages. The design "layers" are documented as specifications (for functions, programs, etc.). These are used to implement program elements (code). Two end products result from this process, namely code and "as-built" program implementation documentation. Figure 4.1 shows what is reusable in the project cycle: intangible design, and tangible code, specifications, and documentation.

Specific techniques for implementation of reusability are presented in Table 4.1. The techniques for generating RUS, which are described in this section, are generally different than current standard practice.

For purposes of discussing reuse issues, reuse of source code implies the reuse of accompanying program implementation documentation, specifications and functional design. Reuse of program specification material (without code reuse) also implies reuse of the underlying functional design. Reuse of functional design alone generally implies that new program specifications and new source code will result.

#### 4.1.1 Functional Design Reuse

Structure and Partitioning. Functional design, to be reusable, must be clearly identified, locatable, and understandable. For maximum benefit, the design must be structured and partitioned such that individual functions become portable software packages. A technology of structuring and partitioning requirements into functions needs to be developed. (At present, requirements allocation and functional partitioning is an art



E080 15226-1A

Figure 4.1 Reusable Software and the Development Process

TABLE 4.1 RUS TECHNIQUES/ISSUES AND CONCEPTS

Functional Design Reuse

- Structure and Partitioning
- Identification

    Requirements Fulfilled
    Generic Function Identification

- Location
    Cataloging Scheme

- Understanding
    Hierarchical Exposition
    Language

Specification and Documentation Reuse

- DID Format
- Specification Language, Standardization
- Investigate Documentation Automatically Generated from Code

- Machine readable interchange media
    Word processable distribution

Code Reuse

- Transferability (Peoplewise, assumes no personal contact)

    Readability
        Package specification
        Interface specification
        Procedure
    Modularity
        Packaging and Information Hiding
        Standards, Conventions, and Style

- Transportation (Systemwise)

    Language Standardization
    Toolset Standardization
    Package/Unit Sizing
    Flexibility
    Interface Standardization (Intermodule and I/O)

        Parametric
            Package Specification
            binding
            Conversion
        Tasking
            Timing Independence
    Assembly Language Considerations

process dependent on the software engineer's skill.)

Since RUS is intended to be put to a new end use after its initial application, a number of special technical considerations affects its implementation. A future end use will not be known at the time of initial implementation. Judgement must be exercised in the partitioning of requirements and allocation of functions to software components to maximize the probability of future reuse.

Identification. A unit of functional design is identifiable when it is described by both (1) the requirements it fulfills and (2) by the generic or specific functions it implements. As part of defining rules and scientific procedures for functional design structuring, identification criteria can be established. Providing consistent identification criteria for each end unit of functional design represents a new task for software designers.

Location. A unit or collection of units of functional design is locatable when the software designer can readily identify pre-existing software packages as candidates for reuse. A tool providing a multidimensional location catalog (index) is needed to provide location by identification criteria described above. Preparation of inputs to the catalog is performed by the functional designers as each design component and software unit is identified.

Understanding. Functional design is understandable when it can be understood from design specifications in separate design environments (e.g., different companies, without personal contact or communications with the original authors). Design which fails to be understandable is not reusable. Techniques for go-no go judgment of design specification understandability need to be established and procedurized.

To be understandable, a top-down hierarchical exposition of the design is mandatory. (Design is presently documented by introductions and functional details, without intervening levels). The time to read and understand design communications material should be minimized by clear top and clear successively detailed descriptions. Figure 4.2 shows how the number of levels needs to become larger than the two (top level and bottom) presently documented. Reuse of portions of design (such as algorithms and subprocesses) will be facilitated by specification material which allows the reader of a unit of reusable design to learn a minimum of application-specific details to understand internal subprocesses and subfunctions, other than those applicable to the specific portion which is the candidate for reuse. In the figure, in the past method, to understand detail 1.2 (presumably a reusable unit candidate), one needs to understand interfaces, which are to 1.1, and 1.3. Thus, much of the system must be understood. But in the future example, to understand 1.1.1.2, one need only examine its neighbors (if interfaced to) and the higher levels 1.1.1, 1.1, and 1.0. This reduces the level of detailed understanding a future software user need gain about interfacing units/functions. Techniques such as ICAM Definition (IDEF) methodology support a hierarchial exposition and should be explored for wider application.

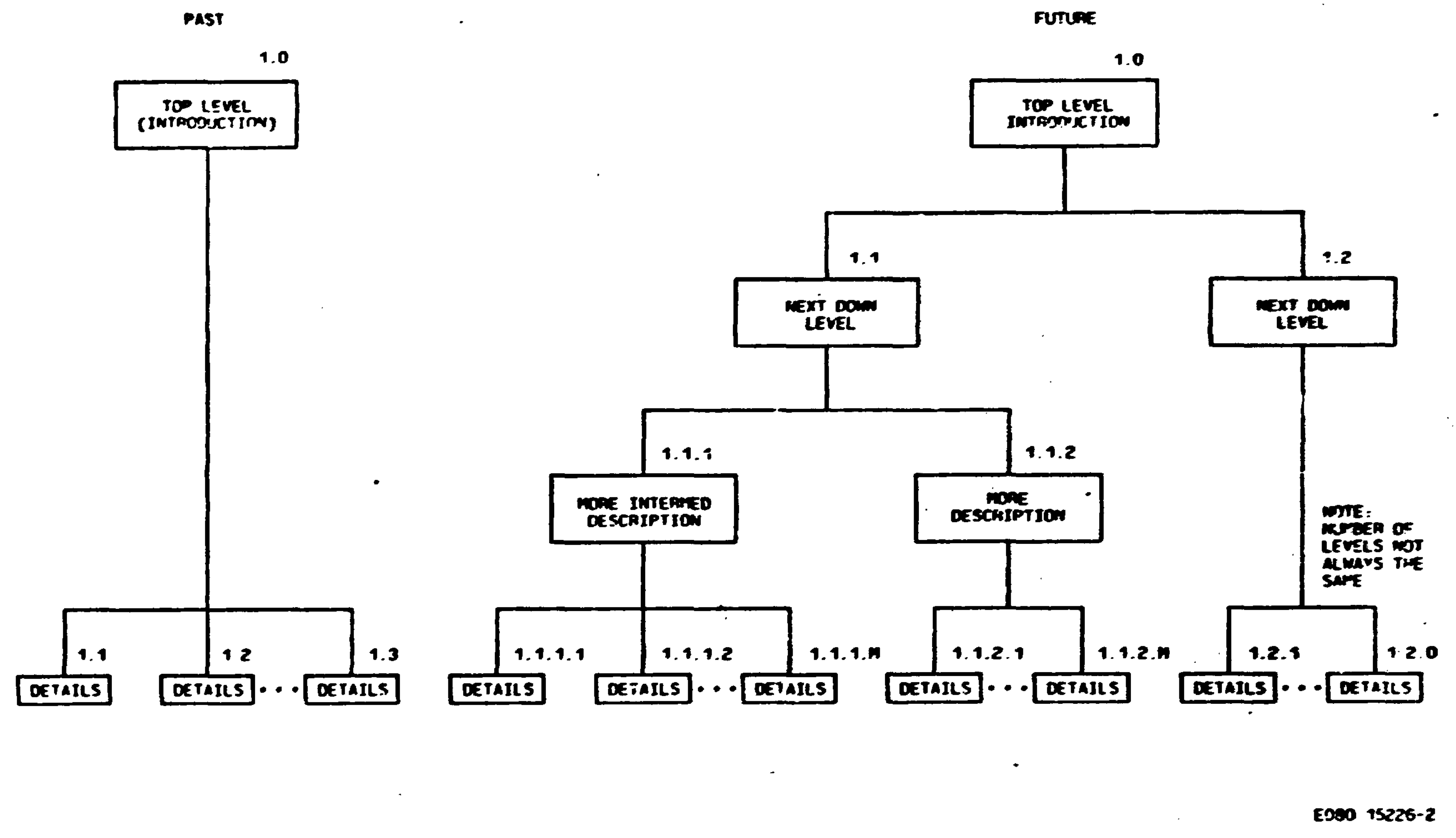


FIGURE 4.2. Top-Down Hierarchical Multilevel Design Description

Language. The textual, or graphical, means of functional design exposition is not at present constrained. For reuse, progress toward consistency or standardization is required. Significant advances are being made in the area of tools which provide languages for design specifications. As these language tools become standardized and accepted, a methodology for their use for hierarchical functional design is needed.

### 4.1.2 Specification Reuse

While specifications and documentation are semantically different, they have common attributes. As used here, specifications describe functional design, they "specify" the implementation of program code; documentation describes how the programs interface and operate, as built. Documentation is covered in Section 4.1.5. Specifications and documentation need to be meaningful conveyances of functional and detailed design, and of software implementation. This is presently difficult to implement on large DOD projects. Research into formats of design exposition (such as hierarchical and multi-level as previously described) is needed to identify tools and techniques which will provide support to reuse. The "writing-illiteracy" of many software designers has caused specification writing to be decoupled from code implementation which is often (even usually) decoupled from program description (documenting). The application of automated specification production tools and graphical computer-aided design documentation is needed to compensate for this problem. Use of specification languages and accompanying tools are a part of this need. The DOD should consider building and providing specification generation and maintenance tools, including a common specification language, to contractors.

Any attempt to reuse specifications of any sort is presently thwarted because (unless written by same contractor, same department), the physical appearance and organization of contents is inconsistent. The present JLC Monterey I and Monterey II efforts to produce a single set of Tri-Service DIDS for software specifications, when brought to a completion, will significantly aid reusability..

Specifications are not readily reusable when distributed on paper (hardcopy). A DOD-wide documentation interchange media is required which is word-processable (for text and "language") and graphically-processable (for non-text techniques, as used commercially). Procedures and techniques for incorporation of reused specification material with mechanical file transfer can then be considered.

### 4.1.3 Code Reuse

Source code reuse requires source code transferability (peoplewise, from an author, who is not in personal contact with reuser, to a reuser) and source code transportability (physical movement, from one host/target combination to another).

4.1.3.1 Transferability. Transferability requires human factors type considerations: readibility, modularity, packaging, information hiding, and use of standards, conventions, and style.

Readability. Present technology of languages and program organization provides two basic sections to a software package: the specification and the body. The package specification defines the naming, calling, and interface to the package's subprograms. The package body includes internal declaration and procedural code implementing the packaged subprocesses. Both the package specification and body need to be readable by reusers, but for different reasons.

The package specification, and in particular the included interface specification sections, needs to be readable by a future software engineer evaluating the package for reuse. A highly understandable description of algorithmic and control parameters is critical to the suitability for reuse. The package specification readability must address interfacing to new and other reused packages.

The package body is not meant to be examined by the average reuser. For DOD system maintainers, the package body may be centrally controlled by an issuing agency which may not even distribute source code to using agencies and contractors. If distributed, it may be only of use to a scientific, communications, or special discipline expert. Thus, the sophistication of the code body may be higher than that of a package specification.

In the real world, even if the package body is intended to be reused by machine code capture without source code, access to its body portion is needed during test, integration, and maintenance. Given that testing, or post-deployment retest, uncovers a problem with the package operation, a readable package body is necessary. The degree of sophistication and knowledge of the "body" reader will be higher than that of the package specification (interfacing) reuser, thus allowing sophisticated techniques of programming to be used so long as the transferability of procedure understanding can be conveyed in the absence of personal author-to-reuser contact.

Modularity. Modularity refers to the partitioning of a collection of functional operations within a reusable package. Modularity is an important transferability characteristic which encourages the physical partitioning of program code into small comprehensible units (representing, for example, single ideas). Limits on the number of lines of source code in a package enhance the ability of a reuser to comprehend its total scope. Limits on number of functions performed in a package encourage simplicity. Given compact size and simple functions, understanding and reuse is facilitated. Modularity of a package will result in modularity of the package specification (the user desription and interface section) and modularity of the code internal to the package body.

Packaging and Information Hiding. Given functional modularity and good partitioning of the package's internal operations, the reuser will have access to individual functions as needed. Information hiding is implemented by strict separation of package specifications from bodies and by judicious unit partitioning. Given that a package is a distributable collection of software subprograms, careful attention must be given by a system's designers to the partitioning of functions to packages. Like functions (which may map to diverse requirements) should be candidates for co-packaging. Machine-dependent operations should be candidates for mandatory separate packaging from generically transportable functions. A formal methodology for packaging must be developed in order for reusable software to be developed.

