AD-A040 467 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/2
IMPACT OF MPP ON SYSTEM DEVELOPMENT.(U)
MAY 77 J R BROWN F30602-76-C-0095
UNCLASSIFIED TRW-29115-6001-RU00 RADC-TR-77-121 NL

1 OF 3
AD
A040467

12

NW

RADC-TR-77-121
Final Technical Report
May 1977

IMPACT OF MPP ON SYSTEM DEVELOPMENT

TRW Defense and Space Systems Group

Approved for public release; distribution unlimited.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report contains a large percentage of machine-produced copy which is not of the highest printing quality but because of economical consideration, it was determined in the best interest of the government that they be used in this publication.

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

This report has been reviewed and is approved for publication.

APPROVED: Roger W. Weber

ROBER W. WEBER
Project Engineer

APPROVED: Robert D. Krutz

ROBERT D. KRUTZ, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-77-121	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: IMPACT OF MPP ON SYSTEM DEVELOPMENT.	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. Dec 75 — Jan 77,
6. PERFORMING ORG. REPORT NUMBER: TRW-29115-6001-RU00	
7. AUTHOR(s): John R. Brown	8. CONTRACT OR GRANT NUMBER(s): F30602-76-C-0095
9. PERFORMING ORGANIZATION NAME AND ADDRESS: TRW Defense and Space Systems Group, One Space Park, Redondo Beach CA 90278	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F, 55810261
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISIM), Griffiss AFB NY 13441	12. REPORT DATE: May 1977
	13. NUMBER OF PAGES: 203
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: Same	15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

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)*

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer:
Roger W. Weber (ISIM)

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Modern Programming Practices, Project Organization and Management Procedures, Documentation Standards, Programming Standards, Computer Software, Cost and schedule control, reliable software, quality control, system development

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

This document presents the results of an evaluation of the impact of modern programming practices (MPP) applied to TRW's Ballistic Missile Defense (BMD) Systems Technology Program (STP) Software development. The report provides an overview of STP including a reconstructed chronology of significant events and a description of the STP software development environment. It also presents detailed results of: identification, selection and definition of appropriate STP practices to be evaluated; investigation of the impact of selected MPP on software development cost, schedule and quality; as well

OVER

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

as definition and preliminary application of techniques and tools comprising methodology for comparing TRW practices with those used by other contractors on other projects.

PREFACE

This document is the final technical report (CDRL Item A002) for the Impact of MPP on System Development, Contract F30602-76-C-0095. It presents the results of an evaluation of the impact of modern programming practices (MPP) applied to TRW's Ballistic Missile Defense (BMD) Systems Technology Program (STP) software development. The report provides an overview of STP including a reconstructed chronology of significant events and a description of the STP software development environment. It also presents detailed results of:

- Identification, selection and definition of appropriate STP practices to be evaluated
- Investigation of the impact of selected MPP on software development cost, schedule and quality
- Definition and preliminary application of techniques and tools comprising methodology for comparing TRW practices with those used by other contractors on other projects.

The report was prepared by J. R. Brown with significant contributions by J. W. Dowdee, D. E. Hacker, G. R. Keludjian, D. S. Lee, M. Lipow, G. R. Paxton and T. R. Savage. Additional assistance and guidance came from the MPP Project Review Committee which included B. W. Boehm, J. M. Dreyfus, F. S. Ingrassia, E. C. Nelson and F. G. Spadaro.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

EVALUATION

This report describes the software development technology and management practices employed on a large and complex system development program by TRW.

The intent of the RADC program to which this document relates, TPO V/3.4, is to describe and assess software production and management tools and methods which significantly impact the timely delivery of reliable software.

The study contract is one of a series of six with different firms having the similar purpose of describing a broad range of techniques which have been found beneficial.

RADC is engaged in promoting utilization of Modern Programming Technology, also called Software Engineering, especially in large complex Command and Control software development efforts.

Roger W. Weber

ROGER W. WEBER
Project Engineer

1.0 INTRODUCTION AND SUMMARY

1.1 Background

For a number of years TRW and other contractors have been developing ways to do a better job of software development. The primary impetus for this effort has come from increasing demands for error-free (or at least highly reliable) software, from the need for software which is easily (i.e., quickly and inexpensively) modified to meet newly imposed requirements, and from recognition that a cost-effective software engineering discipline (not the "black art" of old) can be achieved through identification, application and evaluation of improved production practices [1, 2, 3, 4].

It is generally believed that the quest for improved Modern Programming Practices (MPP), coupled with many opportunities to apply, evaluate and further improve upon them has brought about a significant advancement of modern software technology [5, 6]. In further pursuit of the goal of a cost-effective software engineering discipline, the Air Force Rome Air Development Center (RADC) recognized the need for an investigation of the actual merits of a broad range of established and newly proposed practices.

In the last few years some trends in DoD procurement philosophy have placed strong attention on the evolution and application of MPP by software contractors [7, 8]. The most noticeable trends are: 1) an unprecedented dual interest in both the cost of producing software and the quality of the end product, and 2) requests for proposals which require bidders to identify and propose application of specific practices that address and alleviate the problems that have plagued past software development activities.

The combined committments of both the DoD and major software contractors to accelerate the definition and application of improved production practices have resulted in a dramatic increase in the use of MPP on some large scale, highly critical software procurements. A prominent example is the TRW development of the Data Processing Subsystem software for the Ballistic Missile Defense Systems Technology Program (BMD STP). This program was previously named and is frequently referred to as Site Defense (SD). The importance of the Data Processing Subsystem to ultimate successful performance of the system has placed extraordinary demands on TRW as the software development contractor. To ensure the production of software that will meet the system requirements, TRW invented a unique, rigorous, and highly disciplined software development process [9]. To overcome many of the usual problems of software development and to ensure the development of error resistant, readily understood and easily maintained software, a comprehensive set of standard programming practices was established and firmly imposed [10]. To ensure adherence to the disciplined development methodology and to support the production of high quality software within reasonable schedule and cost, software developers were supplied with a set of automated tools to aid design, code, debug, test, and documentation activities [11, 12, 13].

In addition, TRW monitored the "vital signs" of project activity in assessing the effectiveness of the development process, the standards and the support tools. This required collection of data on the software development activity and frequent reviews of project performance. Based on these reviews and on analyses of the data, changes were made to improve the development process, the programming standards, and the tools used to support development. This resulted in the evolutionary development of a production methodology most appropriate for the STP software development activity. With the encouragement of the procuring agency (U.S. ARMY/ BMDSCOM) and the STP prime contractor (McDonnell Douglas Astronautics Company), TRW actively took the lead in developing and effectively applying modern programming practices. The investment in improved practices has paid off, and the STP software development is on schedule and within cost, and has experienced a significant reduction in the number of errors found, per line of code, in integration testing of completed increments of the deliverable software.

1.2 Objective, Scope and Problems of MPP Study

In view of the RADC interest in improving the technology applied to future Air Force software development, TRW proposed to conduct a study of MPP, concentrating on those used in the STP software development activity. The overall goal of the study was to provide an evaluation of the impact of STP programming principles, guidelines, standards, tools and management practices to determine their potential for application to Air Force software development. RADC contracted with TRW to study the Impact of MPP on System Development (Contract No. F30602-76-C-0095), and this report constitutes the final contractual deliverable of the completed study.

The primary objectives of the study were to:

- Assess the impact of modern programming practices (MPP) applied to the TRW STP software development activity.
- Develop a methodology for comparison of STP MPP with alternative MPP implementations used on other projects by other contractors.

In order to fully achieve these objectives we needed to obtain valuable insights into the relative merits of individual practices and to devise and demonstrate a technique to evaluate the combined effectiveness of selected practices when applied to projects of varying type, size, and complexity.

In the MPP Study Statement of Work it was hypothesized that: Software requirements definition, design, implementation, evaluation and documentation rules, if rigorously defined and applied, and supported by modern techniques and tools, make possible the production of higher quality software at lower than usual cost. One of the primary objectives of the study was to either confirm or refute this general hypothesis specifically with regard to the extensive application of modern practices to the TRW STP software development activity. That is, TRW was expected to obtain and evaluate sufficient relevant evidence about the effectiveness of applied practices to permit an objective conclusion as to the actual validity of the hypothesis (in the STP case) and the probable validity of the hypothesis in general.

In attempting to carry out this task, TRW was faced with several formidable problems. First, the hypothesis, as stated above, is extremely general and ambiguous. The state-of-the-art of software engineering has not yet produced a useful definition of "higher quality software", and, although much has been done recently toward development of software cost estimation methods, the emerging cost models are as yet imprecise, awkward to use, and limited in scope of application. As a result, there is not yet a commonly understood and accepted meaning of "higher quality software" and "usual cost" upon which a test of the hypothesis can be based.

The second major problem has to do with the availability of relevant data to be used in testing the hypothesis. Although a great deal of data on the STP software development activity was collected and available to support the MPP study, the identification of truly relevant data and translation into credible evidence concerning the hypothesis was seen to be a costly and probably imprecise and inconclusive task unless planned and approached in a highly selective fashion.

1.3 Overview of Study Results

The TRW study of the Impact of MPP on System Development is complete and, with the completion of this report, the requirements of the Statement of Work are satisfied. In striving to overcome the possible adverse effects of the above mentioned problems, TRW defined and implemented an innovative MPP impact assessment approach which included:

- Formulation of many, more definitive sub-hypotheses involving the impact of individual practices on characteristics of software and the software development process.
- Accomplishment of a modified Delphi exercise including several surveys of STP personnel to permit both relative quantification and testing of the sub-hypotheses.
- Highly selective examination of STP records and documentation for further testing of certain sub-hypotheses.
- Evaluation of the overall hypothesis through combination of sub-hypothesis test results.

Each of the surveys was based on a list of practices produced by refining and extending the MPP listed in the TRW MPP Study Proposal. The surveys were designed to achieve a balance of:

- desired objectivity in formulation and testing of the hypotheses, and
- essential reliance on actual experience and use of qualified engineering judgement in evaluating MPP impact.

The participants in the impact evaluation study activity were selected from a cross-section of STP management and performer personnel and other key TRW personnel including representatives from staff groups responsible for software technology planning and research.

The surveys were key to the success of the MPP impact assessment activity. In that they were conducted in the STP context, Section 2 of this report provides a detailed discussion of STP software, computing facilities, personnel and a chronology of significant project events. Section 3 presents the study approach in terms of detailed task activities, discusses the steps taken in selecting a final set of MPP for study, and describes in considerable detail the approach, analyses and results of the surveys. Section 4 follows with a discussion of the need for a methodology to permit comparative evaluation of alternative MPP implementations. Section 4 further discusses the feasibility and difficulty of arriving at an acceptable and widely useful comparison methodology, and presents a proposed comparison model and procedure for using it. Finally we demonstrate the proposed comparison methodology through comparative evaluation of the predicted relative impact of two subsets of TRW MPP in the context of a hypothetical software development activity. Section 5 summarizes and discusses the more significant study conclusions, describes the relationship of TRW MPP to current Air Force practices, and provides recommendations for continued research and increased application of high-impact MPP.

In summary, TRW's study of the Impact of MPP on System Development and the study results reported here represent important progress toward confirmation of the hypothesis, viz:

> Modern Programming Practices, if rigorously defined and conscientiously applied, and if supported by appropriate techniques and tools, help make it possible to produce software of higher than usual quality at lower than usual cost.

2.0 SYSTEMS TECHNOLOGY PROGRAM (STP) ENVIRONMENT

2.1 General

TRW chose STP as the "guinea pig" for studying MPP impact for several reasons. First, software development to date spans almost five years during which many modern programming practices have been applied, most from the beginning but some during only more recent development activity. Second, a great deal of data has been collected and is regularly used to analyze and evaluate the performance of STP developers and the software they produce, and, based upon these analyses, an effort has been made to define new production practices and supply developers with new tools to improve performance.

There is one overriding reason for the large number of MPP used in the STP development activity. At the outset of the project (formerly called the Site Defense (SD) program), there was considerable feeling outside TRW that the SD data processing subsystem, and especially the software could never be built to meet the demanding performance specifications, and certainly not within the projected cost and schedule. To meet this challenge, it was necessary to produce not only the heart of the SD system (the real time software) but also the development support software and test support software required to demonstrate that satisfactory performance had been achieved. The sheer magnitude of the software (nearly 1 million machine instructions) and the project (as many as 400 people) demanded unprecedented rigor in the development process and led to the establishment and enforced application of a variety of both existing practices and new practices unique to STP.

The following subsections provide a brief description and historical account of STP to provide the reader with a general understanding of the environment within which these modern programming practices were applied.

2.2 System Description

Site Defense (SD) was intended to be an anti-ballistic missile terminal defense system. It was designed to possess a performance credibility sufficient to deter an aggressor from a first-strike attack against the Minuteman (MM) force and to ensure that an acceptable number of MMs survive in the event of a first strike. Furthermore, the SD System would be capable of countering attacks of various levels and tactics, and of degrading gracefully in the event of subsystem overloads or failures, or in the event of attacks of greater severity than the design threat parameters. The SD System development has been redefined to be a BMD Systems Technology Program (STP) to provide objective evidence of the performance of the key functions of a tactical SD System. The primary objectives of STP are:

- Validating the data processing subsystem by demonstrating the performance of the engagement software (the tactical applications program (TAP) and the tactical operating system (TOS)) executed by the computer, a CDC 7700, against both real targets and simulated threats.

- Providing the framework for incorporating currently deferred data processing subsystem elements.

- Supporting data gathering during system tests at Kwajalein missile range (KMR).

The STP software being developed by TRW has been organized into engagement software (ESW), test support software (TSS), and development support software (DSS) categories. The software in these categories consists of ten major computer programs and involves a total of nearly 1,000,000 machine instructions.

The ESW is the software necessary to identify and track ballistic re-entry vehicles through the use of STP system resources. The logic and algorithms within the ESW consist of those needed to satisfy functional and performance requirements which support the system engagement functions: detect and designate objects, track objects, and discriminate objects.

The software that actually runs on the CDC 7700 (the STP processor), is referred to as a process. A process is composed of a data base, an operating system, and one or more application programs. The Tactical Application Program (TAP) is composed of tasks, which are composed of several levels of routines. The TOS is a table-driven, real-time operating system designed specifically for the CDC 7700. TAP and the test support programs operate under its control. TOS provides the following basic functions: task supervision, scheduling, dispatching, real-time input and output, system timing, data management, history logging, error detection, error processing, initialization, and termination.

The primary component of the Test Support Software (TSS) is the KMR Test Support Program (KTSP). KTSP is required to test key functions of the engagement process and to support system test operations at KMR. Additional test support functions have been provided in the Data Processing Subsystem Simulator (DPSS) and a variety of test tools used in the generation of test data and evaluation of test results. Development of two other TSS components (i.e., the System Environment and Threat Simulator (SETS) and the System Test Driver (STD)) was initiated, but continued development was recently deferred.

The Development Support Software consists of the Basic Operating System (BOS), the Process Construction Program (PCP), specialized development support tools, and the Data Reduction and Report Generator (DRRG). BOS, the primary operating system used in software development, consists of a specially tailored version of the SCOPE 2 operating system for the CDC 7600/7700 plus the associated loaders, compilers, assemblers, and utilities. The functions of PCP are data base definiton, data base generation, task/routine compilation, process consolidation and process adaptation using a higher-level, FORTRAN-like language which facilitates construction of the real-time software. PCP, using the process designer's input directives and definitions and a library file of coded routines and tasks, assembles the application program, constructs the tables defining

the program operation to the tactical operating system, and links the entire process together.

Specialized tools consist of a number of utility programs that aid developers in the design, execution, and evaluation of SD software. They provide information useful in static and dynamic analysis of the software and relieve developers and testors of many repetitive and tedious tasks.

DRRG is an off-line program used to postprocess in a non-real-time environment the data generated by the test processes. DRRG supports analysis and reduction of real-time execution history logs and generates reports based on user requests.

2.3 STP Programming Environment

2.3.1 The Computing Facility

The major elements of the computer hardware system consist of CDC 6400, 7600, and 7700 computers. The CDC 6400 computer is used primarily as an input and output station for the CDC 7600 and CDC 7700 computers. The CDC 7600 computer was used in the early phase of software development and was replaced by the CDC 7700, which is essentially two CDC 7600's with a shared large core memory. The 7600 and 7700 main frames have been supported by an extensive complement of peripherals driven by the CDC 6400. At the peak of the configuration the system contained two card readers, one card punch, six tape drives, eleven disk packs, six printers, one large disk file, and three operators consoles. Off-line peripherals consisted of one IBM 360-20 and three calcomp plotters.

All STP software development up to July 1976 has been done in batch mode with some standalone testing of the real time system. No time sharing or remote job entry systems have been used. After the CDC 7700 system was shipped to KMR (July 1976) a remote job entry (RJE) system connected to the McDonnell Douglas Astronautics computer facility in Huntington Beach, California has been used by TRW for software maintenance and additional development. At the peak operation (mid-1974) approximately 600 batch jobs per day were processed by the computer facility at TRW. Availability of the hardware grew from 85% (in early 1973) to an average of 97% (mid-1976). Turn around time was greatest (4 hours) in mid 1974 for classified runs and has averaged approximately 40 minutes for all runs from mid 1974 through mid 1976. Throughout 1975, the second 7600 processor was devoted for one full shift to real-time testing of TOS and TOS/TAP. The computer facility was available for 1.5 shifts in 1973, 2.5 shifts in 1974, 2 shifts in 1975, and two shifts for the first 6 months of 1976.

CDC provided a complete set of user documentation for the computer system and vendor supplied software. The early availability of up-to-date, accurate, and well written software documentation provided programmers who were not familiar with the CDC system, a source of readily available documents which were kept current by CDC through their documentation update system. Appendix B contains a list of the major vendor supplied software elements.

A systems support group was established to be responsible for maintaining the CDC supplied software. This group had the responsibility to develop STP unique improvements and additions to the CDC software. A major product of this group was the Basic Operating System (BOS) under which most STP software was developed. BOS is an extension of the SCOPE 2.1 Operating System and provides special features which are required for STP software development.

2.3.2 The People

TRW's experience in large scale software systems development, anti-ballistic missile technology, radar systems, and systems engineering provided a readily available source of highly qualified software managers, programmers, systems analysts, software engineers, and support personnel for assignment to the STP project. The large TRW software personnel pool and the STP proposal team made it possible to staff the STP project with a nucleus of qualified people, augmented by additional people hired in the open job market. At its peak the project staff exceeded 400 personnel. A history of the project population through mid-1976 is shown in Figure 2-1. The figure also illustrates the personnel mix (educational background, degree level and experience) that has remained relatively constant despite the large variation in project size.

The initial set of STP software managers and key technical personnel from the STP proposal team were familiar with the detailed technical issues of the software tasks, and enabled the STP project to operate effectively from the beginning with very little of the typical initial start-up "learning curve" problems. In fact, all of the key management and technical positions in the early project organization (Figure 2-2) were readily filled by members of the TRW STP proposal team. As the STP organization changed in keeping with the growth in project personnel and the progressive phases of development activity, the approach to software management has been to promote from within the project or to use experienced TRW personnel from other projects to fill key positions.

The programmers on the STP project fell into three major categories: 1) CDC operating systems support programmers, 2) real time systems programmers, and 3) application programmers. Since the primary computing facilities at TRW already use CDC 6000 series computers, the STP project had available to it a source of qualified systems programmers who needed little additional training to use the CDC 7000 series computers. Also, since the Tactical Application Program and Test Support Software were written in FORTRAN, the availability of experienced programmers within TRW and the open job market was not a problem. TRW's experience in simulation, radar systems, antiballistic missile technology, test tool development, and test support software provided a nucleus for the applications programming area. In short, obtaining a qualified STP programming staff was not a problem.

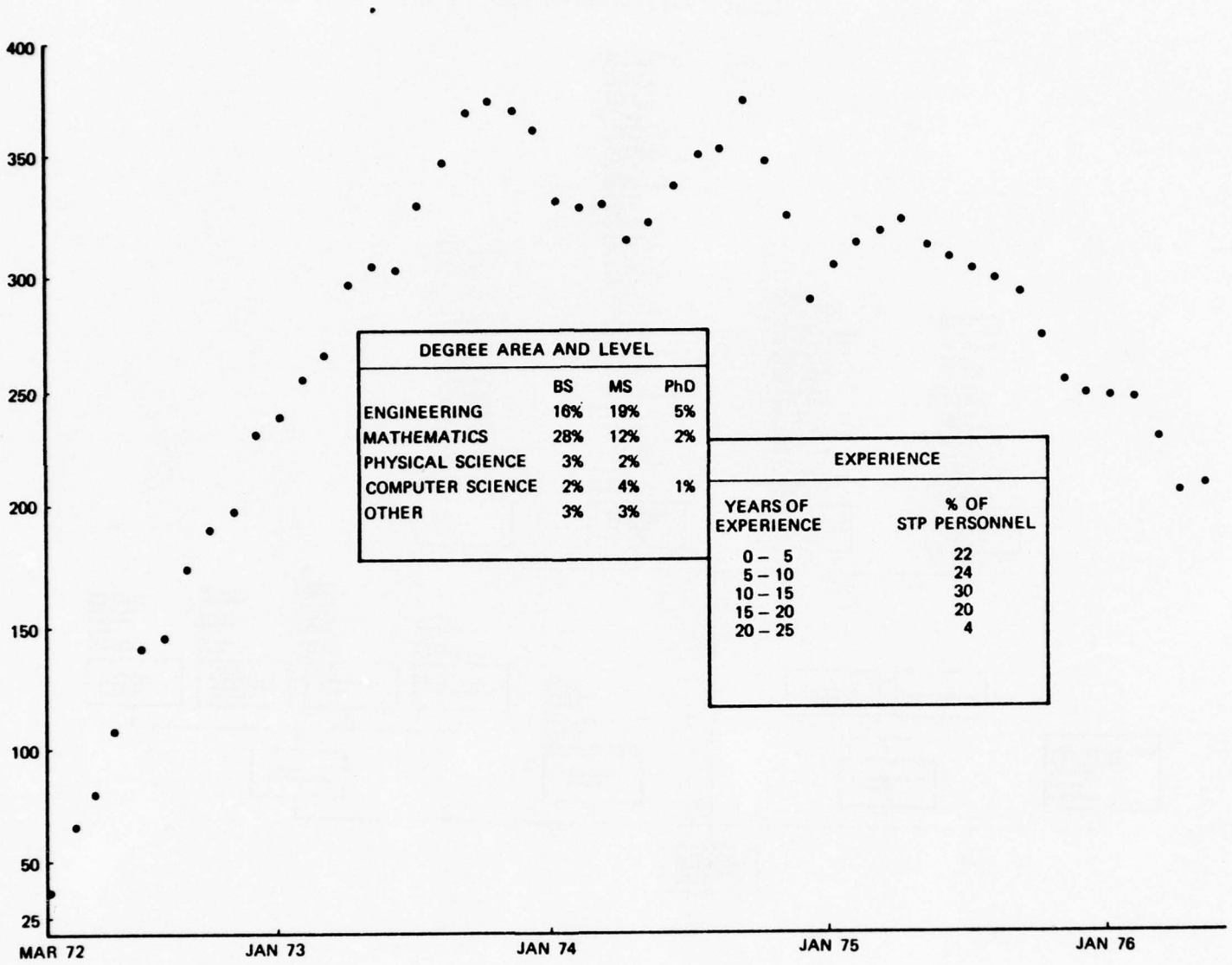


Figure 2-1. STP Staffing

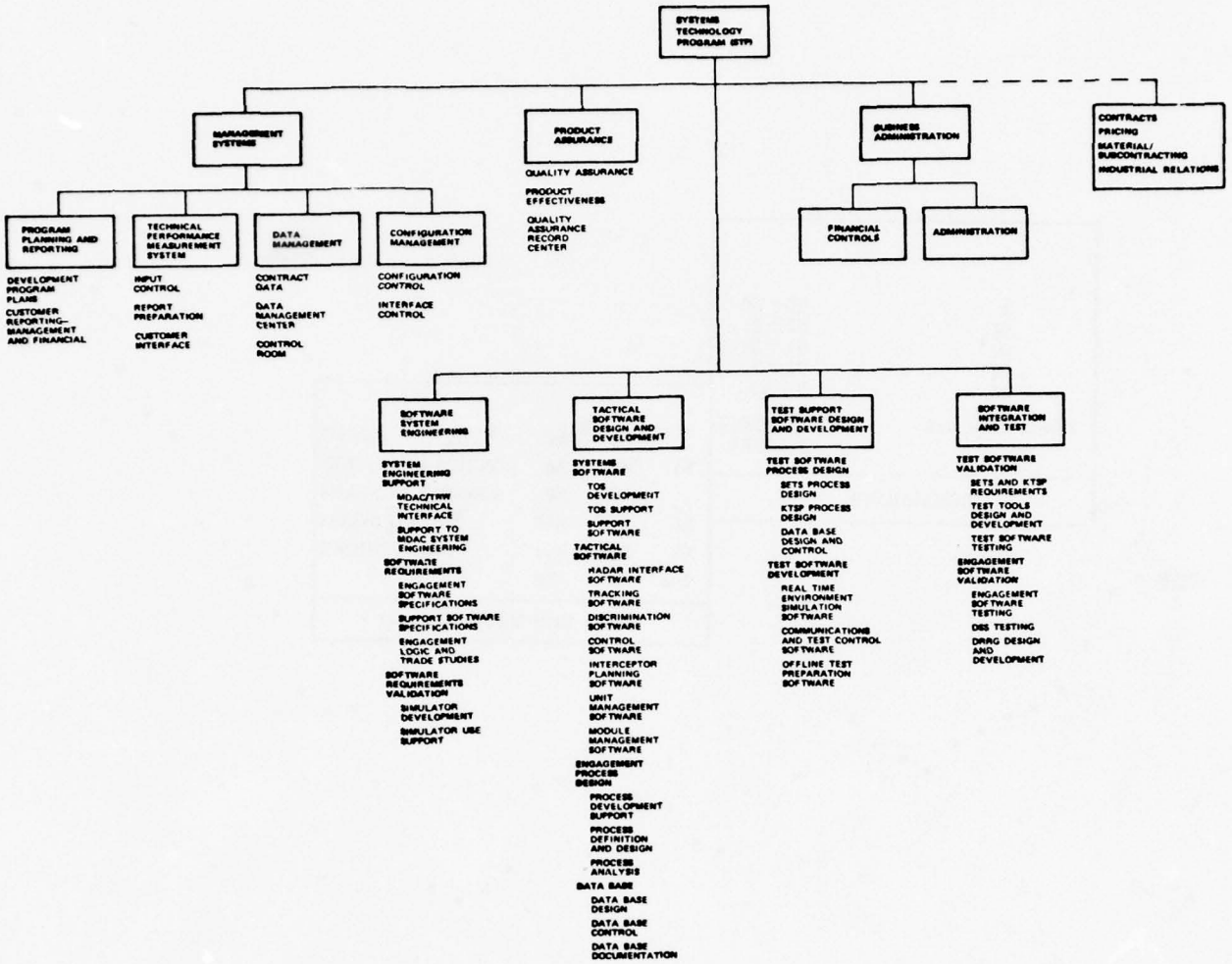


Figure 2-2. STP Organization (1972)

2.4 STP Chronology

The STP Program began with the award of multiple contract definition (CD) phase contracts to three teams of contractors one of which was the team of McDonnell-Douglas, TRW, GE, and CDC. At the end of the CD phase, each team was required to submit proposals for future STP development and the McDonnell-Douglas team won this competition. Since that initial award the program has undergone multiple redefinitions (here called "Reprogramming") mainly as the result of congressional action. The major effect of this reprogramming has been to stretch schedules and redefine the content of the computer program deliverables. The approximate contractual chronology for STP is as follows:

- CD Contract Award - April 1971
- CD Phase Complete/STP Proposal Submitted - November 1971
- STP Contract Award - March 1972
- First Reprogramming (66 month program) - April 1973
- Second Reprogramming (72 month program) - March 1974
- Third Reprogramming (80 month program) - February 1975
- Fourth Reprogramming (86 month program) - March 1976

Owing largely to the effect of the reprogramming on the STP software development schedules, but also due to TRW's phased (Incremental Development) approach, it is extremely difficult to provide a single, yet meaningful illustration of the detailed STP development chronology. For this reason, Figure 2-3 depicts the start and completion dates for increments (called "loops"), versions and critical releases of major ESW, TSS and DSS components. More detailed chronologies are provided in tabular form in Appendix B.

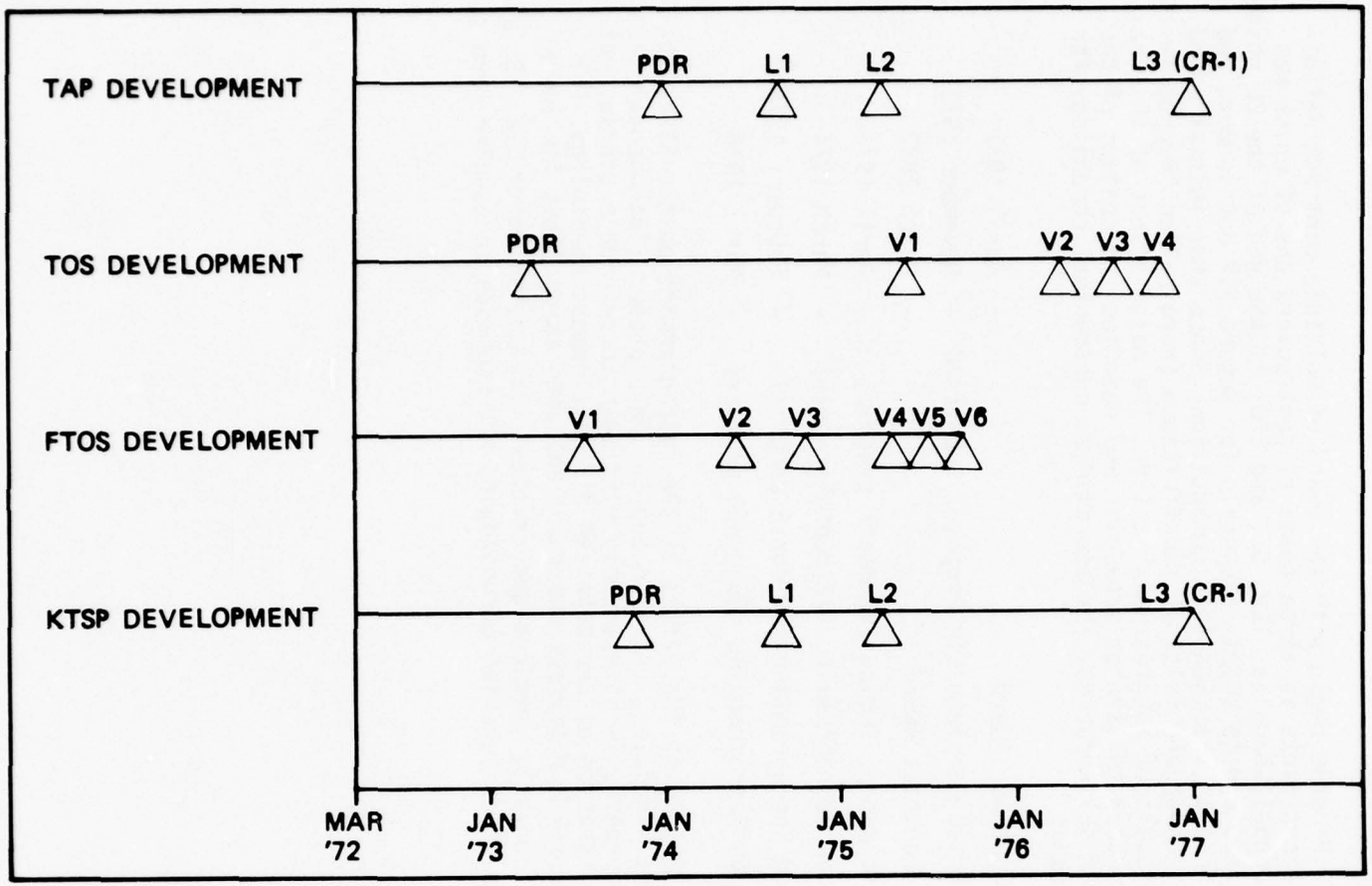


Figure 2-3. STP Software Development Chronology

3.0 MPP IMPACT EVALUATION APPROACH AND RESULTS

3.1 Detailed Technical Approach

In response to the Statement of Work, TRW defined and followed the MPP study approach described below as a sequence of tasks. The tasks, in combination, provided necessary engineering analyses to 1) accomplish an in-depth evaluation of the effectiveness of MPP in the STP software development environment, and 2) develop and demonstrate a methodology for comparative evaluation of alternative MPP implementations. Figure 3-1 illustrates the approximate sequencing of project activities and demonstrates the high interconnectivity of the tasks described below.

Task 1: STP MPP Evaluation

This task included: identification of candidate practices and selection and definition of a final set of practices to be included in the impact evaluation study; collection, organization and analysis of pertinent STP data and reconstruction of the chronology of major milestones in the project development cycle; development of definitions for project terminology; development of a description of the STP software development environment; and detailed analysis to determine the effects of adopting specific programming practices.

Subtask 1.1: MPP Identification, Selection and Definition

This subtask included: review of the candidate MPP list provided in the TRW MPP Study proposal; identification of additional key practices through interviews with STP personnel; refinement of the MPP set through a series of surveys and ranking exercises to permit concentrated study and impact evaluation of a small number of key STP practices; and preparation of both brief and expanded definitions of each selected practice to support the impact evaluation activity.

Subtask 1.2: STP Chronology Reconstruction

This subtask included: identification, collection and review of relevant historical records, briefing materials, meeting reports, progress and status reports and formal software development plans; identification of key STP events; and preparation of a reconstructed chronology of STP events including begin and end dates of distinct development activities for each of the major components of the software, intermediate and final milestones, and dates marking both the availability and first application of specific programming practices.

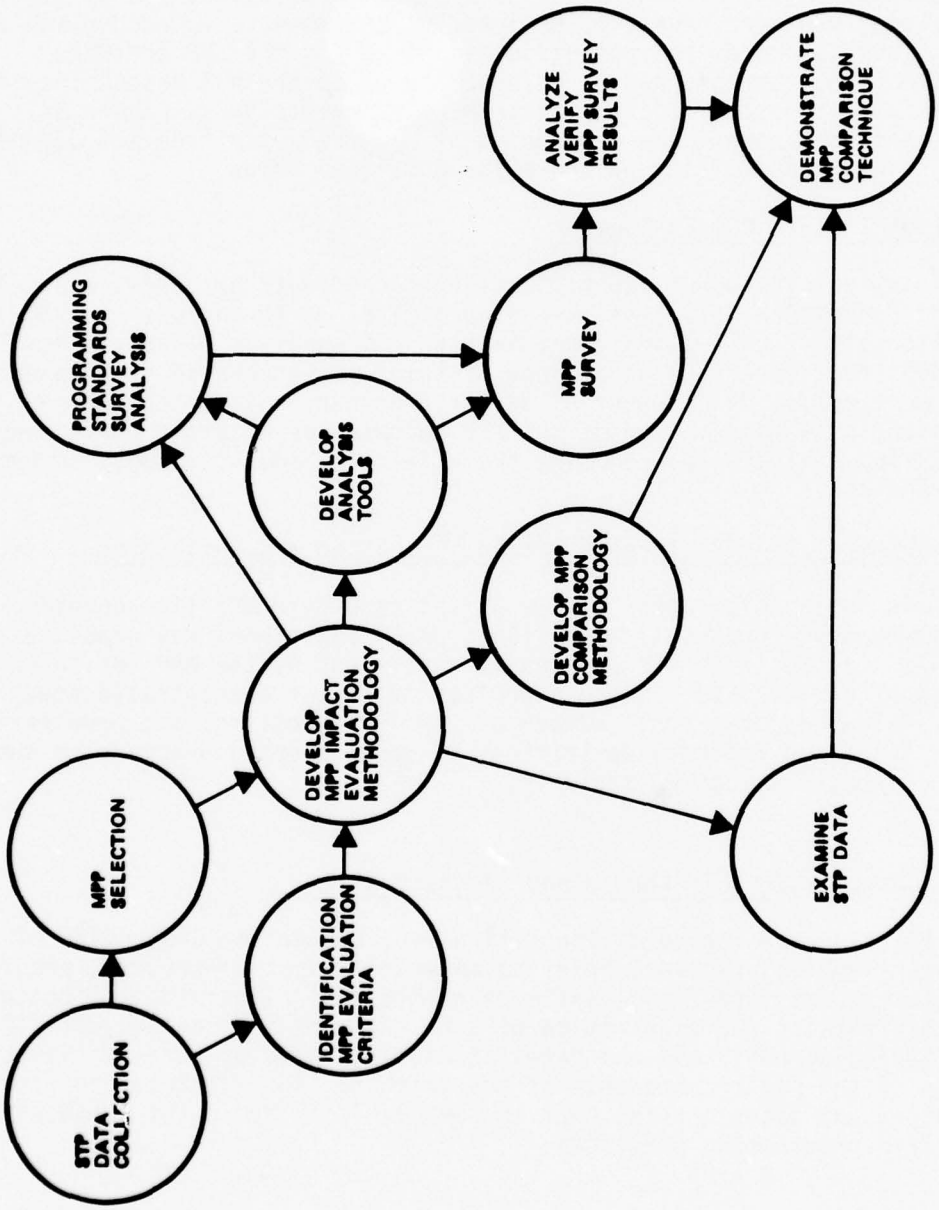


Figure 3-1. MPP Project Activity Network

Subtask 1.3: STP Terminology Definition

This subtask included: identification of both general and specific terms unique to or especially important in the STP and/or MPP impact evaluation context; preparation of brief definitions of all such terms; and production of a glossary of selected definitions for inclusion in this report.

Subtask 1.4: STP Development Environment Description

This subtask included: collection and review of formal system and data processing subsystem documentation and project staffing historical records; and preparation of a description of the STP software development environment including 1) general characteristics of the computer hardware configuration and operational support software (e.g., operating system, library utilities, compilers, assemblers) 2) the availability of computing resources through batch, remote batch and time sharing services at various stages of the software development, and 3) an illustration of project personnel staffing in terms of the availability of specialized skills, fields and levels of education and directly related experience during major phases of the software development activity.

Subtask 1.5: MPP Impact Evaluation

This subtask included:

- design of a framework to provide for the collection and representation of quantitative data on the impact of practices on software quality, cost and schedule
- development of a technique (i.e., a modified Delphi exercise) for obtaining individual assessments of the impact of practices based on detailed analysis of available data by key STP personnel
- preparation for and performance of two surveys of a cross section of STP personnel to obtain impact assessments of 1) detailed Programming Standards on desirable software qualities and 2) MPP on typical problems that have plagued software development
- development and application of statistical analysis algorithms to interpret cumulative response data and formulate composite impact measures
- computation of figures of merit representing the average impact of 1) each practice on all problems, 2) all practices on each problem, and 3) all practices on all problems.

