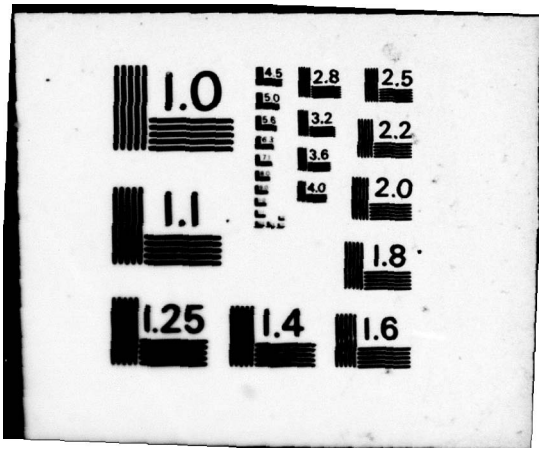
AD-A036 115 SYSTEM DEVELOPMENT CORP SANTA MONICA CALIF F/G 9/2
SOFTWARE DATA COLLECTION STUDY. VOLUME I. SUMMARY AND CONCLUSIO--ETC(U)
DEC 76 N E WILLMORTH, M C FINFER F30602-75-C-0248
UNCLASSIFIED SDC-TM-5542/001/01 RADC-TR-76-329-VOL-1 NL

1 OF 2
ADA036115


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

AD A036115

RADC-TR-76-329, Volume I (of eight)
Final Technical Report
December 1976

SOFTWARE DATA COLLECTION STUDY
Summary and Conclusions

System Development Corporation

DDC
FEB 28 1977
C

Approved for public release;
distribution unlimited.

COPY AVAILABLE TO DDC DOES NOT
PERMIT FULLY LEGIBLE PRODUCTION

ROME AIR DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
GRIFFISS AIR FORCE BASE, NEW YORK 13441

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.

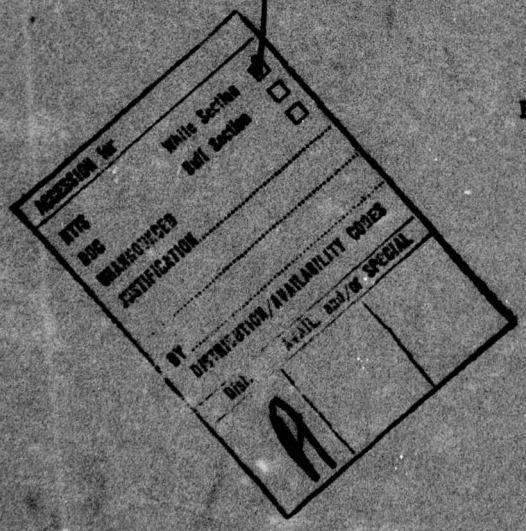
This report has been reviewed and approved for publication.

APPROVED: Richard T. Slavinski

RICHARD T. SLAVINSKI
Project Engineer

APPROVED: Alan R. Barnum

ALAN R. BARNUM
Assistant 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-76-329, Vol I (of eight)
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER (9)

4. TITLE (and Subtitle): (6) SOFTWARE DATA COLLECTION STUDY. Volume I. Summary and Conclusions.
5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. Jun 75 — Jun 76
6. PERFORMING ORG. REPORT NUMBER: (14) SDC TM-5542/001/01

7. AUTHOR(s): (10) N. E. Willmorth, M. C. Finfer, M. P. Templeton
8. CONTRACT OR GRANT NUMBER(s): (15) F30602-75-C-0248

9. PERFORMING ORGANIZATION NAME AND ADDRESS: System Development Corporation, 2500 Colorado Avenue, Santa Monica CA 90406
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 63728F, 55500810

11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISIS), Griffiss AFB NY 13441
12. REPORT DATE: (11) Dec 76
13. NUMBER OF PAGES: 114

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same (12) 122p.
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.

(16) 5550 (17) 08 (18) RADC

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report): Same

(19) TR-76-329-Vol-1

18. SUPPLEMENTARY NOTES: RADC Project Engineer: Richard T. Slavinski (ISIS)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number): Software Development, Data Collection, Software Research

20. ABSTRACT (Continue on reverse side if necessary and identify by block number): The burgeoning costs of software development have centered research interest in software methodology, project productivity and program reliability. However, such research is hindered by the lack of standard, reliable data for an adequate sample of software projects upon which to base conclusions. RADC proposes to establish a repository in which software development data may be accumulated; this study was conducted to generate recommendations concerning a data collection system for that repository.

(cont'd)

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

339900

The objective of the study was to investigate:

- Data collection problems
- Data requirements for productivity, software reliability and cost studies
- Data entry/data management interface
- Specifications for a software data collection and reporting system

This report consists of an executive summary of the investigations made. Data collection problems were found to arise from the lack of standardization, from the effects of "instrumenting" the development process, from resistance to management control and from relunctance to release data. Data requirements for productivity, software reliability and cost studies include environmental parameters, project performance data and product quality characteristics obtained at strategic points in time and place in the software development life cycle. An overview of the collection automation reauirements included a discussion of the advantages and disadvantages of various data base structures, degree of centralization, data management functions, and system hardware. The general specifications for the data collection and reporting system included the recommendations made for data entry, and data management system, with specificity to data types and methods of collection, after considering the alternatives derived during the course of the study. Also included is a brief discussion of the merits/demerits of an independent collection agency consisting of either a civilian contractor, Civil Service, Air Force personnel or combination of the above.

TABLE OF CONTENTS

Section		Page
1.	Technical Problem	1
1.1	Definition of the Problem	1
1.2	Objectives of the Study	2
1.3	General Methodology	3
1.4	Organization of the Report	4
2.	Assumptions and Limitations	7
2.1	Software System Development Concepts	7
2.1.1	Controlling the Developmental Process	8
2.1.2	Project Environment	8
2.1.3	Software Development Life-Cycle Model	12
2.1.4	The Work Breakdown Structure	14
2.1.5	Performance Control	18
2.2	Repository Operational Concept	19
2.2.1	Project Management	19
2.2.2	Research	20
2.2.3	Desensitization, Privacy, and Security	21
2.2.4	Growth Potential	22
2.2.5	Flexibility	22
3.	Data Collection Technology	24
3.1	Problems of Data Acquisition	24
3.1.1	Mensuration Difficulties	25
3.1.2	Instrumentation Effects	26
3.1.3	Unreliability of Measures	27
3.1.4	Reluctance to Release Data	27
3.1.5	Cost Factors	29
3.1.6	Systemic Problems	30
3.1.7	Problem Summary	30
3.2	Current Data Collection Techniques	31
3.2.1	Environmental Characteristics	32
3.2.2	Project Performance Characteristics	32

TABLE OF CONTENTS(cont'd)