Standards and Conventions. The issues of standards, conventions and style is a natural one to code transferability. Examples of standards are: language; interfaces (interprogram and input/output); modularity; design specification; documentation; and programming environment. Standardization of these types of issues enhances understandability of both package specifications and procedural code.

Conventions govern those issues which cannot be edicted by a standard. Examples are: naming conventions, formatting of listing and commenting. Universal standards and conventions will be required in order to transfer software between contractors. However, it is not feasible to consider that all software developers could agree on all standards. Therefore, effort is required to isolate those standards and conventions which most effect transferability.

4.1.3.2 Transportability. Transportability requires mechanical compatibility from original system to reusing system. Standardization of language, toolset, and interfaces facilitates transportation.

Language Standardization. Language standardization is an important consideration for code transportability for both physical movement of code to the computer of a new system and for the inability to translate between languages (the language issue is fully discussed in 4.1.4).

Toolset Standardization. Toolset standardization is needed so that a package's author's environment is available to the reuser. The Ada effort presently is experimenting with a standardized toolset, the Ada Program Support Environment (APSE).

Interface Standardization. This issue addresses creation of package specifications which utilize standardized interfaces. Interface considerations are separated into parametric interfaces (the passing of data values) and tasking interfaces (the timing independence and scheduling coordination, e.g.Ada "rendezvousing") of sections of program code.

Parametric interfacing is enhanced by information hiding, as mentioned above, as well as value format and parameter order conversion. These require use of HOLs which perform parameter conversion, reordering, and subroutine acceptance of parameters somewhat independently of caller specifications. Other parameter interface issues which require attention are naming conventions and parameter conversion such as from fixed to floating numerics or scaler to array.

The reuse of time-dependent software and the problem of coordinating software items within their native task structure environments may result in practical limitations to widespread reuse of such items. Further study is required on this point.

Assembly Language Considerations. Assembly language issues cover ISA standardization, package separation, and common data item reference: ISA standardization is needed if reused packages are to be allowed to include assembly code. Package separation is necessary so that assembly language (machine-dependent code) is physically removed from transportable machine independent RUS.

### 4.1.4 Language Issues

In looking at the technical factors which influence the degree to which software is reusable, it becomes evident that language issues must be considered as a primary factor. The proliferation of languages, dialects of languages, and hence language processors, give rise to various syntactic and semantic differences which preclude, to a degree, the reusability of software. Although it is expected that the transition from the use of existing "standard" languages to the use of Ada will alleviate the problem significantly, the language problem may continue to persist unless very strict controls are implemented and enforced with respect to both the Ada language and its compiler(s). It is incumbent upon the responsible DOD support software agency to take forceful action to ensure that "dialects" of Ada are not permitted to occur. With respect to existing languages DOD program managers must take action to restrict the use of language features in the various dialects to the common subset, if it exists.

Similarly, reusability is affected somewhat by both the magnitude and the frequency of changes which are applied to existing languages to incorporate enhancements or new features. The latter is especially a problem in the event that there exists multiple dialects of the same language, often managed by separate agencies, and enhancements are not incorporated in each dialect. The result clearly is language divergence and, thus, the ability to interchange software elements between projects using the different dialects is greatly diminished.

Another factor which affects RUS is the degree to which the language supports coding of required functions at the high order source level. If the language features are such that the software engineer is forced to resort to the frequent use of assembly language, then reusability is affected adversely by the resultant machine dependencies. It is incumbent upon the responsible DOD support software agency to ensure that the language supports the coding of required functions. Otherwise, and for existing languges which are not adequate, software developers must isolate machine dependencies in order to achieve reusability.

A point is frequently made that the use of assembly language is necessary in order to achieve efficiency. The argument most often prevails with respect to operating system software where inefficiencies give rise to system overhead, in terms of both memory and process[illegible] Although valid in many instances, it is nevertheless the case that coding in assembly language results in machine dependencies which, again, tend to minimize software reusability. With regard to this issue, the DOD Ada Program Office must provide compilers which generate code that is efficient enough to permit the development of time critical processes in HOL.

At the current rate of hardware technology advancement, it may be that the hardware will compensate for tremendous degrees of inefficiency per today's standards a lot sooner than the compilers can make very significnat gains in efficiency. At any rate, the short term solution to this will be for designers to identify and locate packages with the highest probability of reuse and ensure those packages are implemented in higher order language.

Another factor which affects the rapidity with which RUS is achieved, assuming Ada is all it is expected to be, pertains to whether or not a capability is provided to DOD's program managers to incrementally upgrade Ada. Since an enormous amount of software exists today which must be supported for many years to come, it might be worthwhile to consider the implications of supporting mixed-languages in the Ada support environment. However there is a trade-off of providing this capability or hope that the other current standard languages might "die a natural death" much sooner. If the mixed-language concept is not supported in the Ada environment, and if we are not successful in upgrading existing systems to Ada, then the languages currently in existence will require support well beyond the year 2000.

### 4.1.5 Documentation

Documentation requirements in an environment of RUS are more elaborate but less labor intensive than with conventional non-reusable software. The techniques to support the producing of RUS require a number of new items of documentation. They also require consideration of new tools, to support production, distribution, and reuse of documentation.

Documentation considered by this section includes but is not limited to:

Functional Design Specifications

- Functional Performance Specifications
- Hierarchical Design Specifications

Software Implementation Documentation

- Program Design Documentation
- Code Documentation (Embedded Comments)
- Interface Design Documentation
- Traceability Documentation

User Documentation

Testing Documentation

Maintenance Documentation

Tri-Service DIDs. A single set of documentation formats is required to implement reusability of documentation. A single responsible agency (such as the JLC) will need to consider DID evolution in support of an emerging technology for reusability. A primary reason for having a single set of DIDs is so that all services can reuse packages which may have been generated for any other service.

Improved Design Documentation Methodology. Two considerations for improved design documentation standards are hierarchical multi-level design documentation, and use of a standardized language for textual design descriptions. Each package specification segment should have a brief summary as to inputs/outputs required, linking information, who wrote the software, and what it does. Attention must be given to package specification formats which are not subject to design and coding technology variations. Thus, material can be extracted from one application and inserted into the deliverable documentation of another without requiring rewriting.

Computerized Tools. A number of advanced capabilities are required to facilitate reuse of software documentation. These entail machine distribution of word processable text (including language prose, if any) and of graphically processable figures. In addition, tools will provide an automated capability to edit and incorporate revisions to reused documentation sections as they are issued by the agencies responsible for reused material in a new end item.

Documentation Traceability. An absolute essential for RUS is the traceability of documentation. A package of code must have a traceable, identifiable document action history. When a system segment or package is used, documentation segments may also be reused. This will have other benefits that are a requirement for the ability to reuse software. If an analyst is to reuse segments or packages of application software, he must know the function and parameters of that package. Given good documentation and organized cataloging, the reuser will know what is available and the functions the segments and packages perform.

Two items of special interest in traceability and derivation are identified.

a) Traceability of Requirements. If one views the tier of documents as successive levels of requirements, then the purpose of each source statement can be associated with a requirement. For RUS, we need to be able to trace from system requirements to a source statement. This may require some additions to the documentation standards.

b) Derivation of Family Tree. To reduce the documentation that would result when a module is modified for a slightly different function, reusers must keep a very precise record of the derivation of the modified module. This need and the general need for documentation of the smallest modules probably mean that CPCIs will be established at lower levels than normally required.

The two above items may require further work in the areas of CPCI definition and documentation standards.

Automated Test Support Documentation. This topic covers a data base of automated test sequencing, control, and analysis data; the tools which automatically test based on this data, and reports produced by test tools using such data. The test tools should operate under much greater automated control then at present. The purpose of automated support documentation is to facilitate automated retest of both the configured end item (its components and integrated segments) and retest of the reusable system segments delivered to a RUS library. Automated retest should occur automatically upon change to internally authored code and upon receipt of updates to reused segments of code, specification, or design.

### 4.1.6 Tools

Software support tools are an important part of present-day (1981) software projects, but will become even more important in future projects which are both consumers and producers of RUS. Table 4.2 characterizes specific needs for tools identified in this section. It is neither complete nor exhaustive but aims at identifying the minimum set of tools required for software reuse. As noted in the Table, many new tools will be required to effect an efficient transfer of software between developers. Investment will be required to research and develop these tools.

All support tools must themselves be transportable and, when needed to support application of source code (e.g., preprocessors), rehostable. The languages supported (if several) must have the capability to produce object code linkable to that of each others language, and usable on different target machines. Standardizing on a single language would eliminate the need for language compatibility and would eliminate the redundant effort and money expended to maintain and enhance older incompatible translators. By restricting the software development process to the target computer, we naturally inherit its limitations in terms of both hardware resources and support software tools and, thus, frequently provide to our software developers an unsatisfactory environment within which to generate software. This results often in less frequent recompiles and, consequently, greater reliance upon machine-dependent code in the form of patches to resolve software errors. The problem is exaggerated when source libraries are not kept current, when source libraries are maintained at sites remote from the integration facility, and when developers are not forced to incrementally recompile programs. Transportable standard support tools capable of being hosted on a variety of commercial systems will help to alleviate this problem.

The concepts of RUS must be applied in entirety to support tools. With the increased dependence on support tools that reusable application programs will require, it is mandatory that support tools be controlled, and certified in a very thorough manner. The tailoring is inevitable, but it must be controlled such that it does not result in unique machine or application dependencies being inserted into the application programs.

### 4.1.7 Recommendations

There was a consensus among this group that attempts to reuse present-day existing software are likely to meet with failure. Widespread reuse of software will require that new software be developed with a requirement that such new software be manufactured for reuse. Hence, software modules, segments, and items must be identified as candidates for reuse at their initial creation based upon their inherent potential for reuse, the expected requirements, and other economic factors. The comments of

TABLE 4.2

SOFTWARE SUPPORT TOOLS IN REUSE ENVIRONMENT

| | | |
|---|---|---|
| 1. | Catalog and Librarian | Source and recipient of reusable software (Code and Specification). Includes indexing system*. |
| 2. | Computerized Documentation | Standardized text processable and graphically processable distribution needed. Used for functional design*, program description, and user documentation. Allows selected or complete reuse of text and figures from pre-existing documents, with automatically repeatable editing and update* (e.g., so revised reused specs can be automatically reedited and updated). |
| 3. | Configuration Management | Tracking and control of reused components; tracing of elements in reusable system delivery segments; control of automatic retest*, code and documentation update upon internal or reused item changes/updates. |
| 4. | Languages | Standardized transportable design language*. Standardized transportable retargetable programming language. |
| 5. | Standard Utilities | Standardized transportable toolset*. Interhost file and media conversion vehicles*. Standardized interactive command language*. (Includes linkers, binders, and loaders for reused (object) components.) |
| 6. | Verification and Audit | Specific tools to enforce standards and reusability. |
| 7. | Computerized Testing | Computer sequence unit test and integration testing*. Repeatable automatic retesting upon revision of reused packages. Testing both to end item configuration and for deliverable reusable system segments. |
| 8. | System Generation | Automated system generators (for producing configured end-item of software objects). |

*Denotes tools not currently available in present state-of-the-art, they will require new development.

Section 4.1 would then form the methodology by which the new software would be developed to achieve the reusability.

Specific recommendations are:

1. Structure and partitioning methods must be developed including functional design description by hierarchical exposition.

2. Software unit packaging, modularity rules, coding standards and information binding concepts require exploration and standardization.

3. Interface standardization criteria (Intermodule and Input/Output), including parametric interface (data passing, binding, and conversion) and tasking interface (timing/scheduling, flow of control and coordination) require development.

4. A standard for identifying functional units (at the design level) by (1) requirement and (2) generic/specific function is needed. The methodology of using centralized catalogs of reusable software components/functions must be developed.

5. Use of, and standardization of, specification language and accompanying tools is needed.

6. Develop and standardize a support tool environment which emphasizes transportability, multiple target machine code generation, and a standard support tool interface. In some agencies, this is currently underway and should be supported.

7. A rigorous certification of support tools and reusable application software should be performed.

8. Require that support tools be identified during the design phase of application software with consideration given as to what tools exist and the re-application of project-specific tools to other projects. A controlling board or standard should be created to enforce these requirements. Project-specific tools which are needed to reuse source code (e.g., preprocessors) must be transportable and rehostable.