Task 2: MPP Comparison Methodology Development

This task included: development and demonstration of a methodology for comparative evaluation of alternative MPP implementations; and preparation and oral presentation of a project status and comparison methodology definition briefing to RADC.

Subtask 2.1: Methodology Definition

This subtask included: investigation of alternative approaches to MPP comparison; selection of a technique that was both 1) feasible to implement and use for comparative evaluation of many MPP implementations and 2) compatible with the MPP impact evaluation/representation framework developed in Task 1; and demonstration of the comparison methodology via comparative evaluation of two distinct MPP implementations.

Subtask 2.2: Oral Presentation

This subtask included: collection, analysis and refinement of materials relevant to the status of the MPP definition, impact evaluation and comparison methodology development activities; and preparation and presentation of an oral briefing to RADC personnel.

Task 3: Technical Report Preparation

This task included all activities necessary for preparation of this technical report to include:

- Information to describe the TRW Systems Technology Program (STP) software development activity and key modern programming practices used in STP software production,
- A description of the approach taken to obtain quantified assessment of the impact of MPP on software quality, cost and schedule, the results achieved from the impact evaluation studies, and corresponding conclusions and recommendations,
- A description of the methodology for comparative evaluation of alternative MPP implementations and an exemplary demonstration of the methodology and support tools in comparing the impact of two subsets of STP MPP, and
- Rationale and recommendations for 1) Air Force consideration of key practices for inclusion in evolving software development guidelines, and 2) continued research and development tasks necessary to make proper use of TRW MPP study results and achieve improved insights into the contribution of modern programming practices to a disciplined and cost-effective software engineering process.

3.2 Selection of MPP

The term, Modern Programming Practices (MPP), generally refers to standard programming practices, software design, construction, test and documentation techniques and management methods used in the production of software for the purpose of enhancing certain qualities of the software and/or reducing, eliminating or circumventing known software production difficulties. Clearly, a review of almost any large software development activity could be expected to uncover some practices that easily qualify as MPP. Due to the highly critical nature of the STP software development, it is not surprising to find that the set of such practices employed at various stages of the project is large indeed. Given the primary objective of the MPP study (i.e., to evaluate and quantify the impact of MPP on the STP development) and given limited time and funding to complete the task, it was clearly necessary to choose a reasonably small number of MPP to be studied so that sufficient attention could be given to each.

Our approach to obtaining a final list of MPP was straightforward and involved the following:

- Identify a baseline set of practices through extensive interviews with key STP personnel.
- Survey a cross section of STP and non-STP personnel to obtain a relative-importance ranking of the MPP and to identify additions to the baseline set.
- Analyze the expanded list of MPP, delete those of low rank and those highly peculiar to STP and, where appropriate, combine several related individual practices into one more general practice.

Through iterative application of the above steps, we progressed from the original list of 14 candidate practices to the final set of 11 MPP as illustrated by Figure 3-2. Appendix A contains brief descriptions of the 14 candidate practices and detailed definitions of the final 11 MPP.

It is important to note that we sought not only a manageable number of MPP upon which to concentrate the impact analyses but also a representative collection of practices that spanned the major phases of STP development. Of the set of 11 MPP:

- 4 practices apply mainly to the requirements definition/preliminary design phases of software development:
  - Requirement Analysis and Validation
  - Baselining of Requirements Specification
  - Complete Preliminary Design
  - Process Design

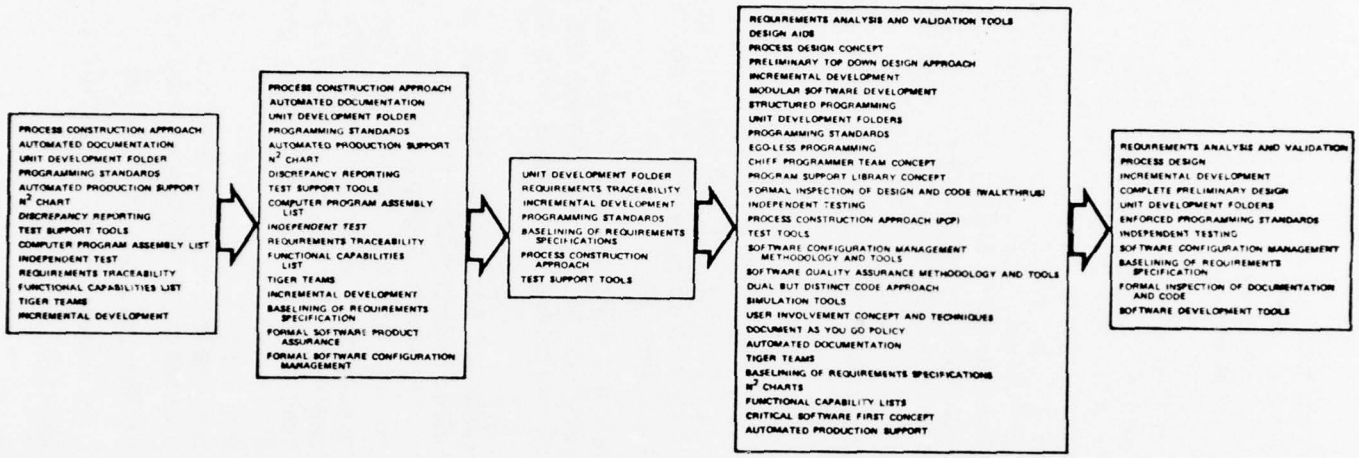


Figure 3-2. MPP Selection

- 3 practices apply mainly to the detail design and coding phases:
  - Enforced Programming Standards
  - Incremental Development
  - Unit Development Folders
- 2 practices apply mainly to the test phase:
  - Independent Testing
  - Formal Inspection of Documentation and Code
- 2 practices apply to all phases of software development:
  - Software Development Tools
  - Software Configuration Management

3.3 Impact Evaluation Study

A primary objective of the Impact of MPP on System Development project was to obtain a quantified assessment of the nature and extent of the effect of modern practices applied to the TRW STP software development. Doing so was expected to provide important evidence with respect to the general hypothesis (Section 1.2) as to the net positive impact of MPP on software quality and cost.

Selection of an impact assessment approach and methodology was strongly influenced by previous TRW experience on several related advanced software technology studies. These were the Quantitative Software Safety Study [14] under contract to McDonnell Douglas Astronautics Company and the Characteristics of Quality Software study [15] for the National Bureau of Standards. Both of these studies involved measuring the quality of software and required identification of explicit characteristics (i.e., detailed qualities)of software and associated metrics with which a quantified quality assessment could be achieved.

Similarly for the MPP study, the approach to evaluating the overall impact of practices on software quality and cost involved 1) establishment of a common set of software characteristics, and 2) development and use of a technique to obtain measures of the nature and extent of the cause effect relationship between each practice and each characteristic.

In view of particular interest in programming standards, two distinct impact evaluation activities were planned and conducted. The first focused on 18 detailed documentation and coding standards from the current version of the STP Software Standards and Procedures Manual [10] and sought an impact assessment relative to 30 characteristics of software and the software development process. The second impact evaluation activity then concentrated on the 11 more general MPP and sought an impact assessment relative to 12 typical problems that have plagued (and all too often characterize) software development. The first evaluation thus called for 540 (i.e., 18 x 30) individual measures of the impact of the practices on the characteristics, while the second required 132 (i.e., 11 x 12) impact measures, one for each practice-problem combination.

Owing to the large number of individual impact measures to be obtained and the attendant magnitude of effort required to collect and evaluate data for each, it was decided that the basic technique for obtaining impact measures should be a modified Delphi exercise involving several surveys of experienced TRW personnel. Use of this technique not only made the acquisition of comprehensive, detailed impact assessment data a practical task, it also brought into the impact evaluation activities a broad spectrum of project knowledge and thus varying impact assessments based on the unique and differing experiences of a cross section of STP personnel. In order to ensure maximum objectivity in the analysis of response data and formulation of a composite impact measure for each practice-characteristic combination, analysis algorithms were established in advance and implemented in the RANK, IMPACT and MERIT support programs.(See Appendices C and D for detailed descriptions of the algorithms and support tools.)

In essence, the impact evaluation approach was:

- Conduct a survey requiring each participant to provide an individual measure (influence rating) of the impact of specific practices on certain characteristics of software development
- Analyze the survey response data as necessary to obtain a composite impact measure for each practice-characteristic combination as well as measures of the impact of 1) each practice on all of the characteristics, 2) all of the practices on each characteristic, and 3) all of the practices on all of the characteristics.

3.3.1 Programming Standards Impact

The first survey concentrated on detailed programming standards and sought measures of their impact on characteristics of software and the software development process. The survey questionnaire was completed by 54 people representing a cross section of STP personnel as follows:

Tactical Software Design & Development	Operating System Software	Test Support Software Design & Development	Software Integration and Test	Product Assurance & Configuration Management	Other
17	9	11	11	3	3

The Programming Standards Impact Evaluation survey questionnaire contained three parts:

1) A two dimensional matrix of 18 programming standards versus 30 software characteristics (i.e., 540 "multiple choice/fill-in-the-blank" questions)
2) A true/false question posed as a restatement of the general MPP hypothesis: "Programming (i.e., documentation and coding) standards, if rigorously defined and systematically enforced with support tools, help make possible the production of software of higher than usual quality at lower than usual cost".
3) A request for a free-form statement (in 25 words or less) as to the general effect of programming standards on STP.

Figure 3-3 illustrates the blank survey questionnaire.

Each survey participant was asked to complete as much of the matrix as possible and, from personal experience, to indicate both the nature and amount of influence of each standard on each characteristic by choosing and entering the most appropriate influence rating from the following:

+2	Strong Positive Influence
+1	Some Positive Influence
0	No Influence
-1	Some Negative Influence
-2	Strong Negative Influence
Blank	Unknown Influence

Upon completion of the survey the mathematical algorithms and support programs were used to analyze and summarize the response data. The IMPACT program provided analysis of the response-frequency- distribution (i.e., the number of +2, +1, 0, -1, and -2 responses) for each element of the matrix, computation of mean and variance, and application of a hypothesis testing procedure (summarized in Table C-III of Appendix C) to derive both summary assessments of impact (i.e., positive, negative or none) and confidence levels (i.e., strong, medium, weak) for assertions:

"Practice i has a positive (negative) influence on characteristic j."

The summarized results of the Programming Standards Impact Evaluation survey are presented in Figure 3-4. The figure also contains a legend describing the abbreviated notation for the possible combinations of assertion strength/influence rating which appear in the elements of the matrix, an overall summary of the matrix contents, and the percentages of true, false and "don't know" responses to the general hypothesis.

The IMPACT program also computed influence ratings for each row (practice), each column (characteristic), and for the total matrix corresponding to 1) the average effect of each practice on the combined set of characteristics, 2) the average effect of the combined practices on each characteristic, and 3) the average effect of the combined practices on the combined set of characteristics. These additional assertion strength/influence ratings can be viewed as a 19^{th} row and 31^{st} column of the matrix as indicated in Figures 3-5 and 3-6.

The assertion strength/influence rating for the combined practices on the combined characteristics was determined to be Weak/Positive. The weakness of the assertion strength is due to the large number (130) of Indifferent (i.e., no effect) ratings for individual practice-characteristic pairs , while the overall influence rating derives from the ratio of positive to negative influence ratings ($292/17 \cong 17.2$) which clearly suggests Positive impact.

PROGRAMMING STANDARDS IMPACT EVALUATION SURVEY

SOFTWARE CHARACTERISTICS

INFLUENCE RATING	
+ 2	Strong positive
+ 1	Some positive
0	None
- 1	Some negative
- 2	Strong negative

		Requirements Traceability	Code Auditability	Documentation Auditability	Personnel Motivation	Interface Consistency	Software Integration	Documentation Means	Coding Means	Retesting	Customer/Contractor Relationship	Testing Thoroughness	Documentation Understandability/Readability	Code Understandability/Readability	Documentation Maintainability/Usability	Code Maintainability/Usability	Testability	Operational Reliability	Cost	Schedule	Consistency	Completeness	Documentability	Codability (ease of coding)	Documentation Error Frequency	Coding Error Frequency	Portability (Computer Independence)	Execution Time	Core Requirements	Logical Organization	Programmer Productivity
STANDARDS FOR DELIVERABLE DOCUMENTATION	Text Format																													
	Text Level of Detail																													
	Flow Chart Format																													
	Flow Chart Level of Detail																													
DETAILED CODING STANDARDS	Statement Label Format																													
	Executable Statement Format																													
	Routine Size (Modularity)																													
	Calling Sequence Arguments																													
	Mixed Mode Arithmetic																													
	DO-LOOP Usage																													
	Computed GO-TO Usage																													
	Labeled Common vs Blank Common Usage																													
	Imbedded Physical Constants Usage																													
	Preface Common Block Requirement																													
	In-line Commentary Requirement																													
	Structured Coding Requirement																													
	Execution of Every Program Branch Requirement																													
	Naming Conventions																													

A. Programming (i.e., documentation and coding) standards, if rigorously defined and systematically enforced with support tools, help to make possible the production of software

	True	False	?
o of higher than usual quality	☐	☐	☐
o at lower than usual cost.	☐	☐	☐

B. (In 25 words or less) STP PROGRAMMING STANDARDS had the following general effect _____

_____

Name: _____

Mail Station: _____

Figure 3-3. Programming Standards Survey Questionnaire

PROGRAMMING STANDARDS IMPACT EVALUATION SURVEY

ASSERTION STRENGTH	INFLUENCE RATING					
	STRONG POSITIVE	POSITIVE	INDIF-FERENT	NEGATIVE	STRONG NEGATIVE	INCON-CLUSIVE
Strong	SSP	SP	I	SN	SSN	SI
Medium		MP		MN		MI
Weak		WP		WN		WI

SOFTWARE CHARACTERISTICS

		1 Requirements Traceability	2 Code Auditability	3 Documentation Auditability	4 Personnel Motivation	5 Interface Consistency	6 Software Integration	7 Documentation Rework	8 Coding Rework	9 Retesting	10 Customer/Contractor Relationship	11 Testing Thoroughness	12 Documentation Readability	13 Code Understandability/Readability	14 Documentation Maintainability/ Usability	15 Code Maintainability/Usability	16 Testability	17 Operational Reliability	18 Cost	19 Schedule	20 Consistency	21 Completeness	22 Documentability	23 Codability (ease of coding)	24 Documentation Error Frequency	25 Coding Error Frequency	26 Portability (Computer Independence)	27 Execution Time	28 Core Requirements	29 Logical Organization	30 Programmer Productivity
STANDARDS FOR DELIVERABLE DOCUMENTATION	1 Text Format	SP	WP	SP	WI	MP	WP	SP	I	WP	SP	WP	SP	WP	SP	WP	WP	WP	MI	MI	SP	MP	SP	I	MP	I	I	I	I	MP	MI
	2 Text Level of Detail	SP	I	SP	MI	MP	MP	SI	WI	WP	SP	WP	SP	WP	MP	WP	WP	WP	SN	MI	MP	SP	SI	WP	MI	I	I	I	I	MP	SI
	3 Flow Chart Format	WP	WP	SP	WI	WP	WP	MP	WP	WP	SP	WP	SP	MP	MP	WP	WP	I	WI	WI	SP	MP	MP	MP	MP	WP	WP	I	I	MP	MP
	4 Flow Chart Level of Detail	MP	WP	MP	MI	MP	MP	SI	MP	MP	MP	MP	SP	MP	SI	MP	WP	WP	SI	MI	MP	SP	MI	SP	MI	MP	WP	I	I	MP	SI
DETAILED CODING STANDARDS	5 Statement Label Format	I	MP	I	WI	I	I	I	MI	WP	I	I	I	SP	I	MP	WP	I	WI	WI	MP	I	WP	MI	I	MP	I	I	I	MP	WI
	6 Executable Statement Format	I	MP	I	WI	I	WP	I	WI	I	WP	I	I	MP	I	MP	WP	I	WN	WN	SP	I	WP	MI	I	MP	WP	I	I	WP	WI
	7 Routine Size (Modularity)	MI	SP	MP	MI	MI	SI	MI	SI	MP	WP	MP	MP	SP	MP	SP	SP	MP	SI	SI	MP	WP	MP	SI	WI	MP	WP	SN	MN	SP	SI
	8 Calling Sequence Arguments	I	WP	I	I	MP	MP	I	WP	WP	WP	I	I	MP	I	MP	WP	WP	WI	WI	MP	I	WP	MI	I	WP	WP	WI	WN	WI	WI
	9 Mixed Mode Arithmetic	I	WP	I	WI	WP	WP	I	WP	WI	I	I	I	WP	I	WP	WP	MP	I	I	MP	I	I	WI	I	MI	MP	WI	WI	I	WI
	10 DO-LOOP Usage	I	MP	I	WI	I	I	I	WP	WP	I	I	I	MP	I	WP	WP	WP	WI	I	MP	I	I	WP	I	WP	WP	WP	WP	WP	WP
	11 Computed GO-TO Usage	I	MP	I	MI	I	I	I	WI	WI	I	WI	I	MP	I	MI	WP	MP	WI	WI	MP	WP	I	SI	I	MI	WI	MI	WI	MP	MI
	12 Labeled Common vs Blank Common Usage	I	MP	I	I	SP	MP	I	WP	WP	I	I	WP	MP	WP	MP	WP	WP	WP	WP	MP	WP	WP	MI	I	MP	WP	I	I	WP	WP
	13 Imbedded Physical Constants Usage	MP	MP	I	I	SP	MP	MP	MI	MP	WP	WP	WP	MP	WP	SP	MP	MP	WP	WP	SP	WP	WP	MI	I	MP	WP	WN	WI	I	WP
	14 Preface Commentary Block Requirement	MP	SP	WP	MI	MP	MP	MI	MI	WP	SP	I	MP	SP	MP	SP	WP	I	MI	MI	SP	MP	MP	MI	MP	WP	I	I	I	I	MI
	15 In-line Commentary Requirement	MP	SP	WP	MI	WP	WP	WI	SI	MP	SP	WP	WP	SSP	WP	SSP	WP	WP	MI	MI	MP	MP	MP	MI	WP	MP	I	I	I	MP	SI
	16 Structured Coding Requirement	WP	SP	WP	SN	I	WP	WI	SI	WP	MP	WP	MP	SP	WP	SI	MP	MP	SN	SN	SP	WP	MP	MN	WP	MI	WP	SN	SN	SP	SN
	17 Execution of Every Program Branch Requirement	WP	WP	I	MI	WP	MP	I	WI	SI	MP	SSP	I	I	I	I	MI	SSP	SN	SN	MP	MP	I	I	I	MP	I	I	I	I	MN
	18 Naming Conventions	WP	MP	MP	WI	MP	SP	WP	WP	I	MP	I	MP	SP	MP	MP	WP	WP	WI	WI	SP	WP	MP	SI	MP	MP	I	I	I	WP	MP

SURVEY SUMMARY	
RATING	FREQUENCY
SSP	4
1	130
SSN	0
SP	45
SI	20
SN	10
MP	114
MI	41
MN	3
WP	129
WI	40
WN	4

A. Programming (i.e., documentation and coding) standards, if rigorously defined and systematically enforced with support tools, help to make possible the production of software:

	True	False	?
• of higher than usual quality	85%	4%	11%
• at lower than usual cost.	28%	50%	22%

B. (In 25 words or less) STP PROGRAMMING STANDARDS had the following general effect: _____

Name: _____

Mail Station: _____

Figure 3-4.

Practice Index	Practice Identification	Assertion Strength/ Influence Rating
1	Text Format	WP
2	Text Level of Detail	MP
3	Flow Chart Format	WP
4	Flow Chart Level of Detail	MP
5	Statement Label Format	WP
6	Executable Statement Format	WP
7	Routine Size (Modularity)	MI
8	Calling Sequence Arguments	WP
9	Mixed Mode Arithmetic	WI
10	Do-LOOP Usage	WP
11	Computed GO-TO Usage	WI
12	Labeled Common vs Blank Common Usage	WP
13	Imbedded Physical Constants Usage	WP
14	Preface Commentary Block Requirement	MP
15	In-line Commentary Requirement	MP
16	Structured Coding Requirement	MI
17	Execution of Every Program Branch Requirement	WI
18	Naming Conventions	WP

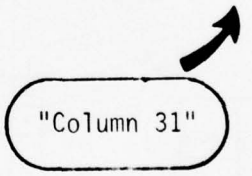


Figure 3-5. · Practice Impact on Combined Characteristics

Characteristic Index	Characteristic Identification	Assertion Strength/ Influence Rating
1	Requirements Traceability	WP
2	Code Auditability	MP
3	Documentation Auditability	WP
4	Personnel Motivation	WI
5	Interface Consistency	WP
6	Software Integration	WP
7	Documentation Rework	WI
8	Coding Rework	MI
9	Retesting	WP
10	Customer/Contractor Relationship	MP
11	Testing Thoroughness	WP
12	Documentation Understandability/Readability	WP
13	Code Understandability/Readability	MP
14	Documentation Maintainability/Usability	WP
15	Code Maintainability/Usability	MP
16	Testability	WP
17	Operational Reliability	WP
18	Cost	MI
19	Schedule	WI
20	Consistency	MP
21	Completeness	WP
22	Documentability	WP
23	Codability (ease of coding)	MI
24	Documentation Error Frequency	WI
25	Coding Error Frequency	MP
26	Portability (Computer Independence)	WP
27	Execution Time	SI
28	Core Requirements	SI
29	Logical Organization	WP
30	Programmer Productivity	MI

"Row 19"

Figure 3-6. Combined Practices Impact on Each Characteristic

There are far too many individual elements of the matrix to permit detailed discussion of the assessed impact of each standard programming practice on each of the software characteristics. Certain of the impact evaluation results are particularly interesting, however, and are deserving of special mention and, in some cases, further explanation. These are:

- The Structured Coding Requirement received a positive influence rating for 18 of the 30 software characteristics, a negative for 7 characteristics, an indifferent rating for one characteristic, and an inconclusive rating for all the rest, including the average rating across all 30 software characteristics. Of the 18 positive ratings, however, only 4 can be asserted with strong confidence, 5 with medium confidence and 9 with only weak confidence. On the other hand, just one of the 7 negative ratings can be asserted with medium confidence and the other 6 with strong confidence. Compared with the other standards which all together received only 10 negative ratings, this indicates a relatively dim view of structured coding on the part of STP personnel. There are some good reasons for this to be the case. First, it is well recognized that structured coding (unlike most of the other standards) generally requires more core and execution time. Additionally, the standard was not explicitly defined, established and enforced until almost two years after STP began, and the requirement to restructure existing, working code was felt to be both unnecessary and counter-productive. Moreover, writing structured code in standard FORTRAN is awkward and introduced some inefficiencies in core and execution time making it difficult to satisfy demanding design and performance requirements levied on the real-time software. Finally, STP has not yet reached the phase during which the major benefits of structured programming (i.e., improved readability and maintainability) are expected to be reaped.

- Approximately one out of every four of the standard-characteristic pairs received an assertion strength/influence rating of Indifferent. The Indifferent rating implies a strong assertion of no impact, in that over 75% of the survey participants had to enter a no-influence (0) rating for the composite rating of Indifferent to be obtained. Although at first the number of Indifferent ratings seems high, careful review of the matrix reveals a very good match with intuition.

- There are eight software characteristics which received no negative or inconclusive impact assessments for any of the 18 standards. These are: Code Auditability, Documentation Auditability, Customer/Contractor Relationship, Documentation Understandability/Readability, Code Understandability/Readability, Operational Reliability, Consistency, and Completeness. On the other hand, Personnel Motivation received 14 inconclusive ratings (showing mixed positive and negative responses from the survey participants), three indifferent ratings and one negative. It appears that the overall impact of standards on Personnel Motivation correlates much better with the preceived impact on Cost and Schedule (i.e., only two positive influence ratings for each) than with the above-mentioned eight characteristics. This says something quite positive about the objectivity of the responses; i.e., in general the survey participants did not let their personal distaste for a standard adversely effect their judgement of its effectiveness.

- The results for the Routine Size (Modularity) programming standard are particularly interesting in that they compare quite well with those of Hoskyns in "Evaluation of Programming and Systems Techniques; Implications of Using Modular Programming." [16] In particular, Hoskyns reports that 89% of those surveyed experienced "easier maintenance" from modular programming. By comparison, the software characteristics most closely related to maintenance (i.e., Code Auditability, Code Understandability/Readability, and Code Maintainability/Usability) all received Strong/Positive ratings. Furthermore, Hoskyns' results on "better program design" and "easier program testing" (85% and 78%, respectively) compare well with ours for Logical Organization and Testability, both of which received Strong/Positive ratings. Notably, fewer of the modular programming users (64%) experienced the benefit of "more reliable programs", again comparing well with our results showing Medium/Positive ratings for both Operational Reliability and Coding Error Frequency. On the negative side, Hoskyns reports that 28% of the users claimed "more computer time required during development" and 27% experienced "less efficient final programs", corresponding to our Strong/Negative and Medium/Negative results for Execution Time and Core Requirements. The only obvious inconsistencies between Hoskyns' results and those reported here are for Programmer Productivity and Schedule. Hoskyns' results for "higher programmer productivity" (64%) and "more likely to meet target dates" (63%) are based, however, on a somewhat relaxed interpretation of modularity (i.e., 20 to 2000 statements), while our Strong/Inconclusive ratings relate to a formal standard restricting module size to a maximum of 100 executable FORTRAN statements.

To complete analysis of the survey responses, the free form responses were carefully examined. From review of all 54 responses, a list of general response categories was prepared and frequencies were derived for the number of times each response category appeared in the total set of responses. Most of the free form responses contained two or more of the response categories (some as many as four). Thus, the sum of the category-appearance-frequencies is much greater than 54, the number of free form responses analyzed.

Frequency of Response Category Appearance	Response Category
17	Code easier to understand, maintain, test, or document, or better documentation
11	Consistent code format
11	Reduced bugs in code, more reliable, higher quality
10	Guided programmers, guidelines for development
8	Standards introduced late, changing standards caused negative impact
7	Constrained programmers' creativity
7	Immediate cost high, schedule impact negative
5	Programmer gripes, disgruntlement
5	Recognition of overall cost decrease
5	Generally positive (unspecific)
4	Disciplined programmers
4	Arbitrary or inconsistent standards
4	Not applicable, no opinion, or wait and see
3	Quality of standards insufficient, could be improved
2	Interfaces better, easier
2	Job done faster and better
1	Generally negative (unspecific)

It can be seen from this view of the free form responses that there were almost twice as many positive comments (67) as there were comments of a negative nature (35). Finally, note that the highest frequency categories represent responses generally consistent with those of the Impact Evaluation, i.e., code and documentation understandability and maintainability, coding error frequency, and consistency are scored generally "positive". However, the categories dealing with the effect on programmers appear to indicate a stronger impact than does the Impact Evaluation. In the latter, personnel motivation and programmer productivity did not seem to be big issues, except for the effect of the structured coding requirement. This particular standard clearly seemed to be the most controversial and motivated many of the free responses.

3.3.2 MPP Impact

The second impact evaluation survey focused on the 11 previously identified MPP and sought an assessment of the extent to which the practices did (in the case of STP) and could (in a hypothetical software development) contribute to elimination of typical problems that have plagued and adversely affected software development activities. The survey questionnaire was completed by 67 people from a cross section of STP personnel. Of the 67 survey participants, 31 held management positions with varying levels of project responsibility.

The MPP Impact Evaluation survey questionnaire was designed as a booklet containing five parts:

1) A series of individual impact rating matrices, one for each of the 11 MPP. Each page contained a brief definition of the practice and a blank 12 x 5 evaluation matrix. Each row of the matrix corresponded to one of the 12 typical development problems, while the columns offered a choice of five influence ratings. Survey participants were required to enter an "A" in the appropriate column of each row to indicate the nature and extent of the influence of the practice on each problem in the actual STP context, and to enter a "T" to indicate the theoretical, optimum impact of the practice under ideal conditions. Figure 3-7 illustrates the impact rating matrix used to evaluate each of the MPP.

2) A list of the 11 practices and a request that they be ranked according to relative importance with respect to their positive contribution to elimination of software development problems.

3) A true/false question posed as a restatement of the general MPP hypothesis.

4) A list of the 12 typical software development problems and a request that they be ranked according to relative significance with respect to their negative effect on quality, schedule and cost of software production.

   (The questionnaire format for items 2,3 and 4 is illustrated in Figure 3-8).

MPP RATING / PROBLEM	HIGH POSITIVE	LOW POSITIVE	NO EFFECT	LOW NEGATIVE	HIGH NEGATIVE
COST OVERRUN					
DEVELOPMENT STATUS INVISIBILITY					
UNRELIABILITY					
UNMAINTAINABILITY					
INADEQUATE SATISFACTION OF REAL REQUIREMENTS					
INEFFICIENT USE OF RESOURCES					
SCHEDULE OVERRUN					
INADEQUATE PLANNING AND CONTROL					
PROJECT MISMANAGEMENT					
LACK OF PROGRAMMING DISCIPLINE					
LACK OF CONCLUSIVE TESTING					
POOR DOCUMENTATION					

Figure 3-7. MPP Survey Questionnaire Sample Rating Sheet

MPP Ranking

MPP	RANK*
Requirements Analysis and Validation	
Process Design	
Incremental Development	
Complete Preliminary Design	
Unit Development Folders	
Enforced Programming Standards	
Independent Testing	
Software Configuration Management	
Baselining of Requirements Specification	
Formal Inspection of Documentation and Code	
Software Development Tools	

General MPP Hypothesis: Rules governing software development, evaluation, and documentation, if rigorously defined and applied, and supported by modern programming practices (techniques and tools), make possible the production:

	True	False	?
• of higher than usual quality software	☐	☐	☐
• at lower than usual life cycle cost	☐	☐	☐

Software Development Problem Ranking

PROBLEM	RANK**
Cost Overrun	
Development Status Invisibility	
Unreliability	
Unmaintainability	
Inadequate Satisfaction of Real Requirements	
Inefficient Use of Resources	
Schedule Overrun	
Inadequate Planning and Control	
Project Mismanagement	
Lack of Programming Discipline	
Lack of Conclusive Testing	
Poor Documentation	

* Relative importance ranking : Use rank of 1 for "most important," 2 for "next most important,"... 11 for"least important."

** Relative significance ranking: Use rank of 1 for "most significant", 2 for "next most significant,"... 12 for"least significant."

Figure 3-8. MPP Survey Questionnaire Ranking Sheet

5) A request for a brief free-form statement as to the need for and benefits derived from MPP as applied to STP and in general.

Analysis of the MPP Impact Evaluation survey response data was accomplished with the same mathematical algorithms and support tools used in the programming standards study. The IMPACT program (Appendix D) was used to compute influence rating averages and frequencies and to apply the hypothesis testing procedure to the response-frequency-distibution for each practice-problem pair. The hypothesis testing was applied independently to both the actual (A) and theoretical (T) responses, yielding two composite assertion strength/influence rating indicators for each practice-problem pair as illustrated in the MPP Impact Evaluation survey summary matrix (Figure 3-9). IMPACT also computed an average response-frequency-distribution for each of the 11 rows and 12 columns and for the complete matrix, thus providing an indication of the cumulative (average) impact of 1) each practice on the combined set of problems, 2) the combined set of practices on each problem, and 3) all the practices on all the problems. The legend in the upper left corner of the figure indicates the appropriate interpretation of the matrix contents. For example at the intersection of row 9 and column 5 there is an SP above the diagonal and an SSP below from which we obtain:

> "We can assert with strong confidence that, in the actual case of the System Technology Program, the practice (Baselining of Requirements Specification) made a positive contribution toward eliminating the problem (*Inadequate Satisfaction of Real Requirements*)."

> "We can assert with strong confidence that, in the theoretical case, the practice (Baselining of Requirements Specification) can and should make a strong positive contribution toward eliminating the problem (Inadequate Satisfaction of Real Requirements)."

The Survey Summary on the right side of the figure indicates the number of times each assertion strength/influence rating appears in the matrix, excluding the 12^{th} row and 13^{th} column. For example, the rating SP (i.e., strong confidence/positive impact) was obtained for 24 practice-problem combinations in the actual STP case and for 54 combinations in the theoretical case.

The figure also illustrates the composite survey response to the general MPP hypothesis true/false question, indicating that 85% of the survey participants agreed that MPP contribute to the production of "higher than usual quality software." On the issue of "lower than usual life cycle cost", agreement as to a positive MPP contribution is not nearly as marked, however, the true responses outnumber the false responses by almost 4 to 1.

In the formulation of the average impact of each practice on all problems (i.e., the assertion strength/influence ratings in column 13 of the matrix), equal significance of all problems was assumed. From the cumulative response-frequency-distributions for all of the practices, it was thus possible to rank the practices in order of their relative impact on the set of uniformly weighted problems. For the purposes of verifying the survey results and further analyzing and estimating the relative value of the practices in differing development activities, several other rankings

MODERN PROGRAMMING PRACTICES IMPACT EVALUATION SURVEY

ASSERTION STRENGTH	INFLUENCE RATING					
	STRONG POSITIVE	POSITIVE	INDIFFERENT	NEGATIVE	STRONG NEGATIVE	INCONCLUSIVE
STRONG	SSP	SP	IND	SN	SSN	SI
MEDIUM		MP		MN		MI
WEAK		WP		WM		WI

SOFTWARE PROBLEMS

PROGRAMMING PRACTICES	1 COST OVERRUN	2 DEVELOPMENT STATUS INVISIBILITY	3 UNRELIABILITY	4 UNMAINTAINABILITY	5 INADEQUATE SATISFACTION OF REAL REQUIREMENTS	6 INEFFICIENT USE OF RESOURCES	7 SCHEDULE OVERRUN	8 INADEQUATE PLANNING AND CONTROL	9 PROJECT MISMANAGEMENT	10 LACK OF PROGRAMMING DISCIPLINE	11 LACK OF CONCLUSIVE TESTING	12 POOR DOCUMENTATION	CUMULATIVE EFFECT OF A PRACTICE ON ALL PROBLEMS
1 REQUIREMENTS ANALYSIS AND VALIDATION	SP / SSP	MP / SP	MP / SP	MP / MP	SP / SSP	MI / SP	MI / SP	MP / SP	WP / MP	WP / WP	SP / SP	MP / SP	MP / SP
2 PROCESS DESIGN	SP / SP	MP / SP	MP / SP	WP / MP	SP / SP	MP / SP	MP / SP	MP / SP	WP / MP	WP / WP	WP / MP	WP / MP	MP / SP
3 INCREMENTAL DEVELOPMENT	SI / SP	SP / SSP	SP / SP	MI / MP	MP / MP	SI / MP	SI / SP	SP / SP	MP / SP	MI / MP	SP / SP	MI / MP	SP / SP
4 COMPLETE PRELIMINARY DESIGN	SP / SSP	SP / SP	MP / SP	MP / MP	SP / SP	MP / SP	MP / SP	MP / SP	MP / SP	WP / MP	WP / MP	MP / SP	MP / SP
5 UNIT DEVELOPMENT FOLDERS	SI / SP	SP / SSP	MP / MP	SP / SP	MP / MP	MI / MP	MI / SP	MP / SSP	MP / MP	SP / SP	MP / SP	SP / SSP	MP / SP
6 ENFORCED PROGRAMMING STANDARDS	SN / MI	IND / WP	MP / SP	SP / SSP	IND / IND	SN / MI	MN / MI	WP / WP	WI / WP	SP / SSP	IND / WP	MP / SP	MI / MP
7 INDEPENDENT TESTING	SI / MI	WP / MP	SP / SSP	WP / WP	SP / SSP	MI / MP	MI / MP	WI / WP	WI / MP	WP / WP	SP / SSP	WP / WP	MI / MP
8 SOFTWARE CONFIGURATION MANAGEMENT	MI / MP	MP / MP	MP / SP	MP / SP	WP / WP	MI / MP	MI / MP	MP / SP	MP / SP	WP / MP	WP / MP	WP / SP	MI / MP
9 BASELINING OF REQUIREMENTS SPECIFICATION	SI / SSP	MP / MP	MP / SP	WI / MP	SP / SSP	MI / SSP	SI / SSP	MI / SP	MI / SP	IND / IND	MP / MP	MP / MP	MI / MP
10 FORMAL INSPECTION OF DOCUMENTATION AND CODE	MI / SP	MP / MP	MP / SP	WP / MP	MP / SP	MI / SI	MI / MP	WP / MP	WP / MP	MP / SP	WP / MP	MP / SP	MP / MP
11 SOFTWARE DEVELOPMENT TOOLS	SI / SP	WP / MP	SP / SP	MP / SP	MP / MP	SI / SP	MI / MP	WP / MP	WP / MP	MP / SP	SP / SP	MP / MP	MP / MP
CUMULATIVE EFFECT OF ALL PRACTICES ON A PROBLEM	SI / SP	MP / MP	MP / SP	MP / MP	MP / SP	MI / MP	MI / MP	MP / MP	MP / MP	MP / MP	MP / MP	MP / MP	MP / MP

A = ACTUAL (STP)

T = THEORETICAL

SURVEY SUMMARY	
RATING	FREQ. (A / T)
SSP	0 / 18
IND	4 / 2
SSN	0 / 0
SP	74 / 54
SI	9 / 1
SN	2 / 0
MP	45 / 44
MI	20 / 4
MN	1 / 0
WP	23 / 11
WI	4 / 0
WN	0 / 0

GENERAL MPP HYPOTHESIS: RULES GOVERNING SOFTWARE DEVELOPMENT, EVALUATION, AND DOCUMENTATION, IF RIGOROUSLY DEFINED AND APPLIED, AND SUPPORTED BY MODERN PROGRAMMING PRACTICES (TECHNIQUES AND TOOLS), MAKE POSSIBLE THE PRODUCTION

	TRUE	FALSE	?
• OF HIGHER THAN USUAL QUALITY SOFTWARE	85%	4%	11%
• AT LOWER THAN USUAL LIFE CYCLE COST	49%	15%	36%

Figure 3-9. MPP Impact Evaluation Survey Results

were obtained. The RANK program (Appendix D) was used to average the relative-importance-practice rankings provided by the survey participants and produce a composite rank ordering of the 11 MPP with a mean rank and standard deviation for each. Similarly, RANK produced a composite relative-significance-problem ranking (with means and standard deviations) using as inputs the individual rankings from the 67 survey questionnaires. The mean ranks for the problems were used to derive the problem weights shown in Figure 3-10. Using these weights to appropriately factor each practice-problem response-frequency-distribution, the MERIT program (Appendix D) computed an average distribution for each practice on the set of weighted problems. MERIT also applied the statistical hypothesis test to the response-frequency-distributions for the theoretical case, producing the assertion strength/influence ratings and the relative impact ranking for the practices in which the varying significance of software production problems is reflected. The results of the three separate rankings of the practices are shown in Figure 3-11. Notice that, as a result of weighting the problems, the assertion strength/influence rating for Baselining of Requirements Specification changed from Medium/Positive to Strong/Positive and the relative impact rank changed from 4 to 2. That is, in a software development activity for which the 12 typical problems are of equal (or very nearly equal) significance, it can asserted with medium confidence that:

> "The practice, Baselining of Requirements Specification, makes a positive contribution toward elimination of the problems, and only 3 other practices have more positive impact."