Section		Page
3.2.3	Configuration Characteristics	33
3.2.4	Quality Control Data	34
3.2.5	Summary of Current Military Practices	34
3.3	Data Collection Monitor Systems	35
3.3.1	IMPACT	35
3.3.2	SIMON	35
3.3.3	BMDATC Quantitative Data Base	36
3.3.4	IBM Management Data Collection and Reporting System	36
3.3.5	TRW Software Reliability Study	37
3.3.6	Evaluation of Software Monitors	37
3.4	Evaluation of Reporting Formats and Data Types	38
3.4.1	Environmental Data	38
3.4.2	Performance Data	40
3.4.3	Configuration Data	41
3.4.4	Quality Control Data	42
4.	Data Requirements for Productivity and Reliability Studies	43
4.1	Productivity Analyses	43
4.2	Reliability Anayses	48
4.3	Software Development Cost Studies	51
5.	Data Collection Automation Requirements	54
5.1	Data Structures	54
5.1.1	Degree of Centralization	54
5.1.2	Data Base Structure	57
5.1.2.1	Physical Structure	57
5.1.2.2	Access Methods	58
5.1.2.3	Logical Structures	60
5.2	Data Management Functions	61
5.2.1	Data Definition Language	61
5.2.2	Data Base Generation and Maintenance	62
5.2.3	Data Base Query and Information Retrieval	62
5.2.4	Data Base Integrity	62

TABLE OF CONTENTS (cont'd)

Section		Page
5.2.5	Security	63
5.2.6	Accounting	64
5.2.7	Data Base Restructuring	64
5.2.8	Core Management	64
5.2.9	Reporting	65
5.2.10	Scheduling	65
5.2.11	Coaching	65
5.2.12	Data Management System Capabilities	65
5.3	System Hardware Configuration	66
5.3.1	Data Base Storage Media	66
5.3.2	Data Entry Methods	67
5.3.3	Processors	70
6.	Software Data Collection System Specifications	72
6.1	Purpose	72
6.2	Assumptions and Limitations	72
6.3	Data Acquisition Procedures	75
6.4	Data Requirements	78
6.5	Data Entry Configuration	103
6.6	Data Base and Data Base Management	106
6.7	Repository Operations Management	109
7.	Implications for Further Research	113

EVALUATION

The objective of this effort was to investigate the general area of software data collection and determine data criteria necessary to assist project management, to assess evolving programming techniques, and to support continued software technology research and development.

This effort is part of TPO 3.V.A.1.4, Quality Control. Immediate results of this study will be used to establish a pilot facility at RADC which will function as a nucleus or baseline for the development of a fully operational central repository of software development data.

This investigation successfully addressed all major program objectives. Notable accomplishments, initially viewed as critical problems facing the development of a large software data base, include:

a. The definition, classification, and categorization of data parameters required to support productivity, reliability, and cost analyses.

b. The formulation of data collection forms, modularized to accomodate diverse projects and allow expansion of data parameters based upon evolving research goals and future software quality metrics.

Recommendations resulting from this effort and a parallel study conducted by the Illinois Institute of Technology Research Institute (IITRI), contract F30602-75-C-0257, "Software Data Repository," are currently being integrated into RADC plans for implementing a localized pilot facility. Technical decisions/direction for a fully operational repository will evolve from continued evaluation of this facility.

Richard T. Slavinski

RICHARD T. SLAVINSKI
Project Engineer

1. TECHNICAL PROBLEM

The overall objective of the Data Collection Study was to conduct an investigation into the formation of a software data repository that would assist RADC in the development of a viable and effective research program in data processing technology. Accurate, reliable and valid data are required to provide credible evaluations of proposed innovations in software development methodology and to provide deeper insight into the software development process that could result in improvements in programming productivity, software reliability, and software development costs.

1.1 DEFINITION OF THE PROBLEM

The burgeoning costs of software development, the lag in software productivity behind hardware productivity, and the continuing uncertainty of predicting the cost, time and difficulty of software production are matters of great concern to the data processing community. At the present time there are a number of current development technologies that are alleged to result in increased programmer productivity and software reliability. However, despite the optimistic claims advanced for these tools and techniques, there is at present few, if any, really trustworthy data to support the claims. There has been little effort expended in the past in compiling objective historical data concerning software development projects and the techniques employed by those projects. Most previous studies of the software development process have been based upon subjective, after-the-fact assessment of project parameters. Current claims are based upon experimental trials using the new methodology in the absence of adequately defined control trials upon which to base comparisons. Without such information, conclusions drawn concerning the effectiveness of the new methodology lack validity. Establishment of a software development data repository will provide a valuable service in collecting the data upon which to base comparisons and form conclusions.

A continuing repository and data collection system will go a long way toward supporting and, perhaps, resolving many of the difficulties encountered in performing studies of data processing technology. Developmental projects have an ephemeral existence; they are created, perform as required, and are discon-

tinued. The people working on a project are dispersed, assigned to other projects and are no longer responsible for preserving data concerned with that project. Without any immediate interest in the accumulated data, it is soon lost or discarded. Once delivered, the responsibility for operating and maintaining a system often shifts to an O&M group. Since many of the current claims to increased productivity are based upon alleged increases in reliability, transportability and maintainability during the life of the system, continuity of data collection is essential to the demonstration of the truth of such claims. Only by providing a repository for software development data independent of the transitory concerns of the various groups associated with the life of a system can such data be preserved for use in technology assessments.

1.2 OBJECTIVES OF THE STUDY

The objectives of the proposed RADC Software Development Data Repository are to:

- Preserve software development data collected over many projects and under many conditions for further evaluation.
- Generate further insight into the software development process.
- Provide the basis for comparative studies of software development methodologies and techniques.
- Provide data services for project management.

The study is being pursued by two contractors. SDC, the author of this study, is emphasizing components of data collection for the repository, while ITTRI, the other contractor, is emphasizing the retrieval aspects of the repository.

The objectives of this study are to:

- Investigate the problems associated with the collection of accurate, reliable and valid software development data.
- Determine the data required to serve the needs of project management and methodological research for software system development.

- Investigate data collection and data entry methods and techniques that could be used to acquire data for the repository on an ongoing basis.
- Determine the impacts of data base structure and data management system functions on the software data collection system.

To attain these objectives, the study determined the data needed to support a variety of research areas and to support the effective management of a project. The research goals that were considered include investigations of the impacts of tools and techniques on programmer productivity and program reliability, investigations of the impacts of environmental factors on productivity and reliability, and investigations of the relationships of other indices of program quality to project characteristics. Research was also done into project management techniques and tools as well as defining the basic data requirements of a project management system.