9. A study should be initiated to identify new management support tools needed for control of the developed software intended for reuse such as computerized generation and dissemination of specifications, verifications and certification of adherence to reuse standards.

10. For existing standard languages having multiple dialects, restrict software development to a common language subset.

11. For the ultimate set of standard languages, the responsible government agency should maintain strict configuration control to ensure that multiple dialects are not permitted to occur. Also, the agencies should ensure that coding of all required functions is adequately supported at the source level (without having to resort to direct code).

12 Determine the feasibility of supporting multiple languages in the Ada Support Environment (APSE) and thereby permit incremental upgrade of current systems to the ultimate language set.

13. Compare maintenance documentation requirements to reusability documentation requirements to determine what additional requirements exist.

14. Investigate and develop traceability methods/tools such that:

    a) System requirements can be traceable down to a source statement.

    b) A precise record of the module derivation can be kept (parent-offspring relationships).

## 4.2 Managing a "Ported" Development

The purpose of this subpanel was to identify the differences that exist between the management of a development project employing some amount of reusable software and the management of a development project employing no RUS. All areas of project management that required special emphasis were identified. In the subpanel, all project management functions were analyzed, and it was determined that there were differences or special emphasis required in the following project management areas:

- o Project organization
- o Project control
- o Configuration management
- o Quality assurance
- o Testing
- o Documentation
- o Library

Three conditions of RUS were considered. They are as follows.

1. The new development project reuses software developed within it's own company.

2. The new development project reuses software supplied by the Government as GFE.

3. The new development project reuses software developed and resident within a company other than the company using it.

### 4.2 1 Reusable Software Organization (RUSO)

Within each system or software producing entity, both industry and government, an organizational structure must be established which specifically addresses the reuse of software. The objectives of the RUSO are to establish an effective company-wide reuse program. The goals are: reduced development time; reduced duplication of effort; improved system reliability; and reduced costs. An effective RUS program will ensure that productivity on software development projects is improved and the cost of software is reduced. The effective RUS program will also cause generated software components to be likely candidates for future reuse.

Initially, the RUSO should review current software development practices and determine where standardization for reusability would have the greatest payoff in future reuses. In the beginning, software packages containing procedural routines usually offer the highest payoff for standardization efforts.

Upon identifying areas for standardization, the RUSO should evaluate support software in use for its appropriateness as a "standard development environment". In the likely event adequate development environment tools do not exist to support reuse, the RUSO shall undertake to procure, develop, and otherwise install the required tools.

As the RUSO matures in its function, serious consideration should be given to developing applications software components in anticipation of future systems/programs needs. This requires that the RUSO coordinate with business planning and marketing groups to identify areas in which to concentrate the building of reusable components.

The system documentation of designated tools should be maintained by RUSO. Such a practice helps to ensure software maintenance in the event original software authors depart the organization. The documentation of reusable components, usually in the form of the user guides, will be critical to the overall success of the RUSO. The documentation will be the primary mechanism for transferring reusable software throughout the programming community. The RUSO should employ a professional technical writer to ensure that the documentation quality is of a level that will be readily accepted by the organizations' programming community.

The effectiveness of the RUSO program should be measured on a routine basis to ensure that the RUSO is meeting pre-established goals. Productivity measurements may be obained by developing baseline estimates on the amount of time required for the various stages and functions of an application development. For example,

Man Days for Data Handling Routines (Storage & Retrieval)

| Complexity | Design | Code | Debug |
|---|---|---|---|
| Low | 1 | 1 | 1 |
| Middle | 1 | 2 | 2 |
| High | 2 | 2 | 4 |

Baseline estimates for comparison would be from past performance on prior contracts for each specific application. A specific set of baseline estimates must be prepared for each discrete application stage or function. All future projects should be monitored for their performance against these baselines to determine the productivity gains.

### 4.2.2 Project Control

The actual cost of employing RUS on a new development is valuable information to have from both an historical perspective and from a software cost control perspective. It is anticipated that there will be a cost savings by employing RUS, however, the actual savings is not known at this time. The Project Manager of the new development requires up-to-date information on each RUS component used so that he can make informed continuing estimates of project cost and schedule. Additionally, from a corporation or government standpoint, it is very useful to have a history of the actual costs of various RUS elements.

With these requirements in mind, it becomes imperative that the RUS to be used in the new development be included in the WBS in such a way as to allow the tracking and collection of RUS costs. For instance, each piece of RUS should be identified as a separate work package. Thus, historical information will be gathered which could be used to estimate future costs and cost savings.

### 4.2.3 Configuration Management

Prior to the start of any project using RUS, the corporate CM office and the project CM office must prepare for the RUS environment. Existing CM plans, policies, and procedures must be revised to accommodate the employment of RUS. This effort will be accomplished in coordination with the RUSO in addition to normal coordinating efforts such as with other company offices.

The RUSO should establish a separate and distinct library for the RUS (see Section 4.2.7). Procedures and controls must be established, as described in 4.1, such that anyone can access the contents of the library sufficiently to determine if there are library elements which would be of use to the new development. The library also addresses recapture of the new software components developed for reuse on future projects. Of paramount importance in the library is ease of access by future development projects, as covered in 4.1.1, under "Location".

The RUS library components should be identified by the referenced location scheme. Changes made to an RUS component will cause the changed element to be re-identified. This facilitates automatic retest of systems employing the changed component.

RUS supplied to a contractor by the Government would be identified by the RUSO using the same identification/location scheme used to identify other GFE.

The prime responsibility for configuration control rests with the RUSO. This is accomplished by establishment of a RUS Change Control Board (RCCB) whose charter would be to maintain the integrity of the RUS and its library. Any change required to be made to RUS by a project employing the RUS must be submitted to the RCCB in the form of a change proposal prior to the change being made. The RCCB is the sole authority for the approval, disapproval, or deferral of proposed changes. Figure 4.3 identifies the change processing flow for RUS changes required and

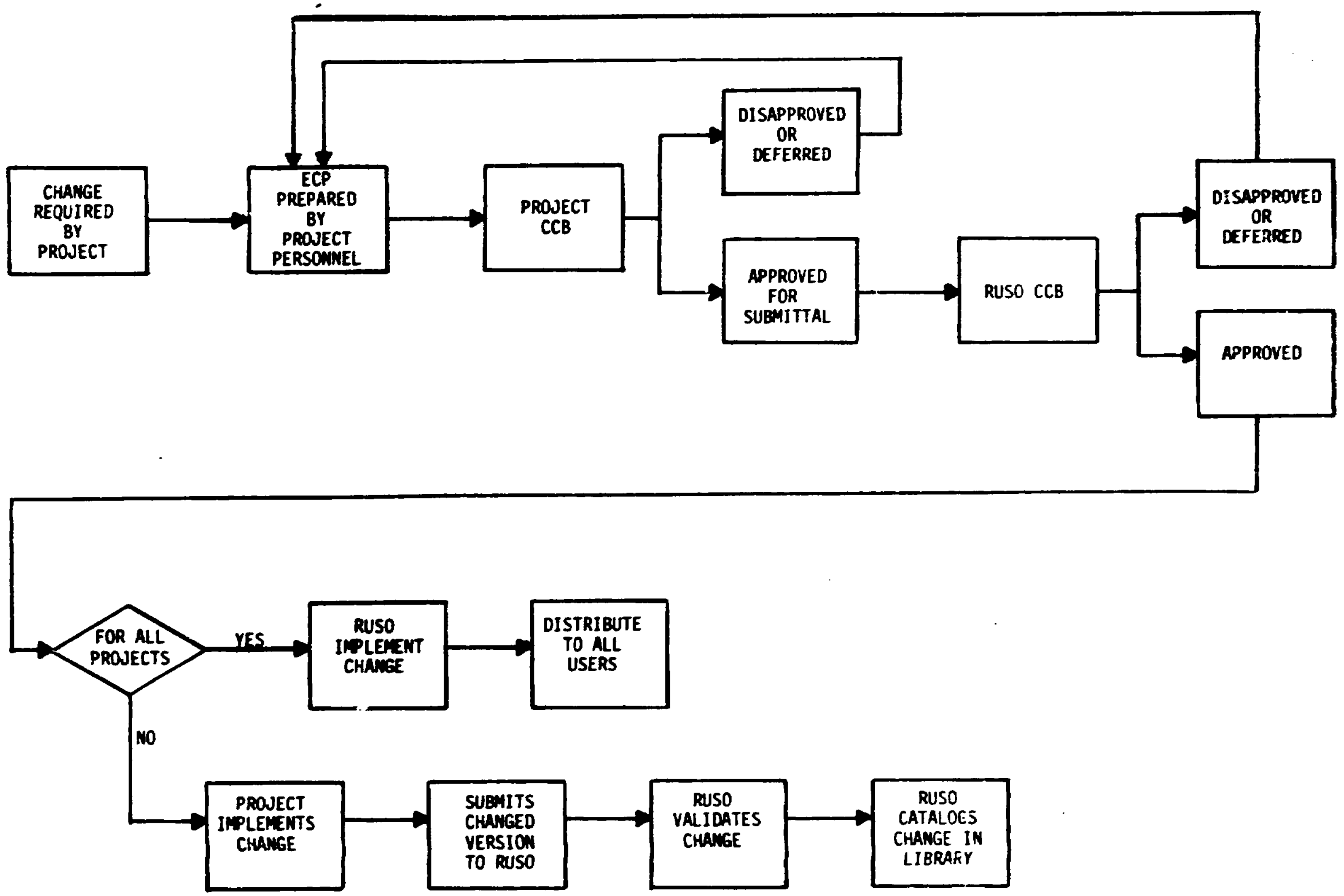


FIGURE 4.3. CHANGE PROCESS FLOW

proposed by the project employing the RUS. If the change to be implemented is to be made to every project employing the RUS, the RUSO will accomplish the change and submit the new version to the users. If the change to be implemented is to be made only to the single project RUS, the project will implement the change and submit a copy of the changed RUS to the RUS library. The RUSO should validate the change and then incorporate it into the library.

It should be noted that if a project proposed change is disapproved by the RUSO, the project can still implement the change but the RUSO will probably then not continue to support the RUS.

### 4.2.4 Quality Assurance

Software Quality Assurance (SQA) functions remain largely the same in the RUS environment. However, additional plans, policy, and procedures may be required to accommodate certain differences. The company SQA office, in coordination with the RUSO, CM, and the software department, all should assist in establishing SQA criteria and attributes which will qualify software to be identified in RUS.

The SQA Group should review, evaluate, and verify the RUS package for completeness and compliance with the company-established SQA criteria and attributes for RUS prior to placement of any elements into the library. Upon receipt of an RUS package from a source external to the company, the SQA Group should participate in the evaluation of the RUS package for completeness and performance as stated.

### 4.2.5 Testing

One of the benefits of employing RUS should be reduced component-level testing. The degree of reduced testing will vary depending on several factors. The source from which the RUS comes appears to be one factor which will affect the amount of testing conducted. If received from the government, the RUS is more likely to have been subjected to rigorous testing and therefore will likely require less testing by the recipient.

The documentation and tests available to the RUSO should include a complete test package from the development organization which initially produced the RUS. This package should include the RUS test plans, procedures, results and reports. It may also include test programs and drivers, test scenario, etc. This allows the reuser to assure that the tests cover the intended scenario of operation.

In the case in which the project company receives the RUS from another company, additional testing of the RUS is probably required to establish a performance benchmark or system baseline. The test group will probably have a bigger task in adapting and integrating the other company's RUS test plans, procedures, etc., into the project test plan and procedures. The size of the task will depend upon the amount and quality of test documentation received in the RUS package. Inter-company standardization of test plans and procedures would help alleviate this expense.

The approach to formal testing of RUS should be that of testing the RUS at the highest level of testing first. If the RUS portion of the project software fails at the higher level of testing, then regressive testing would be conducted on the RUS to isolate the erroneous elements.

### 4.2.6 Documentation

A key responsibility of the RUSO must be to provide documentation for the RUS which is suitable for adapting to each project. Thus, a standard set of documents must be provided for each RUS element. The documents should be a superset of those normally required by the companies' customer base. Included in the documentation should be an abstract designed to quickly orient the potential RUS user to the specific element. This documentation required by the RUSO with each RUS element should be determined and identified in RUS standards and procedures developed by the RUSO. Section 4.1 discussed specific documentation practices to promote RUS. When the RUS is changed because of project requirements, the documentation must be submitted to the library along with the copy of the changed code.