On the other hand, the Strong/Positive rating and the rank of 2 imply that, in a software development activity for which the problems have varying significance (and are assigned the weights of Figure 3-10), it can be asserted with strong confidence that:

> "The practice, Baselining of Requirements Specification, makes a positive contribution toward elimination of the problems, and only 1 other practice has more positive impact."

Analysis of the MPP impact evaluation survey summary results matrix, together with the above three practice rankings, prompts the following general observations:

- There is strong agreement among STP personnel as to the four MPP of greatest importance and impact and strong agreement on their relative ranking:
  - Requirements Analysis and Validation
  - Baselining of Requirements Specification
  - Complete Preliminary Design
  - Process Design
- There is strong agreement on the importance and positive impact of the next three most highly ranked MPP, but the relative ranking among them is less clear:
  - Incremental Development
  - Unit Development Folders

PROBLEM	INDIVIDUAL PROBLEM RANKING	ASSIGNED PROBLEM WEIGHTS
COST OVERRUN	1	69
DEVELOPMENT STATUS INVISIBILITY	9	43
UNRELIABILITY	6	52
UNMAINTAINABILITY	10	42
INADEQUATE SATISFACTION OF REAL REQUIREMENTS	2	68
INEFFICIENT USE OF RESOURCES	7	48
SCHEDULE OVERRUN	3	58
INADEQUATE PLANNING AND CONTROL	5	53
PROJECT MISMANAGEMENT	4	55
LACK OF PROGRAMMING DISCIPLINE	12	22
LACK OF CONCLUSIVE TESTING	8	47
POOR DOCUMENTATION	11	35

Figure 3-10. Assignment of Problem Weights

PRACTICE	PROBLEMS UNWEIGHTED		PROBLEMS WEIGHTED		INDIVIDUAL PRACTICE RANKING
	ASSERTION STRENGTH/ INFLUENCE RATING	PRACTICE RANKING	ASSERTION STRENGTH/ INFLUENCE RATING	PRACTICE RANKING	
REQUIREMENTS ANALYSIS AND VALIDATION	STRONG/POSITIVE	1	STRONG/POSITIVE	1	1
PROCESS DESIGN	STRONG/POSITIVE	4	STRONG/POSITIVE	4	4
INCREMENTAL DEVELOPMENT	STRONG/POSITIVE	5	STRONG/POSITIVE	6	5
COMPLETE PRELIMINARY DESIGN	STRONG/POSITIVE	2	STRONG/POSITIVE	3	3
UNIT DEVELOPMENT FOLDERS	STRONG/POSITIVE	3	STRONG/POSITIVE	5	6
ENFORCED PROGRAMMING STANDARDS	MEDIUM/POSITIVE	11	MEDIUM/POSITIVE	11	9
INDEPENDENT TESTING	MEDIUM/POSITIVE	9	MEDIUM/POSITIVE	10	8
SOFTWARE CONFIGURATION MANAGEMENT	MEDIUM/POSITIVE	10	MEDIUM/POSITIVE	9	10
BASELINING OF REQUIREMENTS SPECIFICATION	MEDIUM/POSITIVE	6	STRONG/POSITIVE	2	2
FORMAL INSPECTION OF DOCUMENTATION AND CODE	MEDIUM/POSITIVE	8	MEDIUM/POSITIVE	8	11
SOFTWARE DEVELOPMENT TOOLS	MEDIUM/POSITIVE	7	MEDIUM/POSITIVE	7	7

Figure 3-11. Comparison of Practice Rankings

- Software Development Tools

- There is strong agreement on the importance and positive impact of the four lower ranked MPP, but the relative ranking among them is not at all clear:
  - Independent Testing
  - Enforced Programming Standards
  - Software Configuration Management
  - Formal Inspection of Documentation and Code

Detailed analysis of the 67 free form MPP survey responses addressing the need for and benefit of MPP on STP and in general resulted in identification of 12 response categories. Many of the free form responses fell into more than one category. The following describes the response categories and identifies the number of comments in the 67 free form responses which fell into each category.

Frequency of Response Category Appearance	Response Category
22	Generally benefical. Benefit in proportion to size of program. Good management practices and good source of management information. Using MPP changed undisciplined process to produce reliable, consistent software. Should result in lower maintenance costs.
15	Neutral or no comment.
11	MPP should be tailored for the application. Modes of application should be flexible. A subset of MPP could be cost-effectively used in unique applications.
11	MPP can be costly to establish and enforce. Can increase work and decrease productivity. Some MPP emphasized low leverage activities. Some standards inferior to normal practices.
10	Other factors more important than MPP, e.g., work environment, stability of software requirements, professional experience and management expertise.
8	Disciplined use of MPP not strongly enforced. Lack of committment to achievement of long range MPP benefits. MPP application neither extensive, nor consistent nor early enough causing some increased cost due to rework. Independent testing not very effective.

5	Need for better documentation and training on the proper application of MPP to achieve more uniformity, productivity and discipline.
4	Requirements changes are most important software development problem that MPP alone can't solve.
4	Unit Development Folders essential for required management visibility of software development status. Process Construction Program and Code Auditor necessary to ensure code compliance with standards.
2	Quality Assurance personnel not sufficiently qualified (technically) for good communication with developers. Configuration Management system too slow in responding to change; required fast-response informal change control system.
1	Review of Unit Development Folders (UDF) requires extra time and effort; UDF control requires a large bureaucracy.
1	STP afforded excellent opportunity to experiment (define, apply, evaluate, improve) on a variety of modern practices, thus very beneficial for future programs.

From the above it can be seen that 38 of the 94 comments (about 40%) were positive in nature, while 27 (about 29%) were negative and 29 (about 31%) were more or less in the middle. More important than the percentages, however, the overall message in the comments appears to confirm some of our earlier findings, and prompt new ones namely:

- The proper handling (i.e., baselining and controlling changes) of software requirements is of critical importance with respect to software quality, cost and schedule, and MPP which encourage and support proper handling (such as Baselining of Requirements Specification and Requirements Analysis and Validation) are most needed and have most beneficial impact.
- Although there is substantial agreement on the overall beneficial impact of the 11 STP MPP, there is clearly room for improvement, especially regarding the four least highly ranked MPP (Independent Testing, Enforced Programming Standards, Software Configuration Management, and Formal Inspection of Documentation and Code). This is consistent with the MPP Impact Evaluation summary which generally indicates higher positive MPP impact (in the theoretical case) than that reported in the actual STP software development to date.
- Most of the positive attitude toward and general acceptance of MPP is expressed in terms of positive impact on software quality (particularly reliability and maintainability), while it is apparently felt that MPP contribute less directly to software development cost and schedule. Furthermore, if MPP are introduced

late in the development process, or if they are poorly defined or poorly understood and inconsistently applied, they can adversely effect rework, productivity, and personnel motivation with possible negative impact on cost and schedule.

3.3.3 Correlation of Study Results

The preceding sections reported upon two independent but similar impact evaluation activities. The first provided an assessment of the impact of detailed coding and documentation standards on desirable characteristics of end item software and the software development process. The second provided an assessment of the impact of higher level modern programming practices on avoidance or elimination of typical problems that plague software development. The two activities were similar in that they sought quantitative data to permit completion (i.e., filling in the blank) of statements of the general form:

> "It can be asserted with_____ confidence that practice X has_____ impact on software characteristic (or problem) Y."

For example, from the Programming Standards Impact Evaluation we obtained an assertion strength/influence rating of Medium/Positive as the perceived average impact of programming standards on the software characteristic "Code Maintainability/Usability". From the MPP Impact Evaluation we obtained two impact ratings for the practice "Enforced Programming Standards" on the typical software production problem "Unmaintainability". These were Strong/Positive (based on actual STP experience) and Strong/Strong Positive (estimating the theoretical impact under optimum circumstances).

There were enough similarities between the surveys to permit a direct comparison and simple illustration of the highly correlated results. That is, it is possible to compare the average impact of programming standards on a software characteristic with both the actual and theoretical impact of Enforced Programming Standards on a problem strongly related to the characteristic. In the illustration of the comparison (Figure 3-12.), the letter "C" indicates the composite (i.e., average) assertion strength/influence rating from row 19 of the Programming Standards Impact Evaluation matrix. The letters "A" and "T" indicate the ratings from corresponding column elements in row 6 of the MPP Impact Evaluation matrix. The y-axis contains an ordering of the possible impact ratings from very positive to very negative with all Inconclusive ratings combined as a single point on the scale, and the x-axis contains seven areas of commonality between the characteristics and problems in the two surveys.

The apparent correlation of the survey results is very good, especially in view of several important differences in the surveys;

- The "C" ratings represent the average influence of all 18 standards on each characteristic, while the "A" and "T" ratings are for the total influence of Enforced Programming Standards on each problem.
- The first survey was primarily concerned with initial development cost and schedule, while the MPP survey called explicitly for impact assessments with respect to life cycle cost and schedule concerns.

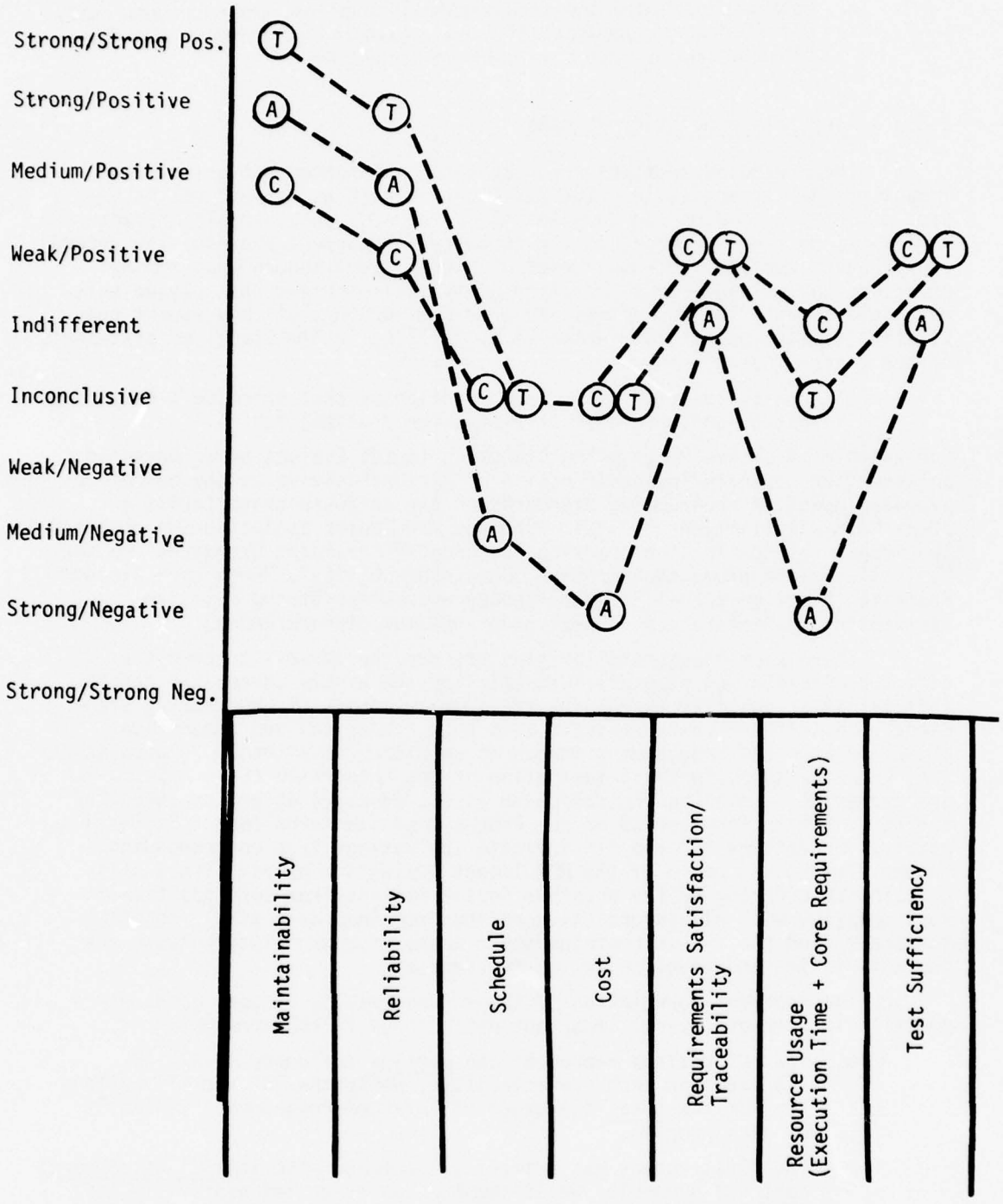


Figure 3-12. Correlation of Survey Results for "Enforced Programming Standards"

4.0 MPP COMPARISON METHODOLOGY

4.1 Objective and Approach

One of the very important aims of The Impact of MPP on System Development study was to develop a framework within which various practices and overall MPP implementations could be compared. To begin to solve the problem of selecting appropriate practices for a particular application, it was necessary to provide an MPP comparison methodology that would permit:

- detailed comparative analysis of two or more individual practices, and
- overall comparison of the combined effectiveness of alternative MPP implementations.

It was expected that such a methodology would represent significant progress toward more scientific and systematic evaluation and selection of specific practices which address the range of problems that have historically plagued software production. Unfortunately, in the past, very little attention has been given to determining the actual nature of the effect of individual practices on distinct software production problems. What has been missing is an appraisal of the strengths and weaknesses of practices and the use of the appraisal to guide selection of appropriate practices for application to software projects of varying type, size and complexity.

Clearly, all modern programming practices are not equally effective in all respects. It is recognized that different people have different strengths and weaknesses, and the same is true for modern programming practices. In fact, this phenomenon is especially evident for practices since they are usually invented for a special reason (e.g., to enhance some particular software quality or to solve some particular production problem) without regard for other possible benefits and side effects.

The approach to development of both the MPP impact evaluation methodology and the MPP comparison methodology was strongly influenced by the above consideration. The approach was to identify software production areas, within which comparison of practices could be accomplished in terms of individual strengths and weaknesses. This involved the development of a model that would serve to both definitize and normalize the impact of practices on software development. It also involved the definition of a procedure for using the model to compare MPP implementations to aid in the selection of appropriate practices for different software projects.

4.2 The Comparison Model and Procedure

The basic element of the comparison model is a generalized version of the impact evaluation matrix. Specific examples of the matrix are given elsewhere in this report in presenting the results of the Programming Standards and MPP Impact Evaluation activities. In both cases, each row of the matrix corresponded to a practice, and each column corresponded to a characteristic of software production(i.e., a quality of developed software, an attribute of the software development process, or a typical problem plaguing software development). The generalized version of the matrix for

comparative analysis of MPP implementations can be viewed as a combination of the impact evaluation matrices for two sets of MPP as illustrated in Figure 4-1. Each element of the matrix contains five numbers (i.e., a distribution of frequencies) corresponding to the influence categories:

- High positive
- Low positive
- No contribution
- Low negative
- High negative

The procedure for performing a comparative analysis of the two MPP implementations involves computation of an overall figure of merit for each MPP set. In order to permit comparative evaluation of selected MPP in the context of differing types of software development activities, the procedure also involves an assignment of weights to each column (characteristic) and each row (practice). Each column weight is a number between 1 and 99 which reflects the relative importance of the characteristic in a particular software development context. Each row weight is either 1 or 0 to indicate whether or not the influence of the corresponding practice is to be used in computing an overall figure of merit for the MPP set.

Given a set of column weights (CW_i for i=1,....,N) a gross comparison of the two MPP implementations can be obtained from two applications of the hypothesis test performed by the MERIT support program. A figure of merit for the first MPP implementation (FOM_1) is obtained by setting the first M_1 row weights equal to 1 and the remaining row weights equal to 0. Similarly, FOM_2 is obtained by setting RW_i=0 for i=1,....,M_1 and the remaining row weights equal to 1. In addition to the overall FOM, MERIT produces a composite FOM for each row (i.e., the summary influence of each practice on all characteristics) and for each column (i.e., the summary influence of all practices on each characteristic). Implicit in the calculation of the figures of merit is the basic assumption that the cumulative impact of several practices is strictly additive (i.e., not mutually synergistic or overlapping). A comparative analysis of the two MPP implementations is accomplished through review of the two sets of MERIT output and direct comparison of the overall figures of merit as well as those for each column. In addition, comparison of figures of merit for the rows provides an indication of the relative influence of alternative practices on all characteristics.

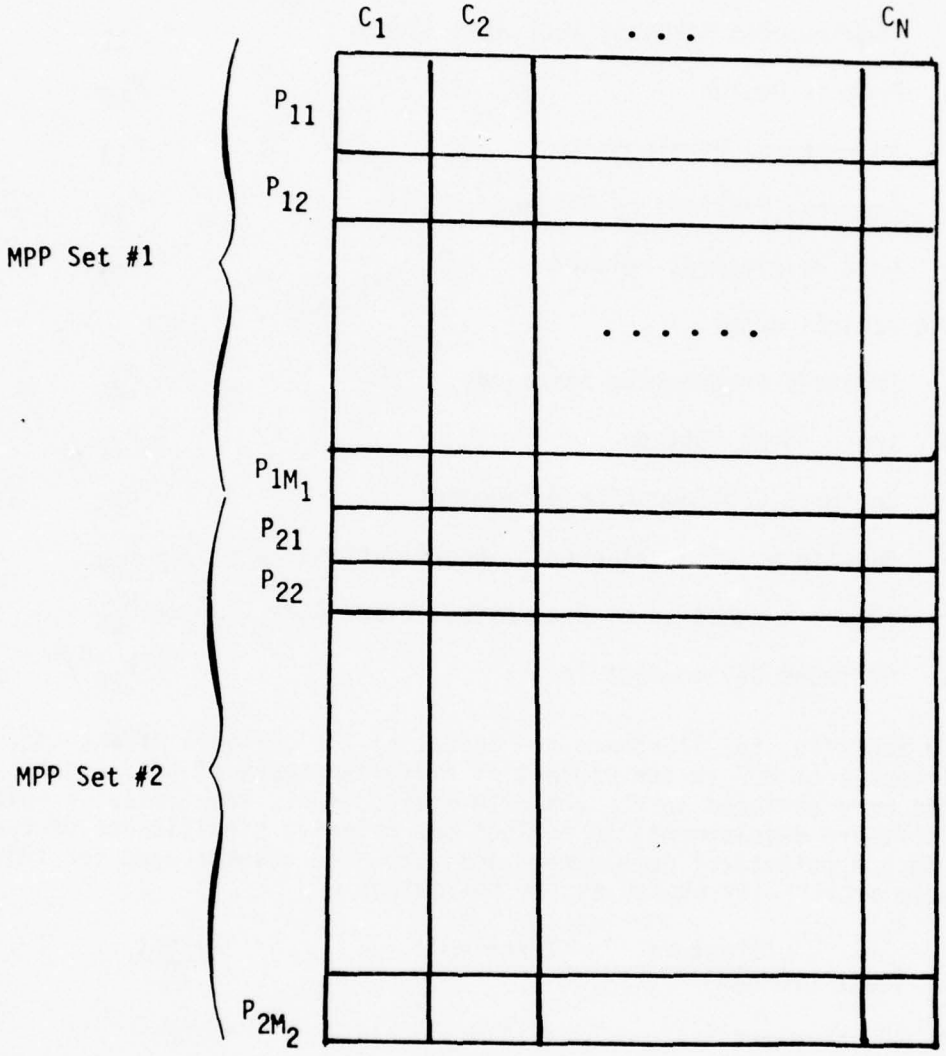


Figure 4-1. : MPP Implementation Comparison Matrix

4.3 An Example of the Comparison Methodology

To illustrate the comparison methodology, the set of 11 STP modern programming practices was arbitrarily divided into two MPP implementations with MPP Set #1 including:

- Requirements Analysis and Validation - P_{11}
- Process Design - P_{12}
- Incremental Development - P_{13}
- Complete Preliminary Design - P_{14}
- Unit Development Folders - P_{15}

and MPP Set #2 including:

- Enforced Programming Standards - P_{21}
- Independent Testing - P_{22}
- Software Configuration Management - P_{23}
- Baselining of Requirements Specifications - P_{24}
- Formal Inspection of Documentation and Code - P_{25}
- Software Development Tools - P_{26}

In addition, to illustrate the potential for analysis of the potential impact of MPP in the context of differing types of projects, new weights were assigned to the characteristics (i.e., the typical problems plaguing software development) to reflect the relative significance of the problems in a hypothetical development activity. The weights used for this example were arbitrarily chosen as the following:

Characteristic (Problem)	Weight
C_1 (Cost Overrun)	99
C_2 (Development Status Invisibility)	50
C_3 (Unreliability)	75
C_4 (Unmaintainability)	25
C_5 (Inadequate Satisfaction of Real Requirements)	50
C_6 (Inefficient Use of Resources)	75
C_7 (Schedule Overrun)	75

C_8 (Inadequate Planning and Control)	1
C_9 (Project Mismanagement)	1
C_{10} (Lack of Programming Discipline)	1
C_{11} (Lack of Conclusive Testing)	25
C_{12} (Poor Documentation)	25

The MERIT program was used to apply the appropriate row and column weights to the individual influence distributions, compute composite distributions for each row and column and for the entire matrix, and perform the hypothesis test distributional algorithm in deriving for each set of MPP:

- an FOM (assertion strength/influence rating) for each practice on the weighted problems,
- an FOM for all practices included in the particular MPP set on each problem, and
- an overall FOM for the combination of all pratices included in the particular MPP set on the combined set of weighted problems.

The key analysis results are shown below, and complete MERIT output is in Appendix E.

Some striking differences are seen in comparison of the asserted impact of the combined practices of Set 1 versus the combined practices of Set 2 and in comparison of the overall figures of merit for the two MPP implementations. The summary figures of merit (i.e., assertion strength/influence rating) obtained from the hypothesis test performed by MERIT were as follows:

Characteristic (Problem)	MPP Set #1	MPP Set #2
C_1 (Cost Overrun)	Strong/Positive	Strong/Positive
C_2 (Development Status Invisibility)	Strong/Strong Positive	Medium/positive
C_3 (Unreliability)	Strong/Positive	Strong/Positive
C_4 (Unmaintainability)	Medium/Positive	Medium/Positive
C_5 (Inadequate Satisfaction of Real Requirements)	Strong/Positive	Medium/Positive
C_6 (Inefficient Use of Resources)	Strong/Positive	Medium/Positive
C_7 (Schedule Overrun)	Strong/Positive	Medium/Positive

C_8 (Inadequate Planning and Control)	Strong/ Positive	Medium/Positive
C_9 (Project Mismanagement)	Medium/Positive	Medium/Positive
C_{10} (Lack of Programming Discipline)	Medium/Positive	Medium/Positive
C_{11} (Lack of Conclusive Testing)	Strong/Positive	Medium/Positive
C_{12} (Poor Documentation)	Strong/Positive	Medium/Positive

The most noticeable difference is between the assertion strength/influence rating for the MPP Sets on Characteristic 2 (Development Status Invisibility) which prompt the following:

- We have strong confidence that the 5 practices of MPP Set 1, in combination, make a strong positive contribution toward eliminating the problem of Development Status Invisibility.

- We have medium confidence that the 6 practices of MPP Set 2, in combination, make a positive contribution toward eliminating the problem of Development Status Invisibility.

The result of weighting the individual influence distributions in computing summary FOMs for each practice and a composite FOM for the two MPP sets is shown below:

Practice	Weighted Figure of Merit
P_{11} - Requirements Analysis and Validation	Strong/Positive
P_{12} - Process Design	Strong/Positive
P_{13} - Incremental Development	Strong/Positive
P_{14} - Complete Preliminary Design	Strong/Positive
P_{15} - Unit Development Folders	Strong/Positive
Composite (MPP Set 1)	Strong/Positive
P_{21} - Enforced Programming Standards	Medium/Positive
P_{22} - Independent Testing	Medium/Positive
P_{23} - Software Configuration Management	Medium/Positive
P_{24} - Baselining of Requirements Specifications	Strong/Strong Pos.

Practice	Weighted Figure of Merit
P_{25} - Formal Inspection of Documentation & Code	Medium/Positive
P_{26} - Software Development Tools	Strong Positive
Composite (MPP Set 2)	Medium/Positive

The composite figures of merit for the two MPP implementations are indeed different, and, in the context of a hypothetical software development activity (for which the weights are relevant):

- We have strong confidence that the 5 practices of MPP Set 1, in combination, make a positive contribution toward eliminating the problems.
- We have medium confidence that the 6 practices of MPP Set 2, in combination, make a positive contribution toward eliminating the problems.

4.4 Limitations of the Comparison Methodology

It was indicated earlier that the MPP comparison model could be used for the selection of a set of appropriate practices for a proposed software development activity. This involves computation of an overall FOM for candidate combinations of MPP and selection of the combination with the best FOM. It is important to note, however, that the validity of the FOM computation is dependent upon the make up of the individual matrix elements (i.e., the influence rating frequency distributions) which are summed to form an overall distribution. In particular, the sources of the impact evaluation ratings may be different and, unless the frequency distributions are normalized, the influence of one practice may unduly dominate the composite FOM. This problem can be avoided by appropriately factoring the individual frequency distributions as necessary to cause the sum of the frequencies in each matrix element to be a constant. For example, the three unnormalized distributions below can be normalized as shown:

	Unnormalized Frequency Distributions			Composite	%
High Positive	14	4	1	19	15.2
Low Positive	26	8	4	38	30.4
No Contribution	22	20	14	56	44.8
Low Negative	7	3	1	11	8.8
High Negative	1	0	0	1	.8
Total	70	35	20	125	

	Normalized Frequency Distributions			Composite	
High Positive	14	8	3.5	25.5	12.1
Low Positive	26	16	14	56	26.7
No Contribution	22	40	49	111	52.9
Low Negative	7	6	3.5	16.5	7.9
High Negative	1	0	0	1	.4
Total	70	70	70	210.0	

The importance of normalizing the contents of the impact evaluation matrix is more apparent from comparison of the differing percentages for the unnormalized and normalized composite frequency distributions in the above example.

An alternative and easier way to avoid adding "apples and oranges" is to use the percentage figures computed for each frequency distribution in formulating the composite FOM, divide by the number of matrix elements in the sum (i.e., compute average percentages), and apply the hypothesis test to the percentages rather than to the summed frequencies. The MERIT program does not currently provide this capability but could be easily modified to do so.

A secondary limitation on the utility of the MPP comparison methodology arises from the strict dependence on the completed impact evaluation matrix. There are, of course, alternative methods for acquiring impact evaluation data including the modified Delphi/survey/selective validation approach used in this study, experimentation and data collection on parallel development activities, and detailed analysis of a variety of software developments using various MPP combinations. All of these methods are aimed at isolating and measuring the effect of individual practices on software production, and all are difficult and time consuming to apply.

Finally it should be noted that the two sets of characteristics used in this study (i.e., the desirable qualities of software and the development process and the typical problems plaguing software development) are certainly not exhaustive. Although the 12 typical problems served well as metrics for evaluating the impact of selected STP modern programming practices, it is possible that the greatest strength (or weakness) of some practice cannot be readily represented without expanding the list of characteristics. Thus, if many comparisons of MPP implementations are anticipated, it is necessary to first establish a comprehensive set of common characteristics upon which the detailed assessment of a wide variety of practices can be based as necessary to permit a fair comparison of relative impact.

5.0 CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

Of the more than 30 candidate modern programming practices considered during this study, the following four were determined to have the strongest positive impact on software quality and productivity:

1. Requirements Analysis and Validation
2. Baselining of Requirements Specification
3. Complete Preliminary Design
4. Process Design

This outcome is consistent with data available from other projects [5,17] indicating that increased rigor and engineering discipline in the requirements definition and design phases have the highest leverage on improving software production.

The following three MPP were determined to have the next highest impact:

5. Incremental Development
6. Unit Development Folders
7. Software Development Tools

The following four MPP completed the list of the highest-impact practices:

8. Independent Testing
9. Enforced Programming Standards
10. Software Configuration Management
11. Formal Inspection of Documentation and Code

All eleven of these MPP were determined to have a generally positive impact on attaining desirable qualities of software and achieving more productive developer effort through elimination of typical software production problems. A special survey and analysis of the impact of 18 detailed programming standards on 30 specific software qualities showed a generally positive impact on software quality and a mixed impact on software development productivity. Noting, however, that improved software quality strongly correlates with long range benefits such as easier software maintenance, the net result is an overall positive impact of the standards on software life-cycle productivity.

The detailed results provide hundreds of impact assessments for individual MPP that are difficult to abstract here, but which provide detailed support to decision makers concerned with selection, definition, and implementation of appropriate standards and procedures for individual projects. Clearly, the assessment of a large number of individual interactions in the analysis of candidate combinations of MPP can be impractical to handle by manual methods. This problem was addressed in the study by developing several easy-to-use computer programs to support cumulative MPP impact assessments and comparative evaluation of MPP implementations.

5.2 Discussion of Conclusions

It should not be surprising that the overall evaluation of the STP MPP turned out to be positive, mainly because STP was never intended to be a trial experiment with a set of practices having unknown effect. Indeed, a major emphasis of the project was to evolve an improving set of practices to eliminate a wide range of software production problems, to enhance specific software qualities and to reduce the life cycle cost of the software.

This evolutionary approach has had several effects. The first and most obvious is that an evaluation of any set of relatively mature practices will almost certainly produce a positive impact rating, because, in general, the practices are designed, applied, examined, and refined with that goal in mind. Another more subtle but important observation concerns the evaluated impact of individual practices.

Before a newly implemented practice becomes mature (i.e., during its initial application), its value is strongly governed by two competing phenomena: the Hawthorne effect [18], and a resistance to change/learning curve effect. The first results in emphasizing (often overemphasizing) the strengths of the practice while resistance to change has generally the opposite effect. This competition is healthy since it leads to steady improvement of the practice through refinements which maximize the strengths and minimize the weaknesses. The evolutionary process involves many people, however and it is sometimes painful and usually slow. Thus, the premature evaluation of a practice, if objective, will often be mixed, having positive impact in some areas, negative impact in others. In some areas the impact may be inconclusive (i.e., neither clearly positive nor negative) due to differing experiences and insufficient data.

The best example of the latter phenomenon occurred in the evolution of the structured coding requirement. It was the most recently adopted STP standard programming practice. The evaluated impact of the structured coding practice was (as shown in Section 3) either negative or inconclusive with respect to one third of the desirable qualities of software and the development process. Among the factors which also influenced the structured programming requirement were:

- The formal requirement for structured coding was established two years after the start of the project, after a large amount of code had been developed,

- The structuring standard was imposed on coding in either FORTRAN or assembly language, neither of which easily support writing structured programs, and
- The phase of the project during which the major benefits of structured programming are expected to be reaped (i.e., during operations and maintenance) has not yet begun.

Another important feature of the study results is the grouping of the eleven MPP into three broad categories. The four highest ranked practices (Requirements Analysis and Validation, Baselining of Requirements Specifications, Complete Preliminary Design, and Process Design) fell into one group; the three middle ranked practices (Incremental Development, Unit Development Folders, and Software Development Tools) in another; and the four lowest ranked practices (Formal Inspection of Documentation and Code, Software Configuration Management, Independent Testing, and Enforced Programming Standards) fell in the bottom group. The significant point to be noted is that the highest ranked group contains practices that are generally applied at the front end of the software development activity, the middle ranked group applies to the middle stages of development, while the lowest ranked group applies mostly to the latter stages.

Notice also that the first group, and, to some extent, the second group of practices are viewed as providing assistance to the human performance of tasks generally thought of as complex, thought-provoking and highly creative. The last group, on the other hand is more often viewed as the implementation of policing functions aimed at checking up on developers and pointing out their mistakes. It is likely that a practice-popularity-poll would correlate well with the actual ranking of the groups according to their relative positive impact on software development.

The relative impact ranking of the groups of practices was also strongly influenced by their history of application in the STP chronology (Section 2). In particular, the software development activity has been influenced by the first group of practices for the longest period of time, and those practices, therefore, have had more opportunity to make their positive impact felt. This observation is consistent with recent software engineering research findings (e.g., CCIP-85 and the recently completed TRW Software Reliability Study for RADC) which point out 1) the large fraction (as high as 70%) of total software problems which occur early in the development cycle, and 2) the unusually high value of practices that help find problems as early as possible, before they propagate confusion, additional work, and expense throughout subsequent project phases [17,19].

Another important conclusion of the study is that the isolation of the effect and evaluation of impact of any given practice within a large, complex and dynamic project environment can be exceedingly difficult. Fortunately this conclusion was anticipated at the outset of the study, and great care was taken to identify a select subset of the many STP practices such that:

- The number of practices was small enough to permit thorough study of each,
- Each practice was generic enough (i.e., not strongly oriented toward unique features of STP) to permit relatively straightforward inclusion in Air Force development guidelines with broad applicability,
- Each practice had been used extensively enough to ensure a reasonable base of project data and experience to support impact evaluation, and
- The set of practices, in combined application, spanned the development process and addressed each major development phase.

The careful selection of MPP limited the number of practices a lot and limited their scope a little, but it permitted sufficient depth of the impact analyses and sufficient effort for comparison methodology development to make it possible to ultimately conclude:

- The STP MPP (and some practices in particular) contribute significantly to enhancement of important qualities of software, and generally the investment required to implement and apply MPP pays off in overall life-cycle cost reductions.
- There are significant differences among MPP in both the areas to which they contribute and the magnitude of the contributions they make.
- It has been demonstrated that a methodology can be developed and used to obtain a comparative evaluation of alternative MPP implementations.
- The MPP comparison methodology (including the support tools developed for this study) can be used to both select and evaluate candidate combinations of MPP for projects of varying type, size and complexity.

5.3 Relation of Conclusions to Current Air Force Practices

Some of the MPP determined in this study to have high positive impact are already fairly well covered by established Air Force regulations such as MS-483 and MS-490. These are:

#2. Baselining of Requirements Specification

#10. Software Configuration Management

Other of the MPP have been covered in some of the more recent Air Force and DoD initiatives such as AFR 800.14, DoDD 5000.29, MS-52779, and the recent RADC standards effort. These are:

#1. Requirements Analysis and Validation

#3. Complete Preliminary Design

#8. Independent Testing

#9. Enforced Programming Standards

#11. Formal Inspection of Documentation and Code

Some of the MPP are sufficiently well-formalized and general-purpose to be covered in new Air Force policy initiatives. These are:

#5. Incremental Development

#6. Unit Development Folders

#7. Software Development Tools (partly --e.g., test tools, code auditing tools)

Still other of the MPP need and deserve further formalization and/or investigation of their general applicability prior to being established in policy initiatives. These are:

#4. Process Design

#7. Software Development Tools (partly--e.g., highly specialized simulators and post processors)

5.4 Recommendations

The overall objective of the MPP study was to assess the effects of MPP on a large DoD system software development so that those practices determined to have a favorable effect could be recommended to and considered by the Air Force for adoption on future developments. In light of this goal and owing to the determination of positive impact for all eleven of the STP MPP, it is reasonable to recommend that all be considered for inclusion in Air Force software development guidelines, procurement practices, and policy initiatives. Of these MPP, the four most highly ranked should receive especially strong consideration in view of the pressing need for improved methods for defining, expressing and allocating software requirements and for maintaining their traceability throughout the development process. On the other hand, it is clear that not all software development activities are alike, and few systems compare with the size, complexity and real time performance requirements of the software being developed by TRW on the Systems Technology Program.

Based in part on the above consideration as well as on the detailed MPP study findings documented fully in this report, it is recommended that the Air Force:

1. Continue to emphasize those high impact MPP listed above which are now covered or will be covered in existing and new Air Force and DoD policies
2. Emphasize the following additional high impact MPP in future Air Force policy initiatives:
   - Incremental Development
   - Unit Development Folders
   - Software Development Tools (especially those with demonstrated capability and broad applicability)

3. Promote research to formalize and generalize other high impact MPP:
   - Process Design
   - Software Development Tools (especially requirements definition and design aids, higher order languages and diagnostic language processors, and advanced test tools)
4. Support continued investigation necessary for further confirmation of the results of this study and comparison with other results on other projects
5. Develop standardized definitions and measures (e.g., productivity, modularity, cost) critical to accurate and comparable assessments of impact of alternative MPP
6. Pursue evaluation of MPP used by other contractors on other projects and build upon the impact evaluation/comparison framework using the tools and data base developed in this study
7. Establish ground rules and a methodology for relating software qualities to software operations and maintenance costs to support better assessment of life-cycle MPP impact
8. Initiate research and development of techniques for selecting an appropriate subset of high impact practices commensurate with the type, size, complexity, and available resources of planned software procurements.

6.0 REFERENCES

1. Brown, J. R. , "Getting Better Software Cheaper and Quicker", Practical Strategies for Developing Large Software Systems, Addison-Wesley, 1975, pp. 131-154.

2. Kessler, M. M. and Kister, W. E., "Software Tool Impact", Structured Programming Series, RADC-TR-74-300, Vol. XIV, May, 1975, (A015795).

3. Knuth, D. E., The Art of Computer Programming, Addison-Wesley, 1968.

4. Dahl, O. J., Dijkstra, E. W. and Hoare, C.A.R., Structured Programming, Academic Press (London), 1972.

5. Boehm, B. W., "Software Engineering", IEEE Transactions on Computers, Vol. C-25, No. 12, December, 1976, pp. 1226-1241.

6. Brown, J. R., "Improving Quality and Reducing Cost of Aeronautical Systems Software through Use of Tools", Proceedings of Air Force Aeronautical System Software Workshop, April, 1974.

7. Kosy, D. W., "Air Force Command and Control Information Processing in the 1980s: Trends in Software Technology", Rand Report No. R-1012-PR, June, 1974.

8. Manley, J. H., "Findings and Recommendations of the Joint Logistics Commanders Software Reliability Work Group (SRWG Report)", Vol. I (Executive Summary), November, 1975.

9. Williams, R. D., "Managing The Development of Reliable Software", Proceedings of the International Conference on Reliable Software, April, 1975, pp. 3-8.

10. Software Standards and Procedures Manual, TRW Systems Technology Program Report No. 10780000D, July, 1973, Revised December 1976.

11. Manthey, F. C., "Software Development Tools: Technology", Proceedings of the TRW Symposium on Reliable, Cost-Effective, Secure Software, TRW-SS-74-14, March 1974.