The Software Data Repository must interact with the data collection system to accept and store these data, to protect the stored data from unauthorized access or modification, and to answer complex queries and perform requested analyses. The structure of the repository partially depends upon the variety and composition of the data, partially upon the adopted concepts of data acquisition and storage, and partially upon research requirements.

1.3 GENERAL METHODOLOGY

The investigation of software data collection problems tapped several sources for the identification and evaluation of problems and the prescription of solutions to them. These include:

- The literature
- SDC project managers
- Military program management offices
- Software data repositories

The literature survey included books on programming management, programming methodology and program reliability , plus papers from the technical press, military and governmental repositories. Summaries of the previous studies will be found in Volumes 2, 3, and 4. All references to material used or quoted are found in the bibliographies of the individual volumes.

The SDC project managers with relevant experience were contacted personally and via questionnaire. Military program management offices of the three services were contacted by questionnaire. The results of the questionnaire are analyzed in Volume 5.

A conference was held at SDC which included representatives from the proposed RADC Software Data Repository, the BMDATC Quantitative Data Base and the USAF Satellite Control Facility Computer Program Development Library. The proceedings of the conference are contained in Volume 6.

1.4 ORGANIZATION OF THE REPORT

The total report for this study is organized into a series of volumes and appendices. This volume, Summary and Conclusions, states the problem, the assumptions and limitations upon which the study is based, and an executive summary of the investigations and conclusions of the study. References are made through this volume to the other volumes that support the conclusions.

The volumes in the report series are:

Volume/001 Summary and Conclusions.
This volume summarizes the study and encompasses the recommendations that are made for the RADC Software Development Data Collection System.

Volume/002 An Analysis of Software Data Collection Problems and Current Capabilities.
This volume addresses the current state of the art in software data collection. It looks at the difficulties that surround the collection of reliable and valid data, including the standardi-

zation of measurements and developer reluctance to release data. It also examines current military data collection practices and the potentials of the automatic data collection tools that are under development. Operations management of the data collection facility is considered.

Volume/003 Data Requirements for Productivity and Reliability Studies.
This volume reviews the studies that have been done in the areas of project and programmer productivity, software reliability and other indices of program quality. As a result of the review, augmented by the other activities and analyses conducted by the project, requirements for the parameters necessary to study this phenomena are derived. Detailed description of data items is relegated to Volume 007, a Compendium of Procedures and Parameters.

Volume/004 Data Management System Interface.
This volume reports the survey of data entry and data management methodology conducted to define the interface requirements that exist between data acquisition and the data storage. Requirements for data entry formatting and processing may be derived from this survey.

Volume/005 Survey of Project Managers.
This volume reports the results of a survey of project managers and military program office personnel that was conducted to isolate problems and data requirements.

Volume/006 Proceedings of the Data Collection Problems Conference.
This volume reports the results of an SDC sponsored conference attended by personnel associated with three software data repositories on the problems associated with software data collection and potential solutions.

Volume/007 Compendium of Procedures and Parameters

This volume is in essence an appendix for Volume 003. It contains descriptions of data parameters, proposed data base structure, and data collection forms and instructions. The forms and data base elements were derived using criteria of collection priority and of the principles of modularity, including "internal strength" and "relative independence" as well as size limitations.

Volume/008 Glossary of Data Collection Terminology

This volume consists of a glossary of the terms used in the above reports. The glossary does not seek to repeat definitions of most commonly used data processing terms, but only those pertinent to these reports.

2. ASSUMPTIONS AND LIMITATIONS

The recommendations for the data collection study are based upon several assumptions concerning the structure and organization of the phenomena to be measured, the aims and concepts of operation of the proposed data repository, and the problems inherent in collecting reliable and valid data from a software development project. These assumptions were not "given", but were developed during the study and formed a major portion of it. These assumptions govern the design alternatives available to the software data collection system and the content and structure of the data requirements that have been developed.

2.1 SOFTWARE SYSTEM DEVELOPMENT CONCEPTS

An understanding of the nature, composition and functioning of the software development process is necessary if a thorough analysis of information requirements conducive to the understanding, control and improvement of the process is to be accomplished. This understanding is advanced through the adoption of a series of models of various aspects of the software development process, including:

- Process control model
- Project environment model
- Software development life-cycle model
- Work Breakdown Structures
  - Product configurations
  - Function configurations
- Management control models
  - Configuration management
  - Performance management

2.1.1 Controlling the Developmental Process

In terms of a software development system, software data collection operates at two levels: 1) the level of direct management control over the developmental process; 2) the higher level of performing methodological research to improve the software development process itself. (See Figure 1). At the management "quality assurance" level, only that information is necessary that enables the manager to ascertain the quality of performance of the software product. The efficiency of the total process may be of some interest, but the day-to-day control of operations is primary.

At the system "quality control" level, more far-reaching information is necessary. The software development system must have operated enough times to obtain a stable estimate of its average performance and enough manipulation of the system must have occurred to understand the influence of manipulations upon its behavior. For management control purposes, "black box" measurement is adequate; for quality control purposes, the internal operations of the system must be known so that they may be improved. In short, although quality control may not need as frequent or as fine information on the immediate operations of the system, it needs additional measures on product quality, system performance and the effectiveness of the management control process that the 'quality assurance' level does not need. Insofar as the accumulation of these additional measures forms a burden on the project, project resistance to their collection may be expected. Since the precision and accuracy of measures for one use differ from those for the other, some problem exists. On the other hand, there is great overlap in the information to be collected for the two purposes and it would be inefficient to use duplicate data collection systems if one will suffice.

2.1.2 Project Environment

There are a great many forces in the environment of software development projects that influence their performance. If an assessment of that performance is to be made, information about these forces is necessary. Figure 2 depicts some of the classes of information required.