When providing RUS to a contractor, the government has the responsibility to provide comprehensive documentation with the software elements. As updates are made by the government's original source, the documentation on these changes must be passed on as well.

NOTE: In general, whenever the government puts themselves into the position of providing software to contractors, they must assume the responsibility for on-going baseline control and dissemination of that same software to every re-use application.

### 4.2.7 Reusable Software Library

A RUS Library must be established by the RUSO to retain master copies of each piece of the RUS along with copies of the necessary documentation. As each RUS is changed, the library will ensure that updated documentation and updated masters replace the existing masters in the library. Additionally, the library must ensure that a historical record of the RUS is kept.

The RUS Library is responsible for the distribution and dissemination of RUS required by individual projects. Additionally, as RUS is revised by the RUSO, the revised RUS and revised documentation is distributed to projects employing the RUS.

One of the tasks that the RUS Library must accomplish is the development of a system for storage, numbering, indexing, etc., to ensure not only an organized filing system but also an organized and efficient method of retrieval. The RUS Library will index the RUS stored in the library. The library should periodically distribute throughout the company the index of RUS to ensure that company personnel are aware of the RUS library contents.

### 4.2.8 Recommendations

The following specific recommendations resulted from this subpanel:

1. A Reusable Software Organization should be established within each large organizational entity, to effect the reuse of software. Their responsibilities should include standards and methods for developing reusable software, enforcing documentation standards and controlling the reusable software. In addition, the RUSO should be concerned with performance measurement to assure the reuse applications are cost effective.

2. Project control mechanisms must be adjusted to account for RUS. Specifically, the WBS must allow for visibility into the RUS.

3. Configuration management for RUS must be centralized with the RUSO. As changes are needed, they should be individually approved and incorporated by the central organization, disseminated to all users, and accounted for as distinct changes. All of the RUS must be retained within a central master library administered by the RUSO. The library would be responsible for control and dissemination of all packages and for historical records related to reuse of the software.

4. Testing and documentation methods should be examined in RUS applications. Automated regression tests and high level testing have an increased importance. A standard modular set of documents must be available for each RUS package. Testing documentation needs to be portable along with the software itself.

## 4.3 Learning from Past Experience

Over the years, some members of the commercial business sector have had excellent success in utilizing the reusability technique in the development and maintenance of business application software. This section examines the lessons learned from these experiences; addresses contractual considerations for the planning, award and performance of the system and software contracts; looks at documentation needs and user groups; and provides recommendations to the DOD community for achieving greater reuse of software.

A case study of the reuse of business application software is presented in Appendix B. In brief, this specific experience is as follows:

> During the last five years, one division of a large company developed, implemented and perfected a reusable design and code technique supported by software packages, tools and services. The concept behind this methodology was to accelerate application development through the elimination of redundancy in the software cycle. Techniques of functional modularity are employed to prepare modules for use in multiple applications. These modules are designed, coded, tested, documented, certified, and stored in a library for later use. To date, a central library, containing over 2500 certified reusable modules composed of 76,000 lines of code, has been built. The payback for this one time investment is close to a million lines of reusable code. This technique has largely been responsible for the development of eight new data base management applications with an average of 60 percent reusable code. In addition, the methodology imposes a top-down approach to development

in which high-level logic is designed and coded first. The result is the production of logic structures, prewritten for each of six types of business application programs. Each logic structure serves as a "working shell" to be completed by reusable modules and unique code. Over 3000 new programs have averaged 42 percent reusable code through the use of logic structures. By applying these techniques, it was found that 40 to 60 percent of the redundancy in business application development can be eliminated.

This implementation experience could serve as a logical model for future reusability programs for the DOD. Most of the experiences described below come from this successful program and from information exchanges among businesses which have implemented or are contemplating implementing reusability programs.

### 4.3.1 RUS Development Philosophy Lessons

The implementation of RUS methodology forms a system to develop systems. Because it is a system itself, the RUS methodology should pass through the same development activities as any other system - from identifying the project goals to installing the system. The main project goal for the development of an RUS methodology is to formally integrate structured technology (analysis, design, coding, testing, and documentation techniques) into a set of practical guidelines - a system, if you will, for developing automated software systems.

Experience in implementing these structured techniques has indicated that these guidelines must be used in order for the reusability concept to work within an organization. It is difficult, if not impossible, to create and reuse software unless structured technology is utilized. However, experience has also shown that some business systems analysts and programmers find it difficult to adapt to structured technology principles. The reason behind this problem is that their past training and experience have conditioned them to think in a procedural fashion rather than in a hierarchical, functional way. One solution is to identify and train personnel who can adapt to this kind of structured environment and then use them in a pilot project. Another strategy is to institute an entry-level training program. The program should concentrate heavily on structured and reusability techniques. It was found that an entry-level training program is one of the best investments an organization can make when trying to overcome the inherent problems of using structured technology and reusability concepts.

In sum, the RUS methodology used to standardize system development activities should impose a structured, disciplined approach on the development process and enhance communication between all members of the project team and organization.

### 4.3.2 Installation Considerations

More important than the formulation of project goals is the proper installation of RUS methodology. (If a system is not correctly installed, then it may not be used at all!) There are four major factors to consider when installing RUS methodology. Listed in order of importance, they are: political/psychological factors; installation and training; the training medium; ongoing support and monitoring; and consulting. In the following paragraphs, these considerations are discussed briefly.

Political and Psychological Factors. These factors can profoundly affect user's acceptance of RUS methodology. The most important consideration is the commitment by high level management to the installation of RUS methodology. In order to enlist their backing, managers must be educated in the advantages of using a new methodology. It is important that the methodology be adopted throughout the organization, because isolated groups using the methodology, or part of it, will not be able to communicate effectively.

The second consideration is the potential feeling of loss of freedom or of power, as well as the natural resistance to change, on the part of new users of RUS methodology. To combat this resistance, the methodology must appear simple to use. This, of course, depends on the training, tools, and services (such as an automated data dictionary) provided to support the methodology. The quantity of forms and control procedures that a company assigns to go along with the methodology will also affect its complexity. It's best to keep procedures to a minimum until personnel become accustomed to the methodology.

Finally, there is the possibility that users' acceptance of RUS methodology will be extreme, and that the methodology will be taken too literally. Some data processing personnel may look upon it as a "cookbook" solution to all their problems, instead of as an aid to their own thought processes. This could potentially lead to the problem of the developers supporting the methodology, as opposed to the methodology supporting the development process.

Aware of these considerations, the following steps can be taken to minimize their effect:

- o Classify the methodology as a guideline rather than as a standard and allow users to customize it whenever problems are found in its use, thereby allowing the methodology to be viewed as a help, not as a restriction.
- o Set up a review group consisting of managers whose departments will use the methodology, thus giving them ownership and a part in the decision to accept the methodology (beware, however, that reluctant managers may make any meeting of this group their political battleground).
- o Prove that the methodology works by offering statistics from other users of the methodology or results of a pilot project.
- o Send key personnel to reusability seminars and conferences. For example, the IBM Guide User's Conference has a project on reusability.
- o Carefully consider methods of training new users to apply the

methodology.

Installation and Training. There are four general strategies that can be used to install a methodology and to train its eventual users.

- The Prototype Project Strategy - The installation is tested by applying the techniques on a reasonably-sized prototype project, and the resulting documentation from the project is used as support for adoption of the methodology. The developers of the prototype project are then used to assist or even train members of other projects. Experience has indicated that this usually is the best approach for gaining acceptance of the reusability concept in most organizations.

- The Walk-Before-You-Can-Run Strategy - Training in the methodology consists of teaching only those techniques that will be used immediately by members of new projects. For example, a course in structured design techniques is held before structured design activities; a reusable design course before reusable design activities, and so on. This training is repeated for each new project team.

- The Functional Job Description Strategy - Training courses are held for each category of employee (analyst, designer, coder, database developer), starting with the course applicable to the specific role and proceeding to courses on activities with which the employee may need to interface. A programmer, for instance, would take a course in reusable coding techniques first, and a design overview course later. This is a realistic strategy for groups that are functionally organized.

- The Educate-the-Masses Strategy - In this approach, all available staff members are trained when a course in any of the techniques is scheduled, regardless of whether they will need to use the techniques in the near future. This strategy also advocates training every employee in all of the techniques at once. The approach is usually not successful unless the personnel can apply the techniques on a project at the end of the training.

As with any system development effort, a plan should be developed to monitor and control the installation of RUS methodology and its associated training. Realistically, a combination of the above strategies will be needed for most (large) environments.

The following list denotes some training requirements for each level of participant:

- High-Level Management - Requires an overview of the reusability concept.

- Middle Management - Requires an overview and possibly an introduction to the structured techniques

- Project Leaders - Require a working knowledge of the complete methodology.

- o Technical Project Staff Members - Require a detailed working knowledge of the activities for which they are responsible and possibly an overview of other activities.

Training Medium. How RUS methodology is taught will greatly affect its understandability and, therefore, its acceptability. Although cost will probably determine the training medium used, some options include a workshop course, a lecture course, and a self-study audio-visual course.

A workshop course is one of the best forms of training, as it provides the staff with hands-on experience in applying a technique with immediate feedback to questions. Experience has demonstrated that an entry-level workshop training program can provide the foundation required to implement a successful reusability program. Personnel graduating from an entry-level training program can then be assigned to experienced personnel who will believe in reusability techiques. This approach has worked out extremely well and is cost effective. A lecture course also provides immediate answers to questions and is usually easier and cheaper to present, but doesn't offer participants the chance to practice using the techniques. A self-study audio-visual course allows easy assimilation of material and is a user-friendly medium.

All of these media can be backed up with assistance from outside consultants and on-the-job advice of experienced personnel. When using this approach, the best results come from using a combination of training vehicles, such as a workshop course with follow-up visits from the consultant/advisor.

On-Going Support and Monitoring. RUS methodology, like any system, should be maintained to keep it from becoming out-of-date. A support group or possibly one person must be responsible for this task. The importance of the task should not be minimized for the data processing industry is developing rapidly; new software development and support techniques are being introduced in virtually every phase. For the methodology to survive, it needs to accommodate pertinent innovations, but the process of incorporating them must be controlled according to the same principles used in the original methodology. This rule applies to any customizing of the methodology, either for the organization as a whole or for a particular project. The effectiveness of RUS methodology should be tracked and compared with results using the old technique to determine the cost-effectiveness of the methodology. Based on these results, adjustments to the methodology should be made when necessary. Consequently, measurement data must be collected before total installation of the methodology has taken place.

Consulting - Finally, a point to stress is the need for support consulting. An expert in the methodology (someone who has used the techniques, in a project or in the original training group) should be available to help out on new projects to ensure that the methodology is being used correctly. This can be accomplished simply be having the advisor accessible during reviews or walkthroughs.

In summary, the installation (and on-going support) of any system requires a great commitment of cost and effort. A RUS methodology is no exception.

### 4.3.3 Contractual Considerations

The reusability of software has an impact on the planning, award, and performance of system and software contracts. The following are some of the existing or anticipated problems having contractual implications:

- Adequacy of Documentation - A meaningful competitive proposal cannot be prepared or evaluated without the availability of thorough documentation of the software which must or may be reused. This is true whether the reuse of software is by the government in the Request for Proposal (RFP) or initiated by the contractor in a proposal to gain a competitive edge. If initiated by the government, a complete documentation package must be made available during the solicitation period and consideration should be given to extending this period to allow a thorough examination by potential bidders. The RFP should also indicate that if a bidder initiates the reuse of software, a complete documentation package must be made available for the evaluation and any proprietary rights clearly identified. The documentation package must include detailed test procedures and reports.

- Contractual Direction - Problems arise in performing a contract reusing government-furnished software if the contract does not specify the rules of how it is to be used. It must be made clear if reuse of the software is mandatory or at the discretion of the contractor. It also must be made clear whether the software, if used, can be modified by the contractor with or without formal government approval and participation in Configuration Control Board activities.

- Liability/Responsibility - The area of liability or responsibility must be clearly addressed in the contract if software will be provided by the government either by direction or at the request of the contractor. Contract requirements without a means to measure compliance, usually cannot be enforced or lead to government-contractor conflict.