12. Brown, J. R., "Software Test Tools: Technology", TRW-Software Series-74-14, March, 1974.

13. Mullin, F. J., "Software Test Tools: Project Experience", TRW-Software Series-74-14, March, 1974.

14. Brown, J. R. and Buchanan, H. N., The Quantitative Measurement of Software Safety and Reliability: Quantitative Software Safety Study (QSSS) Final Report, TRW Technical Report No. SDP-1776, August, 1973.

15. Boehm, B. W., Brown, J. R., et al, Characteristics of Software Quality, TRW Technical Report No. 25201-6001-TU-00, TRW-Software Series-73-09, December, 1973.

16. Hoskyns, J., "Evaluation of Programming and Systems Techniques: Implications of Using Modular Programming", UK Central Computer Agency Guide No. 1, Her Majesty's Stationery Office, 1973.

17. Thayer, T. A., Software Reliability Study Final Technical Report, TRW Report No. 76-2260.1.9-5, March 1976.

18. Roethlisberger, F. J. and Dickson, W. J., Management and the Worker, Experiments Conducted at Western Electric Company (Hawthorne Plant), Harvard University Press, 1939 (Eleventh printing 1956).

19. Boehm, B. W., "Software and Its Impact: A Quantitative Assessment", Datamation, May, 1973, pp. 48-59.

APPENDIX A
DEFINITIONS

A.1 MPP Definitions

This section provides 1) brief definitions of the candidate modern programming practices (MPP) included in the TRW proposal for the study, 2) brief definitions of the 18 standard programming practices studied in the Programming Standards Impact Evaluation activity, and 3) more detailed definitions of the final 11 MPP upon which primary study effort was concentrated.

A.1.1 Candidate MPP

Process Construction Approach: Use of high level meta-language for rapid configuration of pre-coded modules into a real-time program.

Automated Documentation: Text editors, reformatters, and flowcharters supporting initial preparation and maintenance of detailed software specifications.

Unit Development Folder: Incremental transformation from design specification to product documentation, with the UDF cover sheet as a formal mechanism for unit development planning, scheduling and continuous status accounting.

Programming Standards: Explicit, enforceable standard programming practices with specific emphasis on those which require the production of structured code.

Automated Production Support: Programming support library and program maintenance support facilities. Audit trail of program change history.

N^2 - Chart: A technique for compact illustration of data flow through a process to support program design and impementation through explicit identification of module inputs/outputs and inter-module data dependencies.

Discrepancy Reporting: A formal, closed loop procedure to ensure proper documentation, analysis, corrective action and closure of all detected software anomalies including automated functions of discrepancy report storage, retrieval, distribution and status accounting.

Test Support Tools: A comprehensive set of support tools which 1) help with code inspection for compliance with programming standards (Code Auditor, Product Assurance Evaluator (PACE) STRUCT, Process Construction Program (PCP)), 2) provide automated assistance in preparation, handling, execution and analysis of test cases and test results including automated retest results comparison and test sufficiency measurement (PACE, Logic Reduction Analyzer, Automated Test Case Comparator (ATCC), Data Reduction and Report Generation (DRGG)), and 3) permit simulation and detailed analysis of expected performance characteristics and resource requirements for candidate design solutions (Data Processing Subsystem Simulator (DPSS) and Analytical Load Formulation (ALF)).

Computer Program Assembly List: A management planning technique and automated tool to ensure early identification and detailed definition of total software system structure complete with execution time and core storage allocations and responsible personnel for each program element; use of the CPAL for periodic status reporting of actual versus estimated computer resources and percent complete at the unit level and above.

Independent Test: A formal organization within the project (but separate from design and development organizations) with the responsibility for planning and performing an independent test and evaluation of software for satisfaction of requirements as well as responsibility for direction of software system integration.

Requirements Traceability: Use of documentation techniques and functional processing thread diagrams to establish and maintain forward and reverse traceability from requirements through design to code and from requirements to validation tests.

Functional Capabilities List: A technique used to provide objective assurance of sufficient scope and depth of unit testing done by original developers. Identifies functions recognized in detailed review of design documentation and specifies tests to assure that corresponding capabilities are present in as-built code.

Tiger Teams: Special small teams of highly skilled system engineers and designers organized to focus needed early attention on critical algorithms and reduce the likelihood of serious problems later in the development

process. Typically concentrate on validation of requirements and preliminary design approach. Design and code candidate algorithms to support evaluation of performance characteristics.

Incremental Development: Overall software production approach involving the definition of progressively more complete increments (i.e., "loops") of system capability. Each loop provides an independently testable increment of the evolving program containing prototype code (stubs) in place of functions to be fully developed in a subsequent increment. Builds within loops, carries the incremental development approach to more detail and further reduces the chance of downstream problems.

A.1.2 Programming Standards

Text Format: The requirement to organize and document descriptive information on software modules (subroutines, routine, tasks, and programs) in accordance with the prescribed format of MIL-STD 490, Specification Practices.

Text Level of Detail: The requirement to provide descriptive information on software modules to fully and clearly illustrate their purpose, function and interfaces with other modules in accordance with the prescribed content of MIL-STD 490.

Flow Chart Format: The requirement to develop diagrams depicting the functional hierarchy and execution flow of all software modules using the flow charting symbols prescribed in "The International Organization for Standardization Draft Recommendation on Flowchart Symbols and Their Usage in Information Processing (ANSI X-3.5.1970)".

Flow Chart Level of Detail: The requirement for 1) program and task level flow charts to depict overall composition and hierarchical structure down to the lowest routine level with a single block for each routine containing the routine name and a brief functional description, and 2) routine level flow charts to illustrate in detail the basic function of the routine including identification of all external files used, operating system service requests, decision logic and branches, calculations, calls to subroutines, and entry and exit points.

Statement Label Format: All statement labels to be right justified in Columns 2-5 and all labels to appear in ascending order throughout the extent of the module.

Executable Statement Format: All executable FORTRAN statements to start in Column 7 except where indentation is required to illustrate code segments and nested structures.

Routine Size (Modularity): Each routine to contain a maximum of 100 executable source statements from the time it is initially coded until turnover for independent testing with the limit subsequently raised to 150 to permit incremental addition of new capability without major software breakage.

Calling Sequence Arguments: Arguments in calling statements to subroutines and function subprograms not to contain arithmetic or logical expressions, and routines in the real-time software to have no calling sequence arguments.

Mixed Mode Arithmetic: Mixed mode arithmetic expressions (excluding integer exponentiation) not allowed on the right side of an equal sign.

DO-LOOP Usage: Do-Loops not to exceed six levels of nesting, and Do-Loop index parameters to be expressed only as integer constants or variables.

Computed GO-TO Usage: All computed GO TO statements to be immediately preceded by a validity (range) test of the variable switch parameter. Tests revealing invalid parameter values to initiate error processing. Statement labels used as branch addresses to appear after (i.e., below) the GO TO statement in the sequence in which they appear in the GO TO statement parameter list, with the exception of redundant statement labels in the parameter list.

Labeled COMMON vs Blank COMMON Usage: Only labeled COMMON blocks to be used; use of blank COMMON not permitted.

Imbedded Physical Constants Usage: Literal physical constants (e.g., 3.14159 for the value of pi) not to be imbedded in the code.

Preface Commentary Block Requirement: Each module to contain a standardized block of comment statements immediately following the module declaration statement. The block to contain: 1) module name and version identifier,

2) functional description, 3) inputs, 4) outputs, 5) key local variables, 6) usage restrictions, 7) identification of modules called by this one, and 8) description of abnormal return conditions and actions.

In-Line Commentary Requirement; Each module to contain sufficient in-line commentary in the source code to identify the purpose of every statement which conditionally alters a data value or which alters the sequential execution of program statements.

Structured Coding Requirement: Each module to be structured in accordance with a prescribed, restricted list of segment structures including: 1) sequence of two operations, 2)conditional branch and rejoin, 3) loop construct with condition tested prior to execution of loop body, 4) loop construct with condition tested after execution of loop body, and 5) loop construct allowing for premature escape from unnecessary loop repetition. Permits nesting of segment structures to any level and error-condition transfers from any segment structure to a common point in the module.

Execution of Every Program Branch Requirement: The requirement to cause the execution of every program branch statement and to force the transfer of control to each of the possible branch addresses during unit testing of each module.

Naming Conventions: The requirement to formulate the names of processes, programs, tasks, routines, real-time events, and global data files and parameters in accordance with established patterns prescribing the use of alphanumeric characters identifying item type and function.

A.1.3 Detailed Definitions of Final 11 MPP

Requirements Analysis and Validation: The formal decomposition and allocation of system requirements into structured software requirements and use of automated techniques to analyze and validate the requirements. Software requirements engineering is the discipline for developing a complete, consistent, unambiguous specification which can serve as a requirements baseline for common agreement among all parties concerned, and describing "what" the software product will do. Requirements analysis and validation is the practice through which a given software requirements specification can be analyzed to check for completeness and consistency and validate its content

prior to design and coding. It is a general concensus that it is extremely cost effective to analyze thoroughly the given requirements and detect errors early in the software development cycle. The lack of a good re-requirements specification makes a top down design impossible and testing meaningless, obscures management monitoring capability and prevents good communication between user and developer.

The first step in the software development process is the definition of data processing, and subsequently software, functional and performance requirements which represent the primary input to the subsequent process design activity. Requirements from the system specification which the Data Processing subsystem must perform in any operational situation are allocated to hardware and software. They are iterated with the system level requirements to assure completeness and traceability.

Once all the Data Processing subsystem related requirements have been identified, the generation of the requirements specification is the next order of business. Special attention must be given during this activity as to what makes a good requirement and specification. For real-time systems, the Data Processing subsystem functional requirements are best expressed in terms of:

1) Input
2) Processing Requirements
3) Output

That is for each external stimulus, the subsystem must perform certain processing functions in order to produce the desired output. Requirements expressed in such a manner aid in assignment of performance requirements, establish a one-to-one correspondence between requirements and interface specifications, facilitate the verification of requirements and preliminary design, and facilitate testing.

Equally important are the properties a specification should have. The attributes which are most important are completeness, consistency, traceability, and testability. By careful definition of Data Processing subsystem requirements adhering to the general attributes described above, the resultant requirements specification will enhance software design solution.

Interface definition parallels the Data Processing Subsystem requirements generation. Since the processing requirements are organized by stimuli, it is absolutely essential that for each external interface a complete message catalog be established. Furthermore, compatibility between interface and functional requirements must be assured; that is, the one-to-one correspondence discussed above.

Once the Data Processing Subsystem functional requirements have been identified and documented in a traceable fashion, they must be analyzed and verified for their adherence to the critical attributes. The requirements analysis group utilizes a formalized method of describing requirements in terms of paths. The requirements are functionally grouped and then decomposed in flow chart forms which illustrate the interaction between them. These flow charts, called Requirements Networks or R-NETS, consist of:

1) A set of nodes representing processing steps called alphas.
2) A set of nodes identifying processing conditions, called "OR-nodes".
3) A set of nodes identifying "don't care" sequences of alphas, which could be performed in any sequence or even in parallel ("AND-nodes").
4) A set of interface nodes where external messages are received or transmitted.
5) A set of initial and terminal nodes.
6) A set of arcs joining the nodes to form a network.
7) A set of events and conditions for R-NET enablements.
8) A set of validation points used to uniquely identify transitions.

Each alpha is associated with two subsets of data identifiers which represent data input to and output from the processing step. These, together with the definition of the interface data and the identification of the choice variables, form the basis for consistency checking of the paths. On the other hand, if an R-NET processes data from an interface, the net shows all possible processing paths emanating from the interface. Thus, the processing of each message for each possible condition is identified.

This provides a significant aid in verifying completeness of requirements.

Performance requirements are assigned to validation points providing a connection between these and processing requirements. Each validation point is associated with a set of data identifiers. If performance requirements are expressed in terms of these identifiers, then the requirements will be testable.

A collection of R-NETS define a set of Data Processing functional requirements which can be checked for completeness, consistency and testablility.

A functional simulator is utilized to further analyze and validate the requirements. Simulators model the functional characteristics of the candidate Data Processing Subsystem and the way its resources are utilized. Significant stimuli, hardware characteristics as well as critical processing paths from the R-NET are modeled.

Via functional simulation, the requirements analysis team can demonstrate that the given Data Processing subsystem requirements are implementable on the selected hardware configuration. In the event incompatibility exists, system/subsystem trade-off studies are conducted until a viable baseline is established. An important by-product of the use of the functional simulation is that it provides another method by which the viability and compatibility of the Data Processing Subsystem requirements are demonstrated.

Once a viable hardware configuration has been established, the requirements are allocated to hardware and software. It is important that a complete traceability be maintained from the system to the hardware and software specifications. Interface specifications are also established at this point of the preliminary design phase. From the message catalog defined earlier, interface specifications are generated which define functional requirements and quality assurance provisions for each Data Processing Subsystem interface. They specify communications in terms of external messages, format, rate and the data transfer protocol applicable to the interface.

Process Design: The structuring of functional requirements and operational logic into distinct software functions prior to allocation to software units, and the performance of special tasks at the overall software system level such as the estimation of timing, accuracy and storage requirements, the evaluation of processing limitations, and the analysis of system behavior and computational loading characteristics. Design is performed at the process level where the objective is to ascertain that the entire process fits together and can perform the required tasks within the given time limits and accuracy requirements, that all interfaces are properly specified, and that the data base is correctly defined. Subpractices within process design include the allocation of software requirements to software units, the specification of all data exchanges via N^2 charts, a functional simulation to analyze timing and loading behavior, a preliminary sizing of all data processing resources, an identification of critical issues and process limitations, a specification of the execution sequence for all logic threads.

Process design is the activity that establishes a candidate preliminary design baseline. The three major inputs to this design activity are the baselined and validated Data Processing Subsystem functional and interface requirements and the candidate Data Processing hardware characteristics.

The first step in top-down process design activity is the "mapping" of functional and interface requirements to operating system and application program. These requirements are then further allocated downward into software architectural elements. The purpose of this mapping is to define the required operating system support services and to subdivide the application program into a number of independently schedulable modules which perform specific major processing functions.

Along with the definition of the software architectural elements, the definition of the process structure which will best meet given performance requirements is required. This activity defines how the process (which consists of the operating system and application programs) will perform as a result of external/internal stimuli. Software analysis determines intermodule communications, scheduling/dispatching criteria,

priority servicing, load handling, error processing and recovery. The impact of external intermodule communications upon the data base structure is folded into the data base design activity.

Once the modules of a program have been defined, functional and interface requirements are further allocated downward to the lowest, meaningful entities of the software structure; that is, to routines.

Two important guidelines are followed in this series of downward allocation of requirements. Firstly, complete traceability of requirements must be maintained to the routine level. A traceability matrix must be maintained to the routine level. That is, a traceability matrix must be generated which identifies the allocation of each software and interface requirements paragraph (subparagraph) to software modules and routines. The second important guideline is that performance budgets of storage, accuracy and timing must also be allocated down to the module and routine level. These budgets are derived from software performance specification, hardware constraits and software analysis. Their judicious allocation to software architectural elements can assure that software performance requirements are controllable within the given hardware limitations.

An important activity which supports the generation of realistic routine budget estimates is the key algorithm design. This design activity is conducted to solve critical areas before detailed design and production coding begins. Candidate critical areas are identified with as much detail as possible during process design. Prototype software is developed to find appropriate techniques for solutions.

The data base design is also accomplished as part of the top-down process design activity. The top-level, functional data base is defined using N^2-chart. For a process designer, this chart provides a top-level base from which data flow and design modularity for the entire Data Processing Subsystem can be assessed. Utilizing the N^2-chart, the global data base is defined by the process designers in terms of data files. The preliminary data base design activity establishes the general category of the data base parameters for each file with preliminary definition of the data base constants and variables, formats, units, etc. Size of data files are estimated and fed back to the budget allocation process for modules and routines.

In summary, the process design allocates Data Processing Subsystem requirements and interface requirements in a traceable manner to hardware and various programs that comprise the entire software. Furthermore, the software architecture is defined and requirements and performance budgets allocated to the module/routine level. The process interaction and control have been established and data base structure designed. Consequently, a candidate development baseline has been established.

Incremental Development: The identification, design, coding and testing of functionally meaningful subsets (i.e., increments) of the overall software system wherein each increment 1) provides a self sufficient, executable and testable portion of the complete software capability and 2) adds to and builds upon the preceding increment. Incremental development is an overall software production approach involving the definition and implementation of progressively more complete increments (or "loops") of system capability. Each increment is self sufficient and provides an independently testable increment of the evolving program containing real deliverable code as well as dummy code for those functions that are to be fully developed in subsequent increments. The final increment includes all the deliverable code as a minimum. Builds within each increment carry the incremental development approach to more detail and further reduce the chance of downstream problems. Incremental development reduces risk by developing critical software first, evaluating system performance gradually and allowing an early start to test and integration activities.

Continuation of top down design concepts in conjunction with an incremental implementation approach is accomplished through identification and development of a dummy process to begin with, followed by successive updates replacing dummy software modules with actual code. This incremental approach is supported by earlier development of the overall software structure during preliminary design. Within that structure it is possible to identify selected portions of the software (e.g., processing threads involving critical functions) which can be developed and tested as independent increments. With care, these increments (or loops) can be chosen to provide a complete increment of system capability within which high visibility testing of critical functions can be achieved with relative ease. Each completed increment thus represents not only 1) a meaningful version of

the evolving system containing more capability than the previous increment but also 2) a testbed (including the process structure and test information) with which interface/integration testing of subsequently completed modules can be done.

Complete Preliminary Design: The definition, documentation and baselining of a complete preliminary design prior to detailed design and coding, and including, as a minimum, the top-level software structure, data base definition, interface characteristics, scheduling criteria, and analysis of critical algorithms. The preliminary design specification contains all the conceptual design information needed to support the detailed definition of the individual software system components and, upon completion of the Preliminary Design Review, becomes the work statement and design baseline for development of the detailed design specification used in coding. The preliminary design specification has a previously agreed upon standard format, identifies the mapping from requirements to software design, identifies the organizations responsible for the software to be developed, defines the software structure, specifies required processing resources, specifies the data base organization and makes uses of the results furnished by the process design to specify the estimated resource budgets.

During the process design a candidate preliminary design baseline is established. A significant step in modern programming practice is the demonstration that the preliminary design baseline is viable. The software preliminary design baseline is considered complete when the following conditions are met:

1) All applicable system level requirements have been allocated to the software structure.

2) Interface control has been defined and implemented via interface specifications.

3) Allocation of stated software and interface requirements into basic software structural elements has been completed.

4) Storage, execution timing and accuracy budgets have been established for each software module/routine.

5) The data base has been defined in sufficient detail to demonstrate software operability.

6) Satisfactory design approach for the entire software process and its elements has been demonstrated.

7) Software performance requirements can be met under required operational loading conditions.

Primary methods of demonstrating the above points, that is the completeness of preliminary design, are the use of functional simulation, thread analysis and preliminary design review.

The functional simulator models the characteristics of the Data Processing hardware, external stimuli, operating system and each module of the application program. Each module is modeled by its service requests (data and operational) to the operating system, its logical structure, the execution time for each segment in the logic structure and data base interaction (both external and internal). Performance budgets established during process design are utilized in the simulator. This detailed level of modeling allows an evaluation and insight into the software preliminary design and its complex interaction. The simulator is used by the process designers to examine the sensitivities of computer capacity, timing responsiveness, loading conditions and port-to-port timing. Process design issues such as proper allocation of requirements to software modules, data base interaction efficiency, and recording capacity can be studied. Expected operational conditions are simulated to verify that the software can meet its performance requirements.

Preliminary design validation via the use of functional simulation is an extremely important method during the preliminary design review process for real-time software. This review provides to the software designers and the design reviewers confidence that the preliminary design baseline meets the software and interface requirements, that compatibility exists between hardware/software, that the design is a workable solution and that it meets its performance requirements under required operational conditions.

Thread analysis is another method of demonstrating the completeness of the preliminary design. This analysis identifies each unique input to the DP Subsystem and, for each, a thread is generated identifying the applicable software module involved in processing this input. The unique

software requirements which are involved in such processing, the data base interaction (at the file level) associated with the given input processing and the output produced are identified on these threads. Port-to-port timing requirements are also identified and are verified by the use of the functional simulator.

Thread analysis is employed by the process designers to provide additional assurance that:

1) Each unique input is properly processed and the required output is produced.

2) The requirements are complete.

3) All requirements have properly been allocated to software modules.

4) Inter-module interactions and data base definition are complete.

At the completion of the preliminary design phase a Preliminary Design Review is conducted. This review provides the media by which the soundness and completeness of the preliminary design can be demonstrated. Simulator results, thread analysis and other material are presented to this effect. A successful Preliminary Design Review becomes the foundation from which software detailed design and code production can emanate.

Unit Development Folders: The generation, maintenance and review of a document that is an integral element of a "document as you go policy" and that 1) serves as a collection point for all pertinent development and test information (requirements, design data, code, flow diagrams, test plans, test results, etc.) for each identified software unit, and 2) requires completion of a cover sheet specifying a detailed development plan (i.e., programmer estimates of completion dates for intermediate milestones) and providing a means for indicating actual completion dates and review events to depict the status of each software unit.

A Unit Development Folder (UDF) is prepared and maintained for each software unit in order to provide an organized accessible collection of all requirements, design data, code and test data pertaining to that unit, as these data are produced, and to collect unit-level schedules and status information. The folders are established within one week after the completion of PDR and are maintained until the final as-built software documentation is delivered.

Each UDF contains a cover sheet identifying the contents and schedule of each section of the folder. It is signed off by the originator upon completion of each section. The sections of the UDF are briefly described below:

- Section 0 (Cover Sheet and Schedule) contains a cover sheet delineating scheduled due dates, actual completion dates, assigned originators and reviewer sign-offs and dates plus a change log recording the UDF revision history.
- Section 1 (Requirements) identifies the baseline requirements specification, enumerates the requirements which are allocated for implementation in the software unit, and shows the mapping (by paragraph number) to the requirements specification.
- Section 2 (Design Description) contains the current, working version of the design including, at the appropriate times, the preliminary design and successively more detailed design data and flow charts suitable as a "code to" specification.
- Section 3 (Functional Capabilities List) contains a list of the testable functions performed by the unit obtained from an independent review of the detailed design by someone other than the unit designer.
- Section 4 (Unit Code) contains current source code listings for the unit and a change log of post-baseline updates to the unit code.
- Section 5 (Unit Test Plan) describes the overall testing approach and each test case, identifies test tools or drivers used for unit testing, and illustrates the FCL/Test Case Matrix relating test cases to capabilities tested.
- Section 6 (Unit Test Plan Review) documents the findings from an independent review of the Plan to provide assurance that unit test cases will adequately test branch conditions, logic paths, input and outputs, and error handling procedures.

- Section 7 (Test Case Results) contains a compilation of test case results and analyses necessary to demonstrate that the unit has successfully passed the tests prescribed by the Unit Test Plan.
- Section 8 (Problem Reports) contains status logs and copies of all Design Problem Reports, Design Analysis Reports and Discrepancy Reports which document design and code changes pertaining to the unit.
- Section 9 (Notes) contains relevant memoranda, informal reports and other notes which provide supplementary detail expanding on other UDF contents.

Each UDF is reviewed by the assigned reviewer, other than the originator, following completion of each UDF section. At completion of unit testing, the manager responsible for the unit reviews the entire UDF for technical adequacy and completeness and signs and dates the cover sheet to indicate approval. In addition, periodic audits of UDFs are conducted to ensure compliance with project standards.

Enforced Programming Standards: The establishment and enforcement of strict rules to direct the efforts of developers in the production of code and documentation which meets or surpasses adopted standards involving the 1) definition of certain programming practices which must be followed, 2) identification of other practices which are not permitted, and 3) establishment of an active review/inspection procedure to ensure compliance with the rules and achievement of the standards.

The term, "Programming Standards", generally refers to both 1) approved procedures to be used in the production of code and associated documentation and 2) minimum acceptable standards against which the completed products are measured.* All such standards are, to the extent possible, specified early in the project and documented in a Software Standards and Procedure Manual. One of the primary objectives of the standards is to

* Section A.1.2 provides brief descriptions of 18 programming standards established for the Systems Technology Program software development.

guarantee a high degree of uniformity across all individual software system components which are to be integrated and must interface properly for the complete system to work. This uniformity is especially advantageous when someone other than the original developer is required to understand the purpose, function, intricacies and limitations of a software component during integration, test and maintenance activities.

Establishment of standard programming practices, however, is only the beginning. If the standards are to have the desired effect, they must be followed; if they are to be consistently followed, they usually must be enforced. Enforcement of strict adherance to project standards is accomplished through a combination of informal and formal reviews of code and documentation. Performance of code reviews for compliance with standards is aided substantially by source code scanners including the Process Construction Program, Code Auditor, and STRUCT.**

Independent Testing: The performance of formal testing by a team of testing specialists which is organizationally separate from the group or groups responsible for software design, code, debug, and development testing. It involves analysis of software requirements, planning, preparation, and performance of tests, review of test results for satisfaction of requirements, discrepancy reporting, and retesting after discrepancies are resolved and fixes are made. It also includes support to the developers through review of design documentation and designation of distinct functional capabilities to be exercised during development testing.

The independent test group determines whether or not the software works (not how it works), evaluates requirements and design to assure they are testable, prepares comprehensive test plans, supports quality assurance, and plans the development of test tools. The independence of the test group from the design group is desirable in order to introduce an unbiased and different point of view in the attempt to assess the capability of the developed software.

** These tools are described below in Software Development Tools.

The formal testing activity is briefly characterized as follows:

- The primary objective is to verify that the software satisfies the requirements levied upon it.
- The testing is planned and directed and, in general, completely performed by the independent test group.
- Software must meet specified standards prior to turnover to the test group (e.g., the requirement to exercise every branch condition in a module during unit level testing).
- Software is placed under formal configuration control prior to initiation of formal testing and subsequent changes are subject to established change control policies and procedures.
- Formal testing is audited by the independent quality assurance group for sufficiency and compliance with test plans, procedures and specifications.
- Errors are documented in discrepancy reports, deficiencies are analyzed and corrected by designers, and quality assurance audits are performed to ensure timely and satisfactory problem resolution.
- Test tools are extensively used to supplement the testor's efforts by completely handling otherwise tedious, error-prone and difficult tasks.

Software Configuration Management: The establishment and practice of formal software management principles to set policies and procedures for the review and acceptance of contractual and internal baselines, and including 1) configuration identification and documentation, 2) configuration control to monitor changes to the established baseline specification, and 3) configuration status accounting to report discrepancies and record software implementation status.

Configuration management establishes a series of baselines and methods of controlling changes to these baselines throughout the development and test phases. A configuration management plan, addressing the requirements of this practice, is prepared by the project and approved by the project manager prior to the first software design review. This plan specifies:

- the baseline structure to be used by the project (requirements, design, test, product)
- the baseline definitions (products, events)
- the configuration identification ground rules including the types of products to be controlled and the rules for identifying these products
- the configuration control mechanisms including the definition of change and approved processes for controlled products and the handling of waivers and deviations
- the configuration status accounting system, including the records and reports required to assure traceability of changes to controlled products and to provide a basis for communication of configuration information within the project
- the configuration verification approach including periodic audits to assure that products are developed and maintained according to the groundrules defined in the plan.

The configuration management activity:

- establishes and operates a product development library in which the controlled products are maintained
- establishes and maintains a regular problem/discrepancy reporting system
- monitors the issuance, retention, change control, packaging and delivery of the computer products for conformance to current baselines.

Baselining of Requirements Specifications: The review, customer acceptance and documentation of the software requirements specification to be used as a formal, approved baseline for software development.

The baselining of requirements occurs after the preparation of a software requirements specification and prior to the first software design review. It is intended to achieve a mutual understanding and written agreement as to proper interpretation of mandatory provisions of the requirements specifications which are the basis for software end-item acceptance prior to delivery.

Prior to baselining the software and interface requirements, it is necessary to demonstrate that:

- The Data Processing Subsystem (DPS) is compatible with the overall system definition.
- All applicable system level requirements have been allocated, in a traceable manner, to the DPS and subsequently to hardware and software.
- Software requirements are complete, consistent and testable.
- Consistent interface control has been established.
- The DPS requirements can be implemented on the selected hardware configuration.

In preparing for and conducting the review activities that precede baselining of the specification, the requirements are analyzed and evaluated for technical and contractual acceptability and any identified problems are documented. All issues and problems are discussed and resolved and all agreements and action items resulting from the review are documented prior to baselining of the requirements specification.

Formal Inspection of Documentation and Code: A formal, efficient and economical method for conducting independent and unbiased review and assessment of the quality of software documentation and code. Includes formal inspection techniques such as peer group critiques and walkthroughs, quality assurance reviews and audits, and design reviews.

Formal inspection of documentation and code facilitates the early detection of design errors. A design walkthrough, for example, is accomplished by having the software unit design reviewed by one or more individuals other than the originator. The scope, technique and degree of formality of the walkthrough is established by the project manager prior to PDR. The review includes, as a minimum, checks for responsiveness of design to requirements, design completeness and consistency, flow of data through input/output interfaces, error recovery procedures, modularity, simplicity and testability. The technique used for walkthroughs consists of either a

presentation of the design and the code by the originator in the presence of the reviewers, or a review of the documentation followed by a discussion of review comments involving all participants. Problems detected during the walkthrough are identified in a written summary and made available to the managers responsible for the software.

Independent quality assurance audits are conducted at key intervals during the development and evaluation of software through requirements definition, design, coding, testing and operation. These quality assurance audits are:

- System Requirements Audit; To determine whether planned testing will ensure satisfaction of requirements and to make sure that requirements are traceable to the next higher-level specification.
- Product Specification Audit; To determine whether product specifications conform to established standards of format, content, completeness, and level of detail.
- Interface Verification Audit; To examine interface requirements, interface design and program specifications to identify and resolve interface problems.
- Product Specification (Engineering Design Document) Audit; To examine detailed design specifications and the data base definition to ensure that all routine input/output is properly defined, the design data is current, the structure of flow diagrams conforms with project standards, and explicit traceability exists to higher-level specifications.
- Unit Development Folder (UDF) and Code Audit; To incrementally 1) audit UDFs as they are prepared in verifying that changes to unit requirements, functional design description and flowchart and interface definitions are current, 2) manually (plus with the automated Code Auditor) scan the unit source code for compliance with programming standards, and 3) measure thoroughness of unit level testing and identify, if necessary, additional testing requirements.

- Pre-Turnover Audit; To assess the adequacy of development testing of products prior to their internal delivery to the independent test group through examination of the products and test results for 1) actual versus expected output, 2) completeness, content, organization and approval status of UDFs, 3) compliance with programming standards, 4) testing thoroughness (e.g., execution of every branch condition), and 5) assurance that all reported discrepancies have been corrected and tested.

- Testing Audit; To supplement formal testing done by the independent test group to ensure 1) established configuration management procedures are followed, 2) test specifications are maintained current, 3) test reports are properly prepared, 4) test procedures explicitly define tests to be conducted and test results comply with acceptance criteria in the test procedure, and 5) test data package contents are complete and comply with approved formats.

Software Development Tools: The planned provision and usage of general purpose and specially developed support programs at various points during the software development to assist requirements analysis, software design, code generation, debug and checkout, formal software testing, and documentation in a partially or fully automated fashion. This is a general practice involving the use of automated tools for differing purposes and at various stages throughout the software development process. Prominent examples of software development and test tools are:

- Data base building tools; To build and maintain data definition libraries and data value libraries, check data definitions, and supply complete data specifications to all using software modules.

- System construction tools; To permit use of a high-level, special purpose language in describing the desired system organization and subsequent use of a tool to automatically merge system elements into a unified whole. Also to provide an illustration of system structure, module hierarchy and cross-references. (The STP Process Construction Language and Program (PCL and PCP) are system construction tools.)

- Code-checking tools; To automatically check every line of code in the system to 1) identify and report on instances in which project programming standards have not been followed, 2) identify potential singularities such as division by zero, 3) verify units consistency for each usage of each parameter, and 4) identify parameters that are used before set, set but not used, etc. (For example, PCP counts the executable source statements in a module to check for compliance with the routine size modularity programming standard, STRUCT analyzes the detailed logic structure of a module to check for compliance with the structured coding standard, and Code Auditor automatically checks every line of code for compliance with most of the other detailed STP programming standards.)

- Test Tools; To aid generation, accumulation and evaluation of testing information and, specifically, to 1) support identification of problems early in the development cycle, 2) identify required test data that relates requirements to software capabilities and ensures thorough testing, 3) select test cases to be used in retesting changed software, 4) check actual test results against previously obtained or expected results, and 5) drive and/or monitor software test execution and measure testing thoroughness. (For example, the Code Auditor and Units Consistency Analyzer (UNIC) "test" code for standards compliance and parameter usage consistency; the Product Assurance Confidence Evaluator (PACE) analyzes code, identifies overall logical structure, inserts "test hooks" into the code, and subsequently monitors test execution and reports on the extent to which the software structural components are exercised; the Data Reduction and Report Generator (DRRG) provides essential support in analysis of voluminous, detailed data logged by the real-time process for subsequent post processing and test results evaluation).

A.2 Definitions and Terminology

This section contains 1) definitions of the thirty characteristics (i.e., qualities of software and the software development process) used in the Programming Standards Impact Evaluation survey, 2) descriptions of the twelve "typical problems plaguing software development" used in the MPP Impact Evaluation survey, and 3) brief definitions of other MPP study terminology.

A.2.1 Software Characteristics

The following definitions of the thirty software characteristics used in the Programming Standards Impact Evaluation survey reflect common usage of this terminology in the surveyed environment.

Requirements Traceability: The ability to demonstrate the allocation, implementation and test verification of the individual requirements from the requirements specification to the design specification and to the specification of test requirements.

Code Auditability: The ability to audit code for compliance with design specifications and any pre-designated coding standards.

Documentation Auditability: The ability to audit documentation for compliance with a pre-designated set of documentation formats, standards and requirements.

Personnel Motivation: The interest and enthusiasm of personnel to accomplish the required work in a timely and effective manner with due regard for its quality and for management directions and requirements.

Interface Consistency: The operational compatibility of software modules with each other and with the external software/hardware system.

Software Integration: The process of combining and testing individual software units into larger entities for the purposes of demonstrating interface compatibility and testing for satisfaction of higher level requirements.

Documentation Rework: The revision of documentation to reflect updates or to correct errors.

Coding Rework: The revision of code to reflect updates or to correct errors.

Retesting: The execution of previously run successful tests to assure consistency and non-degradation, or to assess the impact of changes made to the product.

Customer/Contractor Relationship: The state of the environment that exists between the customer and the contractor.

Testing Thoroughness: A measure of the thoroughness of testing based on some established criteria, e.g., percentage of code executed; percentage of requirements tested; percentage of capabilities demonstrated, etc.

Documentation Understandability/Readability: The relative ease with which a program document may be read and understood by someone who has a general familiarity with the subject.

Code Understandability/Readability: The relative ease with which program code may be read and understood by someone who understands the language and has a general familiarity with the subject.

Documentation Maintainability/Useability: The relative ease with which a program document may be updated and used in tracking changes to the program code. This is a function of both the use of documentation tools and the format and standards used in the documentation.

Code Maintainability/Useability: The relative ease with which program code may be updated and used in the development process. This is a function of both the design and programming standards imposed.

Testability: The degree to which code may be tested and test results evaluated under conditions approximating its destined operational environment and the degree to which satisfaction of requirements may be verified.

Operational Reliability: The probability that the software will satisfy the stated operational requirements for a specified time interval or a unit application in the operational environment.

Cost: A measure of the amount of resources used to accomplish a particular task.

Schedule: A plan for the amount of elapsed time needed to accomplish a particular task.

Consistency: The degree to which separate components of a system abide by the same standards and procedures and are uniform with respect to format, content and interfaces.

Completeness: The degree to which the design and documentation define and describe all of the essential elements of a system.

Documentability: The ease with which readable and correct documentation can be produced.

Codability: The ease with which a design solution can be coded.

Documentation Error Frequency: The number of errors per unit measure of documentation.

Coding Error Frequency: The number of errors per unit measure of code.

Portability: The relative independence of a software product from the computer system on which it is designed to operate.

Execution Time: The amount of computer time actually used in executing a particular set of functions.

Core Requirements: The amount of immediate access memory required to contain the code and data for a particular set of functions.

Logical Organization: The clear and progressive delineation of flow and function in design and code.

Programmer Productivity: The amount of time and resources needed to produce, test and document a given unit of code.

A.2.2 Typical Software Development Problems

Cost Overrun: Exceeding the contractual price for accomplishing a stated amount of work or exceeding one's allocated budget.

Development Status Invisibility: The inability to gauge the amount of work performed for the amount of resources expended or the progress being made relative to the schedule.

Unreliability: The inability of the software to perform in accordance with its functional requirements or its unavailability for use.

Unmaintainability: The relative difficulty of debugging and correcting software errors or of making minor modifications or adding new capabilities to a delivered product.

Inadequate Satisfaction of Real Requirements: The inability of the software to perform in an operational environment in accordance with how it was envisioned based on the stated requirements.

Inefficient Use of Resources: The undue expenditure and waste of manpower or computer resources in accomplishing a given task.

Schedule Overrun: Exceeding the contractual time for accomplishing a stated amount of work or exceeding one's allocated time budget.

Inadequate Planning and Control: The lack of visibility and control over budgets, schedules, staffing or resource allocation and performance.

Project Mismanagement: Failure of project management to allocate and control resources necessary to achieve the desired results within the negotiated budget and schedule.

Lack of Programming Discipline: The non-existence or non-adherence to a software development methodology containing standards for software design, coding, documentation, testing, configuration management and quality assurance.

Lack of Conclusive Testing: The non-existence or non-adherence to a documented test plan and procedures which adequately define the test environment, objectives and acceptance criteria or adequately trace test cases and results to requirements and capabilities.

Poor Documentation: The inadequacy of the documentation to describe the design, contents and use of the software at the level required for maintenance and operational application.

A.2.3 Other Terminology Used

Discrepancy: Any of a defined set of problems or conditions which require the generation of a problem report and resultant action independent of its severity.

Executable Instruction (Statement): A computer instruction or statement which requires action by the central processing unit (CPU).

Failure: The result of encountering an error of the type that causes a computer program to abort, halt or perform inadequately.

Line of Code: An individual line of a code listing including executable statements, commentary and other non-executable statements or directives.

Module: A collection of one or more software entities which perform a discrete, definable function.

Practice: A preferred or advisable approach or method which is acknowledged to be generally beneficial or useful in achieving a desired result.

Process: A collection of one or more application programs and their data bases and the operating system configured for a particular hardware configuration and operational objective.

Routine (Subroutine): A set of instructions and/or statements that exist as an identifiable entity and carry out some well defined operation or set of operations. A routine may call or is callable by other routines.