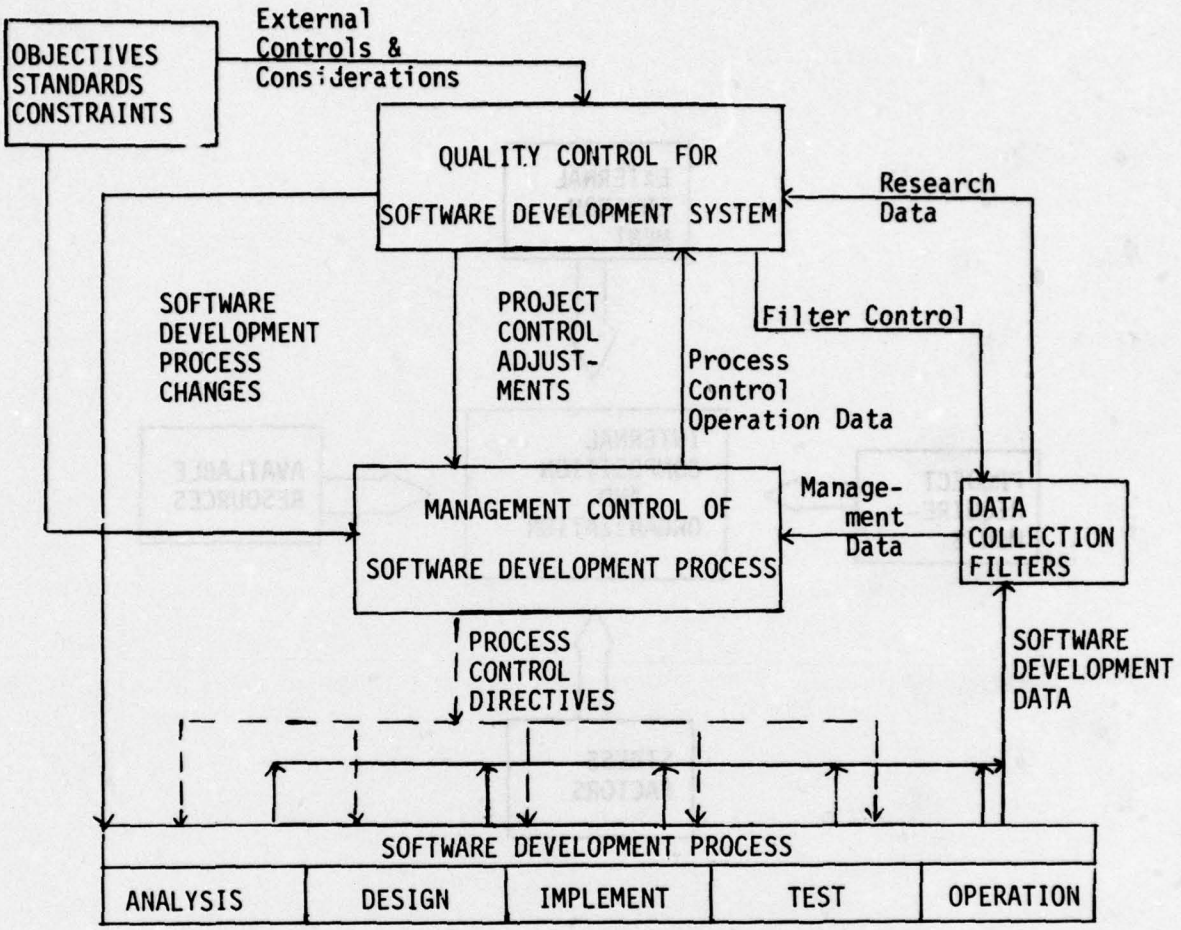


Figure 1. - SYSTEMIC INTERRELATIONSHIPS OF
SOFTWARE MANAGEMENT CONTROL AND
METHODOLOGICAL RESEARCH

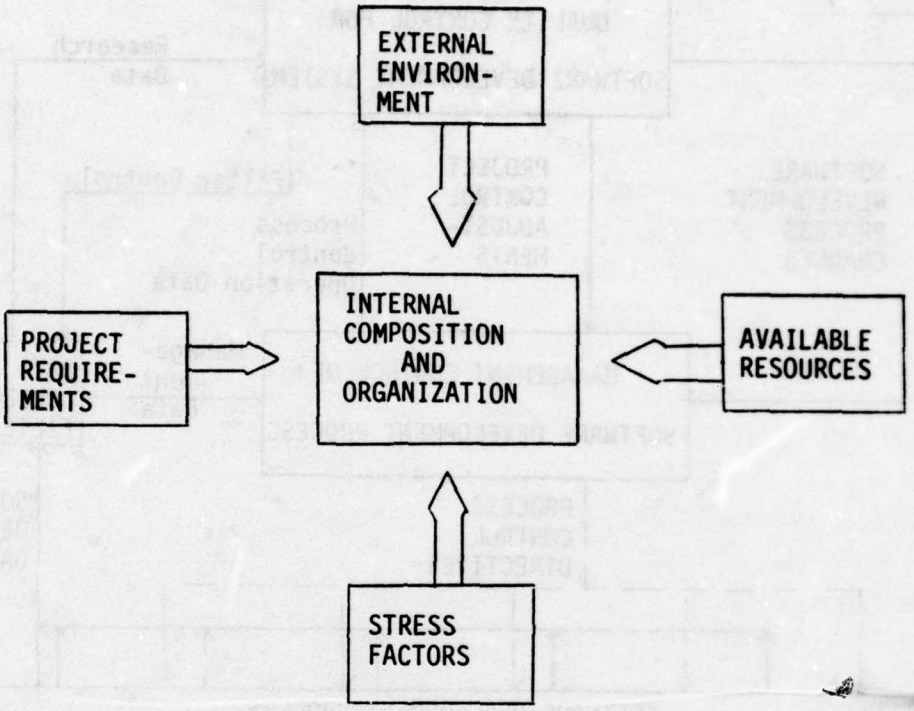


Figure 2. Software Development Project Environment

The internal composition and organization of the project strongly reflect the impacts of the other forces. Evaluations of the suitability of the composition and organization in meeting the demands of these impacts may be the basis of interesting investigations. Internal managerial, working and communication relationships are also of interest.

The external environment for the project consists largely of higher management, project monitors, user organizations and subcontractors. Contract provisions set working conditions. Relations with the customer and with subcontractors can materially affect the amounts of communication and cooperation that is experienced in getting information, getting concurrence and arriving at decisions. The closeness of supervision by both management and project monitors influences project behavior, and there may be many other implications for relative project success in its relationships with its environment, not the least being the project reporting requirements that this study seeks to define.

The primary determinants of the size and composition of the project are the size, complexity and difficulty level of the software to be produced. Well-understood, simple programs demand much less of the project than do programs requiring innovations, high performance criteria and complex interactions.

The primary determinant of the ease and efficiency with which the project meets its work requirements is the level of the resources available to it. Resources include the full range of manpower, equipment, work facilities, tools and techniques, stores of information, and numerous other items and services that it takes to perform the software development job. There is a great deal of information that can be gathered about resources. Manpower alone has many attributes such as various skills and skill levels, training, experience, and knowledge of the application area and of the customer that might influence how well the project meets its requirements. The speed, power and capacity of the computer and the conditions of use certainly affect efficiency, as does the convenience and comfort of the working facility. The tools and techniques, both manual and automated, also contribute to project effectiveness. Methodology evaluation is somewhat intangible, as is

an evaluation of the customer-supplied information available to the project in doing its job. Studies using the repository may provide the service of placing such evaluations on a more objective basis.