- Incentives - New technologies and development procedures are best encouraged by providing a profit motive for their introduction. Incentive fees in contracts may be a workable mechanism for this, but care must be exercised to preclude arbitrary reuse of code without proper regard for cost effective benefits to the government. However, an incentive plan must not overlook the need for government investment in software tools, aids, services, training and library development for the enhancement of reusability. Investment in these areas should be included in the DOD Computer Resources Technology Plan. Other methods of funding experiments in the reuse of software should be explored by the JLC-JPCG-CRM. These include the cost-plus contracts where reuse is required or parallel engineering developments where reuse is specified in one development and new design in the second.

### 4.3.4 Documentation

"Code breakage" between prototype systems and first-item production units has demonstrated some necessary but not sufficient requirements for code reusability. Changing requirements, even though minor, significantly hamper the use of prototype code in a production item. This difficulty is further complicated if the original code is not in an HOL and is poorly documented. Although this "breakage" may be acceptable in the prototyping environment, when code is being developed with the intent that it is to be reusable, the requirements for programming in an HOL with proper and thorough documentation are essential. JLC-JPCG-CRM should continue to support the development and use of HOLs and the definition of appropriate documentation standards.

### 4.3.5 User Groups

Specific previous experience in transferring both hardware and software technology has provided some insight into the necessary elements of an environment for reusability. History has demonstrated that transfer of technology can be accomplished by the establishment of User Groups. In the area of software, many User Groups exist such as DECUS and SHARE/7. Software exchange and reuse frequently take place between individuals in these groups. Necessary ingredients of this transfer are the meeting of individuals working on similar problems in similar environments with follow-up technical support. In the examples mentioned above, the follow-up support is informal and normally carried on by telephone or computer network communications. JLC-JPCG-CRM should sponsor the formation of User Groups in the software areas which they consider candidates for cost-effective reusability. NOTE: The IBM Guide Conference User Group is currently sponsoring a reusable design and code productivity technique project.

### 4.3.6 Recommendations

The following specific recommendations resulted from this subpanel:

1. The CRM should define a technology transfer program which incorporates lessons learned in business applications to Weapon System Programs. Included would be a pilot project to gain the necessary experience needed to implement reusability on a global level. This demonstration project could also be used as a logical model for DOD environments similar to the business model referenced in Appendix B.

2. JLC should provide a government investment plan for reusable software technology development. Included should be incentives to invest IR&D and overhead funds in reusable software technology and tool development, respectively. Also, training programs and an organizational structure for their implementation. A plan for library support services on a DOD-wide basis should be addressed.

3. JLC should determine the feasibility (possibly under contract) of standardizing on a language and methodology for system and software requirement analyses. This study should identify applications that are made amenable to the use of requirements analysis methodologies

and evaluate the possibility of DOD adopting a single methodology or methodologies for various applications. The study would also address the impact of adopting a methodology on government/contractor relationships and contractor performance on DOD contracts.

4. Existing DOD guidebooks should be updated to address technical problems and remedies in the planning, award and performance of contracts where the reuse of software is anticipated. Standards and acceptance criteria must be developed to determine compliance with contract requirements to reuse software or to generate reusable software.

5. CRM should sponsor the formation of User Groups in areas considered candidates for reusability.

## 4.4 Impact on DOD Policy and Acquisition Practice

The use of digital computers continues to pervade military systems. A major limitation to realizing the full potential of this technology is the time, cost and risk of software acquisition. The concept of reusable software holds the dual benefits of significantly reducing the development time and life cycle cost of military systems, as well as increasing the labor pool available for development, operations, and maintenance of computer software. A DOD Software Reusability Program should include the following goals:

a) Reduce software implementation personnel requirements.

b) Reduce the maintenance contribution to life cycle costs.

c) Reduce the elapsed time from identification of an operational requirement to system deployment.

d) Reduce the variety and level of skills required for system operation and maintenance.

e) Reduce overall life cycle risks and costs.

Achieving reuse requires new thinking in DOD policy and acquisition practice as outined in the following paragraphs.

### 4.4.1 Augmentation of DOD Policy

DOD policy on software acquisition must reflect a desire for, and a commitment to, software reusability. This concept should then be made a requirement by integrating the various facets of reusability into existing directives. DODD 5000.29 should be changed to promulgate DOD's policy on software reusability. Next the DSARC Guidebook should be updated to include questions of program managers on the various areas of reusability (i.e., can this requirement be satisfied with existing software? Will this new software have application in other, later developments, etc.?). Also required is a DOD Instruction 5000.RUS which would be a companion [illegible] 5000.31 and 5000.5X, stating DOD's requirements on reuse. The indivi[illegible] services must then promulgate their own implementing instructions.

The DOD acquisition process must contain the appropriate incentives for getting both program managers and contractors to carry out the intent of the official policy. They must be convinced that to develop RUS, and to include in their projects the reuse of existing software is in their best interests. In the case of government program managers, the DSARC leverage cited above will be effective only in a limited sense; they must also be enlightened and convinced that RUS will provide them with very specific benefits during the acquisition cycle of their system. Items like lower risk, reduced costs and reduced development time are examples of the payoffs that they are most interested in. Contracts can be incentivised through the source selection and award fee processes. Proposal evaluation criteria should contain weighting factors that reward offerers for capturing existing software. This requirement should then be integrated into the government's monitoring, review and auditng procedures to ensure continued compliance. Finally, an award fee should contain a factor for the amount of software actually reused (this can be directly measured).

To effectively implement the DOD policy on RUS, action must be taken to modify, where appropriate, the various contractual vehicles that can influence the various aspects of reusability. The following is a partial list of contract mechanisms that must be analyzed for impact and then changed accordingly to support the concept.

- o Defense Acquisition Regulations
- o Military Standards & Specifications
- o Statement of Work Preparation
- o Work Breakdown Structure Preparation
- o Solicitation Process
- o Proposal Evaluation Criteria
- o Source Selection Process
- o Contract/Project Incentives

It is imperative to involve the industry in implementing the concept of RUS throughout the DOD. They must be frequently briefed on the status of DOD initiatives in the area. Their review and comments must be solicited in order to capitalize on their experience and expertise. All documents promulgated should be coordinated with the industry through the various associations such as EIA, EDPA, NSIA and AIA.

The government's thinking regarding the specification of contract deliverables must be revised. When a contractor is tasked to develop a specific item of software that will ultimately be reusable, that item must be documented, designed, tested, etc., to a greater degree. This must be incorporated in Project Plans, must be budgeted for, and QA and CM must be augmented.

The DOD should coordinate their efforts with other organizations, both nationally and internationally, that potentially have similar or related interests. Candidates are: ANSI, NBS, Foreign Military Sales Offices, NATO, ISO.

## 4.4.2 Areas of Necessary Standardization

If software reusability is to be a viable objective, DOD efforts to promote and promulgate standardization will have to be expanded beyond the current areas of high order language (HOL) and instruction set architectures (ISA), to the entire programming and operational support environment. The primary areas of additional standardization deal with the interfaces that any RUS element must have with the overall software system.

High Order Language. Reusability at the source code level makes the use of a standard HOL an absolute necessity. In this light, the on-going DOD activities to restrict the proliferation of programming languages and, further, to ultimately arrive at a single, common high order language (i.e., Ada) is the necessary prerequisite for making RUS a reality.

Instruction Set Architecture. DOD activities directed toward standardization of computer instruction set architectures can also be viewed as supportive of the concept of RUS. The benefits derived from standardization at this level do not apply as much to reusability of applications software since reusability at the object code level tends to be less viable than reusability at source code, design, or algorithm levels. Rather, instruction set architecture standardization supports reusability of support software tools that are applied in the development process.

Operating System/Executive Interfaces. The concept of RUS would be greatly enhanced if the DOD standardization activities could be extended to address the idea of a standard operating system or executive. However, in light of the variety of operating envionments and consequent demands imposed on the operating systems and/or executives to function in those environments, a standard operating system is probably infeasible. It is feasible, however, to consider standardizing the way in which an element of software interfaces with the operating system (e.g., the way a package is invoked by the operating system). Standardization at this level would eliminate the need to modify an element of software to achieve compatibility with the operating system or executive being used for a given application.

Data Base Interfaces. For reasons similar to those stated for standardizing the operating system interface, the way that software elements store and retrieve data from a global data base should be standardized. Data handling provisions of Ada will aid in this area, however, if substantial modification to achieve data base compatibility is to be avoided, additional standardization is required.

Inter-Function Interfaces. The operating system and data base interface standardization activities 'described above, coupled with the procedure definition provisions of Ada will, in substantial measure, standardize the functional inter-relationships among elements of software. Whether in fact, these prove to be sufficient to effect this standardization is unclear at present. This is an area for additional study.

Inter-Computer/External Interfaces. An obvious candidate for DOD standardization activity is in digital communications. Some work has already been done in this area (e.g., MIL-STD-1553: data busses). This has primarily addressed avionics applications to date. This activity needs to be broadened to encompass other forms of digital communications for applications such as CCCI and fire control systems.

### 4.4.3 Additional General Comments Regarding Practices

A number of additional issues loom as significant in addition to the above two major areas. These are: support requirements, data collection and dissemination, configuration management, and promoting acceptance.

Support Requirements. For software to be effectively reused, the support requirements must be examined. Software development practices such as modern programming techniques must be reviewed to determine what the lower-level elements will look like. The over twenty Integrated Logistic Support Plan items will have to be tailored to support reuse. Quality Assurance programs will have to be expanded to audit for reusability. For example, several (i.e., correctness, maintainability, efficiency, testability) of the eleven commonly applied quality factors will be directly impacted by RUS. Reliablity measurements that also insure reusability must be developed.

Data Collection/Distribution. In order to make RUS a reality on anything but an intra-company basis, it will be necessary to estabish a mechanism for the accumulation, maintenance and distribution of RUS. If done properly, this RUS library, implemented at the DOD level, would serve as a repository for descriptive data for all software identified as being potentially reusable together with a pointer identifying the location of the code, documentation and test data for each software element. A data storage retrieval system needs to be established with data retrievable on a key-word or relational basis to respond to contractor or program office requests for existing software that is potentially applicable to a new program.

An alternative approach would be to establish a RUS library within each service to exchange top-level information (key-word lists, software element names, etc.) and to provide a cross-reference to the other services' library lists. Documentation will assume a more important role; without adequate documentation, reuse will be limited to applications within the same company. Greater emphasis on tri-service standards and documentation contents will be required.

Configuration Management. While the concept of RUS has definite configuration management implications, configuration management need not have direct impact on the establishment or maintenance of the RUS library. It will, however, be necessary to devise appropriate procedures for keeping the library updated with data reflecting current versions of the software. Also, procedures to maintain current status once the software has been imbedded in multiple new applications are required. Reuse will affect configuration management practices as to the allocation of CPCIs and level of control.

Promoting Acceptance. The introduction of the RUS program to the acquistion process must be multifaceted. The concept is often not an accepted practice for personnel currently involved in computer software design, implementation, integration, and test. Resistance to change to the status quo is likely. Education is the primary means to help balance the perception of risk and benefit occasioned by a RUS program. Methods of education should include familiarization in service schools and briefings to both government and industry. "Road shows" of both high and low detail as well as "guidebooks" should be used to build a consensus for software reusability. Audiences should include sponsors, project managers, source selection authorities, industry leaders, and contracting personnel. Training curricula must be revised to promote adaption of existing software and group collaboration rather than individual effort and zero base approaches to programming projects.

Dual incentives are needed for both industry and government to both produce reusable software and capitalize on existing software when designing new systems.

### 4.4.4 Recommendations

The transition to maximize the amount of RUS will not be easy. The implied significant cost and manpower savings gives DOD a strong motivation. However, this does not motivate the individual program manager or contractor. To be effective, changes must be made in the basic way the government does business. Policy must be updated, standardization fostered and contractual vehicles updated. Education is important; if government and industry recognize the potential of RUS, the mechanics and implementation will follow. How fast will be determined by the acceptance and emphasis by top level management, government, and contractor personnel.

The following specific recommendations are offered:

1. Update the following documents to reflect DOD policy and emphasis on reuse of software: DODD 5000.29, DSARC Guidebook, and specific contractual vehicles. In addition, the individual services must issue supporting implementing instructions.

2. Promulgate a new DODI 5000.RUS on Software Reusability.

3. Develop/provide incentives for DOD PM's and contractors to support the concept. One necessary element of this is to establish an educational program to make both government and industry aware of the potential benefits. A guidebook detailing practices might also be of benefit.