Standard: An unambiguous, explicit description of a desired property of a product or a mandatory practice to be followed.

Subsystem: An assemblage of components which operate as part of a total system and which is collectively capable of performing a specific function within, and as defined by, that system.

Task: The basic unit of software capability dispatched by the operating system, consisting of a task control routine and the required number of lower level routines.

Unit of Code: A manageable element of software architecture containing certain characteristics and encompassed by a Unit Development Folder.

APPENDIX B

STP CHRONOLOGY AND ENVIRONMENT DATA

B.1 Vendor Supplied Support Software

Table B-I lists major components of the development support software provided by CDC to support Systems Technology Program software development. Each component is identified by name and briefly described.

B.2 STP Development Chronology

Tables B-II through B-IV present lists of significant milestone events in the development of major STP software system components:

- Tactical Applications Program (TAP) — Table B-II
- Tactical Operating System (TOS) (including the early, non-real time, functional TOS (FTOS)) — Table B-III
- Test Support Software (TSS) — Table B-IV

B.3 MPP Chronology

Table B-V identifies distinct practices that have been part of the overall STP software development methodology during the period from March, 1972 to the present. Most were conceptually defined as elements of the methodology at the contract outset, but many were either not:

- sufficiently refined for broad application,
- formally established,
- adequately supported with available tools, or
- appropriate for the current project phase

until later in the development process. Table B-V therefore contains two sets of dates to illustrate 1) when each practice became part of the STP development methodology, and 2) when each practice became sufficiently well defined, formally established, supported and appropriate for use by STP developers. An asterisk (*) is used to denote "Before March 1972".

TABLE B-I. CDC Vendor-Supplied Development Support Software

Name	Description
SCOPE 2.1	CDC 7600/7700 Operating System
SCOPE 3.4	CDC 6400/CYBER Operating System
FTN Library	Math library for FTN compiler
FORTRAN Debug	Debug package for use with FORTRAN programs
COMPASS 2.0	Assembly language assembler
COMPASS 3.0	Assembly language assembler with improved MACRO and MICRO capabilities
SYSMOD	Modifies the system nucleus library to add, delete, or replace system overlays and interrupt handlers
SYSLIBE	Modifies the system libraries
SIF Analysis Program	Analyzes System Information File (SIF)
SYSDUMP	Dead dump program
SMM	System Maintenance Monitor (SMM) to support hardware debug
UPDATE/LIBEDIT	Utility for source deck maintenance
SORT/MERGE	Utility for sort/merge
Record Manager	Utility for I/O operations
Loader	Load user programs
Permanent File Manager	Utility for maintaining permanent files
SAS	CDC 7600 CPU Simulator
PPU Simulator	CDC 7000 PPU Simulator

Table B-II. Tactical Applications Program (TAP)

Development Chronology

Begin Engagement Software Design and Development	06 March 1972
Loop 1:	
Preliminary Design Complete	01 February
Detailed Design Complete	25 May 1973
Code and Checkout Complete	11 September
Development Test Complete	01 May 1974
Loop 2:	
Preliminary Design Complete	02 July 1973
Detailed Design Complete	15 February 1974
Code and checkout Complete	01 November 1974
Release 1 (CR-1) or Loop 3:	
Preliminary Design Complete	22 August 1974
Detailed Design Complete	15 January 1976
Code and Checkout Complete	01 August 1976
Process Integration and Evaluation Testing Complete	01 January 1977
Preliminary Release of CR-1 Software	16 August 1976
Final Release of CR-1 Software	01 January 1977
Release 2 (CR-2):	
Preliminary Design Complete	01 October 1976
Detailed Design Complete	01 January 1977

Table B-III. Tactical Operating System (TOS/FTOS) Development Chronology

Begin Engagement Programs	06 March 1972
Preliminary TOS Design Complete	20 March 1973
Version 1 FTOS:	27 July 1973
Version 2 FTOS:	31 May 1974
Detailed TOS Design Complete	26 July 1974
Version 3 FTOS:	01 October 1974
Version 4 FTOS:	01 April 1975
Version 1 TOS (Preliminary Capability):	01 May 1975
Version 5 FTOS:	14 June 1975
Version 6 FTOS:	01 Aigist 1975
Version 2 TOS (LOEU, GDS Capability):	15 March 1976
Version 3 TOS (RDC Capability):	01 July 1976
Version 4 TOS (Triplex Capability):	15 October 1976

AD-A040 467 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/2
IMPACT OF MPP ON SYSTEM DEVELOPMENT.(U)
MAY 77 J R BROWN F30602-76-C-0095
UNCLASSIFIED TRW-29115-6001-RU00 RADC-TR-77-121 NL

2 OF 3
AD
A040467

Table B-IV. Test Support Software (SETS/KTSP)*

Development Chronology	
Begin TSS Programs Design and Development	06 March 1972
SETS Engineering Requirements Document (ERD) Release	01 September 1972
KTSP Engineering Requirements Document (ERD) Release	02 February 1973
Preliminary Design for KMR SETS Complete	28 February 1973
Preliminary Design for KMR SETS Complete	24 October 1973
KTSP Preliminary Design Review	24 October 1973
SETS Loop 1: Detailed Design, Code and Test Complete	01 May 1974
SETS Loop 2: Detailed Design, Code and Test Complete	01 November 1974
SETS Loop 3:	Deferred Past Current Contract End Date
KTSP Release 1 (CR-1) or Loop 3: Detailed Design, Code and Test Complete	01 January 1977
KTSP Release 2 (CR-2):	01 January 1977

* SETS = System Environment and Threat Simulator
KTSP = KMR Test Support Program

Table B-V. Systems Technology Program Modern Programming Practices (MPP)

Summary of Chronology

MPP	Concept Defined	Available/Used
Process Construction Program (PCP)	*	January 1973
Automated Documentation Tools	*	*
Unit Development Folders (UDFs)	May 1972	Oct-Nov. 1972
Programming Standards	*	May 1972
Quality Assurance Tools	August 1972	November 1973
Structured Programming	April 1974	April 1974
Top-down Design Concept	*	April 1973
Independent Test Organization	*	*
Incremental Development Approach	*	*
Complete Preliminary Design	*	*
Data Processing Subsystem Simulation (DPSS)	*	*
Automated Production Support Facilities	*	*
Discrepancy Reporting System	*	April-May 1973
Requirements Analysis and Validation	*	*
Process Design	*	*
Test Tools	*	August 1973
Document-as-you-go Concept	*	*
Software Configuration Management	*	*
Quality Assurance Audits	*	March 1973
N^2 Chart	*	February 1973

* Before March 1972

APPENDIX C

MATHEMATICAL ALGORITHMS

The following are detailed descriptions of the statistical algorithms developed for the analysis of the survey raw data. These algorithms have been implemented in the software program IMPACT. A listing of IMPACT is included in Appendix D. An A x B matrix of practice versus characteristic is assumed. The (i,j) th element of this matrix represents a measure of the influence that practice i has on characteristic j. Let N_{ij} be the number of survey responses determined for the (i,j)th element. Since each response is a discrete variable ranging from -2 to +2, a distribution function, d_{ij}, can be specified for each element.

C.1 Quantitative Algorithms

Assume that each distribution function d_{ij} is the sample distribution associated with a random variable. The sample mean can be computed as:

$$\mu_{ij} = \frac{\sum_{k=1}^{N_{ij}} X_{ijk}}{N_{ij}} \qquad \begin{matrix} i=1,\ldots A \\ j=1,\ldots B \end{matrix} \qquad \text{(C.1-1)}$$

where X_{ijk} is the kth response for practices-characteristic pair (i,j) and N is the number of non-blank responses for practice-characteristic pair (i,j).

The sample variance, σ^2_{ij}, can be computed as:

$$\sigma^2_{ij} = \frac{\sum_{k=1}^{N_{ij}} (X_{ijk} - \mu_{ij})^2}{N_{ij}} \qquad \begin{matrix} i=1,\ldots A \\ j=1,\ldots B \end{matrix} \qquad \text{(C.1-2)}$$

In a similar manner, the overall effect that a practice has on the complete set of characteristics can be determined. The overall mean u_i and variance σ_i^2 for practice i are given by:

$$\mu_i = \sum_{j=i}^{A} N_{ij}\, \mu_{ij}/N_i \qquad \text{(C.1-3)}$$

$$\sigma_i^2 = \sum_{j=1}^{A} \sum_{k=1}^{N_{ij}} \frac{(X_{ijk} - \mu_i)^2}{N_i} \qquad \text{(C.1-4)}$$

where

N_i = Total number of responses obtained for practice i,

$$N_i = \sum_{j=1}^{A} N_{ij}$$

The mean values for each practice of the Programming Standards Survey are depicted by the "dots" in Figure C-1. The shaded areas in the figure indicate the plus and minus one sigma (σ_i) variations about the means. As seen from the figure, all practices (or standards) have a positive mean value indicating that on the average the standards generally have an overall positive influence on the complete set of characteristics. It should also be noted that the one sigma variations about the mean are generally rather large. Therefore, even though the mean is positive, it cannot be concluded with high confidence that the standard actually has a positive effect. In order to make more definitive statements concerning the overall influence of the standard, the shape of the distribution function must be considered. The evaluation of the distribution functions will be discussed in detail in Section C.2.

Another consideration of interest is what effect the complete set of practices has on a given characteristic. In a manner analogous to equations C.1-3 and C.1-4, the overall mean, μ_j, and variance σ_j^2 for characteristic j can be computed as:

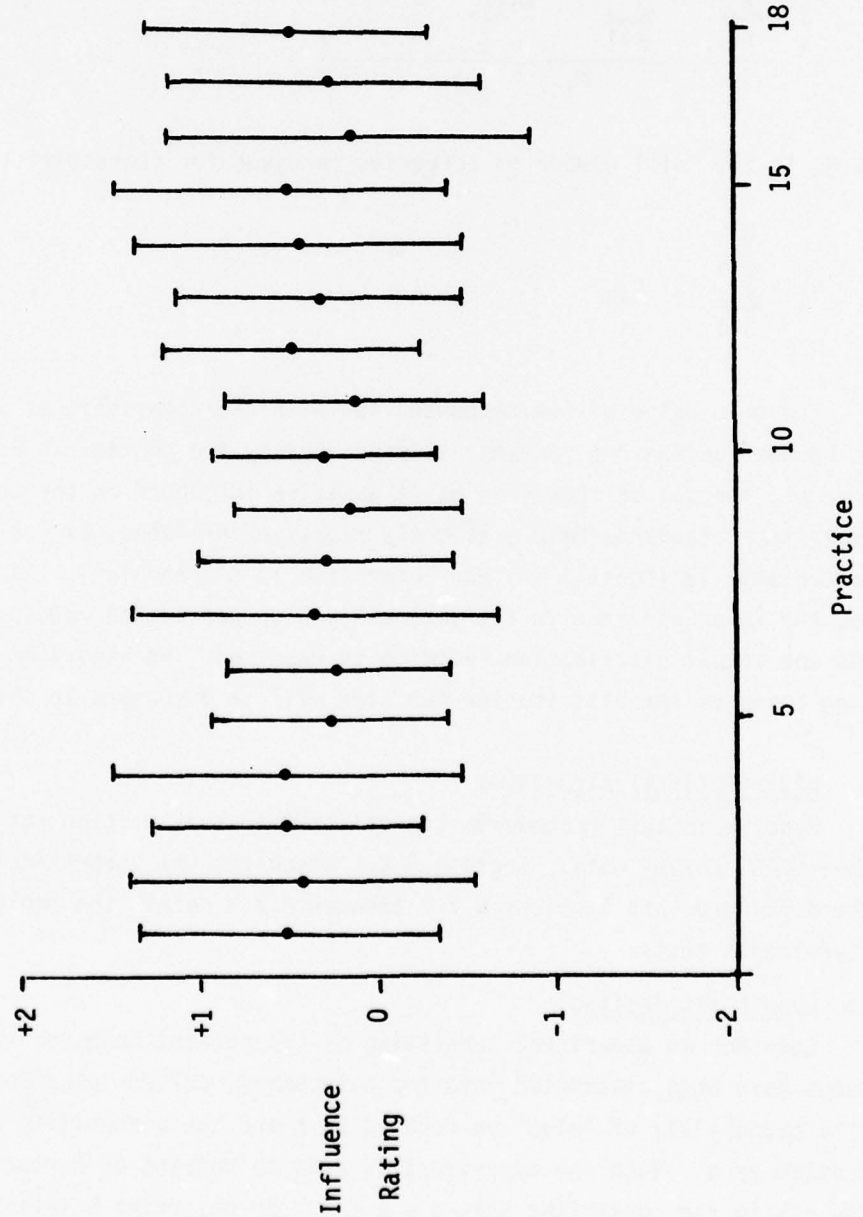


Figure C-1. Practice Overall Means and Variances (Programming Standards Survey)

$$\mu_j = \sum_{i=1}^{B} \frac{N_{ij}\,\mu_{ij}}{N_j} \qquad \text{(C.1-5)}$$

$$\sigma_j^2 = \frac{\sum_{i=1}^{B} \sum_{k=1}^{N_{ij}} (X_{ijk} - \mu_j)^2}{N_j} \qquad \text{(C.1-6)}$$

where N_j is the total number of responses received for characteristic j, i.e.,

$$N_j = \sum_{i=1}^{B} N_{ij} \qquad \text{(C.1-7)}$$

The mean value of the responses for each characteristic as well as the one sigma bounds for the Programming Standards Survey are plotted in Figure C-2. In general, the set of standards has a positive influence on the characteristic. However, some standards have a slightly negative influence, as for example, on characteristic 18 ("Cost") and characteristic 19 ("Schedule"). As previously noted, the large variance in the data makes high confidence conclusions difficult unless the actual distribution function is examined. An algorithm for hypothesis testing based on the distribution function will be discussed in the next section.

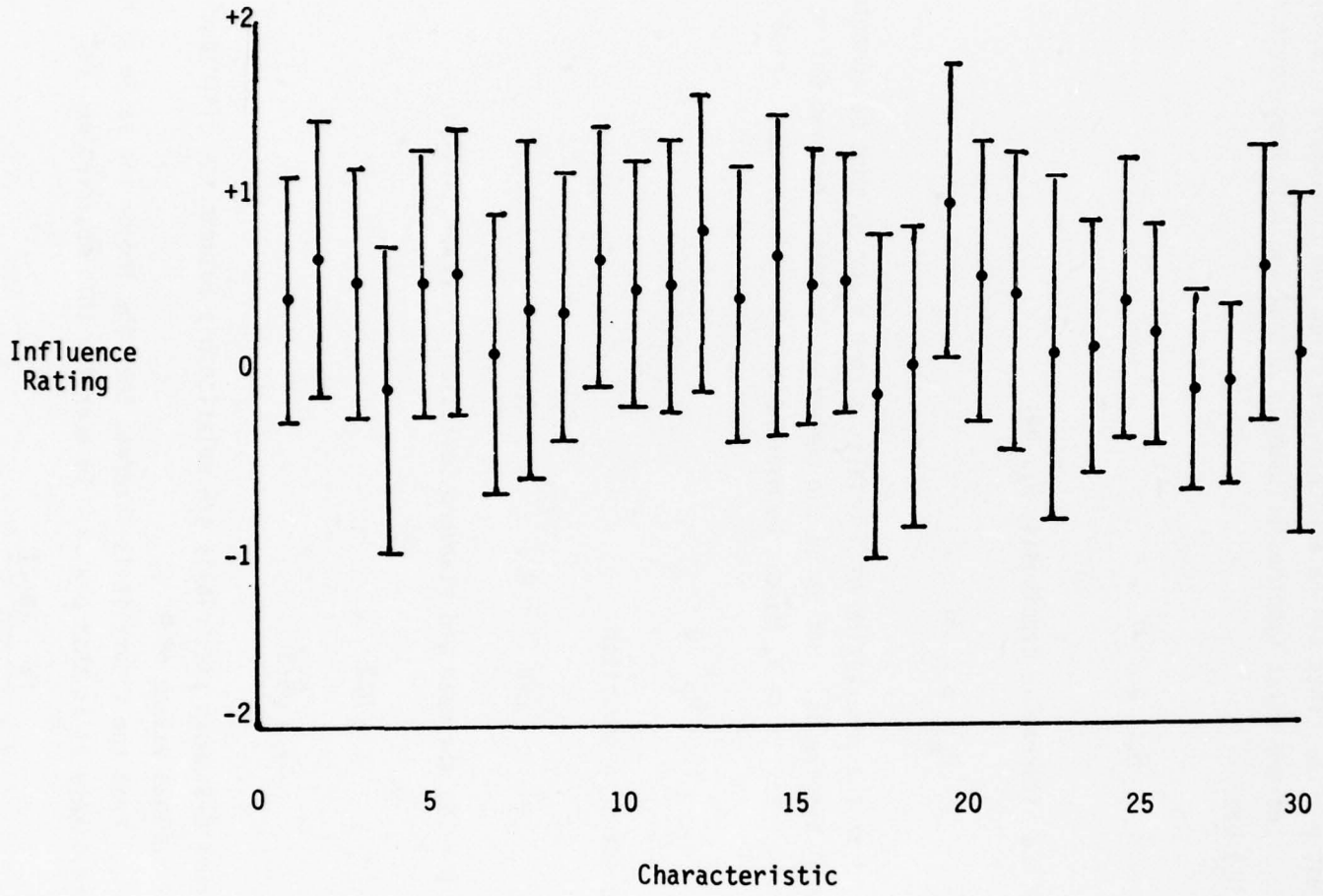
C.2 Distributional Algorithms

Hypothesis test procedures are utilized in this section for the analysis of the IMPACT survey data. Section C.2.1 discusses the theoretical foundations of the procedures and Sections C.2.2 through C.2.5 detail the implementation of the hypothesis tests.

C.2.1 Hypothesis Testing

Consider an experiment consisting of independent repeated trials whose outcomes have been classified into two categories, called "plus" and "minus". Let the probability of "plus" be denoted by p and the probability of "minus" be denoted by q. Such an experiment is said to consist of Bernoulli trials and is subject to the constraint that $p + q = 1$. By observing N trials of this

Figure C-2. Characteristic Overall Means and Variances (Programming Standards Survey)



experiment, it can be determined (within some probability measure) whether or not the probability of pluses exceeds the probability of minuses, i.e., whether or not $p \geq .5$. This can be accomplished using the procedures discussed below.

Let the first hypothesis (usually termed the null hypothesis and denoted by H_o) be:

$$H_o: \quad p < .5 \quad ,$$

and the alternative hypothesis, H_A, be:

$$H_A: \quad p \geq .5 \quad .$$

Based on the observation of N trials, either H_o or H_A must be accepted as the "true" hypothesis. Let Np be the number of "pulses" observed during the N trials. For large N, Np can be modeled as a Gaussian random variable with mean:

$$\mu = Np \quad ,$$

and standard deviation

$$\sigma = [Np\ (1 - p)]^{1/2}$$

If $p = .5$, the mean and standard deviation are respectively:

$$\mu = N/2 \quad ,$$

$$\sigma = \sqrt{N/2} \quad .$$

Figure C-3 shows pictorially the relationship between the distribution of Np for various values of p.

From the figure it is observed that the larger the value of Np, the more likely it is that $p > .5$. To quantify this observation, let

$$Z = \frac{Np - (N/2)}{\sqrt{N/2}} \qquad (C.2.1)$$

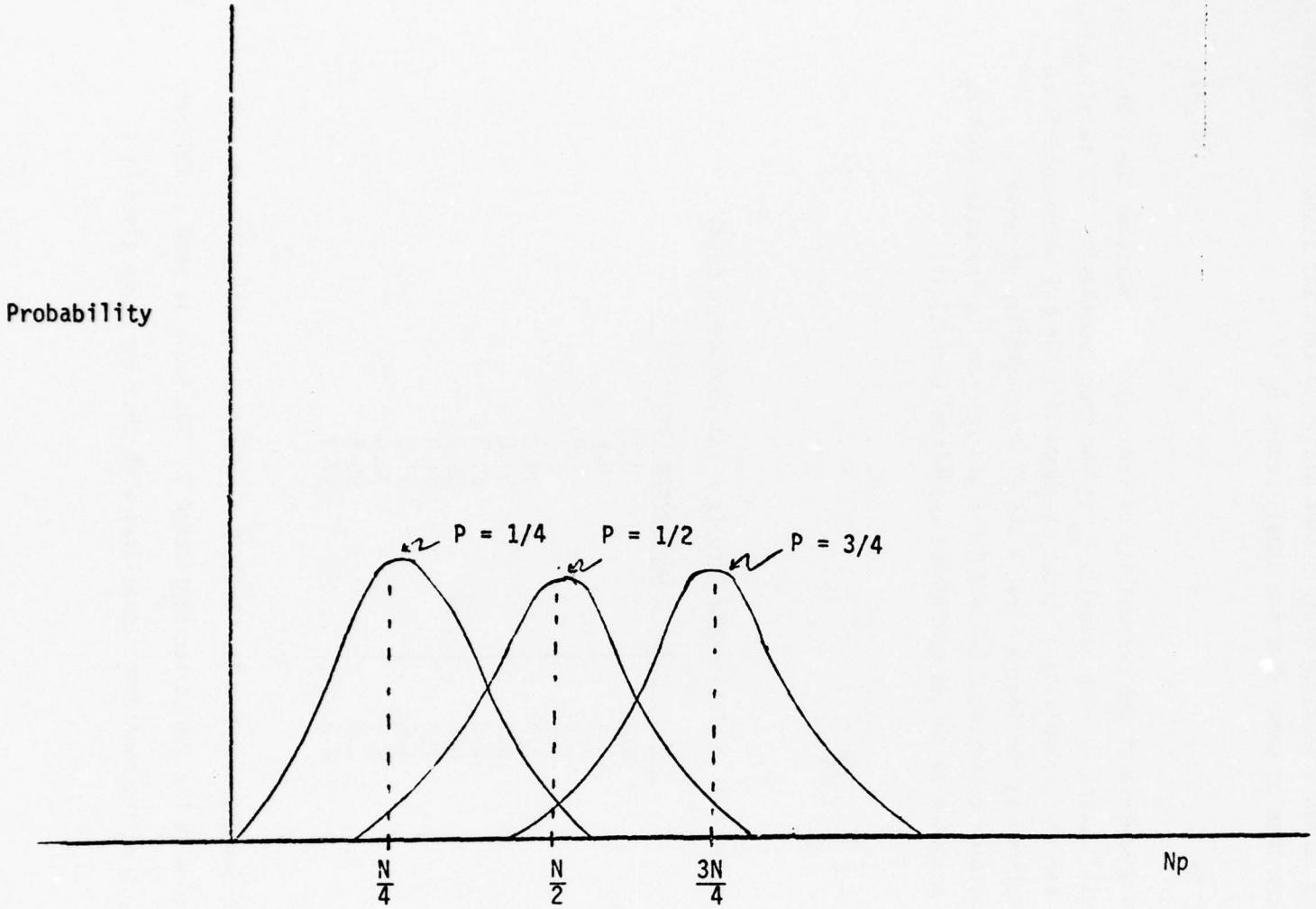


Figure C-3. Distribution of Np for the Indicated Value of P

When p = .5, Z has a Gaussian distribution with mean zero and standard deviation one. It can then be shown that one should accept H_A if

$$Z \geq k, \qquad (C.2\text{-}2)$$

where k is a measure of the probability of error, i.e., k measures the probability that one will accept H_A when actually H_o is the true hypothesis. The relationship between k and the probability of error is shown in Table C-I where condifence level is defined as confidence level = 100 (1 - probability of error).

Combining equations C.2-1 and C.2-2 and rearranging indicates that H_A should be accepted (with the confidence associated with k) if

$$\frac{Np}{N} \geq \frac{1}{2} + \frac{k}{2\sqrt{N}}$$

Table C-I. Relationship Between k and Confidence Level

k	Confidence Level (%)
-3.0	0.13
-2.5	0.6
-2.0	2.0
-1.5	7.0
-1.0	16.0
-0.5	31.0
0.0	50.0
0.5	69.0
1.0	84.0
1.5	93.0
2.0	98.0
2.5	99.4
3.0	99.9

For example, let $N \approx 36$. Table C-II gives the cutoff value, x, for $\frac{Np}{N}$ together with the associated confidence C. The table is read as follows: if $\frac{Np}{N} \geq x$, then the confidence associated with choosing H_A is given by C.

From the table it is seen that if Np/N is greater than .75, one may choose H_A and have extremely high confidence that the "pluses" are truly in the majority.

So far confidence levels have been determined only if H_A is accepted. In an analogous manner, the probability of error associated with choosing H_o can be derived, i.e., the probability that H_o is accepted when H_A is in reality the true hypothesis.

If H_A is rejected (i.e., H_o accepted), then Np/N is less than the cutoff value x. This implies that NM/N > (1 - x) where Nm is the number of "minuses" observed. Thus the confidence associated with accepting H_o is obtained from Table C-II as the confidence associated with the cutoff value (1 - x). For example, if

$$\frac{Np}{N} = x = .25 \ ,$$

then one has very low confidence (.13%) if H_A is accepted. However, if H_o is accepted, the confidence that a correct decision has been made is 99.9%, (the confidence associated with the 1 - x = .75 entry of Table C-II). Thus for this data, H_o would be accepted with a high level of confidence.

The data in Table C-II will be utilized in the following sections to determine the confidence in the assertions associated with the analysis

Table C-II. Example Confidence Level for N = 36

Cut-off Value x	Confidence Level C %
0.25	0.13
0.50	50.0
0.75	99.9

of the Impact Evaluation Survey data. The assertions will be of the form:

"Practice i has a positive (or negative) influence on characteristic j."

To determine if the assertion is supported by the data, a several stage hypothesis testing procedure is utilized. First a hypothesis test is constructed to determine if practice i does influence characteristic j. This determination is described in the next section. If it is concluded that the practice does influence the characteristic, Section A.2.3 will describe the algorithm to determine if this influence is positive, negative, or inconclusive. The overall procedure will be presented in summary form in Section A.2.4.

C.2.2 Assertion Confidence

For each practice-characteristic pair (as well as the overall data for both practices and characteristics), the following hypotheses were tested:

H_o: The respondents to the impact evaluation survey were in agreement that the practice i did have an influence on the characteristic j (i.e., influence rating $\neq$ 0).

H_A: The respondents to the impact evaluation survey were in agreement that the practice i did not have an influence on the characteristic j (i.e., influence rating = 0).

These hypotheses were tested based on the statistic $N_R(i,j)$ defined as:

$$N_R(i,j) = \frac{N_o(i,j)}{N_{ij}} ,$$

where,

$N_o(i,j)$ = Number of indifferent (0) responses observed for the (i,j)th practice-characteristic pair, and

N_{ij} = Total number of responses for the (i,j)th practice characteristic pair.

Since N_{ij} was usually in the 30 to 40 range, Table C-II can be utilized to give an indication of the confidence level associated with the hypothesis test.

If $N_R(i,j) > .75$, Table C-II indicates that H_A can be accepted with a high level of confidence. On the other hand, if $N_R(i,j) < .25$, H_o is the

high confidence choice. Values of $N_R(i,j)$ between .25 and .75 are in an uncertainty zone where neither hypothesis can be selected with high confidence.

For presentation of the survey results in this report the level of confidence is denoted by the following terms:

Strong Assertion: A very high level of confidence that the assertion is correct. Cutoff values are as follows:

$N_R(i,j) > .75$ - highly confident that H_A is correct, i.e., the i^{th} practice does not influence the j^{th} characteristic.

$N_R(i,j) \leq .25$ - highly confident that H_o is true, i.e., the i^{th} practice does influence (either positively or negatively) the j^{th} characteristic.

Medium Assertion: Indications are that the practice does influence the characteristic. However, the data does not support a clear cut verification. Assertions will be termed medium when $.25 < N_R(i,j) \leq .50$.

Weak Assertion: The practice probably has only a weak (if any) influence on the characteristic. The majority of the respondents gave an indifferent rating to the practice-characteristic pair (i.e., $.50 < N_R(i,j) \leq .75$).

C.2.3 Assertions (Influence Rating)

Once it has been determined that practice i does have an influence (strong, medium, or weak) on characteristic j, the next step is to determine if the influence is positive or negative. To accomplish this the following hypotheses were tested:

H_o: Respondents to the survey agreed that practive i had a positive influence on characteristic j (i.e., influence rating +1 or +2).

H_A: Respondents to the survey agreed that practice i had a negative influence on characteristic j (i.e., influence rating -1 or -2).

These hypotheses were tested based on the statistic $RNN_{i,j}$, defined as

$$RNN(i,j) = \frac{NN(i,j)}{NR(i,j)},$$

where,

NN(i,j) is the number of respondents who indicated that practice i had a negative (-1 or -2) influence on characteristic j, and NR(i,j) is the total number of respondents who indicated that practice i had an influence (-2, -1, +1, or +2) on characteristic j.

If RNN(i,j) exceeded .75, hypothesis H_A is accepted with high confidence. On the other hand, if RNN(i,j) was less than .25, H_o was selected. If .25 < RNN(i,j) < .75, neither hypothesis can be selected with high confidence and the result of the experiment is inconclusive.

For strong assertions which were not inconclusive, an additional hypothesis test was performed to pinpoint the influence rating. This additional hypothesis test was not utilized for medium or weak assertions since their confidence level is low. For strong assertions for which a positive influence was established the following test was made.

H_o: For the respondents who agreed that practice i had a positive effect on characteristic j, the majority agreed that the influence was medium (+1).

H_A: For the respondents who agreed that practice i had a positive effect on characteristic j, the majority agreed that the influence was strong (+2).

This hypothesis was tested by utilizing the statistic

$$N_{+2}(i,j)/N_{+}(i,j) \qquad \text{(C.2-3)}$$

where,

$N_{+2}(i,j)$ is the number of respondents who indicated that practice i had a +2 effect on characteristic j, and

N_+ (i,j) is the number of respondents who indicated that practice i had a positive influence (+1 or +2) on characteristic j.

If the ratio given by equation C.2-3 exceeded .5, hypothesis H_A was accepted; if not, H_0 was chosen. An analogous hypothesis test was performed for strong assertions for which a negative influence was established.

C.2.4 Summary

Table C-III provides a summary of the hypothesis test algorithm in tabular form.

TABLE C-III. Hypothesis Test

ASSERTION STRENGTH/ INFLUENCE RATING	ABBREVIATION	NUMBER OF	RATIO	
STRONG/STRONG NEGATIVE	SSN	ZEROS $\leq$ 25%	NEG/POS $\geq$ 3	-2/-1 $\geq$ 2
STRONG/NEGATIVE	SN	ZEROS $\leq$ 25%	NEG/POS $\geq$ 3	-2/-1 $<$ 2
STRONG/POSITIVE	SP	ZEROS $\leq$ 25%	POS/NEG $\geq$ 3	+2/+1 $<$ 2
STRONG/STRONG POSITIVE	SSP	ZEROS $\leq$ 25%	POS/NEG $\geq$ 3	+2/+1 $\geq$ 2
STRONG/INCONCLUSIVE	SI	ZEROS $\leq$ 25%	NEG/POS $<$ 3	
MEDIUM/NEGATIVE	MN	25% $<$ ZEROS $\leq$ 50%	NEG/POS $\geq$ 3	
MEDIUM/POSITIVE	MP	25% $<$ ZEROS $\leq$ 50%	POS/NEG $\geq$ 3	
MEDIUM/INCONCLUSIVE	MI	25% $<$ ZEROS $\leq$ 50%	NEG/POS $<$ 3	
WEAK/NEGATIVE	WN	50% $<$ ZEROS $\leq$ 75%	NEG/POS $\geq$ 3	
WEAK/POSITIVE	WP	50% $<$ ZEROS $\leq$ 75%	POS/NEG $\geq$ 3	
WEAK/INCONCLUSIVE	WI	50% $<$ ZEROS $\leq$ 75%	NEG/POS $<$ 3	
STRONG/INDIFFERENT	IND	ZEROS $>$ 75%		

APPENDIX D

IMPACT EVALUATION/COMPARISON TOOLS

D.1 General

Three programs were developed to support the analyses performed in the MPP study. This appendix provides a brief description of the purpose and function and a source listing for each of the programs:

- IMPACT
- MERIT
- RANK

These programs were coded in the FORTRAN extended language for use on the CDC Cyber 74/174 (TRW/TSS) computer system.

D.2 IMPACT

D.2.1 Purpose

The IMPACT program was required to analyze impact evaluation survey response raw data and produce composite impact assessments.

D.2.2 Input

IMPACT accepts as input the coded influence ratings (+2, +1, 0, -1, -2) contained in the impact evaluation matrices for N survey questionnaires. A rating value of 3 is used to indicate a blank in the evaluation matrix.

D.2.3 Function

IMPACT performs the quantitative and distributional algorithms described in Appendix C.

For each element of the impact evaluation matrix, IMPACT examines the input ratings from the N survey questionnaires and derives a response-frequency-distribution (i.e., the number of +2 responses, +1 responses, etc) and a corresponding rating-percentage-distribution. The mean rating and standard deviation are also computed. IMPACT performs the hypothesis test (distributional algorithm) on the response-frequency-distributions and derives a composite assertion strength/influence rating for each matrix element. In addition, IMPACT computes and analyzes summary response-frequency-distributions for each matrix row and column and for the entire matrix and derives corresponding summary measures of assertion strength/influence rating.

D.2.4 Output

IMPACT output consists of the response-frequency-distributions, percentages, mean and standard deviation statistics, and the composite assertion strength/influence ratings. Examples of IMPACT output (MPP theoretical results) are presented in the sections following.

D.2.4.1 Mean and Standard Deviation

IMPACT determines the mean, number of non-blank responses, and standard deviation (variance) for each matrix element (i,j pair). The results presented in Table D-III are to be interpreted in the following manner:

a. A row index, (i.e., "practice" index), is printed vertically along the page. (The relationship between the practice and its index is presented in Table D-I).

b. A column index, (i.e., "problem" index), is printed horizontally across the page. (The mapping of problem versus index is presented in Table D-II).

c. For each i,j pair three numeric values are derived. These values are as follows:

```
* * * * * * * * * * * * * *
*   sample mean             *
*   number of non-blank responses  *
*   sample variance         *
* * * * * * * * * * * * * *
```

Table D-I Practice Indices

Index	Practice Identification
1	Requirements Analysis and Validation
2	Process Design
3	Incremental Development
4	Complete Preliminary Design
5	Unit Development Folders
6	Enforced Programming Standards
7	Independent Testing
8	Software Configuration Management
9	Baselining of Requirements Specification
10	Formal Inspection of Documentation and Code
11	Software Development Tools

Table D-II Problem Indices

Index	Problem Identification
1	Cost Overrun
2	Development Status Invisibility
3	Unreliability
4	Unmaintainability
5	Inadequate Satisfaction of Real Requirements
6	Inefficient Use of Resources
7	Schedule Overrun
8	Inadequate Planning and Control
9	Project Mismanagement
10	Lack of Programming Discipline
11	Lack of Conclusive Testing
12	Poor Documentation

For example, the mean, number of non-blank responses, and variance of the "Process Design" practice (row index of 2) and "Unreliability" problem (column index of 3) pair is shown on Table D-III to be:

```
* * * * *
*  1.24 *
*    49 *
*   .74 *
* * * * *
```

It should be noted that following column 12 is a column containing the sample mean, number of non-blank responses, and sample variance summary data for each row. Similarly, the row following row 11 contains summary data of each column.

D.2.4.2 Response-Frequency-Distributions and Percentages

IMPACT determines the response-frequency-distributions and percentages for each matrix element (i,j pair). The results presented in Table D-IV are to be interpreted in the following manner:

a. Row and column indices are printed with the same meaning as stated in D.2.4.1-a and D.2.4.1-b.

b. For each i,j pair, two columns of five numbers each are derived. These columns have the following interpretation:

Column 1 Rating Frequency	Column 2 Rating Percentage
Number of Strong Positive (+2) influence ratings	Percentage of +2 responses
Number of Positive (+1) influence ratings	Percentage of +1 responses
Number of No (0) influence ratings	Percentage of 0 responses
Number of Negative (-1) influence ratings	Percentage of -1 responses
Number of Strong Negative (-2) influence ratings	Percentage of -2 responses

For example, the distribution of responses showing the effect that the practice "Unit Development Folders" had on the problem "Development Status Invisibility" is obtained as follows: "Unit Development Folders" has a row index of 5 and "Development Status Invisibility" has a column index of 2. Column 1 of entry (5,2) of Table D-IV shows that the following distribution of responses was obtained for this i,j pair:

Number of Responses	Responses
46	Strong Positive (+2) influence rating
6	Positive (+1) influence rating
1	No (0) influence rating
1	Negative (-1) influence rating
0	Strong Negative (-2) influence rating

Similarly, Column 2 of entry (5,2) shows that the percentage distribution of responses is as follows:

Percentage of Responses	Responses
85	Strong Positive (+2) influence rating
11	Positive (+1) influence rating
2	No (0) influence rating
2	Negative (-1) influence rating
0	Strong Negative (-2) influence rating

The 13th column contains summary data of each row. The 12th row contains summary data of each column.

D.2.4.3 Composite Assertion Strength/Influence Ratings

IMPACT determines the composite assertion strength/influence ratings for each matrix element (i,j pair). The results presented in Table D-V are to be interpreted in the following manner:

a. Row and column indices are printed with the same meaning as stated in D.2.4.1-a and D.2.4.1-b.

b. For each i,j pair, an assertion strength/influence rating indicator is derived. The formulation of this indicator is discussed in detail in Appendix C. Table C-III summarizes all of the possible rating indicators.

From Table D-V, the assertion strength/influence rating indicator of row index 3 and column index 1 is SP. By referencing Table D-I, D-II, and C-III, the following can be concluded about the SP rating of i,j pair (3,1): "We can assert with strong confidence that, in the theoretical case, the practice (Incremental Development) can and should make a positive contribution toward eliminating the problem (Cost Overrun)".