Perhaps of more importance than the absolute values of the various environmental parameters is the relationships between them. If resources are not adequate to meet the demands of the work, stressful conditions result. If time and dollars are short, if there are too few people available or if their skill levels do not meet the demand for innovation, if there is not enough computer power or time available, if the customer makes unreasonable demands and puts pressure on for deliveries, performance may suffer. No stress may be equally debilitating. That is, some pressure and some challenge is desirable to motivate the project. The challenge for repository studies is to find the points of equilibrium that lead to superior project performance, optimal product quality, and effectiveness in allocation and expenditure of resources.

2.1.3 Software Development Life-Cycle Model

Although current philosophy of a top-down approach to software system implementation has complicated the issue, there is a more-or-less standard model of the software system development process that is assumed for most studies. Although the model is subject to both some compression and expansion depending upon system size and difficulty, the closeness of managerial control desired, and the vantage point of the analyst, it is generally accepted that the system development model involves a requirements analysis phase; a system design phase, normally divided into preliminary and detailed design steps; a coding and debugging phase; an integration and test phase; and an installation phase.* An expansion of this model is shown in Figure 3. Here the developmental phase is a portion of the entire life-cycle

*The Guidelines for Managing Information Processing Systems, USGAO, compressess this process considerably by viewing it from the viewpoint of total system development, including lengthy system concept formation phases prior to the allocation of functions to a data system. Willmorth in System Programming Management expands each phase into a set of sub-activities and decision points.

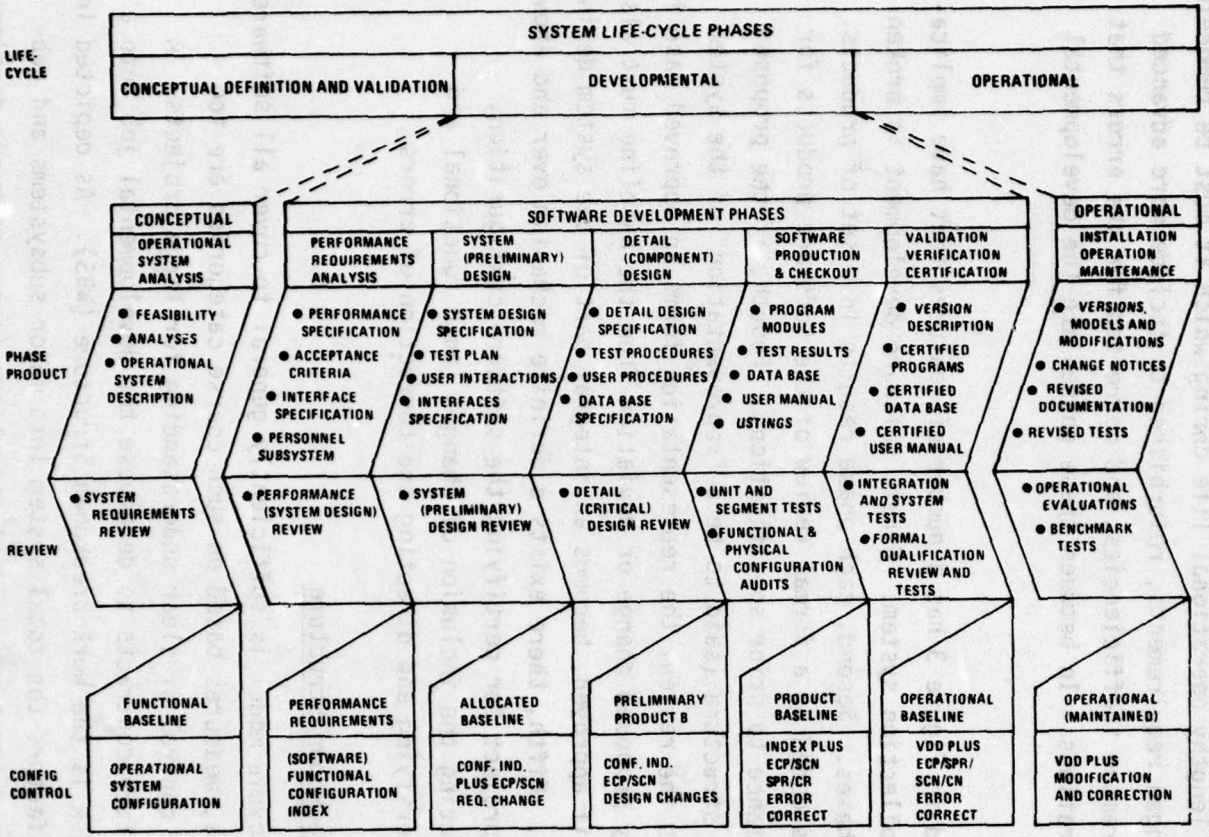


Figure 3. Software Development Life Cycle Model.

of a system, and software development is a portion of all developmental activities for a system. The conceptual or system definition phase that precedes software development defines the operational requirements for the system including those for subsystems. After development, the system is expected to have a lengthy operational life during which it must be remodeled to meet new or changed requirements, refurbished to include more advanced technology and/or remove inefficiencies, and corrected for the errors that show up during operations. In between these phases are the developmental phases.

The model presented in Figure 3 has a number of features that have implications for a data collection system. First, software development is broken into a series of phases. Second, each phase results in a set of products. Third, the phase is ended by a formal review of the software products for quality and conformance to prior specifications. Fourth, if the proposed system content and structure (said to be a "representation" of the system at that point) pass the review, the representation forms an approved baseline configuration. Any proposed change or deviation from the baseline must also be reviewed, and, if approved, becomes an integral part of the system definition at that stage. Fifth, there exists a definite mechanism over and above the developmental project for certifying the software configuration, approving and directing the inclusion of changes (both functional and structural), and verifying and directing the correction of errors.

2.1.4 The Work Breakdown Structure

Although the life-cycle model is sufficiently general to cover all software development efforts, measures based on such coarse categories are too granular for close control or clear understanding for large projects. A model used on military contracts to decompose the developmental job into a more manageable task is the Work Breakdown Structure (WBS). As depicted in Figure 4, the WBS factors the total system into major subsystems and subcomponents, relates these to functional operations (e.g., software development phases), and ties the phases to finer, scheduled work packages or major tasks. This model essentially ties together the project in terms of its organizational elements and the two major aspects of project performance:

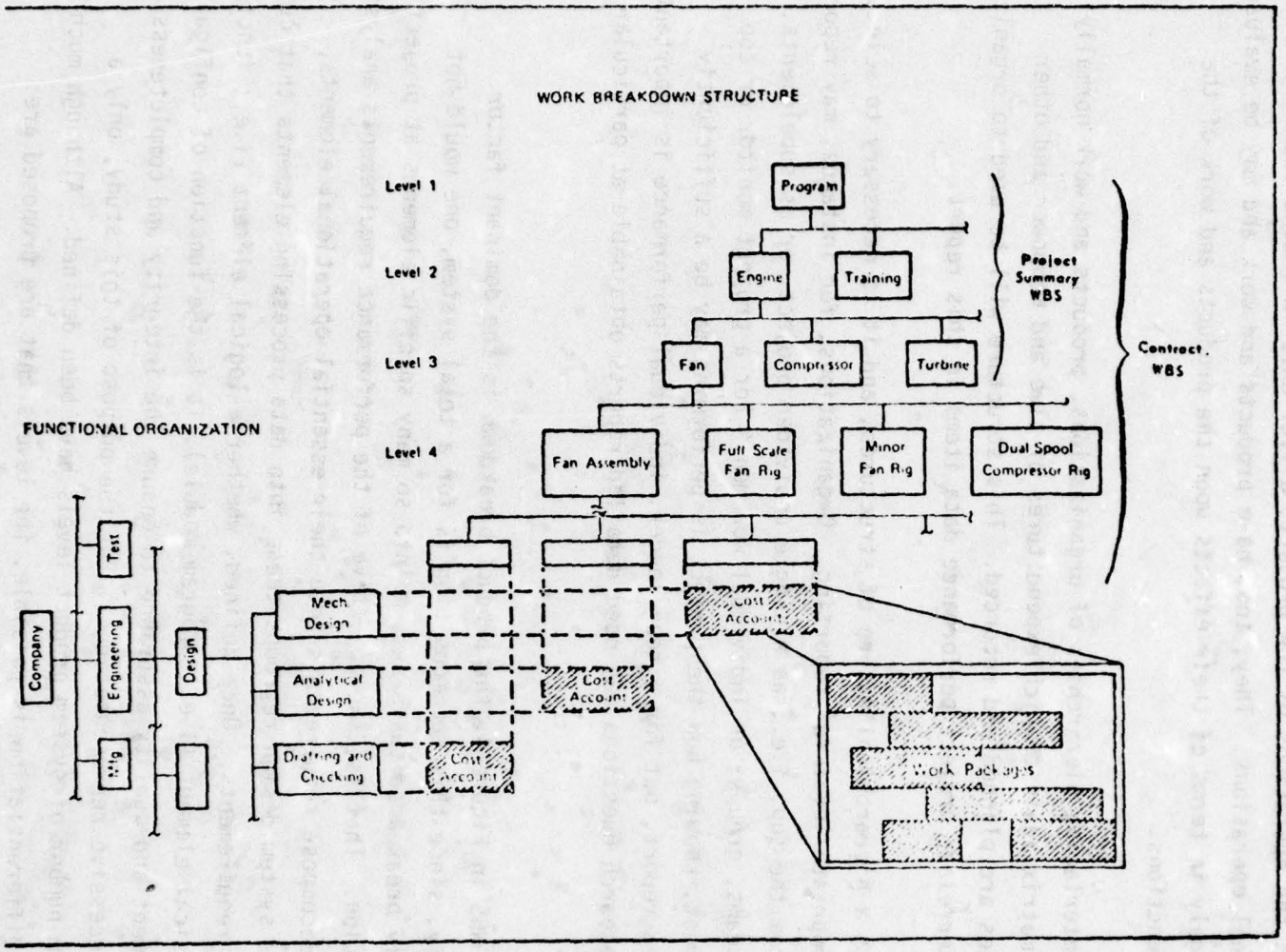


Figure 4. Integration of WBS and Organizational Structure.

*Source: FASCP/AFLCP 173-5, 31 March 1972.

products and work. Two other aspects, the quality assurance program that evaluates the quality of products and work and the managerial functions that controls the operations of the project, differ only in intent from other functional operations. They, too, have products and work and may be evaluated, but largely in terms of their effects upon the products and work of the other functions.

These interlacing hierarchies of organizations, products and work normally form a matrix against which expenditures of time and manpower and other resources are planned and recorded. This structure will be used to organize and interrelate project performance data items in this report.

There is a hierarchical nature of structures, and it is necessary to select an appropriate level for reporting. Organizations, for instance, may report only from the top, i.e., as a single undivided project, or as subelements, i.e., teams, groups, or individual workmen. For a project monitor or top management, knowing how the project is performing may be a sufficiently detailed report, but for a team leader, individual performance is important. Some research questions may need detailed reports obtainable at particular levels.

In the WBS in Figure 4, the product breakdown is the dominant factor. Actually, since this decomposition is for a total system, one would not normally break a software system into so many specific elements at project initiation. That is, the objective of the performance requirements analysis is to decompose requirements into their essential operational elements, whereas system design regroups these into data processing elements that cut across requirements. Once defined, whether a logical element (i.e., function) or physical element (i.e., program module), it is the function of configuration management and quality assurance to ensure the integrity and completeness of the successive representations. For the purpose of this study, only a minimum number of system product levels have been defined. Although much finer differentiation is possible, the levels that are proposed are:

System: An organized set of software modules and/or subsystems, data base elements, and user procedures created to perform a set of specific functions.

Subsystem: A subordinate system consisting of one or more interacting modules, usually capable of operating independently of, or asynchronously with, a controlling system. More than one level of subsystems may exist, but are not normal in a software system.

Module: A software entity that is discrete and identifiable with respect to designing, compiling, loading, and combining with other modules. Several levels of modularization may exist.

A similar situation holds for the organization of work. On small projects, a breakdown by project phase may be fine enough but a finer breakdown is often desired. In structuring data for the repository, this study makes provision for receiving data at several levels of detail. The proposed levels are:

Project: All the work of the project as a single entity.

Phase: One of the recognized developmental phases of a software development project: analysis, design, code, test.

Task: Normally, any developmental phase involves several lesser tasks, some of which may be coterminus with the phase, or breakdowns of it, and some of which may be independent of project phasing.

Activity: A major task that is usually broken into several subtasks or activities, which in turn may be broken into further subtasks.

Job: A single sustained effort or operation, usually of short duration, such as a program compilation or preparation of a report, usually scheduled and budgeted as a class or as part of an activity, but not planned for a specific date or frequency of occurrence.

In planning and performing work, work elements (all work breakdown entities) receive a schedule and a budget (resource allocations). In practice, these are not always isomorphic with organizational, product and work structure as implied by Figure 4. That is, account numbers may cut across rather than being specific to a given product, organization element and task. Provisions must be made for this contingency but a standard structuring is desirable.

2.1.5 Performance Control

The objectives of management and quality assurance are to ensure the efficiency and quality of performance. Performance is evaluated by comparing the following values that are established for all items reflecting project performance:

a. An estimated, projected or planned value established before the work is performed. This value is input initially and/or updated periodically, depending upon the reporting period.

b. An actual, real or experienced value established after the work is performed or as accumulated to date, depending upon the reporting period.