4. Re-evaluate deliverable requirements to be consistent with reuse by organizations different than the developer.

5. Re-evaluate support requirements for reusability, to include development, ILSP requirements, Q.A., reliability and configuration management practices.

6. Conduct a study to identify and define a feasible approach for reusable software data collection and distribution.

7. Expand DOD standardization activities to include standard software

package interface techniques (operating system, data base, and external communication).

8. Training curricula in DOD for PM's and in industry for software designers should be revised to promote reusable software.

9. Legal questions concerning reused software as it relates to competition must be examined.

## 5.0 RECOMMENDATIONS

The panel deliberations resulted in a large number of recommendations for subsequent action and a number of areas requiring special attention by DOD and/or industry. These are all reported in Section 4.0. The major recommendations, have been extracted and are presented here to consolidate the conclusions of Panel E.

1. Standards, instructions, directives, guidebooks, training criteria, regulations, and contractual vehicles must be revised to identify the changes which must be made to include the concept of reusability. The JLC should foster preparation of new policy documents to encourage and enforce reusability.

2. There must be provisions within both industry and government for focusing on the establishment of a reusable software (RUS) program. Corresponding investments in new techniques, tools and training should be applied by both government and industry. A dedicated Reusable Software Organization may be required within each company to standardize and control software designated for reuse.

3. Incentives should be provided for DOD PM's and contractors for compliance with reusability concepts.

4. Reassessment of the deliverables currently specified in DOD contracts must be based on added information needed to effect reuse of the software. Additional data collection and distribution of information will need explicit attention. Support requirements must also be reevaluated for applicability (e.g., QA, CM, ILSP) in a reuse environment.

5. The success of RUS in business applications has been much greater than that of embedded computer systems. A more thorough analysis of the management, techniques, and tools used to achieve this will yield valuable insight to individuals operating in the DOD environment. Pilot projects in proven reuse techniques should be considered to establish a DOD methodology.

6. The JLC should foster investigation into methods for effecting the transfer of software between agencies and contractors. This may include library and information access techniques available to all interested parties and user groups.

7. Standards and methods for requirements analysis and formal specification languages should be pursued.

8. Functional design techniques must be defined to focus on identification, structure and partition for reuse. The design must be cataloged and be accessible independent of the code. Modularity guideliens must be reviewed for appropriate packaging and package specifications for reuse.

9. Ne [illegible] sign and coding techniques for interface standards will need attention. Binding between components and with operating

systems/executives, convertibility for moving to new architectures, avoidance or control of timing and sequence relationships are, all examples of these areas. DOD or system family-wide standards may be required. Further, coding standards with greater emphasis on readibility, individual style restrictions, built-in flexibility for change, and parametric bindings are needed.

10. Research and development of new support tools specifically to promote reusability are needed. Such new areas include specification generation and dissemination, verification, and certification of reused components, library methods and tools for cataloging and access of software components, and management tools for project control.

11. Development and standardization of a support tool environment must be initiated to emphasize transportability, address multiple target machines and consider standard support tool interfaces. Support tools must be rigorously validated and certified. In some agencies, this is currently underway and should be supported.

12. Exclusive use of higher order language is a single very strong necessary precedent for reuse. To the degree that multiple dialects (subsets or supersets) are allowed, reusability will be adversely affected.

13. Further attention is required to the transition to Ada from current languages. A large investment in CMS-2, FORTRAN and JOVIAL programs will not be discarded for many years to come. One consideration would be to have the Ada PSE support multiple languages. Further, some languages (e.g., Atlas, COBOL) may not be replaced by Ada; these languages will need continued attention to ensure that all required functions can be implemented in the higher order language.

14. Special investigation is required into current documentation standards to achieve reusability. This includes allowing for a clear traceability of requirements through the entire document hierarchy (so when accessing a reusable component, the associated documents are easily identified and extracted). Also, investigation is required of the relationship between CPCI's and reusable units (i.e., CPCI's may be defined to a lower level than at present to facilitate reuse).

Brief consideration was given to microprocessor software by the panel. Hardware-intensive applications would not be areas for consideration for design for reuse. Software-intensive applications might be candidates for RUS, however, these applications tend toward machine-dependence by plan. Hence, attempts to apply reuse principles to non-complex microprocessor software should be carefuly weighed to ascertain if it is economically advisable.

Risks. All new endeavors generally have risks associated with them. A large scale effort to reuse software in new applications is no exception. In the process of deliberation, Panel E did identify certain risks attendant to developing and reapplying software in new applications. These are summarized below.

- Hardware trends (VLSI, VHSIC) are moving towards standalone hardware devices which are quite capable and have distinct functional capabilities. The world of reuse in the future may center around configuring systems with such functional hardware elements. As such, the software interfaces will become hardware interfaces and software reuse would require adjustment accordingly.

- If software is to be reused after the initial development, considerable attention must be paid to the proper control and support of the fielded software. That is to say, once the development is done, the software has transitioned to a maintenance environment. If someone else is going to extract the software from that maintenance environment for reuse, the maintenance staff must be very careful to make sure that all of the development documentation is in fact sustained and all of the development tests are available to the reuser. This comment would apply whether the reusable software is being provided by DOD or is just being retained within the organization of a single contractor.

- Reusability must be addressed as an objective of the development process. One cannot build software and then decide (after the fact) they intend to reuse it. At the time the development is started, specific guidelines for methods and standards must be established to support the intended reuse.

- A corollary to this is that not all software will be able to be reused. Certain designs will not lend themselves to new applications. Thus, even though software has been developed with reuse as an objective, if the application is different enough or the interfaces are different enough, the reuse may not become a reality. There has to be a sensitivity to such incompatibility; reuse cannot be arbitrarily forced.

- A certain amount of generality must be injected to the software to support its reuse. For example, standard interfaces and standard programming languages readily come to mind. This generality may translate into inefficiency of operation and limitations of accuracy. When one goes to the next application, if the inefficiency or inaccuracy is intolerable, then the reuse has been lost. Thus, in order to achieve the reuse, some considerations may be required to allow for less than total optimum performance of a new application.

- In the short term of this Panel's deliberation, there was not an adequate dialog as to the cost attendant with the development of reusable software. How much more would it cost to develop software which is intended for reuse than to develop it for single application? There definitely is an increased cost and a specific return on investment curve could be plotted. One must be careful to ascertain that the cost reduction in the new application will compensate for the increased cost of the original development.

- Legal (or quasi-legal) questions relative to the impact of reuse practices on contract competition might present some concern to DOD. Practices must be addressed to both present software (or allow access to software) to contractors when reuse is an expected part of the procurement; and also to assure no single contractor receives unfair preferential treatment in procurements where reuse is expected.

## APPENDIX A - PANEL E PARTICIPANTS

| | NAME | AFFILIATION |
|---|---|---|
| | Mr. F. Barricelli | U.S. Army/CECOM |
| | Mr. H. Carpenter | Teledyne-Brown Engr. |
| | Mr. J. Cooper | Anchor Software |
| | Mr. J. Farrell | MRJ, Inc. |
| | Mr. J. Ferrell | GEISCO, Inc. |
| | Mr. H. Fischer | Litton Data Systems |
| | Ltc R. Goodman | U.S. Navy/NSSC |
| Subpanel 3 Ldr | Mr. R. Lanergan | Raytheon Missile Sys. Div. |
| | Mr. R. Lugo | USAF/SA-ALC/MME |
| | Dr. T. Martin | RCA/Gov't Systems Div. |
| Chairperson | Mr. P Mauro | Hughes Aircraft Company |
| Subpanel 1 Ldr | Dr. F. Maxwell | Aerospace Corporation |
| | Mr. M. Mesecher | Sperry Systems Mgmt. Co. |
| Subpanel 2 Ldr & Co-Chair | Mr. R. Morraele | Computer Sciences Corp. |
| Subpanel 4 Ldr | Ltc E. Myers | USAF/SD/AQM |
| | Mr. L. Osborn | Boeing Aerospace Corp. |
| | Ms. S. Peele | U.S. Navy/FCDSSA |
| | Mr. W. Scherer | U.S. Army/ARRDCOM |
| | Prf N. Schneidewind | U.S. Naval Post Grad Sch. |

## Appendix B

# A SUCCESSFUL APPROACH TO THE REUSE OF BUSINESS APPLICATION SOFTWARE: REUSABILITY MODEL

by

Robert G. Lanergan, Manager of Information Services

and

Denis K. Dugan, Manager of Information Processing Systems and Configuration/ Data Management

Raytheon Company
Missile Systems Division
Hartwell Road
Bedford, MA 01730

# A SUCCESSFUL APPROACH TO THE REUSE OF BUSINESS APPLICATION SOFTWARE: REUSABILITY MODEL

## INTRODUCTION

Picture yourself with the following problem.

You are the Systems and Programming Manager responsible for eight Programmer Analysts. Your prime responsibility is maintaining a large dock-to-stock manufacturing system with 90 to 100 terminals. The system runs on a Univac 1106. You are using a data management system developed by your company. Your management, for economy and centralization reasons tells you to rewrite the system for an IBM/370, located 25 miles from your plant. The system will use IBM's Information Management System. You and your people know nothing about this new environment. However, one of your eight Programmer-Analysts has had previous experience in a IBM O/S environment. You are given four months to rewrite the system. At the end of this time, your Univac 1106 will be removed. There is one positive note, however, your eight Programmer-Analysts have a good grasp of the present system. What development approach do you use under these circumstances?

Needless to say, a project with these conditions has all the earmarks for disaster and under the circumstances your best bet would be to get your resume polished up and move on.

In 1975, a group of eight Programmer Analysts in our Raytheon Missile Systems Divisiôn located in Bedford, Massachusetts did not polish up their resumes. Instead, faced with these conditions, they carried structured techniques to their next logical step, reusable cobol source copy code.

These eight dedicated believers in reusable code techniques developed sixty-two programs with an average of 65% reusable source lines of code and over 800 reusable COBOL source code modules. The system was completed on time with minimum testing and impact on our user community and has run with minimum maintenance and problems over the last four years.

Due to this remarkable achievement, a new era in software development began in Raytheon Missile Systems Division in April, 1976. At that time, our business data processing management initiated a three man, four month study to determine how to implement this reusable code technique in three plants and to make recommendations for additional software tools and aids to increase our programming productivity.

## OUR ENVIRONMENT

The environment studied was a business applications environment using IBM's ANSI COBOL with full extensions and, where applicable, IBM's Information Management System. The work force is comprised of approximately 120 Programmer Analysts located in three different physical locations with three different Data Processing Managers.

## STUDY APPROACH

Our approach to the problem of HOW to globalize this reusable code concept was to determine what type of programming work was produced in each of the three locations, what tools were required to improve this work, what kind of training was required, how much training was required and, how to measure and monitor our progress.

After examining work records for a two year period and interviewing three Managers, twelve Supervisors, and other key personnel, we determined the following about our work profile, personnel and programming environment.

## SYSTEMS PROGRAMMING AND ADMINISTRATION WORK PROFILE FINDINGS

To evaluate the type of work our 120 people performed, the study team developed a work profile that consisted of eight categories of work.

1. NEW DEVELOPMENT - The development of a new application not presently on the computer.

2. REPAIR - The fixing of operational bugs.

3. ENHANCEMENT - The addition or modification of code or program to an existing computerized application.

4. CONVERSION - The transformation of an existing computerized application from one language to another or from one computer to another.

5. REPLACEMENT - The development of a computer application that takes the place of one already existing.

6. SYSTEMS SUPPORT - Manpower dedicated to total user support of a given application.

7. CONTRACT MANPOWER - People not tied to any specific application but supplied by the hour to do what is required by the contracting organization.

8. OVERHEAD ACTIVITIES - Information systems manager and staff engaged in administrative and support activities.

A percentage work breakdown for the above categories was as follows:

| | | |
|---|---|---|
| 1. | New Development | 15.0 |
| 2. | Repair | 7.5 |
| 3. | Enhancement | 30.3 |
| 4. | Conversion | 9.7 |
| 5. | Replacement | 14.5 |
| 6. | System Support | 6.7 |
| 7. | Contract Manpower | 8.3 |
| 8. | Overhead/Administrative Activities | 8.0 |
| | | 100.0 |

The above profile showed that 15% of our effort over a two year period was expended on new application development, 8% on overhead and 77% on what we defined as maintenance.