TABLE D-III. IMPACT Output - MPP Survey Mean and Variance Results

	1	2	3	4	5	6
	1.62	1.12	1.38	.98	1.80	1.14
1	55	52	53	53	54	50
	.65	.85	.76	.81	.45	.77
	1.37	1.32	1.24	.96	1.44	1.29
2	49	50	49	48	50	49
	.75	.76	.74	.91	.67	.78
	1.15	1.59	1.19	.74	.90	.78
3	52	54	52	50	48	51
	.97	.87	.86	1.02	.94	1.02

TABLE D-III. IMPACT Output - MPP Survey Mean and Variance Results (Continued)

	1	2	3	4	5	6
	1.56	1.43	1.24	1.02	1.54	1.16
4	52	54	54	53	52	51
	.72	.71	.66	.84	.66	.75
	.73	1.80	.82	1.51	.66	.66
5	49	54	50	51	50	47
	.94	.56	.84	.64	.82	.86
	.31	.23	1.28	1.50	.18	.06
6	49	48	53	52	51	51
	.81	.47	.74	.82	.51	.87

TABLE D-III. IMPACT Output - MPP Survey Mean and Variance Results (Continued)

	1	2	3	4	5	6
	.27	.63	1.81	.55	1.67	.40
7	49	48	52	49	51	47
	.92	.67	.48	.78	.55	.96
	.63	1.12	1.12	1.15	.67	.40
8	46	49	50	47	48	47
	.87	.82	.77	.77	.77	.73
	1.65	1.04	1.10	.85	1.77	1.28
9	51	49	48	48	53	47
	.74	.83	.77	.79	.42	.82

TABLE D-III. IMPACT Output - MPP Survey Mean and Variance Results (Continued)

	1	2	3	4	5	6
	.66	1.19	1.31	.90	1.21	.43
10	44	47	49	42	48	44
	1.00	.82	.71	.75	.76	1.05
	1.10	.76	1.39	1.10	1.06	1.10
11	48	49	51	49	48	48
	.82	.80	.72	.74	.85	.87

TABLE D-III. IMPACT Output - MPP Survey Mean and Variance Results (Continued)

1	2	3	4	5	6
1.02	1.13	1.27	1.03	1.20	.80
544	554	561	542	553	532
.97	.86	.77	.86	.85	.96

TABLE D-III. IMPACT Output - MPP Survey Mean and Variance Results (Continued)

	7	8	9	10	11	12	
	1.27	1.33	1.06	.59	1.48	1.04	1.24
1	51	54	50	51	52	52	627
	.79	.77	.79	.84	.69	.88	.82
	1.15	1.27	1.04	.70	.89	.84	1.13
2	46	51	46	47	47	50	582
	.75	.86	.75	.80	.78	.85	.82
	1.12	1.37	1.02	.55	.88	.47	.99
3	50	51	51	49	51	49	608
	.91	.77	.83	.83	1.00	.99	.97

TABLE D-III. IMPACT Output - MPP Survey Mean and Variance Results (Continued)

	7	8	9	10	11	12	
	1.40	1.35	1.14	.78	.76	1.23	1.21
4	48	51	50	50	50	52	617
	.70	.65	.77	.86	.76	.82	.78
	.94	1.23	1.10	1.21	1.22	1.80	1.17
5	49	52	50	53	51	54	610
	.93	.87	.85	.79	.77	.59	.88
	.16	.40	.42	1.79	.41	1.29	.68
6	50	48	50	53	51	51	607
	.76	.70	.70	.53	.63	.77	.92

TABLE D-III. IMPACT Output - MPP Survey Mean and Variance Results (Continued)

	7	8	9	10	11	12	
	.36	.53	.81	.54	1.88	.59	.85
7	50	47	48	48	51	49	589
	.79	.77	.70	.73	.32	.75	.92
	.58	1.13	1.27	.78	.76	1.21	.90
8	48	48	48	49	46	47	573
	.91	.75	.76	.79	.79	.71	.84
	1.50	1.39	1.24	.31	1.14	1.02	1.20
9	50	49	49	48	51	49	592
	.73	.72	.74	.62	.82	.77	.82

TABLE D-III. IMPACT Output - MPP Survey Mean and Variance Results (Continued)

	7	8	9	10	11	12	
	.48	.87	.89	1.33	.96	1.27	.97
10	44	45	45	46	46	45	545
	.84	.78	.71	.69	.78	.77	.86
	.89	.65	.67	1.16	1.43	1.06	1.04
11	47	46	46	49	49	50	580
	.88	.73	.75	.82	.67	.88	.83

TABLE D-III. IMPACT Output - MPP Survey Mean and Variance Results (Continued)

*	*	*	*	*	*	*	*	*
*	7	8	9	10	11	12		
*	*	*	*	*	*	*	*	*
**********								
*	.90	1.06	.97	.89	1.08	1.08		
*	533	542	533	543	545	548		
*	.93	.84	.80	.87	.84	.88		

	MEAN	SIGMA	O/O POSITIVE	O/O NEGATIVE
COST	.34	.72	49.25	14.93
QUALITY	.81	.50	85.07	4.48

TABLE D-IV. IMPACT Output - MPP Survey Cummulative Results

	1	2	3	4	5	6
1	39 71* 21 40*	28 53*	16 30*	44 81*	19 38*	
	11 20* 17 33*	18 34*	21 40*	9 17*	19 38*	
	5 9* 13 25*	6 11*	15 28*	1 2*	12 24*	
	0 0* 1 2*	1 2*	1 2*	0 0*	0 0*	
	0 0* 0 0*	0 0*	0 0*	0 0*	0 0*	
2	25 51* 25 50*	21 43*	17 35*	27 54*	24 49*	
	18 37* 16 32*	19 39*	14 29*	18 36*	15 31*	
	5 10* 9 18*	9 18*	15 31*	5 10*	10 20*	
	1 2* 0 0*	0 0*	2 4*	0 0*	0 0*	
	0 0* 0 0*	0 0*	0 0*	0 0*	0 0*	
3	23 44* 42 78*	22 42*	15 30*	16 33*	14 27*	
	19 37* 6 11*	21 40*	13 26*	14 29*	18 35*	
	6 12* 2 4*	6 12*	16 32*	15 31*	15 29*	
	3 6* 4 7*	3 6*	6 12*	3 6*	2 4*	
	1 2* 0 0*	0 0*	0 0*	0 0*	2 4*	

TABLE D-IV. IMPACT Output - MPP Survey Cummulative Results (Continued)

	1	2	3	4	5	6
	32 62*	29 54*	20 37*	19 36*	32 62*	18 35*
	15 29*	20 37*	27 50*	16 30*	17 33*	24 47*
4	4 8*	4 7*	7 13*	18 34*	2 4*	8 16*
	1 2*	1 2*	0 0*	0 0*	1 2*	1 2*
	0 0*	0 0*	0 0*	0 0*	0 0*	0 0*
	11 22*	46 85*	12 24*	30 59*	13 26*	8 17*
	20 41*	6 11*	19 38*	17 33*	18 36*	19 40*
5	12 24*	1 2*	17 34*	4 8*	18 36*	16 34*
	6 12*	1 2*	2 4*	0 0*	1 2*	4 9*
	0 0*	0 0*	0 0*	0 0*	0 0*	0 0*
	1 2*	0 0*	22 42*	34 65*	1 2*	2 4*
	23 47*	12 25*	26 49*	13 25*	9 18*	14 27*
6	15 31*	35 73*	3 6*	2 4*	39 76*	21 41*
	10 20*	1 2*	2 4*	3 6*	2 4*	13 25*
	0 0*	0 0*	0 0*	0 0*	0 0*	1 2*

TABLE D-IV. IMPACT Output - MPP Survey Cummulative Results (Continued)

	1	2	3	4	5	6
	4 8*	5 10*	44 85*	9 18*	36 71*	6 13*
	17 35*	20 42*	6 12*	9 18*	13 25*	15 32*
7	16 33*	23 48*	2 4*	31 63*	2 4*	20 43*
	12 24*	0 0*	0 0*	0 0*	0 0*	4 9*
	0 0*	0 0*	0 0*	0 0*	0 0*	2 4*
	7 15*	20 41*	18 36*	18 38*	9 19*	3 6*
	20 43*	15 31*	20 40*	18 38*	14 29*	17 36*
8	14 30*	14 29*	12 24*	11 23*	25 52*	23 49*
	5 11*	0 0*	0 0*	0 0*	0 0*	4 9*
	0 0*	0 0*	0 0*	0 0*	0 0*	0 0*
	40 78*	18 37*	17 35*	12 25*	41 77*	24 51*
	5 10*	15 31*	19 40*	17 35*	12 23*	12 26*
9	5 10*	16 33*	12 25*	19 40*	0 0*	11 23*
	1 2*	0 0*	0 0*	0 0*	0 0*	0 0*
	0 0*	0 0*	0 0*	0 0*	0 0*	0 0*

TABLE D-IV. IMPACT Output - MPP Survey Cummulative Results (Continued)

		1		2		3		4		5		6
	9	20	21	45	22	45	9	21	20	42	6	14
	18	41	14	30	20	41	21	50	18	37	18	41
10	11	25	12	26	7	14	11	26	10	21	11	25
	5	11	0	0	0	0	1	2	0	0	7	16
	1	2	0	0	0	0	0	0	0	0	2	5
	17	35	11	22	26	51	16	33	19	40	19	40
	21	44	15	31	20	39	22	45	13	27	17	35
11	8	17	23	47	4	8	11	22	16	33	10	21
	2	4	0	0	1	2	0	0	0	0	2	4
	0	0	0	0	0	0	0	0	0	0	0	0
	208	38	238	43	252	45	195	36	258	47	143	27
	187	34	156	28	215	38	181	33	155	28	188	35
	101	19	152	27	85	15	153	28	133	24	157	30
	46	8	8	1	9	2	13	2	7	1	37	7
	2	**	0	0	0	0	0	0	0	0	7	1

TABLE D-IV. IMPACT Output - MPP Survey Cummulative Results (Continued)

	7	8	9	10	11	12	
	24 47*	27 50*	17 34*	9 18*	30 58*	18 35*	292 47*
	18 35*	19 35*	19 38*	14 27*	18 35*	20 38*	203 32*
1	8 16*	7 13*	14 28*	27 53*	3 6*	13 25*	124 20*
	1 2*	1 2*	0 0*	0 0*	1 2*	0 0*	6 **
	0 0*	0 0*	0 0*	1 2*	0 0*	1 2*	2 **
	17 37*	25 49*	14 30*	10 21*	12 26*	14 28*	231 40*
	19 41*	17 33*	20 43*	13 28*	18 38*	15 30*	202 35*
2	10 22*	8 16*	12 26*	24 51*	17 36*	20 40*	144 25*
	0 0*	0 0*	0 0*	0 0*	0 0*	1 2*	4 **
	0 0*	1 2*	0 0*	0 0*	0 0*	0 0*	1 **
	20 40*	26 51*	15 29*	7 14*	16 31*	8 16*	224 37*
	20 40*	20 39*	24 47*	17 35*	19 37*	15 31*	206 34*
3	6 12*	3 6*	11 22*	21 43*	11 22*	20 41*	132 22*
	4 8*	2 4*	0 0*	4 8*	4 8*	4 8*	39 6*
	0 0*	0 0*	1 2*	0 0*	1 2*	2 4*	7 1*

TABLE D-IV. IMPACT Output - MPP Survey Cummulative Results (Continued)

```
      *      *      *      *      *      *      *      *
      *   7  *   8  *   9  *  10  *  11  *  12  *
      *      *      *      *      *      *      *      *
*************************************************************
    * 25 52* 23 45* 19 38* 13 26* 10 20* 22 42*262 42*
    * 17 35* 23 45* 19 38* 14 28* 18 36* 22 42*232 38*
  4 *  6 13*  5 10* 12 24* 22 44* 22 44*  7 13*117 19*
    *  0  0*  0  0*  0  0*  1  2*  0  0*  0  0*  5  **
    *  0  0*  0  0*  0  0*  0  0*  0  0*  1  2*  1  **
*************************************************************
    * 15 31* 26 50* 20 40* 21 40* 21 41* 47 87*270 44*
    * 20 41* 13 25* 16 32* 24 45* 21 41*  4  7*197 32*
  5 * 11 22* 12 23* 13 26*  6 11*  8 16*  2  4*120 20*
    *  2  4*  1  2*  1  2*  2  4*  1  2*  1  2* 22  4*
    *  1  2*  0  0*  0  0*  0  0*  0  0*  0  0*  1  **
*************************************************************
    *  1  2*  5 10*  4  8* 44 83*  4  8* 24 47*142 23*
    * 16 32* 10 21* 15 30*  8 15* 13 25* 19 37*178 29*
  6 * 23 46* 32 67* 29 58*  0  0* 34 67*  7 14*240 40*
    * 10 20*  1  2*  2  4*  1  2*  0  0*  1  2* 46  8*
    *  0  0*  0  0*  0  0*  0  0*  0  0*  0  0*  1  **
```

TABLE D-IV. IMPACT Output - MPP Survey Cummulative Results (Continued)

	7		8		9		10		11		12					
	2	4	7	15	8	17	7	15	45	88	8	16	181	31		
	22	44	12	26	23	48	12	25	6	12	13	27	168	29		
7	18	36	27	57	17	35	29	60	0	0	28	57	213	36		
	8	16	1	2	0	0	0	0	0	0	0	0	25	4		
	0	0	0	0	0	0	0	0	0	0	0	0	2	**		
	9	19	17	35	22	46	11	22	10	22	18	38	162	28		
	15	31	20	42	17	35	16	33	15	33	21	45	208	36		
8	19	40	11	23	9	19	22	45	21	46	8	17	189	33		
	5	10	0	0	0	0	0	0	0	0	0	0	14	2		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	32	64	26	53	21	43	4	8	21	41	15	31	271	46		
	11	22	16	33	19	39	7	15	16	31	20	41	169	29		
9	7	14	7	14	9	18	37	77	14	27	14	29	151	26		
	0	0	0	0	0	0	0	0	0	0	0	0	1	**		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

TABLE D-IV. IMPACT Output - MPP Survey Cummulative Results (Continued)

	7	8	9	10	11	12	
	5 11	11 24	9 20	21 46	13 26	20 44	166 30
	16 36	17 38	22 49	19 41	18 39	18 40	219 40
10	18 41	17 38	14 31	6 13	15 33	6 13	138 25
	5 11	0 0	0 0	0 0	0 0	1 2	19 3
	0 0	0 0	0 0	0 0	0 0	0 0	3 *
	14 30	7 15	7 15	20 41	26 53	21 42	203 35
	16 34	16 35	18 39	18 37	18 37	11 22	205 35
11	15 32	23 50	20 43	10 20	5 10	18 36	163 28
	2 4	0 0	1 2	1 2	0 0	0 0	9 2
	0 0	0 0	0 0	0 0	0 0	0 0	0 0

TABLE D-IV. IMPACT Output - MPP Survey Cummulative Results (Continued)

```
*      *      *      *      *      *      *      *
*   7  *   8  *   9  *  10  *  11  *  12  *
*      *      *      *      *      *      *      *
**********************************************************
*164 31*200 37*156 29*167 31*208 38*215 39**04
*190 36*183 34*212 40*162 30*180 33*178 32**87
*141 26*152 28*160 30*204 38*150 28*143 26**31
* 37  7*  6  1*  4  **  9  2*  6  1*  8  1*190
*  1  **  1  **  1  **  1  **  1  **  4  ** 18
```

TABLE D-V. IMPACT Output - Assertion Strength/Influence Ratings

	1	2	3	4	5	6
1	SSP	SP	SP	MP	SSP	SP
2	SP	SP	SP	MP	SP	SP
3	SP	SSP	SP	MP	MP	MP
4	SSP	SP	SP	MP	SP	SP
5	SP	SSP	MP	SP	MP	MP
6	MI	WP	SP	SSP	IND	MI
7	MI	MP	SSP	WP	SSP	MP
8	MP	MP	SP	SP	WP	MP

TABLE D-V. IMPACT Output - Assertion Strength/Influence Ratings (Continued)

9	*	SSP	*	MP	*	SP	*	MP	*	SSP	*	SSP	*
10	*	SP	*	MP	*	SP	*	MP	*	SP	*	S1	*
11	*	SP	*	MP	*	SP	*	SP	*	MP	*	SP	*
	*	SP	*	MP	*	SP	*	MP	*	SP	*	MP	*

TABLE D-V. IMPACT Output - Assertion Strength/Influence Ratings (Continued)

	7	8	9	10	11	12	
1	SP	SP	MP	WP	SP	SP	SP
2	SP	SP	MP	WP	MP	MP	SP
3	SP	SP	SP	MP	SP	MP	SP
4	SP	SP	SP	MP	MP	SP	SP
5	SP	SSP	MP	SP	SP	SSP	SP
6	MI	WP	WP	SSP	WP	SP	MP
7	MP	WP	MP	WP	SSP	WP	MP
8	MP	SP	SP	MP	MP	SP	MP

TABLE D-V. IMPACT Output - Assertion Strength/Influence Ratings (Continued)

9	SSP		SP		SP		IND		MP		MP		MP	
10	MP		MP		MP		SP		MP		SP		MP	
11	MP		MP		MP		SP		SP		MP		MP	
	MP		MP		MP		MP		MP		MP		MP	

TABLE D-V. IMPACT Output - Assertion Strength/Influence Ratings (Continued)

SSP = 16	SP = 54	IND = 2	SN = 0	SSN = 0	SI = 1
	MP = 44		MN = 0		MI = 4
	WP = 11		WN = 0		WI = 0

TABLE D-VI. IMPACT Source Listing

```
      PROGRAM IMPACT (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE7)
      DIMENSION IRATE(30,18,70) , ISR(30,18) , RB(30,18) , RTS(18)
      DIMENSION RBTS(18) , RTC(30) , RBTC(30) , SIG(30,18) , SIGS(18)
      DIMENSION SIGC(30) , IC(70) , IQ(70) , NT(30,18) , NTS(18)
      DIMENSION ISR2(30,18) , RB2(30,18) , RTC2(30) , RTS2(18) , NTC(30)
      DIMENSION N2P(30,18) , N1P(30,18) , NC(30,18) , N1N(30,18)
      DIMENSION N2N(30,18) , NAME(70) , IPRINT(4) , P2N(30,18)
      DIMENSION P2P(30,18) , P1P(30,18) , PC(30,18) , P1N(30,18)
      DIMENSION NTP(30,18) , NTN(30,18) , IHYP(30) , IA(2) , IB(2)
      DIMENSION N2PC(30) , N1PC(30) , NOC(30) , N1NC(30) , N2NC(30)
      DIMENSION N2PS(18) , N1PS(18) , NOS(18) , N1NS(18) , N2NS(18)
      DIMENSION P2PC(30) , P1PC(30) , POC(30) , P1NC(30) , P2NC(30)
      DIMENSION P2PS(18) , P1PS(18) , POS(18) , P1NS(18) , P2NS(18)
      DIMENSION NTPC(30) , NTNC(30) , NTPS(18) , NTNS(18) , IHYPC(30)
      NAMELIST / INPUT / IRATE , NP , NC , NS , IC , IQ , NAME , ITAPE ,
     1 IPRINT
      DATA NP / 70 / , NC / 30 / , NS / 18 /
*
*     INITIALIZE ALL SURVEYS TO BLANKS
*
      DO 10 K = 1 , 70
      DO 10 J = 1 , 18
      DO 10 I = 1 , 30
      IRATE(I,J,K) = 3
      IC(K) = 3
      IQ(K) = 3
   10 CONTINUE
*
*     READ DATA - IRATE(I,J,K) = AN INDIVIDUAL RANKING
*                 NP = NO. OF SURVEYS (MAX. DEFAULT = 70)
*                 NC = NO. OF CHARACTERISTICS (MAX. DEFAULT = 30)
*                 NS = NO. OF STANDARDS (MAX. DEFAULT = 18)
*                 IC = COST QUESTION ANSWER
*                 IQ = QUALITY QUESTION ANSWER
*                 NAME = PERSON NAME/ROOM LOCATION
*                 ITAPE = TAPE7 WRITE FLAG
*                 IPRINT = PRINT OPTION FLAG
*
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      READ (5,INPUT)
*
*     INITIALIZE DATA
*
      DO 70 I = 1 , NC
      DO 60 J = 1 , NS
      NP = 0
      ISR(I,J) = 0
      ISR2(I,J) = 0
      NO(I,J) = 0
      N1P(I,J) = 0
      N2P(I,J) = 0
      N2N(I,J) = 0
      N1N(I,J) = 0
      RB(I,J) = 3.
*
*     LOOP EXAMINING AN INDIVIDUAL RANKING FOR EACH SURVEY
*
      DO 40 K = 1 , NP
*
*     DETERMINE IF THIS RANK IS BLANK
*
      IF ( IRATE(I,J,K) - 3 ) 20 , 30 , 30
*
*     COMPUTE - ISR(I,J) = SUM OF RANKS
*               ISR2(I,J) = SUM OF RANKS SQUARED
*
   20 ISR(I,J) = IRATE(I,J,K) + ISR(I,J)
      ISR2(I,J) = IRATE(I,J,K) * IRATE(I,J,K) + ISR2(I,J)
*
*     TALLY THE VALUE OF THIS RANK
*
      IF ( IRATE(I,J,K) ) 22 , 21
   21 NO(I,J) = NO(I,J) + 1
      GO TO 40
   22 IF ( IRATE(I,J,K) - 1 ) 24 , 23
   23 N1P(I,J) = N1P(I,J) + 1
      GO TO 40
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      24 IF ( IRATE(I,J,K) - 2 ) 26 , 25
      25 N2P(I,J) = N2P(I,J) + 1
         GO TO 40
      26 IF ( IRATE(I,J,K) + 1 ) 28 , 27
      27 N1N(I,J) = N1N(I,J) + 1
         GO TO 40
      28 N2N(I,J) = N2N(I,J) + 1
         GO TO 40
      30 NB = NB + 1
      40 CONTINUE
         IF ( NB - NP ) 50 , 60
*
*     COMPUTE - NT(I,J) = THE NUMBER OF NON-BLANK SURVEYS FOR THIS RANK
*               RB(I,J) = MEAN FOR THIS RANK
*               RB2(I,J) = MEAN SQUARED
*               P2P(I,J) -- P0(I,J) = PERCENTAGES OF EACH TALLY
*               NTP(I,J) = NO. OF POSITIVE TALLIES
*               NTN(I,J) = NO. OF NEGATIVE TALLIES
*
      50 NT(I,J) = NP - NB
         RB(I,J) = FLOAT( ISR(I,J) ) / ( NT(I,J) )
         RB2(I,J) = FLOAT( ISR2(I,J) ) / NT(I,J)
         P2P(I,J) = N2P(I,J) * 100. / NT(I,J)
         P1P(I,J) = N1P(I,J) * 100. / NT(I,J)
         P0(I,J) = N0(I,J) * 100. / NT(I,J)
         P1N(I,J) = N1N(I,J) * 100. / NT(I,J)
         P2N(I,J) = N2N(I,J) * 100. / NT(I,J)
         NTP(I,J) = N2P(I,J) + N1P(I,J)
         NTN(I,J) = N2N(I,J) + N1N(I,J)
      60 CONTINUE
      70 CONTINUE
         DO 98 K=1,NP
*
*     DETERMINE IF SURVEY INPUT DATA SHOULD BE PRINTED
*
         IF ( IPRINT(1) ) 72 , 100
      72 CONTINUE
*
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
115      *      PRINT EACH INDIVIDUAL SURVEY
116      *
117             WRITE (6,73)
118          73 FORMAT (1H1,/)
119      *
120      *      DETERMINE RESULTS FOR PRINT OF COST AND QUALITY QUESTIONS
121      *
122             IF ( IC(K) ) 74 , 76 , 78
123          74 IB(1) = 6H FALSE
124             IB(2) = 6H
125             GO TO 80
126          76 IB(1) = 6H DO NO
127             IB(2) = 6HT KNOW
128             GO TO 80
129          78 IB(1) = 6H TRUE
130             IB(2) = 6H
131          80 CONTINUE
132             IF ( IQ(K) ) 81 , 82 , 83
133          81 IA(1) = 6H FALSE
134             IA(2) = 6H
135             GO TO 84
136          82 IA(1) = 6H DO NO
137             IA(2) = 6HT KNOW
138             GO TO 84
139          83 IA(1) = 6H TRUE
140             IA(2) = 6H
141          84 CONTINUE
142             WRITE (6,85) K , NAME(K)
143          85 FORMAT (1HO,40X,*PERSON NUMBER*,I4,5X,A7,///)
144             WRITE (6,89) (J,J=1,30)
145          89 FORMAT (1HO,10X,30(1X,I2))
146             WRITE (6,90)
147          90 FORMAT (1HO, 7X,31(2X,1H*))
148             DO 94 J=1,NS
149             WRITE (6,92) J , (IRATE(I,J,K),I=1,NC)
150          92 FORMAT (1HO,5X,I3,1X,1H*,30I3)
151          94 CONTINUE
152             WRITE (6,95)
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
153          95 FORMAT (///,1HO, 6X,25HRESPONSE TO QUESTION A. -,45X,
154            1 16HINFLUENCE RATING)
155             WRITE (6,96) IA(1) , IA(2)
156          96 FORMAT (1HO,31X,17HHIGHER QUALITY - ,2A6,10X,
157            1 18H2  STRONG POSITIVE,/, 71X,16H1  SOME POSITIVE)
158             WRITE (6,97) IB(1),IB(2)
159          97 FORMAT (32X,12HLOWER COST -,5X,2A6,10X,
160            1 30HO  INDIFFERENT (INSIGNIFICANT),/, 70X,
161            217H-1  SOME NEGATIVE,/, 70X,19H-2  STRONG NEGATIVE,/, 71X,
162            323H3  (BLANK) - UNFAMILIAR)
163          98 CONTINUE
164         100 CONTINUE
165      *
166      *      INITIALIZE CHARACTERISTIC COLUMN DATA
167      *
168             DO 180 I = 1 , NC
169             RTC(I) = 0.
170             RTC2(I) = 0.
171             NTC(I) = 0
172             N2PC(I) = 0
173             N1PC(I) = 0
174             NCC(I) = 0
175             N1NC(I) = 0
176             N2NC(I) = 0
177             DO 160 J = 1 , NS
178             SIG(I,J) = 0.
179             NB = 0
180             DO 120 K = 1 , NF
181             IF ( IRATE(I,J,K) - 3 ) 105 , 110 , 110
182      *
183      *      COMPUTE SIG(I,J) = PARTIAL COMPUTATION OF SIGMA FOR THIS RANK
184      *
185         105 SIG(I,J) = SIG(I,J) + ( IRATE(I,J,K) - RB(I,J) ) ** 2
186             GO TO 120
187         110 NB = NB + 1
188         120 CONTINUE
189      *
190      *      COMPUTE RTC(I) = SUM OF RANKS
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
*               RTC2(I) = SUM OF RANKS SQUARED
*               NTC(I) = NUMBER OF NON-BLANK SURVEYS
*               N2PC(I) -- N2NC(I) = NO. OF TALLIES
*
      RTC(I) = RB(I,J) * NT(I,J) + RTC(I)
      RTC2(I) = RB2(I,J) * NT(I,J) + RTC2(I)
      NTC(I) = NT(I,J) + NTC(I)
      N2PC(I) = N2P(I,J) + N2PC(I)
      N1PC(I) = N1P(I,J) + N1PC(I)
      NOC(I) = NO(I,J) + NOC(I)
      N1NC(I) = N1N(I,J) + N1NC(I)
      N2NC(I) = N2N(I,J) + N2NC(I)
      IW = 6
      IF ( NB - NP ) 130 , 140
*
*     RECOMPUTE SIG(I,J) = SIGMA FOR THIS RANK
*
  130 SIG(I,J) = SQRT ( SIG(I,J) / NT(I,J) )
      GO TO 160
  140 SIG(I,J) = 3.
  160 CONTINUE
*
*     COMPUTE P2PC(I) -- P2NC(I) = PERCENTAGE OF CHARACTERISTIC
*                                  COLUMN TALLY DATA
*               RBTC(I) = MEAN RANK
*               RTC2(I) = MEAN RANK SQUARED
*               NTPC(I) = NO. OF POSITIVE TALLIES
*               NTNC(I) = NO. OF NEGATIVE TALLIES
*               N2PT -- N2NT = OVERALL TOTAL SURVEY TALLY DATA
*               NTT = TOTAL NO. OF VALID DATA POINTS IN THE SURVEY
*               NTPT = TOTAL NO. OF POSITIVE DATA POINTS
*               NTNT = TOTAL NO. OF NEGATIVE DATA POINTS
*
      P2PC(I) = N2PC(I) * 100. / NTC(I)
      P1PC(I) = N1PC(I) * 100. / NTC(I)
      POC(I) = NOC(I) * 100. / NTC(I)
      P1NC(I) = N1NC(I) * 100. / NTC(I)
      P2NC(I) = N2NC(I) * 100. / NTC(I)
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      RBTC(I) = RTC(I) / NTC(I)
      RTC2(I) = RTC2(I) / NTC(I)
      NTPC(I) = N2PC(I) + N1PC(I)
      NTNC(I) = N2NC(I) + N1NC(I)
      N2PT = N2PT + N2PC(I)
      N1PT = N1PT + N1PC(I)
      NOT = NOT + NOC(I)
      N1NT = N1NT + N1NC(I)
      N2NT = N2NT + N2NC(I)
      NTT = NTT + NTC(I)
      NTPT = NTPT + NTPC(I)
      NTNT = NTNT + NTNC(I)
  180 CONTINUE
*
*     COMPUTE P2PT -- P2NT = PERCENTAGE OF TOTAL SURVEY TALLY DATA
*
      P2PT = N2PT * 100. / NTT
      P1PT = N1PT * 100. / NTT
      POT = NOT * 100. / NTT
      P1NT = N1NT * 100. / NTT
      P2NT = N2NT * 100. / NTT
      DO 240 I = 1 , NC
*
*     COMPUTE SIGC(I) = CHARACTERISTIC COLUMN DATA SIGMA
*
      SIGC(I) = SQRT ( RTC2(I) - ( RBTC(I) * RETC(I) ) )
  240 CONTINUE
*
*     INITIALIZE STANDARD ROW DATA
*
      DO 300 J = 1 , NS
      RTS(J) = 0.
      RTS2(J) = 0.
      NTS(J) = 0
      N2PS(J) = 0
      N1PS(J) = 0
      NOS(J) = 0
      N1NS(J) = 0
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      N2NS(J) = 0
      DO 280 I = 1 , NC
*
*     COMPUTE RTS(J) = SUM OF RANKS
*             RTS2(J) = SUM OF RANKS SQUARED
*             NTS(J) = NO. OF NON-BLANK SURVEYS
*             N2PS(J) -- N2NS(J) = NO. OF TALLIES
*
      RTS(J) = RB(I,J) * NT(I,J) + RTS(J)
      RTS2(J) = RB2(I,J) * NT(I,J) + RTS2(J)
      NTS(J) = NT(I,J) + NTS(J)
      N2PS(J) = N2P(I,J) + N2PS(J)
      N1PS(J) = N1P(I,J) + N1PS(J)
      NOS(J) = NO(I,J) + NOS(J)
      N1NS(J) = N1N(I,J) + N1NS(J)
      N2NS(J) = N2N(I,J) + N2NS(J)
  280 CONTINUE
*
*     COMPUTE P2PS(J) -- P2NS(J) = PERCENTAGE OF STANDARDS ROW
*                                  TALLY DATA
*             RBTS(J) = MEAN RANK
*             RTS2(J) = MEAN RANK SQUARED
*             NTPS(J) = NO. OF POSITIVE TALLIES
*             NTNS(J) = NO. OF NEGATIVE TALLIES
*
      P2PS(J) = N2PS(J) * 100. / NTS(J)
      P1PS(J) = N1PS(J) * 100. / NTS(J)
      POS(J) = NOS(J) * 100. / NTS(J)
      P1NS(J) = N1NS(J) * 100. / NTS(J)
      P2NS(J) = N2NS(J) * 100. / NTS(J)
      RBTS(J) = RTS(J) / NTS(J)
      RTS2(J) = RTS2(J) / NTS(J)
      NTPS(J) = N2PS(J) + N1PS(J)
      NTNS(J) = N2NS(J) + N1NS(J)
  300 CONTINUE
      DO 320 J = 1 , NS
*
*     COMPUTE SIGS(J) = STANDARD ROW DATA SIGMA
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      *
            SIGS(J) = SQRT ( RTS2(J) - ( RBTS(J) * RPTS(J) ) )
      320 CONTINUE
      *
      *     COMPUTE COST AND QUALITY QUESTION DATA
      *             ICN = TALLY OF FALSE COST DATA
      *             ICP = TALLY OF TRUE COST DATA
      *             IQN = TALLY OF FALSE QUALITY DATA
      *             IQP = TALLY OF TRUE QUALITY DATA
      *             ISIC = SUM OF THE COST DATA
      *             ISIQ = SUM OF THE QUALITY DATA
      *
            ICN = 0
            ICP = 0
            IQN = 0
            IQP = 0
            ISIC = 0
            ISIQ = 0
            NB = 0
            DO 380 I = 1 , NF
      *
      *     DETERMINE IF THE COST QUESTION WAS BLANK
      *
            IF ( IC(I) - 3 ) 325 , 365
      325 ISIC = IC(I) + ISIC
            ISIQ = IQ(I) + ISIQ
      *
      *     DETERMINE THE VALUE OF THE COST QUESTION
      *
            IF ( IC(I) ) 330 , 350 , 340
      330 ICN = ICN + 1
            GO TO 350
      340 ICP = ICP + 1
      *
      *     DETERMINE THE VALUE OF THE QUALITY QUESTION
      *
      350 IF ( IQ(I) ) 360 , 380 , 370
      360 IQN = IQN + 1
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
343             GO TO 380
344         365 NB = NB + 1
345             GO TO 380
346         370 IQP = IQP + 1
347         380 CONTINUE
348       *
349       *     COMPUTE N = NO. OF DATA POINTS
350       *             CB = COST MEAN RANK
351       *             QB = QUALITY MEAN RANK
352       *             SIGCB = COST SIGMA
353       *             SIGQB = QUALITY SIGMA
354       *             CNP = PERCENTAGE OF FALSE COST
355       *             CPP = PERCENTAGE OF TRUE COST
356       *             QNP = PERCENTAGE OF FALSE QUALITY
357       *             QPP = PERCENTAGE OF TRUE QUALITY
358       *
359             N = NP - NB
360             CB = FLOAT (ISIC) / N
361             QB = FLOAT (ISIQ) / N
362             SIGCB = 0.
363             SIGQB = 0.
364             DO 390 I = 1 , NP
365             IF ( IC(I) - 3 ) 385 , 390
366         385 SIGCB = SIGCB + ( IC(I) - CB ) ** 2
367             SIGQB = SIGQB + ( IQ(I) - QB ) ** 2
368         390 CONTINUE
369             SIGCB = SQRT ( SIGCB / N )
370             SIGQB = SQRT ( SIGQB / N )
371             CNP = ICN * 100. / N
372             CPP = ICP * 100. / N
373             QNP = IQN * 100. / N
374             QPP = IQP * 100. / N
375       *
376       *     DETERMINE IF MEAN AND SIGMA DATA SHOULD BE PRINTED
377       *
378             IF ( IPRINT(2) ) 395 , 500
379         395 CONTINUE
380             NC2 = NC / 2
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      N1 = NC2 + 1
      NS3 = NS + 2 / 3
      J = 0
      DO 450 K = 1 , NS3
      WRITE (6,400)
  400 FORMAT (1H1)
      WRITE (6,410)
  410 FORMAT (1HO,4X,16(1H*,6X))
      WRITE (6,420) (I,I=1,NC2)
  420 FORMAT (1HO,4X,1H*,15(1X,I3,2X,1H*))
      WRITE (6,410)
      WRITE (6,430)
  430 FORMAT (1HO,110H**********************************************************
     1**********************************************************)
      DO 450 L = 1 , 3
      J = J + 1
      WRITE (6,440) (RE(I,J),I=1,NC2)
  440 FORMAT (1HO,4X,1H*,15(1X,F4.2,1X,1H*),1X,F4.2)
      WRITE (6,445) J , (NT(I,J),I=1,NC2)
  445 FORMAT (1HO,1X,I2,1X,16(1H*,1X,I4,1X))
      WRITE (6,440) (SIG(I,J),I=1,NC2)
      WRITE (6,430)
      IF ( J - NS ) 450 , 455
  450 CONTINUE
  455 CONTINUE
      WRITE (6,400)
      WRITE (6,410)
      WRITE (6,420) (I,I=1,NC2)
      WRITE (6,410)
      WRITE (6,430)
      WRITE (6,440) (RETC(I),I=1,NC2)
      WRITE (6,420) (NTC(I),I=1,NC2)
      WRITE (6,440) (SIGC(I),I=1,NC2)
      J = 0
      DO 460 K = 1 , NS3
      WRITE (6,400)
      WRITE (6,410)
      WRITE (6,420) (I,I=N1,NC)
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      WRITE (6,410)
      WRITE (6,430)
      DO 460 L = 1 , 3
      J = J + 1
      WRITE (6,440) (RE(I,J),I=N1,NC) , RBTS(J)
      WRITE (6,445) J , (NT(I,J),I=N1,NC) , NTS(J)
      WRITE (6,440) (SIG(I,J),I=N1,NC) , SIGS(J)
      WRITE (6,430)
      IF ( J - NS ) 460 , 465
  460 CONTINUE
  465 CONTINUE
      WRITE (6,400)
      WRITE (6,410)
      WRITE (6,420) (I,I=N1,NC)
      WRITE (6,410)
      WRITE (6,430)
      WRITE (6,440) (RETC(I),I=N1,NC)
      WRITE (6,420) (NTC(I),I=N1,NC)
      WRITE (6,440) (SIGC(I),I=N1,NC)
      WRITE (6,470)
  470 FORMAT (///,16X,4HMEAN,5X,5HSIGMA,5X,12HO/O POSITIVE,5X,
     1 12HO/O NEGATIVE)
      WRITE (6,480) CB , SIGCB , CPP , CNP
  480 FORMAT (1HO,4X,4HCOST,5X,F6.2,3X,F6.2,9X,F6.2,11X,F6.2)
      WRITE (6,490) QB , SIGQB , QPP , QNP
  490 FORMAT (1HO,4X,7HQUALITY,2X,F6.2,3X,F6.2,9X,F6.2,11X,F6.2)
  500 CONTINUE
*
*     DETERMINE IF CUMULATIVE SURVEY DATA SHOULD BE PRINTED
*
      IF ( IPRINT(3) ) 505 , 580
  505 CONTINUE
      J = 0
      DO 540 K = 1 , NS3
      WRITE (6,400)
      WRITE (6,507)
  507 FORMAT (5X,16(1H*,6X))
      WRITE (6,420) (I,I=1,NC2)
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      WRITE (6,410)
      DO 540 L = 1 , 3
      J = J + 1
      WRITE (6,430)
      WRITE (6,510) (N2P(I,J),P2P(I,J),I=1,NC2)
  510 FORMAT (1HO,4X,15(1H*,I3,1X,F2.0),1H*,I5,1X,F2.0)
      WRITE (6,510) (N1P(I,J),P1P(I,J),I=1,NC2)
      WRITE (6,520) J , (NO(I,J),PO(I,J),I=1,NC2)
      WRITE (6,510)      (N1N(I,J),P1N(I,J),I=1,NC2)
  520 FORMAT (1HO,1X,I2,1X,15(1H*,I3,1X,F2.0),1H*,I5,1X,F2.0)
      WRITE (6,510) (N2N(I,J),P2N(I,J),I=1,NC2)
      IF ( J - NS ) 540 , 545
  540 CONTINUE
  545 CONTINUE
      WRITE (6,400)
      WRITE (6,410)
      WRITE (6,420) (I,I=1,NC2)
      WRITE (6,410)
      WRITE (6,430)
      WRITE (6,510) (N2PC(I),P2PC(I),I=1,NC2)
      WRITE (6,510) (N1PC(I),P1PC(I),I=1,NC2)
      WRITE (6,510) (NOC(I),POC(I),I=1,NC2)
      WRITE (6,510) (N1NC(I),P1NC(I),I=1,NC2)
      WRITE (6,510) (N2NC(I),P2NC(I),I=1,NC2)
      J = C
      DO 560 K = 1 , NS3
      WRITE (6,400)
      WRITE (6,507)
      WRITE (6,420) (I,I=N1,NC)
      WRITE (6,410)
      DO 560 L = 1 , 3
      J = J + 1
      WRITE (6,430)
      WRITE (6,510) (N2P(I,J),P2P(I,J),I=N1,NC) , N2PS(J) , P2PS(J)
      WRITE (6,510) (N1P(I,J),P1P(I,J),I=N1,NC) , N1PS(J) , P1PS(J)
      WRITE (6,520) J , (NO(I,J),PO(I,J),I=N1,NC) , NOS(J) , POS(J)
      WRITE (6,510)      (N1N(I,J),P1N(I,J),I=N1,NC) , N1NS(J) , P1NS(J)
      WRITE (6,510) (N2N(I,J),P2N(I,J),I=N1,NC) , N2NS(J) , P2NS(J)
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
495           IF ( J - NS ) 560 , 565
496       560 CONTINUE
497       565 CONTINUE
498           WRITE (6,400)
499           WRITE (6,410)
500           WRITE (6,420) (I,I=N1,NC)
501           WRITE (6,410)
502           WRITE (6,430)
503           WRITE (6,510) (N2PC(I),P2PC(I),I=N1,NC) , N2PT
504           WRITE (6,510) (N1PC(I),P1PC(I),I=N1,NC) , N1PT
505           WRITE (6,510) (NCC(I),POC(I),I=N1,NC) , NCT
506           WRITE (6,510) (N1NC(I),P1NC(I),I=N1,NC) , N1NT
507           WRITE (6,510) (N2NC(I),P2NC(I),I=N1,NC) , N2NT
508       580 CONTINUE
509     *
510     *     DETERMINE IF HYPOTHESIS FORMULATION RESULTS SHOULD BE COMPUTED
511     *
512           IF ( IPRINT(4) ) 590 , 4200
513       590 CONTINUE
514           L = NC2
515           K = 1
516           ISSN = 0
517           ISSP = 0
518           ISN = 0
519           ISI = 0
520           ISP = 0
521           IMN = 0
522           IMI = 0
523           IMP = 0
524           IWN = 0
525           IWI = 0
526           IWP = 0
527           IND = 0
528       600 CONTINUE
529           WRITE (6,400)
530           WRITE (6,410)
531           WRITE (6,420) (I,I=K,L)
532           WRITE (6,410)
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      WRITE (6,430)
*
*     ANALYZE THE HYPOTHESIS STRENGTH OF EACH I,J PAIR
*
      DO 3100 J = 1 , NS
      DO 2000 I = K , L
*
*     DETERMINE IF THE CONFIDENCE LEVEL IS STRONG
*
      IF ( NO(I,J) - NT(I,J) * .25 ) 605 , 605 , 700
*
*     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
*
  605 IF ( NTP(I,J) ) 610 , 615
  610 IF ( NTN(I,J) / NTP(I,J) - 3 ) 645 , 615 , 615
*
*     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG NEGATIVE
*
  615 IF ( N1N(I,J) ) 620 , 630
  620 IF ( N2N(I,J) / N1N(I,J) - 2 ) 640 , 630 , 630
  630 IHYP(I) = 3HSSN
      ISSN = ISSN + 1
      GO TO 1000
  640 IHYP(I) = 3H SN
      ISN = ISN + 1
      GO TO 1000
*
*     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
*
  645 IF ( NTN(I,J) ) 650 , 655
  650 IF ( NTP(I,J) / NTN(I,J) - 3 ) 690 , 655 , 655
*
*     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG POSITIVE
*
  655 IF ( N1P(I,J) ) 660 , 670
  660 IF ( N2P(I,J) / N1P(I,J) - 2 ) 680 , 670 , 670
  670 IHYP(I) = 3HSSP
      ISSP = ISSP + 1
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
571             GO TO 1000
572         680 IHYP(I) = 3H SP
573             ISP = ISP + 1
574             GO TO 1000
575       *
576       *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
577       *
578         690 IHYP(I) = 3H SI
579             ISI = ISI + 1
580             GO TO 1000
581       *
582       *     DETERMINE IF THE CONFIDENCE LEVEL IS MEDIUM
583       *
584         700 IF ( NO(I,J) - NT(I,J) * .5 ) 705 , 705 , 800
585       *
586       *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
587       *
588         705 IF ( NTP(I,J) ) 710 , 720
589         710 IF ( NTN(I,J) / NTP(I,J) - 3 ) 725 , 720 , 720
590         720 IHYP(I) = 3H MN
591             IMN = IMN + 1
592             GO TO 1000
593       *
594       *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
595       *
596         725 IF ( NTN(I,J) ) 730 , 740
597         730 IF ( NTP(I,J) / NTN(I,J) - 3 ) 750 , 740 , 740
598         740 IHYP(I) = 3H MP
599             IMP = IMP + 1
600             GO TO 1000
601       *
602       *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
603       *
604         750 IHYP(I) = 3H MI
605             IMI = IMI + 1
606             GO TO 1000
607       *
608       *     DETERMINE IF THE CONFIDENCE LEVEL IS WEAK
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
609      *
610        800 IF ( NO(I,J) - NT(I,J) * .75 ) 805 , 805 , 900
611      *
612      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
613      *
614        805 IF ( NTP(I,J) ) 810 , 820
615        810 IF ( NTN(I,J) / NTP(I,J) - 3 ) 825 , 820 , 820
616        820 IHYP(I) = 3H wN
617            IWN = IWN + 1
618            GO TO 1000
619      *
620      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
621      *
622        825 IF ( NTN(I,J) ) 830 , 840
623        830 IF ( NTP(I,J) / NTN(I,J) - 3 ) 850 , 840 , 840
624        840 IHYP(I) = 3H wP
625            IWP = IWP + 1
626            GO TO 1000
627      *
628      *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
629      *
630        850 IHYP(I) = 3H wI
631            IWI = IWI + 1
632            GO TO 1000
633      *
634      *     THERE IS STRONG CONFIDENCE THAT THE HYPOTHESIS
635      *               STRENGTH IS INDIFFERENT
636      *
637        900 IHYP(I) = 3HIND
638            IND = IND + 1
639       1000 CONTINUE
640      *
641      *     ANALYZE THE HYPOTHESIS STRENGTH OF EACH CHARACTERISTIC COLUMN
642      *     DETERMINE IF THE CONFIDENCE LEVEL IS STRONG
643            IF ( NOC(I) - NTC(I) * .25 )  1205 , 1205 , 1200
644      *
645      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
646      *
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      1205 IF ( NTPC(I) ) 1210 , 1215
      1210 IF ( NTNC(I) / NTPC(I) - 3 ) 1245 , 1215 , 1215
     *
     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG NEGATIVE
     *
      1215 IF ( N1NC(I) ) 1220 , 1230
      1220 IF ( N2NC(I) / N1NC(I) - 2 ) 1240 , 1230 , 1230
      1230 IHYPC(I) = 3HSSN
           GO TO 2000
      1240 IHYPC(I) = 3H SN
           GO TO 2000
     *
     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
     *
      1245 IF ( NTNC(I) ) 1250 , 1255
      1250 IF ( NTPC(I) / NTNC(I) - 3 ) 1290 , 1255 , 1255
     *
     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG POSITIVE
     *
      1255 IF ( N1PC(I) ) 1260 , 1270
      1260 IF ( N2PC(I) / N1PC(I) - 2 ) 1280 , 1270 , 1270
      1270 IHYPC(I) = 3HSSP
           GO TO 2000
      1280 IHYPC(I) = 3H SP
           GO TO 2000
     *
     *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
     *
      1290 IHYPC(I) = 3H SI
           GO TO 2000
     *
     *     DETERMINE IF THE CONFIDENCE LEVEL IS MEDIUM
     *
      1300 IF ( NOC(I) - NTC(I) * .5 ) 1305 , 1305 , 1400
     *
     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
     *
      1305 IF ( NTPC(I) ) 1310 , 1320
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
685       1310 IF ( NTNC(I) / NTPC(I) - 3 ) 1325 , 1320 , 1320
686       1320 IHYPC(I) = 3H MN
687            GO TO 2000
688       *
689       *      DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
690       *
691       1325 IF ( NTNC(I) ) 1330 , 1340
692       1330 IF ( NTPC(I) / NTNC(I) - 3 ) 1350 , 1340 , 1340
693       1340 IHYPC(I) = 3H MP
694            GO TO 2000
695       *
696       *      THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
697       *
698       1350 IHYPC(I) = 3H MI
699            GO TO 2000
700       *
701       *      DETERMINE IF THE CONFIDENCE LEVEL IS WEAK
702       *
703       1400 IF ( NOC(I) - NTC(I) * .75 ) 1405 , 1405 , 1500
704       *
705       *      DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
706       *
707       1405 IF ( NTPC(I) ) 1410 , 1420
708       1410 IF ( NTNC(I) / NTPC(I) - 3 ) 1425 , 1420 , 1420
709       1420 IHYPC(I) = 3H WN
710            GO TO 2000
711       *
712       *      DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
713       *
714       1425 IF ( NTNC(I) ) 1430 , 1440
715       1430 IF ( NTPC(I) / NTNC(I) - 3 ) 1450 , 1440 , 1440
716       1440 IHYPC(I) = 3H WP
717            GO TO 2000
718       *
719       *      THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
720       *
721       1450 IHYPC(I) = 3H WI
722            GO TO 2000
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
723      *
724      *     THERE IS STRONG CONFIDENCE THAT THE HYPOTHESIS
725      *               STRENGTH IS INDIFFERENT
726      *
727       1500 IHYPC(I) = 3HIND
728       2000 CONTINUE
729            IF ( K .EQ. 1 ) 3000 , 2100
730      *
731      *     ANALYZE THE HYPOTHESIS STRENGTH OF EACH STANDARD ROW
732      *
733       2100 CONTINUE
734      *
735      *     DETERMINE IF THE CONFIDENCE LEVEL IS STRONG
736      *
737            IF ( NOS(J) - NTS(J) * .25 ) 2105 , 2105 , 2200
738      *
739      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
740      *
741       2105 IF ( NTPS(J) ) 2110 , 2115
742       2110 IF ( NTNS(J) / NTPS(J) - 3 ) 2145 , 2115 , 2115
743      *
744      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG NEGATIVE
745      *
746       2115 IF ( N1NS(J) ) 2120 , 2130
747       2120 IF ( N2NS(J) / N1NS(J) - 2 ) 2140 , 2130 , 2130
748       2130 IHYPS = 3HSSN
749            GO TO 3000
750       2140 IHYPS = 3H SN
751            GO TO 3000
752      *
753      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
754      *
755       2145 IF ( NTNS(J) ) 2150 , 2155
756       2150 IF ( NTPS(J) / NTNS(J) - 3 ) 2190 , 2155 , 2155
757      *
758      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG POSITIVE
759      *
760       2155 IF ( N1PS(J) ) 2160 , 2170
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
761       2160 IF ( N2PS(J) / N1PS(J) - 2 ) 2180 , 2170 , 2170
762       2170 IHYPS = 3HSSP
763            GO TO 3000
764       2180 IHYPS = 3H SP
765            GO TO 3000
766      *
767      *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
768      *
769       2190 IHYPS = 3H SI
770            GO TO 3000
771      *
772      *     DETERMINE IF THE CONFIDENCE LEVEL IS MEDIUM
773      *
774       2200 IF ( NOS(J) - NTS(J) * .5 ) 2205 , 2205 , 2300
775      *
776      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
777      *
778       2205 IF ( NTPS(J) ) 2210 , 2220
779       2210 IF ( NTNS(J) / NTPS(J) - 3 ) 2225 , 2220 , 2220
780       2220 IHYPS = 3H MN
781            GO TO 3000
782      *
783      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
784      *
785       2225 IF ( NTNS(J) ) 2230 , 2240
786       2230 IF ( NTPS(J) / NTNS(J) - 3 ) 2250 , 2240 , 2240
787       2240 IHYPS = 3H MP
788            GO TO 3000
789      *
790      *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
791      *
792       2250 IHYPS = 3H MI
793            GO TO 3000
794      *
795      *     DETERMINE IF THE CONFIDENCE LEVEL IS WEAK
796      *
797       2300 IF ( NOS(J) - NTS(J) * .75 ) 2305 , 2305 , 2400
798      *
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
      *
 2305 IF ( NTPS(J) ) 2310 , 2320
 2310 IF ( NTNS(J) / NTPS(J) - 3 ) 2325 , 2320 , 2320
 2320 IHYPS = 3H WN
      GO TO 3000
      *
      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
      *
 2325 IF ( NTNS(J) ) 2330 , 2340
 2330 IF ( NTPS(J) / NTNS(J) - 3 ) 2350 , 2340 , 2340
 2340 IHYPS = 3H WP
      GO TO 3000
      *
      *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
      *
 2350 IHYPS = 3H WI
      GO TO 3000
      *
      *     THERE IS STRONG CONFIDENCE THAT THE HYPOTHESIS
      *          STRENGTH IS INDIFFERENT
      *
 2400 IHYPS = 3HIND
 3000 CONTINUE
      IF ( K .EQ. 1 ) IHYPS = 3H
      WRITE (6,3050) J , (IHYP(I),I=K,L) , IHYPS
 3050 FORMAT (1HO,1X,I2,1X,16(1H*,1X,A3,2X))
      WRITE (6,430)
 3100 CONTINUE
      *
      *     DETERMINE THE HYPOTHESIS STRENGTH OF THE SURVEY
      *
      IHYPT = 3H
      IF ( K - NC2 - 1 ) 4000 , 3200
 3200 CONTINUE
      *
      *     DETERMINE IF THE CONFIDENCE LEVEL IS STRONG
      *
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
837            IF ( NOT - NTT * .25 ) 3205 , 3205 , 3300
838      *
839      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
840      *
841       3205 IF ( N1PT ) 3210 , 3215
842       3210 IF ( N1NT / N1PT - 3 ) 3245 , 3215 , 3215
843      *
844      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG NEGATIVE
845      *
846       3215 IF ( N1NT ) 3220 , 3230
847       3220 IF ( N2NT / N1NT - 2 ) 3240 , 3230 , 3230
848       3230 IHYPT = 3HSSN
849            GO TO 4000
850       3240 IHYPT = 3H SN
851            GO TO 4000
852      *
853      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
854      *
855       3245 IF ( N1NT ) 3250 , 3255
856       3250 IF ( N1PT / N1NT - 3 ) 3290 , 3255 , 3255
857      *
858      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG POSITIVE
859      *
860       3255 IF ( N1PT ) 3260 , 3270
861       3260 IF ( N2PT / N1PT - 2 ) 3280 , 3270 , 3270
862       3270 IHYPT = 3HSSP
863            GO TO 4000
864       3280 IHYPT = 3H SP
865            GO TO 4000
866      *
867      *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
868      *
869       3290 IHYPT = 3H SI
870            GO TO 4000
871      *
872      *     DETERMINE IF THE CONFIDENCE LEVEL IS MEDIUM
873      *
874       3300 IF ( NOT - NTT * .5 ) 3305 , 3305 , 3400
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
      *
      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
      *
 3305 IF ( NTPT ) 3310 , 3320
 3310 IF ( NTNT / NTPT - 3 ) 3325 , 3320 , 3320
 3320 IHYPT = 3H MA
      GO TO 4000
      *
      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
      *
 3325 IF ( NTNT ) 3330 , 3340
 3330 IF ( NTPT / NTNT - 3 ) 3350 , 3340 , 3340
 3340 IHYPT = 3H MP
      GO TO 4000
      *
      *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
      *
 3350 IHYPT = 3H MI
      GO TO 4000
      *
      *     DETERMINE IF THE CONFIDENCE LEVEL IS WEAK
      *
 3400 IF ( NOT - NTT * .75 ) 3405 , 3405 , 3500
      *
      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
      *
 3405 IF ( NTPT ) 3410 , 3420
 3410 IF ( NTNT / NTPT - 3 ) 3425 , 3420 , 3420
 3420 IHYPT = 3H WA
      GO TO 4000
      *
      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
      *
 3425 IF ( NTNT ) 3430 , 3440
 3430 IF ( NTPT / NTNT - 3 ) 3450 , 3440 , 3440
 3440 IHYPT = 3H WP
      GO TO 4000
      *
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
913      *      THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
914      *
915       3450 IHYPT = 3H WI
916            GO TO 4000
917      *
918      *      THERE IS STRONG CONFIDENCE THAT THE HYPOTHESIS
919      *                STRENGTH IS INDIFFERENT
920      *
921       3500 IHYPT = 3HIND
922       4000 CONTINUE
923            WRITE (6,4110) (IHYPC(I),I=K,L) , IHYPT
924       4110 FORMAT (1HO,4X,16(1H*,1X,A3,2X))
925            IF ( L - NC2 ) 4150 , 4120
926       4120 L = NC
927            K = N1
928            GO TO 600
929       4150 CONTINUE
930            WRITE (6,73)
931            WRITE (6,4170) ISSP , ISP , IND , ISN , ISSN , ISI
932       4170 FORMAT (///,4X,5HSSP =,I3,3X,5HSP = ,I3,3X,5HIND =,I3,3X,5HSN = ,
933           1 I3,3X,5HSSN =,I3,3X,5HSI = ,I3,/)
934            WRITE (6,4180) IMP , IMN , IMI
935       4180 FORMAT (1HO,14X,5HMP = ,I3,14X,5HMN = ,I3,14X,5HMI = ,I3,/)
936            WRITE (6,4190) IWP , IWN , IWI
937       4190 FORMAT (1HO,14X,5HWP = ,I3,14X,5HWN = ,I3,14X,5HWI = ,I3,/)
938       4200 CONTINUE
939      *
940      *      DETERMINE IF TAPE7 SHOULD BE WRITTEN WITH THE RANK TALLY
941      *               AND PERCENTAGE DATA
942      *
943            IF ( ITAPE ) 5000 , 6000
944       5000 WRITE (7,5030) NC , NS
945       5030 FORMAT (2I2)
946            DO 5100 J = 1 , NS
947            WRITE (7,5050) (N2P(I,J),P2P(I,J),I=1,NC2)
948            WRITE (7,5050) (N1P(I,J),P1P(I,J),I=1,NC2)
949            WRITE (7,5050) (NO(I,J),PO(I,J),I=1,NC2)
950            WRITE (7,5050) (N1N(I,J),P1N(I,J),I=1,NC2)
```