Although it is reasonable to evaluate the quality of work, quality assurance usually deals with product quality evaluation. For the software configuration (product breakdown) this evaluation consists of comparing the current representation being reviewed to the past baselined configurations, including all changes and corrections that have been officially included in baselined representations. The configurations may be represented by listed and annotated elements, by detailed specifications or by actual programs and manuals. Deviations (errors) may be detected during formal or informal review or test or during operational use. Both the deficiencies and the corrective actions must be recorded along with costs and elapsed time if appropriate reliability and quality information is to be collected.

Efficiency of performance is normally assessed by noting deviations (variances) of actual performance time and costs from those planned. Both for products and work, however, derived measures are often more interesting. That

is, reliability and maintainability are more meaningful data than raw error rates, and "value earned" is a better index of performance than schedule and cost variance.

2.2 REPOSITORY OPERATIONAL CONCEPT

The manner in which the Software Data Collection System is to be used will be decisive in determining many features of the system. First, the system may be useful for both project management and for methodology research. Second, the information in the repository must be equally available to all qualified users. This requirement has serious implications concerning the privacy of the data and the desensitization of the data prior to its release to users. Third, the system may be in existence over a period of years in order to acquire sufficient data to permit reliable comparisons to be made. The data processing industry is still in a period of rapid growth and technological change and evolution of the repository may be considerable over the years. For instance, at the onset of the repository operation it may be technically feasible to collect only a subset of all the data that might be desirable. As collection techniques improve, as they might well through further instrumentation of production tools, greater automation of the software development process, better monitoring tools, and further product evaluations tools, many more variables (and a much greater data volume) might have to be added to the data base.

The system must be flexible in adjusting to user turnover, different modes of interaction, and unforeseen research demands.

2.2.1 Project Management

The data collected must support the needs of project management even though it appears that no direct support of project management is to be supplied by the repository. The basic goals of project management are to:

- Form and maintain adequate plans for the conduct of the project and for the definition of the product.
- Acquire and deploy personnel and other resources sufficient to achieve the plans.

- Monitor performance to insure that planned performance goals are being met.
- Adjust plans and redeploy forces if performance is unsatisfactory or exceeds expectations.
- Provide a stable and certain working environment in terms of controlling excessive change to work plans, product plans, and resource utilization, e.g., personnel turnover and reassignment, while using well-structured control procedures and decision mechanisms.
- Deliver the promised product on time and within budget.

As the project team level, control is exercised day-by-day if not hour-to-hour. For larger projects engaging a hierarchy of teams and specialized organizational units, management may not require as fine planning, monitoring or controlling. Top management and customer program management offices may need even coarser information. Individuals and corporations resist providing any more operational information than is strictly required. More is regarded as non-productive and an invasion of privacy. The level of detail submitted by the project to the repository has been a prime concern of this study.

2.2.2 Research

The data collection from participating software development projects must support the needs of basic research into programmer productivity, program reliability and the software development process. Note that much of the environmental information that a research program may require for valid comparison of projects is "given" for the individual project and would not be recorded unless a special effort is made to collect it. Also note that much of the close surveillance data needed for management control may not be pertinent to a research program. Many of the items that a research study might want to know, such as the number of day-to-day problems, the rigidity of enforcement of technical and managerial standards, and/or the particular employement of tools and techniques to solve a problem or do a job, are lost

in the daily conduct of business. In view of organizational resistance to the collection of data beyond that necessary to conduct the the project operation, there must be extra motivation, as in the conduct of an experiment, or extra compensation to pay for the data collection effort. Coercion may be exerted but is likely to meet with resistance, if not active sabotage, falsification, and evasion.

2.2.3 Desensitization, Privacy, and Security

The data collected for research purposes must be available to all qualified users, but must not yield information sensitive to a particular supplier. Since anonymity cannot be realisitically applied to data gathered for project management purposes, either some means of desensitizing it must exist before entry into the publicly available store or security measures must be exercised to prevent unauthorized access to proprietary and sensitive information. Since adequate guarantees of absolute security cannot realistically be advanced now or in the foreseeable future, it would appear that prescreening, filtering or purging of sensitive information must be done in advance.

The data base must also be protected from unauthorized and even malicious alteration. That is, persons should not be permitted to enter false data nor to alter, without special permission, data already stored. Some provisions for change must be made to correct errors in data entry and/or to accomplish any purging, summarization or other processing that the data base administrator might need to do to preserve anonymity or remove undesired data.

A configuration that offers a reasonable degree of privacy is to buffer all requests for information from the repository through a support facility at the repository. Data entry might still be done automatically but users would not be permitted direct access to the data. Instead, all requests for service would be submitted to a group of analysts who would be responsible for

ensuring the desensitization of the data before release to a requesting organization. Another alternative would be to provide a desensitized research data base that could be available for public access.

2.2.4 Growth Potential

If data is gathered over a period of years from a large number of projects and if the voluminous data generated by automatic tools is stored, a very large data base will accumulate. Since the data processing industry is still growing rapidly as ever more applications are found, and since continued technological innovation may be expected to create even more automated tools, an even faster growth rate must be expected.

Further, as the volume of data grows, the variety of data types will also expand. The initial data set may be a conservative one due to both technical and economic limits on collection capability. However, the desire to investigate new research areas and the development of new technology may introduce new data types. Data rejected as infeasible in the initial set may readily be economically available as new collection technology is introduced.

Therefore, both natural growth and the development of currently unforeseen demands require the data collection system and the repository to be able to support a much higher level of activity than it initially expects.

2.2.5 Flexibility

The data collection system and repository must be flexible to meet new and varied demands. It is expected that the Software Data Repository will be maintained in a central location and will initially reside in a Honeywell 6180 computer. Data will be entered into the repository from software development projects performed internally by USAF personnel and externally by contractor personnel. Projects will be located throughout the United States and may also be located on foreign soil. The precise number of projects, their likely duration, location, size and intent will vary as projects start and finish

and new contacts are let. Data from some projects will reflect experimental use of advances in techniques and methodologies that include new data types.

Flexibility is also needed because of the changing data requirements. Unforeseen research requirements, new analyses, and new treatments of data will evolve. There may also be shifts in policy concerning the data that is collected; for instance, a class of data may prove to have little or no relativity to productivity or program quality.

In view of the managerial and desensitization issues stated above, there may be a requirement to support project-specific subsets of data, and/or a special project may require, for security reasons or to avoid contamination of the larger data base, special subsets of data. Hence, a flexibility in meeting new requirements, situations, and modifications is required of the data collection system.