PERSONNEL SKILLS STUDY FINDINGS

To determine the work skills of our personnel, a general skills inventory questionnaire was completed by our 120 programmer/analysts. The results were as follows:

1. 3% had received formal training in program design.

2. 12% had received formal training in systems analysis.

3. Less than 10% had received formal training in testing and documentation techniques.

The same study was conducted in three other large companies with close to the same results.

As a result of this study, it was apparent that formal training and assistance after training would be mandatory if the REUSABLE CODE approach was to work.

## OUR PROGRAMMING ENVIRONMENT FINDINGS

Approximately 1200 new programs were written annually and they were categorized into three groups:

1. New application development programs for new systems.

2. One-time programs written for our user community which required quick response, usually less than one week.

3. Programs written to enhance, fix, convert or replace existing systems which also required quick response, usually less than one week.

In addition to our new program findings, we determined that we enhance or fix approximately 2000 programs annually.

After analyzing this data from our programming work environment, we arrived at the following conclusions:

1. Approximately 37% of our programming time was spent writing new programs.

2. Approximately 63% of our time was spent on modifying old programs.

3. Approximately 80% or 1600 of our new programs were batch sequential.

4. Approximately 65% or 1300 of our new programs were required in less than three weeks. The "I need it tomorrow" syndrome.

5. A large percentage of our programming work was redundant in nature and held much potential for the reusable code concept.

## PROCEDURE USED TO TEST OUR REUSABLE CODE THEORY

To validate the conclusion that much of our business applications programming work was redundant in nature and held much potential for the reusable code concept, the study team, managers, supervisors and key personnel discussed the various types of programs written in our organization. They came up with the following types and terms:

| | |
|---|---|
| . Edit or Select Programs | - This consisted of editing data, selecting or extracting data and reformatting data. These programs are classified as utility programs. |
| . Update Programs | - This consisted of updating or modifying data on any type of file. |
| . Report Programs | - This consisted of producing reports from any type of file. |

Over 5000 production COBOL source programs were classified by type using the following procedure:

Each supervisor was given a list of the programs for which he was responsible. This list included the name and a brief description of the program along with the number of lines of code. The supervisor then classified each program using the following categories:

. Edit or selection programs

. Update programs

. Report programs

If a program did not fall into the above three categories then the supervisor assigned his own category name. The result of classification analysis by program type was as follows:

| | |
|---|---|
| 1089 | Edit or Selection Programs |
| 1099 | Update Programs |
| 2433 | Report Programs |
| 247 | Extract Programs |
| 245 | Bridge or Conversion Programs |
| 341 | Data Fix Programs |
| 5454 | Total Programs Classified |

After analysis, the 247 Extract and 245 Bridge or Conversion Programs were placed in the Edit or Selection class and the 341 Data Fix Programs were placed in the Update class. The results of our program classification task after these adjustments were as follows:

| | |
|---|---|
| 1581 | or 29% of our programs were Edit or Selection Programs |
| 1440 | or 26% of our programs were Update Programs |
| 2433 | or 45% of our programs were Report Programs |
| 5454 | Total Programs Classified after Adjustments |

The average lines of code by program type for the 5454 programs were as follows:

| | |
|---|---|
| 626 | Lines of code per Edit or Select Programs |
| 798 | Lines of code per Update Programs |
| 507 | Lines of code per Report Programs |

The supervisors then selected over 50 programs that they felt would be good candidates for validating our redundancy theory. Working with the supervisors, the study team found that approximately 40-60% of the COBOL source code in the programs examined was redundant and could be standardized.

In September, 1976, we presented the following conclusions and recommendations to management:

## CONCLUSIONS

1. Our maintenance and productivity problems are caused by re-writing the same redundant functions using different programmer styles and solutions.

2. Our programming environment consists of three types of programs and that 40-60% of the source code could be standardized and pre-written for these three types.

3. The reusable module source code technique would provide a significant increase in programmer productivity and provide a consistent style required to improve our programming maintenance problem. We used the dock-to-stock inventory system as a success story.

4. Our personnel required additional training to upgrade their skills. We used the skill study to prove our point.

## RECOMMENDATIONS

1. One manager and five key technicians should be assigned to staff this effort. Each location should be represented.

2. A six months R&D effort should be funded, to develop software tools and aids, standards, procedures and a in-house training program.

3. A one year implementation period should be scheduled to give our 120 programmer analysts 5 days of training and follow-up assistance as required in reusable code techniques.

4. A steering committee should be established to audit the progress of the project.

Managment approved our proposal with one provision. They required that we provide for measurement reports on reusable code results.

## OBJECTIVES

The primary objective of the Advanced Software Development (ASD) project was to reverse the upward cost trend of software development and maintenance by using reusable modules and logic structures and implementing new disciplines and aids which would yield a more reliable, maintainable product and increase productivity of our systems and programming personnel.

Other goals which complemented the above objective were to:

- o Design, create and reuse standard modules for new applications development. 60% reusability was the target.

- o Increase the maintainability of all new programs and programs that require modifications.

- o Increase the control over our work products.

- o Decrease the overhead required to maintain our work products.

- o Upgrade the skills of our staff using state of the art technology.

- o Create an awareness and environment that fosters the development of new tools and aids within the organization.

## IMPLEMENTATION OF THIS CONCEPT

### General

Encouraged by the results of our study we decided to further test the concept by proceeding in parallel along two paths. One path was the use of pre-written COBOL copy code source modules. The other path was to develop pre-written COBOL logic structures that contain redundant programming functions.

### Pre-Written Modules

These are modules which are written for a specific purpose. They are walked through, tested, certified, documented and stored in a library. Examples in a scientific environment would be routines for trigonometric functions. In a business applications environment we break these into three groups. The first group is called Universal Routines. These are routines that most companies can use. Examples are:

- Gregorian date edit routine.
- Gregorian to Julian date conversion routine, and vice versa.
- Payroll tax routines.

The second group is called Company Routines. These are routines that are used for our standard systems. Examples are:

- Part number validation routine.
- Manufacturing day conversion routine.
- Edits for data fields used throughout our standard company systems, e.g. employee number, purchase order number, etc..

The third group is called Functional Area Routines. These are routines that are used by functional areas in our company such as manufacturing or accounting. Examples are:

- Customer, vendor, and invoice number routines.

- For those companies doing work on Government Contracts there would be another group. Many data fields are standardized for all Government applications. Examples are Federal Stock Number, Federal Vendor Code, Interchangeability Code, Shelf Life, etc.

From a systems and programming point of view, all these routines fall into several categories. What follows are the descriptions of the categories and the number of modules that exist to date in our reusable code library.

- File description i.e. FDs

- Record descriptions i.e. 01 levels in an FD or in working storage

- Edit Routines i.e. the data area and procedure code to edit a specific data field

- Functional Routines i.e. the data area and procedure code to perform some function, such as left justify and zero fill

- Data Base I/O area specifically IBM's IMS I/O area descriptions

- Data Base interface module, specifically IMS Program control blocks

- Data Base Search Arguments, specifically IMS Segment Search Arguments

- Data Base Procedure Division Calls

Using the reusable code centralized library approach, eight major systems have been developed averaging over 60% reusable code during the last four years.

## KEY CONCEPT BEHIND OUR REUSABLE CODE SUCCESS

The key concept behind this successful methodology is that reusable modules are designed, coded, tested, certified and documented prior to completing programming specifications. When a programmer receives a program specification it specifies what, why, when and how to use modules that are applicable to the program problem. The technique resembles a manufacturing environment in that many standard parts are used in building manufactured products. The only difference in our environment is the program is the product. This approach produces systems that are more reliable with less testing, coding and documentation. The biggest payback however, lies in greatly reduced maintenance since a reusable module is independent of the physical program and need only be modified once even though it's used in many programs.

## LOGIC STRUCTURES

A logic structure has a pre-written Identification Division, Environment Division, Data Division, and Procedure Division. It is not a complete program because some paragraphs contain no code, and some record descriptions are also empty, consisting only of the 01 level. It does, however, contain many complete 01 levels and procedure paragraphs.

We have written six Logic Structures:

The update is designed for the classical, sequential update. There is a version with an embedded sort and a version without. The update is designed for situations where the transaction record contains a transaction type field (add, change or delete). This can be easily overridden for collator type updates. The update logic structure is also designed to accommodate multiple transactions per master record. Error messages to a transaction register are provided for standard errors such as an attempt to add an already existing record. Final totals are also provided, as well as sequence checking.

The report logic structure is also written in two versions, one with and one without a sort of the input records prior to report preparation. Major, intermediate and minor levels of totals are provided for, but more may be added if needed. If multiple sequences of reports are desired the record can be released to the sort with multiple control prefixes. Paragraphs are also provided for editing, reformatting, and sequence checking.

The select logic structure is also written in two versions, with or without a sort of the input records. This logic structure was designed for two purposes. One is the editing of input records. In effect, the input records are examined based on some criteria and written to the selected (good records) or non-selected (error) file. Another use for this logic structure is the selection of records from a file, based on some criteria, for later use in a report. In practice one of its most common uses has proven to be special purpose runs against master files when customers request specific non-standard modifications to existing data.

Combinations: We rank these logic structures in the order-update-report-select- when determining which one should be used for a combined program. The highest included function determines which logic structure to use. For instance, an update with editing of transactions would indicate adding edit code to the update, rather than adding update logic to a select.

## CONSTRUCTION OF LOGIC STRUCTURES

For each type of Logic Structure there is a central supporting paragraph.

For the update program it is the high-low-equal comparison.

For the report program it is the paragraph that determines which level of control break to take.

For the selection program it is the select/non-select paragraph.

Let us consider the report program as an example.

Prior to the control break paragraph we can identify support functions which must occur in order for the control break paragraph to function. Examples are: get-record, sequence-check-record, edit-record-prior-to-sort, and build-control-keys. These are supporting functions. Other functions such as major-break, intermediate-break, minor-break, roll-counters, build-detail-line, print-detail-line, page-headers, etc. are dependent on the control break or central paragraph.

Obviously many of these paragraphs (functions) can be either completely or partially pre-written.

Our report program Logic Structure Procedure Division contains 15 paragraphs in the version without a sort, and 20 paragraphs in the version with a sort.

To further clarify what we mean when we talk about logic structures, it might be helpful to indicate the number of non-comment lines of code in each part of a Report Logic Structure, without an embedded sort.

Identification Division

Environment Division

Data Division

File Section

Working Storage Section

```
01  AA1 -  CARRIAGE-CONTROL-SPACING
01  BB1 -  CONSTANTS-AREA
01  BB2 -  TRANSACTIONS-STATUS
01  BB4 -  FILES-STATUS
01  CC1 -  COUNT-AREA
01  DD1 -  MESSAGE-AREA
01  EE1 -  TRANSACTION-READ-AREA
01  FF1 -  KEY-AREA
01  GG1 -  HEAD-LINE1
01  GG2 -  HEAD-LINE2
01  GG3 -  HEAD-LINE3
01  HH1 -  DETAIL-LINE
01  LL1 -  TOTALS-AREA
01  SS1 -  SUBSCRIPT-AREA
01  TT1 -  TOTAL-LINE-MINOR
01  TT2 -  TOTAL-LINE-INTER
01  TT3 -  TOTAL-LINE-MAJOR
01  TT4 -  TOTAL-LINE-FINAL
```

Procedure Division

```
0010 - INITIALIZE
0020 - MAIN-FLOW
0030 - WRAP-IT-UP
0040 - CHECK-CONTROLS
0050 - FINAL-BREAK
0060 - MAJOR-BREAK
0070 - INTER-BREAK
0080 - MINOR-BREAK
0090 - PRINT-TOTAL
0100 - ROLL-COUNTERS
0110 - FILL-DETAIL-LINE
0120 - WRITE-PRINT-LINE
0130 - NEW-PAGE-HEADING
0140 - GET-TRANSACTION-RECORD
0150 - TRANSACTION-FORMAT-OR-EDIT
0160 - SEQUENCE-CHECK
```

Only two of these paragraphs are totally empty. Some, like GG2-HEAD-LINE2 have a filter to prevent Compute errors if they are not needed, but they are essentially empty. Taken together as a program they provide the programmer with a modular functional structure on which he can build a report program, many parts of which are partially pre-written.