TABLE D-VI. IMPACT Source Listing (Continued)

```
951            WRITE (7,5050) (M2N(I,J),P2N(I,J),I=1,NC2)
952            WRITE (7,5050) (M2P(I,J),P2P(I,J),I=N1,NC)
953            WRITE (7,5050) (M1P(I,J),P1P(I,J),I=N1,NC)
954            WRITE (7,5050) (M0(I,J),P0(I,J),I=N1,NC)
955            WRITE (7,5050) (M1N(I,J),P1N(I,J),I=N1,NC)
956            WRITE (7,5050) (M2N(I,J),P2N(I,J),I=N1,NC)
957       5050 FORMAT (15(I3,F6.2))
958       5100 CONTINUE
959            WRITE (6,5150)
960       5150 FORMAT (1H1,3X,17HTAPE7 WAS WRITTEN)
961       6000 CONTINUE
962            STOP
963            END
```

D.2.5 Listing

The IMPACT source listing is presented in Table D-VI.

D.3 MERIT

D.3.1 Purpose

The MERIT program was required to analyze the effect of non-uniform weighting of the response-frequency-distributions (see D.2.3) in computation of summary measures of assertion strength/influence rating for individual MPP and selected MPP in combination.

D.3.2 Input

MERIT accepts the response-frequency-distributions from the IMPACT program. MERIT also accepts a list of row weights (0 or 1 to indicate which rows are to be included in the computation) and column weights (between 1 and 99 to indicate the appropriate factor for each column of distributions) used in computing the weighted summary response-frequency-distributions.

D.3.3 Function

MERIT performs the distributional algorithm described in Appendix C. MERIT uses the input row and column weights to factor each of the response-frequency-distributions from the IMPACT program. MERIT totals the factored distributions across each row and down each column to obtain weighted summary response-frequency-distributions for each row and column and adds the column summary distributions to obtain a weighted summary response-frequency-distribution for the complete matrix. The hypothesis test (distributional algorithm) is performed in deriving a composite assertion strength/influence rating (i.e., a figure of merit) from each of the summary distributions.

The MERIT program was designed for interactive use, permitting the analyst to input varying sets of row and column weights to investigate the figure of merit for various combinations of MPP in the context of different software development activities.

D.3.4 Output

MERIT output consists of the summary response-frequency-distributions with corresponding percentages and the composite assertion strength/influence ratings (figures of merit). Examples of output are contained in Appendix E. Examples of MERIT output (MPP theoretical results) are presented in the section following.

D.3.4.1 MERIT Output - Unweighted

MERIT obtains the response-frequency-distributions generated by IMPACT and applies weighting to the distributions thus perturbing the row and column summary distributions, and the summary response-frequency-distribution for the complete matrix. The output presented in Table D-VII presents the row and column summary data from MERIT uniform weights (i.e., all row weights and column weights equal to one). The output consists of:

The row/column ("Practice"/"Problem") indices

The row/column identification under the label "Characteristic" and "Practice"

The weight applied to that particular row/column

The summary assertion strength/influence rating for that particular row/column after application of the weights

The summary response-frequency-distribution and percentage for that particular row/column after application of the weights.

Located on the last page of Table D-VII is the weighted result for the complete matrix, i.e., the "Composite Figure of Merit."

D.3.4.2 MERIT Output - Weighted

Table D-VIII shows the results of applying varied weights to the data. Note the changes in the response-frequency-distributions and the assertion strength/influence rating for practice number 9, and for the composite figure of merit. Another example of weighting with MERIT is discussed in Section 4 and the MERIT output is in Appendix E.

TABLE D-VII. MERIT Output - Problem Unweighted

INDEX	CHARACTERISTIC	WEIGHT	ASSERTION STRENGTH/ INFLUENCE RATING	RANK	FREQ.	PCT.
1	COST OVERRUN	1	STRONG/POSITIVE	+2	208	38.2
				+1	187	34.4
				0	101	18.6
				-1	46	8.5
				-2	2	.4
2	DEVELOP-MENT STATUS INVISIBIL-ITY	1	MEDIUM/POSITIVE	+2	238	43.0
				+1	156	28.2
				0	152	27.4
				-1	8	1.4
				-2	0	0.0
3	UNRELI-ABILITY	1	STRONG/POSITIVE	+2	252	44.9
				+1	215	38.3
				0	85	15.2
				-1	9	1.6
				-2	0	0.0
4	UNMAIN-TAIN-ABILITY	1	MEDIUM/POSITIVE	+2	195	36.0
				+1	181	33.4
				0	153	28.2
				-1	13	2.4
				-2	0	0.0
5	INADE-QUATE SAT-ISFACTION OF REAL REQUIREMEN	1	STRONG/POSITIVE	+2	258	46.7
				+1	155	28.0
				0	133	24.1
				-1	7	1.3
				-2	0	0.0
6	INEFFI-CIENT USE OF RESOURCES	1	MEDIUM/POSITIVE	+2	143	26.9
				+1	188	35.3
				0	157	29.5
				-1	37	7.0
				-2	7	1.3

TABLE D-VII. MERIT Output - Problem Unweighted (Continued)

7	SCHEDULE OVERRUN	1	MEDIUM/POSITIVE	+2	164	30.8
				+1	190	35.6
				0	141	26.5
				-1	37	6.9
				-2	1	.2
8	INADEQUATE PLANNING AND CONTROL	1	MEDIUM/POSITIVE	+2	200	36.9
				+1	183	33.8
				0	152	28.0
				-1	6	1.1
				-2	1	.2
9	PROJECT MISMANAGE-MENT	1	MEDIUM/POSITIVE	+2	156	29.3
				+1	212	39.8
				0	160	30.0
				-1	4	.8
				-2	1	.2
10	LACK OF PROGRAM-MING DISCIPLINE	1	MEDIUM/POSITIVE	+2	167	30.8
				+1	162	29.8
				0	204	37.6
				-1	9	1.7
				-2	1	.2
11	LACK OF CONCLUSIVE TESTING	1	MEDIUM/POSITIVE	+2	208	38.2
				+1	180	33.0
				0	150	27.5
				-1	6	1.1
				-2	1	.2
12	POOR DOCUMENTA-TION	1	MEDIUM/POSITIVE	+2	215	39.2
				+1	178	32.5
				0	143	26.1
				-1	8	1.5
				-2	4	.7

TABLE D-VII. MERIT Output - Problem Unweighted (Continued)

INDEX	PRACTICE	WEIGHT	ASSERTION STRENGTH/ INFLUENCE RATING	RANK	FREQ.	PCT.
1	REQUIRE-	1	STRONG/POSITIVE	+2	292	46.6
	MENTS			+1	203	32.4
	ANALYSIS			0	124	19.8
	AND			-1	6	1.0
	VALIDATION			-2	2	.3
2	PROCESS	1	STRONG/POSITIVE	+2	231	39.7
	DESIGN			+1	202	34.7
				0	144	24.7
				-1	4	.7
				-2	1	.2
3	INCREMEN-	1	STRONG/POSITIVE	+2	224	36.5
	TAL			+1	205	33.9
	DEVELOP-			0	132	21.7
	MENT			-1	34	0.4
				-2	7	1.2
4	COMPLETE	1	STRONG/POSITIVE	+2	262	42.9
	PRELIMI-			+1	232	37.6
	NARY			0	117	19.0
	DESIGN			-1	5	.8
				-2	1	.2
5	UNIT	1	STRONG/POSITIVE	+2	270	44.3
	DEVELOP-			+1	197	32.3
	MENT			0	120	19.7
	FOLDERS			-1	22	3.6
				-2	1	.2
6	ENFORCED	1	MEDIUM/POSITIVE	+2	142	23.4
	PROGRAM-			+1	178	29.3
	MING			0	240	39.5
	STANDARDS			-1	46	7.6
				-2	1	.2

TABLE D-VII. MERIT Output - Problem Unweighted (Continued)

7	INDEPEND-ENT TESTING	1	MEDIUM/POSITIVE	+2	181	30.7
				+1	168	28.5
				0	215	36.4
				-1	25	4.2
				-2	2	.3
8	SOFTWARE CONFIG. MANAGEMENT	1	MEDIUM/POSITIVE	+2	162	28.3
				+1	208	36.3
				0	189	33.0
				-1	14	2.4
				-2	0	0.0
9	BASELINING OF RE-QUIREMENTS SPECIFICA-TION	1	MEDIUM/POSITIVE	+2	271	45.8
				+1	169	28.5
				0	151	25.5
				-1	1	.2
				-2	0	0.0
10	FORMAL INSPECTION OF DOCU-MENTATION AND CODE	1	MEDIUM/POSITIVE	+2	186	30.5
				+1	249	40.2
				0	138	25.3
				-1	19	3.2
				-2	3	.6
11	SOFTWARE DEVELOP-MENT TOOLS	1	MEDIUM/POSITIVE	+2	203	35.0
				+1	205	35.3
				0	165	28.1
				-1	9	1.6
				-2	0	0.0

TABLE D-VII. MERIT Output - Problem Unweighted (Continued)

COMPOSITE FIGURE OF MERIT IS MEDIUM/POSITIVE	+2	2404	36.6
	+1	2187	33.3
	0	1731	26.3
	-1	190	2.9
	-2	18	.3

TABLE D-VIII. MERIT Output - Problem Weighted

INDEX	CHARACTERISTIC	WEIGHT	ASSERTION STRENGTH/ INFLUENCE RATING	RANK	FREQ.	PCT.
1	COST OVERRUN	69	STRONG/POSITIVE	+2	14352	38.2
				+1	12903	34.4
				0	6969	18.6
				-1	3174	8.5
				-2	138	.4
2	DEVELOP-MENT STATUS INVISIBIL-ITY	43	MEDIUM/POSITIVE	+2	10234	43.0
				+1	6708	28.2
				0	6536	27.4
				-1	344	1.4
				-2	0	0.0
3	UNRELI-ABILITY	52	STRONG/POSITIVE	+2	13104	44.9
				+1	11180	38.3
				0	4420	15.2
				-1	468	1.6
				-2	0	0.0
4	UNMAIN-TAIN-ABILITY	42	MEDIUM/POSITIVE	+2	8190	36.0
				+1	7602	33.4
				0	6426	28.2
				-1	546	2.4
				-2	0	0.0
5	INADE-QUATE SAT-ISFACTION OF REAL REQUIREMEN	68	STRONG/POSITIVE	+2	17544	46.7
				+1	10540	28.0
				0	9044	24.1
				-1	476	1.3
				-2	0	0.0
6	INEFFI-CIENT USE OF RESOURCES	48	MEDIUM/POSITIVE	+2	6964	26.9
				+1	9024	35.3
				0	7536	29.5
				-1	1776	7.0
				-2	336	1.3

TABLE D-VIII. MERIT Output - Problem Weighted (Continued)

7	SCHEDULE OVERRUN	58	MEDIUM/POSITIVE	+2	9512	30.9
				+1	11020	35.6
				0	8178	26.5
				-1	2146	6.9
				-2	58	.2
8	INADEQUATE PLANNING AND CONTROL	53	MEDIUM/POSITIVE	+2	10600	36.9
				+1	9699	33.8
				0	8036	28.0
				-1	318	1.1
				-2	53	.2
9	PROJECT MISMANAGE-MENT	55	MEDIUM/POSITIVE	+2	8580	29.3
				+1	11660	39.8
				0	8800	30.0
				-1	220	.8
				-2	55	.2
10	LACK OF PROGRAM-MING DISCIPLINE	22	MEDIUM/POSITIVE	+2	3674	30.8
				+1	3564	29.8
				0	4488	37.6
				-1	198	1.7
				-2	22	.2
11	LACK OF CONCLUSIVE TESTING	47	MEDIUM/POSITIVE	+2	9776	35.2
				+1	8460	33.0
				0	7050	27.5
				-1	282	1.1
				-2	47	.2
12	POOR DOCUMENTA-TION	35	MEDIUM/POSITIVE	+2	7525	39.2
				+1	6230	32.5
				0	5005	26.1
				-1	280	1.5
				-2	140	.7

TABLE D-VIII. MERIT Output - Problem Weighted (Continued)

INDEX	PRACTICE	WEIGHT	ASSERTION STRENGTH/ INFLUENCE RATING	RANK	FREQ.	PCT.
1	REQUIRE-MENTS ANALYSIS AND VALIDATION	1	STRONG/POSITIVE	+2	15622	50.3
				+1	9782	31.5
				0	5285	17.0
				-1	295	1.0
				-2	57	.2
2	PROCESS DESIGN	1	STRONG/POSITIVE	+2	11949	41.6
				+1	10210	35.5
				0	6341	22.1
				-1	188	.7
				-2	53	.2
3	INCREMEN-TAL DEVELOP-MENT	1	STRONG/POSITIVE	+2	11476	38.2
				+1	10355	34.5
				0	6015	20.0
				-1	1841	6.1
				-2	337	1.1
4	COMPLETE PRELIMI-NARY DESIGN	1	STRONG/POSITIVE	+2	13573	44.6
				+1	11453	37.6
				0	5124	16.8
				-1	250	.8
				-2	35	.1
5	UNIT DEVELOP-MENT FOLDERS	1	STRONG/POSITIVE	+2	12331	41.2
				+1	9860	33.0
				0	6482	21.7
				-1	1171	3.9
				-2	58	.2
6	ENFORCED PROGRAM-MING STANDARDS	1	MEDIUM/POSITIVE	+2	5344	17.9
				+1	9020	30.2
				0	12903	43.3
				-1	2523	8.5
				-2	48	.2

TABLE D-VIII. MERIT Output - Problem Weighted (Continued)

7	INDEPEND-ENT TESTING	1	MEDIUM/POSITIVE	+2	9369	32.2
				+1	8505	29.2
				0	9623	33.0
				-1	1537	5.3
				-2	96	.3
8	SOFTWARE CONFIG. MANAGEMENT	1	MEDIUM/POSITIVE	+2	7766	27.5
				+1	10246	36.3
				0	9389	33.3
				-1	827	2.9
				-2	0	0.0
9	BASELINING OF RE-QUIREMENTS SPECIFICA-TION	1	STRONG/POSITIVE	+2	14351	50.6
				+1	8221	28.0
				0	6217	21.2
				-1	69	.2
				-2	0	0.0
10	FORMAL INSPECTION OF DOCU-MENTATION AND CODE	1	MEDIUM/POSITIVE	+2	7835	29.1
				+1	10787	40.1
				0	7071	26.3
				-1	1048	3.9
				-2	155	.6
11	SOFTWARE DEVELOP-MENT TOOLS	1	MEDIUM/POSITIVE	+2	9339	34.5
				+1	10151	37.6
				0	8053	28.2
				-1	479	1.7
				-2	0	0.0

TABLE D-VIII. MERIT Output - Problem Weighted (Continued)

COMPOSITE FIGURE OF MERIT IS MEDIUM/POSITIVE	+2	*9955	37.2
	+1	*8590	33.7
	0	82508	25.6
	-1	10228	3.2
	-2	849	.3

TABLE D-IX. MERIT Source Listing

```
      PROGRAM MERIT (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE7,TAPE8)
      DIMENSION ICW(30) , IPW(18) , ICHOL(5,30) , IPHOL(5,18)
      DIMENSION N2P(30,18) , N1P(30,18) , NO(30,18) , N1N(30,18)
      DIMENSION N2N(30,18) , P2P(30,18) , P1P(30,18) , PO(30,18)
      DIMENSION P1N(30,18) , P2N(30,18) , N2PC(30) , N1PC(30) , NOC(30)
      DIMENSION N1NC(30) , N2NC(30) , P2PC(30) , P1PC(30) , POC(30)
      DIMENSION P1NC(30) , P2NC(30) , N2PP(18) , N1PP(18) , NOP(18)
      DIMENSION N1NP(18) , N2NP(18) , P2PP(18) , P1PP(18) , POP(18)
      DIMENSION P1NP(18) , P2NP(18) , NW2P(30,18) , NW1P(30,18)
      DIMENSION NWO(30,18) , NW1N(30,18) , NW2N(30,18)
      DIMENSION NTC(30) , NTP(18) , NTPC(30) , NTNC(30) , NTPP(18)
      DIMENSION NTNP(18)
      DIMENSION IS(3) , IR(12) , ISPC(30) , IRPC(30) , ISPP(18)
      DIMENSION IRPP(18) , IRANK(5)
      DATA IRANK / 2H+2 , 2H+1 , 2H 0 , 2H-1 , 2H-2 /
      DATA IS / 6HSTRONG , 6HMEDIUM , 6H  WEAK /
      DATA IR / 6HSTRONG , 6HPOS.   , 6HPOSITI , 6HVE     , 6HINDIFF ,
     1 6HERENT  , 6HNEGATI , 6HVE     , 6HSTRONG , 6HNEG.   , 6HINCONC ,
     2 6HLUSIVE /
      NAMELIST / INPUT / NC , NP , ICW , IPW
*
*     READ - NC = NO. OF CHARACTERISTICS STORED ON TAPE7 (MAXIMUM IS 30)
*            NP = NO. OF PRACTICES STORED ON TAPE7 (MAXIMUM IS 18)
*
      READ (7,25) NC , NP
   25 FORMAT (2I2)
*
*     INITIALIZE CHARACTERISTIC SUMMARY COLUMNS AND WEIGHTS AND
*              PRACTICE SUMMARY ROWS AND WEIGHTS
*
      DO 40 I = 1 , NC
      N2PC(I) = 0
      N1PC(I) = 0
      NOC(I) = 0
      N1NC(I) = 0
      N2NC(I) = 0
   40 ICW(I) = 1
      DO 60 J = 1 , NP
```

TABLE D-IX. MERIT Source Listing (Continued)

```
      N2PP(J) = 0
      N1PP(J) = 0
      N0P(J) = 0
      N1NP(J) = 0
      N2NP(J) = 0
   60 IPW(J) = 1
      N2PT = 0
      N1PT = 0
      N0T = 0
      N1NT = 0
      N2NT = 0
      NTPT = 0
      NTNT = 0
      NTT = 0
*
*     READ USER DATA - NC = NO. OF CHARACTERISTICS
*                      NP = NO. OF PRACTICES
*                      ICW = CHARACTERISTIC COLUMN WEIGHTS (INIT. TO 1)
*                      IPW = PRACTICE ROW WEIGHTS (INITIALIZED TO 1)
*
      READ (5,INPUT)
*
*     OBTAIN TALLY DATA FROM TAPE7 -
*            N2P(I,J) -- N2N(I,J) = TALLY DATA
*            P2P(I,J) -- P2N(I,J) = PERCENTAGES OF EACH TALLY DATA
*
      NC2 = NC / 2
      N1 = NC2 + 1
      DO 80 J = 1 , NP
      READ (7,75) (N2P(I,J),P2P(I,J),I=1,NC2)
   75 FORMAT (15(I3,F6.2))
      READ (7,75) (N1P(I,J),P1P(I,J),I=1,NC2)
      READ (7,75) (N0(I,J),P0(I,J),I=1,NC2)
      READ (7,75) (N1N(I,J),P1N(I,J),I=1,NC2)
      READ (7,75) (N2N(I,J),P2N(I,J),I=1,NC2)
      READ (7,75) (N2P(I,J),P2P(I,J),I=N1,NC)
      READ (7,75) (N1P(I,J),P1P(I,J),I=N1,NC)
      READ (7,75) (N0(I,J),P0(I,J),I=N1,NC)
```

TABLE D-IX. MERIT Source Listing (Continued)

```
      READ (7,75) (N1N(I,J),P1N(I,J),I=N1,NC)
      READ (7,75) (N2N(I,J),P2N(I,J),I=N1,NC)
   80 CONTINUE
*
*     READ CHARACTERISTIC AND PRACTICE HOLLERITH DESCRIPTION DATA
*
      READ (8,95) ((ICHOL(K,I),K=1,5),I=1,NC)
   95 FORMAT (5A10)
      READ (8,95) ((IPHOL(K,J),K=1,5),J=1,NP)
*
*     COMPUTE - NW2P(I,J) -- NW2N(I,J) = WEIGHTED TALLY DATA
*               N2PC(I) -- N2NC(I) = WEIGHTED CHARACTERISTIC SUMMARY
*                                    COLUMN DATA
*               NTC(I) = TOTAL NO. OF WEIGHTED CHARACTERISTIC DATA
*               N1PC(I) = TOTAL NO. OF POSITIVE CHARACTERISTIC DATA
*               N1NC(I) = TOTAL NO. OF NEGATIVE CHARACTERISTIC DATA
*               P2PC(I) -- P2NC(I) = PERCENTAGE OF CHARACTERISTIC
*                                    COLUMN TALLY DATA
*               N2PT -- N2NT = OVERALL TOTAL TALLY DATA
*               NTT = TOTAL NO. OF DATA POINTS
*               N1PT = TOTAL NO. OF POSITIVE DATA POINTS
*               N1NT = TOTAL NO. OF NEGATIVE DATA POINTS
*
      DO 300 I = 1 , NC
      DO 200 J = 1 , NP
      IF ( IPW(J) ) 190 , 200
  190 NW2P(I,J) = N2P(I,J) * ICW(I) * IPW(J)
      NW1P(I,J) = N1P(I,J) * ICW(I) * IPW(J)
      NW0(I,J) = N0(I,J) * ICW(I) * IPW(J)
      NW1N(I,J) = N1N(I,J) * ICW(I) * IPW(J)
      NW2N(I,J) = N2N(I,J) * ICW(I) * IPW(J)
      N2PC(I) = N2PC(I) + NW2P(I,J)
      N1PC(I) = N1PC(I) + NW1P(I,J)
      N0C(I) = N0C(I) + NW0(I,J)
      N1NC(I) = N1NC(I) + NW1N(I,J)
      N2NC(I) = N2NC(I) + NW2N(I,J)
  200 CONTINUE
      IF ( ICW(I) ) 210 , 300
```

TABLE D-IX. MERIT Source Listing (Continued)

```
      210 NTC(I) = N2PC(I) + N1PC(I) + N0C(I) + N1NC(I) + N2NC(I)
          NTPC(I) = N2PC(I) + N1PC(I)
          NTNC(I) = N2NC(I) + N1NC(I)
          P2PC(I) = N2PC(I) * 100. / NTC(I)
          P1PC(I) = N1PC(I) * 100. / NTC(I)
          P0C(I) = N0C(I) * 100. / NTC(I)
          P1NC(I) = N1NC(I) * 100. / NTC(I)
          P2NC(I) = N2NC(I) * 100. / NTC(I)
          N2PT = N2PT + N2PC(I)
          N1PT = N1PT + N1PC(I)
          N0T = N0T + N0C(I)
          N1NT = N1NT + N1NC(I)
          N2NT = N2NT + N2NC(I)
          NTT = NTT + NTC(I)
          NTPT = NTPT + NTPC(I)
          NTNT = NTNT + NTNC(I)
      300 CONTINUE
      *
      *     COMPUTE - P2PT -- P2NT = PERCENTAGE OF TATAL TALLY DATA
      *
          P2PT = N2PT * 100. / NTT
          P1PT = N1PT * 100. / NTT
          P0T = N0T * 100. / NTT
          P1NT = N1NT * 100. / NTT
          P2NT = N2NT * 100. / NTT
      *
      *     COMPUTE - N2PP(J) -- N2NP(J) = WEIGHTED PRACTICE SUMMARY ROW DATA
      *               NTP(J) = TOTAL NO. OF WEIGHTED PRACTICE DATA
      *               NTPP(J) = TOTAL NO. OF POSITIVE PRACTICE DATA
      *               NTNP(J) = TOTAL NO. OF NEGATIVE PRACTICE DATA
      *               P2PP(J) -- P2NP(J) = PERCENTAGE OF PRACTICE ROW TALLY
      *                                    DATA
      *
          DO 500 J = 1 , NP
          DO 400 I = 1 , NC
          IF ( ICW(I) ) 390 , 400
      390 N2PP(J) = N2PP(J) + NW2P(I,J)
          N1PP(J) = N1PP(J) + NW1P(I,J)
```

TABLE D-IX. MERIT Source Listing (Continued)

```
153           NOP(J) = NOP(J) + NW0(I,J)
154           N1NP(J) = N1NP(J) + NW1N(I,J)
155           N2NP(J) = N2NP(J) + NW2N(I,J)
156         400 CONTINUE
157           IF ( IPW(J) ) 410 , 500
158         410 NTP(J) = N2PP(J) + N1PP(J) + NOP(J) + N1NP(J) + N2NP(J)
159           NTPP(J) = N2PP(J) + N1PP(J)
160           NTNP(J) = N2NP(J) + N1NP(J)
161           P2PP(J) = N2PP(J) * 100. / NTP(J)
162           P1PP(J) = N1PP(J) * 100. / NTP(J)
163           POP(J) = NOP(J) * 100. / NTP(J)
164           P1NP(J) = N1NP(J) * 100. / NTP(J)
165           P2NP(J) = N2NP(J) * 100. / NTP(J)
166         500 CONTINUE
167           K = 1
168           L = 15
169        1000 CONTINUE
170       *
171       *     ANALYZE THE HYPOTHESIS STRENGTH OF EACH I,J PAIR
172       *
173           DO 2000 J = 1 , NP
174           DO 1500 I = K , L
175       *
176       *     ANALYSE THE HYPOTHESIS STRENGTH OF EACH CHARACTERISTIC COLUMN
177       *     DETERMINE IF THE CONFIDENCE LEVEL IS STRONG
178       *
179           IF ( NOC(I) - NTC(I) * .25 ) 1110 , 1110 , 1240
180        1110 ISPC(I) = 1
181       *
182       *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
183       *
184           IF ( NTPC(I) ) 1120 , 1130
185        1120 IF ( NTNC(I) / NTPC(I) - 3 ) 1170 , 1130 , 1130
186       *
187       *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG NEGATIVE
188       *
189        1130 IF ( N1NC(I) ) 1140 , 1150
190        1140 IF ( N2NC(I) / N1NC(I) - 2 ) 1160 , 1150 , 1150
```

TABLE D-IX. MERIT Source Listing (Continued)

```
191      1150 IRPC(I) = 9
192           GO TO 1500
193      1160 IRPC(I) = 7
194           GO TO 1500
195     *
196     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
197     *
198      1170 IF ( NTNC(I) ) 1180 , 1190
199      1180 IF ( NTPC(I) / NTNC(I) - 3 ) 1230 , 1190 , 1190
200     *
201     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG POSITIVE
202     *
203      1190 IF ( N1PC(I) ) 1200 , 1210
204      1200 IF ( N2PC(I) / N1PC(I) - 2 ) 1220 , 1210 , 1210
205      1210 IRPC(I) = 1
206           GO TO 1500
207      1220 IRPC(I) = 3
208           GO TO 1500
209     *
210     *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
211     *
212      1230 IRPC(I) = 11
213           GO TO 1500
214     *
215     *     DETERMINE IF THE CONFIDENCE LEVEL IS MEDIUM
216     *
217      1240 IF ( NOC(I) - NTC(I) * .5 ) 1250 , 1250 , 1320
218      1250 ISPC(I) = 2
219     *
220     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
221     *
222           IF ( NTPC(I) ) 1260 , 1270
223      1260 IF ( NTNC(I) / NTPC(I) - 3 ) 1280 , 1270 , 1270
224      1270 IRPC(I) = 7
225           GO TO 1500
226     *
227     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
228     *
```

TABLE D-IX. MERIT Source Listing (Continued)

```
229      1280 IF ( NTNC(I) ) 1290 , 1300
230      1290 IF ( NTPC(I) / NTNC(I) - 3 ) 1310 , 1300 , 1300
231      1300 IRPC(I) = 3
232           GO TO 1500
233     *
234     *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
235     *
236      1310 IRPC(I) = 11
237           GO TO 1500
238     *
239     *     DETERMINE IF THE CONFIDENCE LEVEL IS WEAK
240     *
241      1320 IF ( NOC(I) - NTC(I) * .75 ) 1330 , 1330 , 1400
242      1330 ISPC(I) = 3
243     *
244     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
245     *
246           IF ( NTPC(I) ) 1340 , 1350
247      1340 IF ( NTNC(I) / NTPC(I) - 3 ) 1360 , 1350 , 1350
248      1350 IRPC(I) = 7
249           GO TO 1500
250     *
251     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
252     *
253      1360 IF ( NTNC(I) ) 1370 , 1380
254      1370 IF ( NTPC(I) / NTNC(I) - 3 ) 1390 , 1380 , 1380
255      1380 IRPC(I) = 3
256           GO TO 1500
257     *
258     *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
259     *
260      1390 IRPC(I) = 11
261           GO TO 1500
262     *
263     *     THERE IS STRONG CONFIDENCE THAT THE HYPOTHESIS
264     *               STRENGTH IS INDIFFERENT
265     *
266      1400 ISPC(I) = 1
```

TABLE D-IX. MERIT Source Listing (Continued)

```
267           IRPC(I) = 5
268      1500 CONTINUE
269     *
270     *     ANALYSE THE HYPOTHESIS STRENGTH OF EACH PRACTICE ROW
271     *     DETERMINE IF THE CONFIDENCE LEVEL IS STRONG
272     *
273           IF ( NOP(J) - NTP(J) * .25 ) 1610 , 1610 , 1740
274      1610 ISPP(J) = 1
275     *
276     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
277     *
278           IF ( NTPP(J) ) 1620 , 1630
279      1620 IF ( NTNP(J) / NTPP(J) - 3 ) 1670 , 1630 , 1630
280     *
281     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG NEGATIVE
282     *
283      1630 IF ( N1NP(J) ) 1640 , 1650
284      1640 IF ( N2NP(J) / N1NP(J) - 2 ) 1660 , 1650 , 1650
285      1650 IRPP(J) = 9
286           GO TO 2000
287      1660 IRPP(J) = 7
288           GO TO 2000
289     *
290     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
291     *
292      1670 IF ( NTNP(J) ) 1680 , 1690
293      1680 IF ( NTPP(J) / NTNP(J) - 3 ) 1730 , 1690 , 1690
294     *
295     *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG POSITIVE
296     *
297      1690 IF ( N1PP(J) ) 1700 , 1710
298      1700 IF ( N2PP(J) / N1PP(J) - 2 ) 1720 , 1710 , 1710
299      1710 IRPP(J) = 1
300           GO TO 2000
301      1720 IRPP(J) = 3
302           GO TO 2000
303     *
304     *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
```

TABLE D-IX. MERIT Source Listing (Continued)

```
305      *
306       1730 IRPP(J) = 11
307            GO TO 2000
308      *
309      *     DETERMINE IF THE CONFIDENCE LEVEL IS MEDIUM
310      *
311       1740 IF ( NOP(J) - NTP(J) * .5 ) 1750 , 1750 , 1820
312       1750 ISPP(J) = 2
313      *
314      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
315      *
316            IF ( NTPP(J) ) 1760 , 1770
317       1760 IF ( NTNP(J) / NTPP(J) - 3 ) 1780, 1770 , 1770
318       1770 IRPP(J) = 7
319            GO TO 2000
320      *
321      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
322      *
323       1780 IF ( NTNP(J) ) 1790 , 1800
324       1790 IF ( NTPP(J) / NTNP(J) - 3 ) 1810 , 1800 , 1800
325       1800 IRPP(J) = 3
326            GO TO 2000
327      *
328      *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
329      *
330       1810 IRPP(J) = 11
331            GO TO 2000
332      *
333      *     DETERMINE IF THE CONFIDENCE LEVEL IS WEAK
334      *
335       1820 IF ( NOP(J) - NTP(J) * .75 ) 1830 , 1830 , 1900
336       1830 ISPP(J) = 3
337      *
338      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
339      *
340            IF ( NTPP(J) ) 1840 , 1850
341       1840 IF ( NTNP(J) / NTPP(J) - 3 ) 1860 , 1850 , 1850
342       1850 IRPP(J) = 7
```

TABLE D-IX. MERIT Source Listing (Continued)

```
343             GO TO 2000
344       *
345       *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
346       *
347        1860 IF ( NTNP(J) ) 1870 , 1980
348        1870 IF ( NTPP(J) / NTNP(J) - 3 ) 1890 , 1880 , 1880
349        1880 IRPP(J) = 3
350             GO TO 2000
351       *
352       *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
353       *
354        1890 IRPP(J) = 11
355             GO TO 2000
356       *
357       *     THERE IS STRONG CONFIDENCE THAT THE HYPOTHESIS
358       *               STRENGTH IS INDIFFERENT
359       *
360        1900 ISPP(J) = 1
361             IRPP(J) = 5
362        2000 CONTINUE
363             IF ( K - 16 ) 2100 , 2200
364        2100 K = 16
365             L = 30
366             GO TO 1000
367       *
368       *     DETERMINE THE HYPOTHESIS STRENGTH OF ALL THE DATA
369       *
370        2200 CONTINUE
371       *
372       *     DETERMINE IF THE CONFIDENCE LEVEL IS STRONG
373       *
374             IF ( NOT - NTT * .25 ) 2210 , 2210 , 2340
375        2210 ISPT = 1
376       *
377       *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
378       *
379             IF ( NTPT ) 2220 , 2230
380        2220 IF ( NTNT / NTPT - 3 ) 2270 , 2230 , 2230
```

TABLE D-IX. MERIT Source Listing (Continued)

```
      *
      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG NEGATIVE
      *
 2230 IF ( N1NT ) 2240 , 2250
 2240 IF ( N2NT / N1NT - 2 ) 2260 , 2250 , 2250
 2250 IRPT = 9
      GO TO 3000
 2260 IRPT = 7
      GO TO 3000
      *
      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
      *
 2270 IF ( NTNT ) 2280 , 2290
 2280 IF ( NTPT / NTNT - 3 ) 2330 , 2290 , 2290
      *
      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS STRONG POSITIVE
      *
 2290 IF ( N1PT ) 2300 , 2310
 2300 IF ( N2PT / N1PT - 2 ) 2320 , 2310 , 2310
 2310 IRPT = 1
      GO TO 3000
 2320 IRPT = 3
      GO TO 3000
      *
      *     THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
      *
 2330 IRPT = 11
      GO TO 3000
      *
      *     DETERMINE IF THE CONFIDENCE LEVEL IS MEDIUM
      *
 2340 IF ( NOT - NTT * .5 ) 2350 , 2350 , 2420
 2350 ISPT = 2
      *
      *     DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
      *
      IF ( NTPT ) 2360 , 2370
 2360 IF ( NTNT / NTPT - 3 ) 2380 , 2370 , 2370
```

TABLE D-IX. MERIT Source Listing (Continued)

```
419       2370 IRPT = 7
420            GO TO 3000
421       *
422       *      DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
423       *
424       2380 IF ( NTNT ) 2390 , 2400
425       2390 IF ( NTPT / NTNT - 3 ) 2410 , 2400 , 2400
426       2400 IRPT = 3
427            GO TO 3000
428       *
429       *      THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
430       *
431       2410 IRPT = 11
432            GO TO 3000
433       *
434       *      DETERMINE IF THE CONFIDENCE LEVEL IS WEAK
435       *
436       2420 IF ( NOT - NTT * .75 ) 2430 , 2430 , 2500
437       2430 ISPT = 3
438       *
439       *      DETERMINE IF THE HYPOTHESIS STRENGTH IS NEGATIVE
440       *
441            IF ( NTPT ) 2440 , 2450
442       2440 IF ( NTNT / NTPT - 3 ) 2460 , 2450 , 2450
443       2450 IRPT = 7
444            GO TO 3000
445       *
446       *      DETERMINE IF THE HYPOTHESIS STRENGTH IS POSITIVE
447       *
448       2460 IF ( NTNT ) 2470 , 2480
449       2470 IF ( NTPT / NTNT - 3 ) 2490 , 2480 , 2480
450       2480 IRPT = 3
451            GO TO 3000
452       *
453       *      THE HYPOTHESIS STRENGTH IS INCONCLUSIVE
454       *
455       2490 IRPT = 11
456            GO TO 3000
```

AD-A040 467 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/2
IMPACT OF MPP ON SYSTEM DEVELOPMENT.(U)
MAY 77 J R BROWN F30602-76-C-0095
UNCLASSIFIED TRW-29115-6001-RU00 RADC-TR-77-121 NL

3 OF 3
AD
A040467

END

DATE
FILMED
7 - 77

TABLE D-IX. MERIT Source Listing (Continued)

```
457       *
458       *      THERE IS STRONG CONFIDENCE THAT THE HYPOTHESIS
459       *                STRENGTH IS INDIFFERENT
460       *
461        2500 ISPT = 1
462             IRPT = 5
463        3000 CONTINUE
464             WRITE (6,3100)
465        3100 FORMAT (1H1,2X,*INDEX   CHARACTERISTIC   WEIGHT   ASSERTION STRENG
466            1TH/   RANK   FREQ.   PCT.*/38X,*INFLUENCE RATING*)
467             DO 4000 I = 1 , NC
468             WRITE (6,3200) I , ICHOL(1,I) , ICW(I) , IS(ISPC(I)) , IR(IRPC(I))
469            1 , IR(IRPC(I)+1) , IRANK(1) , N2PC(I) , P2PC(I)
470        3200 FORMAT (1HO,3X,I2,7X,A10,7X,I2,5X,A6,1H/,2A6,4X,A2,3X,I5,3X,F5.1)
471             WRITE (6,3300) ICHOL(2,I) , IRANK(2) , N1PC(I) , P1PC(I)
472        3300 FORMAT (13X,A10,37X,A2,3X,I5,3X,F5.1)
473             WRITE (6,3300) ICHOL(3,I) , IRANK(3) , NOC(I) , POC(I)
474             WRITE (6,3300) ICHOL(4,I) , IRANK(4) , N1NC(I) , P1NC(I)
475             WRITE (6,3300) ICHOL(5,I) , IRANK(5) , N2NC(I) , P2NC(I)
476        4000 CONTINUE
477             WRITE (6,4100)
478        4100 FORMAT (1H1,2X,*INDEX*,6X,*PRACTICE*,6X,*WEIGHT   ASSERTION STRENG
479            1TH/   RANK   FREQ.   PCT.*/38X,*INFLUENCE RATING*)
480             DO 5000 J = 1 , NP
481             WRITE (6,3200) J , IPHOL(1,J) , IPW(J) , IS(ISPP(J)) , IR(IRPP(J))
482            1 , IR(IRPP(J)+1) , IRANK(1) , N2PP(J) , P2PP(J)
483             WRITE (6,3300) IPHOL(2,J) , IRANK(2) , N1PP(J) , P1PP(J)
484             WRITE (6,3300) IPHOL(3,J) , IRANK(3) , NOP(J) , POP(J)
485             WRITE (6,3300) IPHOL(4,J) , IRANK(4) , N1NP(J) , P1NP(J)
486             WRITE (6,3300) IPHOL(5,J) , IRANK(5) , N2NP(J) , P2NP(J)
487        5000 CONTINUE
488             WRITE (6,5100) IS(ISPT) , IR(IRPT) , IR(IRPT+1) , IRANK(1) , N2PT
489            1 , P2PT
490        5100 FORMAT (1H1,7X,*COMPOSITE FIGURE OF MERIT IS *,A6,1H/,2A6,4X,A2,
491            1 3X,I5,3X,F5.1)
492             WRITE (6,5200) IRANK(2) , N1PT , P1PT
493        5200 FORMAT (1HO,59X,A2,3X,I5,3X,F5.1)
494             WRITE (6,5200) IRANK(3) , NOT , POT
```

TABLE D-IX. MERIT Source Listing (Continued)

```
495             WRITE (6,5200) IRANK(4) , N1NT , P1NT
496             WRITE (6,5200) IRANK(5) , N2NT , P2NT
497             STOP
498             END
```

D.3.5 Listing

The MERIT source listing is presented in Table D-IX.

D.4 RANK

D.4.1 Purpose

The RANK program was required to analyze a large number of individual rankings obtained from the MPP Impact Evaluation survey and produce a composite ranking.

D.4.2 Input

RANK accepts as input N individual rank orderings in the form of vectors containing the integers 1,2,...., M in any order.

D.4.3 Function

For each of the M positions (i.e., the length of the input rank vectors), RANK computes the mean, the variance and upper and lower bounds for the rank deviation. RANK also computes other statistics, including the Kendall Coefficient of Concordance and the Spearman Rank Correlation Coefficient for each pair of ranking vectors, and sorts on the mean rank values to obtain the composite rank ordering.

D.4.4 Output

RANK output consists of the composite rank ordering vector, mean rank values, variances and upper and lower bounds of rank deviation and the correlation coefficients. Two examples of the mean rank and composite rank ordering results for ranked software development problems and ranked MPP are contained in Tables D-X and D-XI.

D.4.5 Listing

The RANK source listing is presented in Table D-XII.

TABLE D-X RANK OUTPUT - PROBLEM RANKING

Problem Rank	Problem Index	Problem Description	Mean	Standard Deviation
1	1	Cost Overrun	4.277	3.571
2	5	Inadequate Satisfaction of Real Requirements	4.369	2.980
3	7	Schedule Overrun	5.477	3.517
4	9	Project Mismanagement	5.862	3.383
5	8	Inadequate Planning and Control	6.077	3.245
6	3	Unreliability	6.123	2.927
7	6	Inefficient Use of Resources	6.615	3.661
8	11	Lack of Conclusive Testing	6.738	2.905
9	2	Development Status Invisibility	7.200	3.119
10	4	Unmaintainability	7.292	2.827
11	12	Poor Documentation	8.031	2.734
12	10	Lack of Programming Discipline	9.462	2.632

TABLE D-XI - RANK OUTPUT - MPP RANKING

MPP Rank	MPP Index	MPP Description	Mean	Standard Deviation
1	1	Requirements Analysis and Validation	3.138	2.705
2	9	Baselining of Requirements Specification	3.185	2.642
3	4	Complete Preliminary Design	3.892	1.962
4	2	Process Design	4.615	2.528
5	3	Incremental Development	5.662	2.868
6	5	Unit Development Folders	5.862	2.625
7	11	Software Development Tools	6.923	2.255
8	7	Independent Testing	7.615	2.558
9	6	Enforced Programming Standards	7.769	2.081
10	8	Software Configuration Management	7.877	2.490
11	10	Formal Inspection of Documentation and Code	8.969	2.386

TABLE D-XII. RANK Source Listing

```
      PROGRAM RANK (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      DIMENSION RMPPR(17,70)  , IOUT(70)  , SIGMA(17)
      DIMENSION SR(17)    , RB(17)  , S(17)  , BL(17)  , BU(17)
      DIMENSION D(70,16,17)  , RS(16,17) , DEL(17) , MPP(17)
      DATA IOUT / 70 * 1 / , MAXP / 70 / , ICON / 1 /
      NAMELIST / INPUT / NIND , IOUT , RMPPR , NR
      NAMELIST / OUTPUT / SR, RB , S , BL , BU , D , RS , RT , RBT ,
     1 DEL , W , CHISQR
      READ (5,INPUT)
      DO 40 I = 1 , NR
      SR(I) = 0.
      MPP(I) = I
      DO 20 J = 1 , NIND
      SR(I) = ( RMPPR(I,J) * IOUT(I) ) + SR(I)
   20 CONTINUE
      RB(I) = SR(I) / NIND
   40 CONTINUE
      RNIND = NIND
      DO 65 I = 1 , NR
      S(I) = 0.
      DO 60 J = 1 , NIND
      S(I) = S(I) + ( RMPPR(I,J) - RB(I) ) **2
   60 CONTINUE
      S(I) = SQRT ( S(I) / NIND )
      SIGMA(I) = S(I) / SQRT (RNIND)
      TEMP = ICON * SIGMA(I)
      BL(I) = RB(I) - TEMP
      BU(I) = RB(I) + TEMP
   65 CONTINUE
   70 CONTINUE
      IFLAG = 0
      N1 = NR - 1
      DO 76 I = 1 , N1
      IF ( RB(I+1) - RB(I) ) 74 , 76 , 76
   74 CONTINUE
      IFLAG = 1
      TEMP = RB(I)
      RB(I) = RB(I+1)
```

TABLE D-XIII. RANK Source Listing (Continued)

```
      RB(I+1) = TEMP
      TEMP = MPP(I)
      MPP(I) = MPP(I+1)
      MPP(I+1) = TEMP
      TEMP = S(I)
      S(I) = S(I+1)
      S(I+1) = TEMP
      TEMP = SIGMA(I)
      SIGMA(I) = SIGMA(I+1)
      SIGMA(I+1) = TEMP
      TEMP = BL(I)
      BL(I) = BL(I+1)
      BL(I+1) = TEMP
      TEMP = BU(I)
      BU(I) = BU(I+1)
      BU(I+1) = TEMP
   76 CONTINUE
      IF ( IFLAG ) 70 , 78
   78 CONTINUE
      NSQR = NIND * NIND
      N = NIND * ( NSQR - 1 )
      DO 160 K = 1 , N1
      M = K + 1
      DO 140 L = M , NR
      RS(K,L) = 0.
      DO 120 J = 1 , NIND
      D(J,K,L) = RMPPR(K,J) - RMPPR(L,J)
      RS(K,L) = RS(K,L) + ( D(J,K,L) * D(J,K,L) )
  120 CONTINUE
      RS(K,L) = 1 - ( 6 * RS(K,L) ) / N
  140 CONTINUE
  160 CONTINUE
      RT = 0.
      DO 180 I = 1 , NR
      RT = RT + SR(I)
  180 CONTINUE
      RBT = RT / NR
      W = 0.
```

TABLE D-XII. RANK Source Listing (Continued)

```
77            DO 200 I = 1 , NR
78            DEL(I) = SR(I) - RBT
79            W = W + DEL(I) * DEL(I)
80        200 CONTINUE
81            W = W / (NSQR * 408)
82            CHISQR = NIND * N1 * W
83            WRITE (6,INPUT)
84            WRITE (6,OUTPUT)
85            WRITE (6,220)
86        220 FORMAT (1H1,7X,3HMPP,6X,2HRB,9X,1HS,7X,5HSIGMA,6X,2HBL,9X,2HBU,/)
87            DO 250 I = 1 , NR
88            WRITE (6,225) I , MPP(I) , RB(I) , S(I) , SIGMA(I) , BL(I) , BU(I)
89        225 FORMAT (1HO,3X,I2,3X,I2,5(3X,F7.3))
90        250 CONTINUE
91            STOP
92            END
```

APPENDIX E

DETAILED MPP COMPARISON RESULTS

E.1 Contents

This appendix contains output of the MERIT support program. MERIT was used to determine the effect of applying a selected set of weights to the raw response-frequency-distribution data from the MPP impact evaluation survey. This was done (as discussed in Section 4 of this report) to support comparative evaluation of two MPP implementations. The MPP comparison methodology was demonstrated via determination of figures of merit for two disjoint subsets of the Systems Technology Program MPP.

E.2 Format

The format of the MERIT output presented on Tables E-I and E-II is described in detail in Section D.3.4 (Appendix D). Table E-I presents MERIT output for MPP Set #1 consisting of the practices numbered 1 through 5, while E-II is for MPP Set #2 consisting of the practices numbered 6 through 11.

TABLE E-I. MERIT Output For MPP Set #1

INDEX	CHARACTERISTIC	WEIGHT	ASSERTION STRENGTH/ INFLUENCE RATING	RANK	FREQ.	PCT.
1	COST OVERRUN	99	STRONG/POSITIVE	+2	12870	50.6
				+1	8217	32.3
				0	3168	12.5
				-1	1089	4.3
				-2	99	.4
2	DEVELOP-MENT STATUS INVISIBIL-ITY	50	STRONG/STRONGPOS.	+2	8150	61.7
				+1	3250	24.6
				0	1450	11.0
				-1	350	2.7
				-2	0	0.0
3	UNRELI-ABILITY	75	STRONG/POSITIVE	+2	7725	39.9
				+1	7800	40.3
				0	3375	17.4
				-1	450	2.3
				-2	0	0.0
4	UNMAIN-TAIN-ABILITY	25	MEDIUM/POSITIVE	+2	2425	38.0
				+1	2025	31.8
				0	1700	26.7
				-1	225	3.5
				-2	0	0.0
5	INADE-QUATE SAT-ISFACTION OF REAL REQUIREMEN	50	STRONG/POSITIVE	+2	6600	52.0
				+1	3800	29.9
				0	2050	16.1
				-1	250	2.0
				-2	0	0.0
6	INEFFI-CIENT USE OF RESOURCES	75	STRONG/POSITIVE	+2	6225	33.5
				+1	7125	38.3
				0	4575	24.6
				-1	525	2.8
				-2	150	.8

TABLE E-I. MERIT Output For MPP Set #1 (Continued)

7	SCHEDULE OVERRUN	75	STRONG/POSITIVE	+2	7575	41.4
				+1	7050	38.5
				0	3075	16.8
				-1	525	2.9
				-2	75	.4
8	INADEQUATE PLANNING AND CONTROL	1	STRONG/POSITIVE	+2	127	49.0
				+1	92	35.5
				0	35	13.5
				-1	4	1.5
				-2	1	.4
9	PROJECT MISMANAGE-MENT	1	MEDIUM/POSITIVE	+2	85	34.4
				+1	98	39.7
				0	62	25.1
				-1	1	.4
				-2	1	.4
10	LACK OF PROGRAM-MING DISCIPLINE	1	MEDIUM/POSITIVE	+2	60	24.0
				+1	82	32.8
				0	100	40.0
				-1	7	2.8
				-2	1	.4
11	LACK OF CONCLUSIVE TESTING	25	STRONG/POSITIVE	+2	2225	35.5
				+1	2350	37.5
				0	1525	24.3
				-1	150	2.4
				-2	25	.4
12	POOR DOCUMENTA-TION	25	STRONG/POSITIVE	+2	2725	42.4
				+1	1900	29.6
				0	1550	24.1
				-1	150	2.3
				-2	100	1.6

TABLE E-I. MERIT Output for MPP Set #1 (Continued)

INDEX	PRACTICE	WEIGHT	ASSERTION STRENGTH/ INFLUENCE RATING	RANK	FREQ.	PCT.
1	REQUIRE-	1	STRONG/POSITIVE	+2	14089	53.4
	MENTS			+1	8041	30.5
	ANALYSIS			0	3968	15.0
	AND			-1	251	1.0
	VALIDATION			-2	26	.1
2	PROCESS	1	STRONG/POSITIVE	+2	10849	44.4
	DESIGN			+1	8682	35.6
				0	4714	19.3
				-1	174	.7
				-2	1	.0
3	INCREMEN-	1	STRONG/POSITIVE	+2	10400	40.6
	TAL			+1	8542	33.3
	DEVELOP-			0	4679	18.3
	MENT			-1	1678	6.5
				-2	325	1.3
4	COMPLETE	1	STRONG/POSITIVE	+2	12273	47.3
	PRELIMI-			+1	9891	38.1
	NARY			0	3485	13.4
	DESIGN			-1	275	1.1
				-2	25	.1
5	UNIT	1	STRONG/POSITIVE	+2	9181	36.6
	DEVELOP-			+1	8633	34.5
	MENT			0	5819	23.2
	FOLDERS			-1	1348	5.4
				-2	75	.3

TABLE E-I. MERIT Output for MPP Set #1 (Continued)

COMPOSITE FIGURE OF MERIT IS STRONG/POSITIVE	+2	56792	44.6
	+1	43789	34.4
	C	22665	17.8
	-1	3726	2.9
	-2	452	.4

TABLE E-II. MERIT Output for MPP Set #2

INDEX	CHARACTERISTIC	WEIGHT	ASSERTION STRENGTH/ INFLUENCE RATING	RANK	FREQ.	PCT.
1	COST OVERRUN	99	STRONG/POSITIVE	+2	7722	27.2
				+1	10296	36.2
				0	6831	24.0
				-1	3465	12.2
				-2	99	.3
2	DEVELOP- MENT STATUS INVISIBIL- ITY	50	MEDIUM/POSITIVE	+2	3750	25.9
				+1	4550	31.4
				0	6150	42.4
				-1	50	.3
				-2	0	0.0
3	UNRELI- ABILITY	75	STRONG/POSITIVE	+2	11175	49.2
				+1	8325	36.6
				0	3000	13.2
				-1	225	1.0
				-2	0	0.0
4	UNMAIN- TAIN- ABILITY	25	MEDIUM/POSITIVE	+2	2450	34.1
				+1	2500	34.8
				0	2125	29.6
				-1	100	1.4
				-2	0	0.0
5	INADE- QUATE SAT- ISFACTION OF REAL REQUIREMEN	50	MEDIUM/POSITIVE	+2	6300	42.1
				+1	3950	26.4
				0	4600	30.8
				-1	100	.7
				-2	0	0.0
6	INEFFI- CIENT USE OF RESOURCES	75	MEDIUM/POSITIVE	+2	4500	21.1
				+1	6975	32.7
				0	7200	33.8
				-1	2250	10.6
				-2	375	1.8

TABLE E-II. MERIT Output for MPP Set #2 (Continued)

7	SCHEDULE OVERRUN	75	MEDIUM/POSITIVE	+2	4725	21.8
				+1	7200	33.2
				0	7500	34.6
				-1	2250	10.4
				-2	0	0.0
8	INADEQUATE PLANNING AND CONTROL	1	MEDIUM/POSITIVE	+2	73	25.6
				+1	91	32.2
				0	117	41.3
				-1	2	.7
				-2	0	0.0
9	PROJECT MISMANAGE-MENT	1	MEDIUM/POSITIVE	+2	71	24.8
				+1	114	39.9
				0	98	34.3
				-1	3	1.0
				-2	0	0.0
10	LACK OF PROGRAM-MING DISCIPLINE	1	MEDIUM/POSITIVE	+2	107	36.5
				+1	80	27.3
				0	104	35.5
				-1	2	.7
				-2	0	0.0
11	LACK OF CONCLUSIVE TESTING	25	MEDIUM/POSITIVE	+2	2975	40.5
				+1	2150	29.3
				0	2225	30.3
				-1	0	0.0
				-2	0	0.0
12	POOR DOCUMENTA-TION	25	MEDIUM/POSITIVE	+2	2650	36.4
				+1	2550	35.1
				0	2025	27.8
				-1	50	.7
				-2	0	0.0

TABLE E-II. MERIT Output for MPP Set #2 (Continued)

INDEX	PRACTICE	WEIGHT	ASSERTION STRENGTH/ INFLUENCE RATING	RANK	FREQ.	PCT.
6	ENFORCED PROGRAM-MING STANDARDS	1	MEDIUM/POSITIVE	+2	3627	14.3
				+1	8695	34.3
				0	9846	38.8
				-1	3119	12.3
				-2	75	.3
7	INDEPEND-ENT TESTING	1	MEDIUM/POSITIVE	+2	7918	31.9
				+1	7305	29.4
				0	7382	29.7
				-1	2039	8.4
				-2	150	.6
8	SOFTWARE CONFIG. MANAGEMENT	1	MEDIUM/POSITIVE	+2	5593	23.4
				+1	8733	36.5
				0	8428	35.2
				-1	1170	4.9
				-2	0	0.0
9	BASELINING OF RE-QUIREMENTS SPECIFICA-TION	1	STRONG/STRONGPOS.	+2	13636	54.8
				+1	6362	25.6
				0	4773	19.2
				-1	99	.4
				-2	0	0.0
10	FORMAL INSPECTION OF DOCU-MENTATION AND CODE	1	MEDIUM/POSITIVE	+2	6507	28.5
				+1	8915	39.0
				0	5726	25.1
				-1	1445	6.3
				-2	249	1.1
11	SOFTWARE DEVELOP-MENT TOOLS	1	STRONG/POSITIVE	+2	9217	37.8
				+1	8781	36.0
				0	5820	23.9
				-1	575	2.4
				-2	0	0.0

TABLE E-II. MERIT Output for MPP Set #2 (Continued)

COMPOSITE FIGURE OF MERIT IS MEDIUM/POSITIVE

+2	46498	31.8
+1	48781	33.4
0	41975	28.7
-1	8497	5.8
-2	474	.3

METRIC SYSTEM

BASE UNITS:

Quantity	Unit	SI Symbol	Formula
length	metre	m	...
mass	kilogram	kg	...
time	second	s	...
electric current	ampere	A	...
thermodynamic temperature	kelvin	K	...
amount of substance	mole	mol	...
luminous intensity	candela	cd	...

SUPPLEMENTARY UNITS:

plane angle	radian	rad	...
solid angle	steradian	sr	...

DERIVED UNITS:

Acceleration	metre per second squared	...	m/s
activity (of a radioactive source)	disintegration per second	...	(disintegration)/s
angular acceleration	radian per second squared	...	rad/s
angular velocity	radian per second	...	rad/s
area	square metre	...	m
density	kilogram per cubic metre	...	kg/m
electric capacitance	farad	F	A·s/V
electrical conductance	siemens	S	A/V
electric field strength	volt per metre	...	V/m
electric inductance	henry	H	V·s/A
electric potential difference	volt	V	W/A
electric resistance	ohm		V/A
electromotive force	volt	V	W/A
energy	joule	J	N·m
entropy	joule per kelvin	...	J/K
force	newton	N	kg·m/s
frequency	hertz	Hz	(cycle)/s
illuminance	lux	lx	lm/m
luminance	candela per square metre	...	cd/m
luminous flux	lumen	lm	cd·sr
magnetic field strength	ampere per metre	...	A/m
magnetic flux	weber	Wb	V·s
magnetic flux density	tesla	T	Wb/m
magnetomotive force	ampere	A	...
power	watt	W	J/s
pressure	pascal	Pa	N/m
quantity of electricity	coulomb	C	A·s
quantity of heat	joule	J	N·m
radiant intensity	watt per steradian	...	W/sr
specific heat	joule per kilogram-kelvin	...	J/kg·K
stress	pascal	Pa	N/m
thermal conductivity	watt per metre-kelvin	...	W/m·K
velocity	metre per second	...	m/s
viscosity, dynamic	pascal-second	...	Pa·s
viscosity, kinematic	square metre per second	...	m/s
voltage	volt	V	W/A
volume	cubic metre	...	m
wavenumber	reciprocal metre	...	(wave)/m
work	joule	J	N·m

SI PREFIXES:

Multiplication Factors	Prefix	SI Symbol
1 000 000 000 000 = 10^{12}	tera	T
1 000 000 000 = 10^{9}	giga	G
1 000 000 = 10^{6}	mega	M
1 000 = 10^{3}	kilo	k
100 = 10^{2}	hecto*	h
10 = 10^{1}	deka*	da
0.1 = 10^{-1}	deci*	d
0.01 = 10^{-2}	centi*	c
0.001 = 10^{-3}	milli	m
0.000 001 = 10^{-6}	micro	μ
0.000 000 001 = 10^{-9}	nano	n
0.000 000 000 001 = 10^{-12}	pico	p
0.000 000 000 000 001 = 10^{-15}	femto	f
0.000 000 000 000 000 001 = 10^{-18}	atto	a

* To be avoided where possible.

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.