For the update logic structure the central paragraph is the hi-low-equal control paragraph. Prior to the central control paragraph there must be supporting functions such as get-transaction, sequence-check-transaction, edit-transaction, sort-transaction, get-master, sequence-check-master, build-keys, etc. As a result of this central paragraph you will have functions such as add-a-record, delete-a-record, change-a-record, print-activity-register, print-page-heading, print-control-totals, etc.

Our update logic structure Procedure Division contains 22 paragraphs in the non-sort version and 26 paragraphs in the version with an embedded sort.

## BENEFITS OF REUSABLE CODE

We believe that logic structures have many benefits.

- They help clarify the programmer's thinking in terms of what he is trying to accomplish.
- They make walk-thru easier.
- They help the analyst communicate with the programmer relative to the requirements of the system.
- They facilitate debugging.
- They eliminate certain error prone areas such as end of file conditions, since the logic is already built and tested.
- They reduce program preparation time since parts of the design and coding are already done.

However, we believe the biggest benefit comes after the program is written, when the user requests modifications or enhancements to the program. Once the learning curve is overcome, and the programmers are familiar with the logic structure, the effect is similar to having team programming with everyone on the same team. When a programmer works with a program created by someone else he finds very little that appears strange. He does not have to become familiar with another person's style, because it is essentially his style.

## STRUCTURED TECHNOLOGY GUIDELINE

Our structured design approach advocates that a program be divided into segments called functional modules. Each of these modules is independent of other modules in the program and performs a separate function. Emphasis is placed on the main functional modules first because they contain the highest-level control logic and are the most critical to the success of the program. These modules are designed, coded, and tested. Then the next lower-level modules are created in the same manner. In this way, most of the details of the design are left until the lowest-level modules are designed. Thus, a program developed by using a top-down design consists of a series of modules created and related in a treelike (hierarchical) structure.

The treelike structural relationship between modules in a top-down designed program can be represented in a STRUCTURE CHART. The structure chart shows each module and its functional relationship to other modules. The flow of control is from the highest-level module to the lowest-level modules (top-down). Each module is called or invoked, by a next-higher-level module.

The logic structures described earlier in the report reflect this kind of design.

## STRUCTURED CODING GUIDELINE

Pure structured program code reflects three basic control structures:

| | |
|---|---|
| Sequence | (ADD A TO B, MOVE C TO D, ETC) |
| Selection | (IF A less than B MOVE C TO D else MOVE D TO E) |
| Looping | (PERFORM ROUTINE A UNTIL THERE IS NO MORE DATA) |

Each of these structures has a single entry and single exit. The program listing is easy to read because there is no random branching using GO TO's (FORWARD OR BACKWARD) from one part of the program to another part. Since control flows from top to bottom, the listing can be read more or less like a book.

Our approach advocates a modified approach to the pure structured coding defined above. The approach advocated using GO TO's that branch to the entry, or exit points of a functional module.

## STRUCTURED WALKTHROUGH GUIDELINE

The purpose of STRUCTURED WALKTHROUGHS is to find errors and to find tham at the earliest possible time by having one or more peer people review the software work products of others.

The advantages of walkthroughs are many: higher reliability, increased maintainability, better dissemination of technical information between technicians, and an increased likelihood that work can be salvaged if someone leaves the project.

The walkthrough technique used in our organization is informal. Usually it consists of one person and the author. The objective is to find errors, not to criticize style.

## STANDARD COBOL PROGRAM FORMATS

The purpose of STANDARD COBOL FORMATS is to improve the readability and understandability of our primary work product - PROGRAMS.

Our organization modifies over 2,000 programs annually. In addition to this, over 1,200 new programs are written by practicing programmers in our organization. Most of this activity centers around enhancing existing systems that have serviced the user community for many years. Most of the programmers who now maintain these old systems are not the original authors. Since programmers must read and understand programs before they can fix or enhance them, a process was developed to improve maintainability.

To satisfy this objective the project team modified software that produces a consistent standard COBOL formatting style. Some of the primary functions of the software are:

- o Attaches paragraph numbers to paragraph names. This increases the productivity of maintenance programmers because logic flow is easier to follow.
- o Provides indentation for the "If Then Else" construct in COBOL.
- o Prints one COBOL statement per line.

- Provides spacing between paragraph names.
- Assigns consistent hierarchical numbers to group and subordinate data names in the data division.

## MECHANICS

The procedure to generate a STANDARD COBOL FORMAT is as follows:

- Before the program is moved back to the CENTRALIZED PRODUCTION SOURCE LIBRARY, the programmer runs the program through the STANDARD COBOL FORMAT SOFTWARE using a standard procedure.

The implementation of this technique addressed the problem of what to do with 7,000 programs written over a period of 10 - 15 years by many programmers using different writing styles.

## CENTRALIZED LIBRARY CONCEPT

The centralized source program library at the Business Computer Center at West Andover, Massachusetts is a collection of functional libraries that are shared in common among several independent customers. The libraries are used to store and modify both test and production source programs and copy code. Each library in the network is functional in the respect that it exists to satisfy a specific need.

## BENEFITS OF A COMMON LIBRARY SYSTEM

A centralized facility offers its customers and the Division the following advantages over a diversified multiple library system.

## COST REDUCTION

A single common facility is maintained by full time individuals who have expertise in library management. This is preferable to the situation where individuals are required to expend man hours on an on-going basis to maintain their own private libraries and whose primary responsibility is something other than library overhead maintenance tasks.

A second significant cost reduction is the elimination of redundant and overlapping facilities.

## INCREASED PRODUCTIVITY

Because application programmers do not have to spend time maintaining private libraries, they have more time to perform additional programming oriented functions. This tends to relieve the application programmer from performing clerical activities and many peripheral programming duties are reduced to standardized tasks.

## SECURITY ENHANCED

Library management by full time experienced professionals greatly reduces the likelihood of modules or libraries being permanently damaged or lost due to human error and/or inadequate backup facilities.

## GLOBAL PROCESSING

Because all source code is systematically stored in one centralized location, it is possible to perform a variety of global functions.

## EXAMPLES

1. Scanning all lines of all modules on the library for specific character string.

2. Making global changes to all (or selected modules) on the library.

3. Insuring that all organizations within the division are complying with established standards.

## OUR MEASUREMENT APPROACH

Since the introduction of the Structured Technologies, the data processing community has heard claims of increases in programmer productivity. These claims range from moderate to fantastic and tend to intensify the concern many installations have about their own productivity. "Will the Structured Technology significantly improve my programmers' productivity and, if so, by how much?", is a question many data processing managers are asking themselves.

Unfortunately, most installations have no real baseline from which their own productivity increases can be measured. In fact, very few know what to measure, much less how to measure it. At the present time, there are no industry standards for measuring either the pertinent aspects of software development or the people's time that goes into producing software. Therefore, any attempt to compare one installation's performance against another, may be futile if not downright dangerous.

Perhaps the greatest danger of all is getting caught up in the magic of measurement for its own sake. Some installations have started to count lines of code and keep track of programming time without really determining why this data was needed or how it would be used. In effect, they are making measurement an end rather than a tool to be used to help meet their goals.

With these considerations in mind, it was the purpose of our software productivity to project to identify factors that should be considered in a measurement effort.

Our Measurement Approach in no way tried to determine THE SINGLE WAY TO MEASURE for no such simplistic solution currently exists.

## MEASUREMENT PHILOSOPHY

At the start of the project, it was decided by management and our project team that it would be counterproductive to our productivity goals to have programmers keep manual records or rely on their memories for reusable code statistics. Instead, we decided to measure reusable code utilization at the program and programmer level. These statistics are produced when a program is moved from our test source library to our production source library. This approach made the gathering of statistics transparent to our managers, supervisors and programmers. In addition, it gives management and supervisors the visibility to monitor their reusable code progress.

## MEASUREMENT PLAN AND OBJECTIVE

Our objective was to produce reliable, maintainable programs and increase the productivity of our practicing programmers by providing 40 - 60% reusable code for new programs. It was felt that this was attainable because of our success with the reusable code concept in test mode in 1977. During this time, a range of 15-85% reusable code was realized for new programs when logic structures and copy code were used.

Our plan for implementing our reusable code technique was composed of three simple steps.

1. Establish a flexible standard which made it mandatory that programmers would use logic structures and copy code unless they could tell us why they should not. This approach allowed us to gain the feedback from our programmers necessary to improve our reusable code technique.

2. Give our programming personnel five days training. This was accomplished by using half day training sessions and by gibing assistance to each programmer as required after the training sessions were completed.

3. By establishing a measurement system to measure the utilization of reusable code by program and programmer.

The major items measured were by program, programmer supervisor and plant. Other items reported on were:

1. Total lines of code for each new program.

2. Total reusable copy code for each new program in terms of modules, lines of code and reusable code percentage.

3. Total number of logic structures used and total number of reusable lines of code within each logic structure. i.e., code that the programmer did not have to modify.

## REUSABLE CODE RESULTS TO-DATE

By eliminating redundant programming and application dependent functions, the following results have been attained over a period of four years:

- o 8 New Systems have averaged over 60% Reusable Code
- o Over 3000 New Programs averaged over 40% Reusable Code
- o Over 6000 Programs are now more maintainable
- o Over 2500 Reusable Modules now exist in a Central Library

## HOW WE APPRAISE OUR PROGRESS

Measurement reports are distributed weekly to all supervisory and management personnel at each location. This gives our managers and supervisors the opportunity to analyze and appraise their progress and the progress of others.

To control our progress and deal with problems of implementation, quarterly status meetings are held with project managers and management. These meetings give us meaningful feedback required to tune our reusable code technique. In addition, these meetings also give recognition to our programming personnel for their reusable code feedback, utilization and contributions.

## SUPPORT PRODUCTIVITY TOOLS, AIDS AND SERVICES

Other disciplines and aids developed and implemented by the ASD project team used to complement REUSABLE PROGRAM MODULES - STRUCTURED PROGRAMMING TECHNOLOGIES - STANDARDIZED COBOL FORMATS - CENTRALIZED PROGRAM LIBRARIES - are:

- Indexing Scheme to foster the usage of reusable code.
- Programming Standards to provide consistency required to support the reusable code concept.
- Where Used System to monitor and analyze reusable code.
- Certification Procedures to validate reusable code.
- Program Change Activity system to provide change control visibility.
- Configuration Control System to provide synchronization of source and load programs.
- Standard Job Control Language Procedures to improve productivity.
- Entry Level Training Program - Entry level programmers are put through a two month full time reusable code training program. They design a complete inventory system using structured technology and reusable code techniques. In addition, they receive training in productivity tools and aids and reach a high level of productivity in three months after the training is completed. The students use a team approach with the instructor acting as the chief technican.

## CONCLUSION

After studying our business programming community for over four years, we have concluded that we do similar work year in and year out and much of this work deals with redundant programming functions. By standardizing those functions in the form of reusable code and logic structures, gains in productivity and reliability can be attained and the programmers can concentrate on the real creative problem rather than the redundant one. In addition to the one time development benefit, the programming community can eliminate much of the programmer's individual styles which are the root of the maintenance dilemma that most business data processing installations are faced with today.

## FUTURE PLANS

Currently, our logic structure programs provide us on the average 40-60% reusable code for new programs. The programmer however, must add, change and delete code to produce a complete program to fulfill the problem specifications.

Examples are:

- Moving data fields from the input area to the detail line image in the report logic structure.

- Moving data fields into the hi - lo - equal key in an update logic structure.

To attain additional productivity the next logical extension is to automate these changes rather than have the programmer manually make the modifications.

We are currently developing software which will automate our logic structure technique by using an INTERACTIVE COBOL Generator process which will automatically generate the COBOL source code necessary to complete a large percentage of our new programs utilizing structure code.

For example, the question "How many levels of control breaks?" would cause the required number of control break paragraphs to be included in the source code. In response to a subsequent question such as "What data field controls the most major break?" could cause that data name to replace the word "major" in all references to that paragraph. Thus, the paragraph "0090-Major-Break" becomes "0090-Department-Break". A sentence which reads "Perform 0100-Inter-Break" would read "Perform 0100-Employee-number-break".

This interactive prompting approach will enable a programmer, an analyst or, in some cases, even a user to produce a working COBOL source program in less than 30 minutes in one interactive session. This, coupled with our powerful text editing capability and other tools and aids will provide our programmers with a full set of tools to minimize the amount of time required to bring an application into production.