3. DATA COLLECTION TECHNOLOGY

The design of an adequate software data collection system is contingent upon the understanding of the current technology in data collection, storage and retrieval, and upon appreciation of some of the problems and issues inherent in such a data collection system. Based upon the models reflecting the scope and environment of the software development process discussed in Section 2, the study examined, first, the problems that are encountered in acquiring reliable software production data; second, the techniques that are currently employed to collect such data for military software development projects; and, third, the opportunities for improvement that might lie in such automatic collection approaches as software implementation monitors.

The guidance to be had from these studies can then be applied to the development of data collection procedures, reporting formats, data types and analytic techniques.

3.1 PROBLEMS OF DATA ACQUISITION

Data collection has, and will continue to encounter, some serious problems that cause software development data to be inaccurate, unreliable or have doubtful validity. One objective of this study was to investigate the impacts of these problems and to suggest solutions to them. To assist in this analysis, the project (a) searched the literature for previous studies of data collection problems, (b) included many queries concerning such difficulties in the survey of project managers, and (c) in December 1975, held a conference of persons associated with the proposed RADC repository, the USAF Satellite Control Facility repository, and the Army Advanced Technology Center repository to discuss data collection problems.

As a result of the literature survey covered in Volume/002, of the project manager survey in Volume/005, and of the conference in Volume/006, the principle problems facing a software data collection system are:

- Mensuration difficulties
- Instrumentation effects
- Unreliability of measures
- Reluctance to release data
- Cost factors
- Systemic problems

3.1.1 Mensuration Difficulties

The problems of mensuration are classes as (a) measurement, (b) instrumentation and (c) interpretation. Measurement involves determining what information is needed concerning the complex of project activities, products, reviews, resources and management controls delineated in the previous section, and defining the measures that may be taken to derive that information. A huge number of measures could be taken; however, not only is too much data costly to collect and analyze, but much of it may be of questionable pertinence. Collecting data just because it is readily available and/or cheap is not sufficient; it must be reliable and useful. The problem is to define a set of data that is possible to collect and that is useful to management and that will support research projects. The main objective of this project is to define such measures.

Instrumentation involves the when, where, and how of data collection - the insertion of probes into the process to be measured. The measurement method, the frequency of recording, the time and location of collection, the fineness of detail, the organizational level and the filtering and combining of data in collection are among the instrumentation problems addressed in this study. The principle objectives are to obtain measures that are as objective as possible and that interfere with the software development process as little as possible. The ultimate method is completely automated collection, but practical considerations require that much more subjective, intuitive measures be accepted.

Interpretation involves determining the meaning, value and importance of the measures in evaluating project performance and product quality. In short, assigning weights to the measures for various purposes. While a great deal of attention was given to the meaningfulness of measures in the compilation of recommended and potential measures, exact weights depend greatly upon evaluation goals and situational factors. Except for collection priority, recommendations concerning the importance of the measures are not made.

3.1.2 Instrumentation Effects

The act of measuring can influence the behavior of the process measured, a phenomenon known as "Heisenberg Effects". The motivating effects of knowing that one is in an experiment that causes participants to work harder and perform better are called "Hawthorne Effects". In short, whether the instrumentation has a deleterious effect such as consuming working time or irritating workmen by its interference or a facilitating effect such as creating additional motivation, these effects destroy the comparability of measures taken under one condition to those taken under another.

Feedback effects from knowing the results of evaluations probably cannot be avoided, but not knowing the results may also influence behavior. Although it may not be possible to avoid all effects, they may be minimized by collecting standard, objective data with as little fanfare or interruption as possible. Hence, the more automation and the more observations by independent auditors that are used the better. In order for measures from different projects to be comparable, they should be taken under equally biasing or motivating circumstance. Emphasis should be placed on evaluations of deliverable products and observable events that can be made without interference with the software development tasks, rather than on preparing reports and justifications of current status.

3.1.3 Unreliability of Measures

Without reliable measures, measures that mean the same thing each time they are taken, valid predictions and valid comparisons cannot be made. Although subjectivity and inaccuracies contribute to error variance, the principle problem in defining the measures to collect lies in a lack of standardization. Differences in the measures collected arise from non-standard terminology, differing definitions of measures, differences in the collection procedures and conditions, and variation in the organization and functioning of the software development facility and process that are measured. (Project differences and variations in the data collected are seen as the most universal bar to software methodology research by project managers.)

Two approaches are possible to attaining more reliable measures: standardization and factorization. Standardization seeks to control variation by holding everything constant; factorization by measuring the factors that cause variation and trying to account for, or partial out, their effects on the measures. In this study, both approaches to more reliable measures are taken. Standard measures are defined and standard terminology is used and advocated. On the other hand, the study cannot force all projects, data collection methods, and environmental conditions into a single mold and indeed it is undesirable to do so. Hence, the study recommends collecting a considerable number of measures of environmental conditions both for the project and for the measurement conditions. As much subjectivity as possible will be removed from the measures; that is, objective measures will be defined where possible, but some measures will continue to be subjective ratings.

3.1.4 Reluctance to Release Data

Resistance to management control, as evidenced by a reluctance to provide information on project performance at all levels of management, is an almost universal problem. Control systems are seen as coercive and threatening (and often are), evoking resistance behavior that varies from reluctance and reduced cooperativeness to outright falsification of data and sabotage of the collection system. Project personnel feel their privacy is being

invaded and that punitive measures will result from communicating bad news. Corporations are equally reluctant to invite punishment and fear that proprietary information and trade secrets will be revealed. Very real problems exist in obtaining honest data if heads roll as a result of giving it.

There are some suggested procedures for minimizing resistance; while effective when performed by project monitors and managers, it is difficult to ensure their effectivity by any data collection system however well designed. The members of the development project should understand the goals of the data collection effort and the utility of each measure. To the greatest extent possible, the project members should participate in the setting of project goals, and at the least should know what is expected of them and what the rewards and punishments are for meeting or not meeting those goals. Reporting errors and performance variance should not be a finger-pointing, blame-assigning process, but rather an effort to solve problems, avoid trouble and improve the tools and techniques used in performing a project. To the extent possible, projects and project members should be absolved of personally reporting information that would reflect, favorably or unfavorably, on their performance. That is, reporting should be as automatic and as impersonal as possible.

Obviously, these are measures that seek to make the data collection system as non-threatening and as objective as possible. It is equally obvious that the measures depend upon the project monitor and the management of the project to initiate and carry them out. Not all managers are equally skilled in doing so and many managers believe that the only effective management is tough-minded and coercive, punishing any infractions with decisive action. Under these circumstances, even though project performance may be improved by the strong management, reporting data may be expected to reflect the efforts of projects and project personnel to avoid punishment.

3.1.5 Cost Factors

Although the overt costs of a formal project and configuration management system are quite small (estimated generally at 3% of project costs), the hidden costs in terms of project member time to prepare reports and of interference with technical work may be much larger. Disruptions to the train of thought may not be as serious an interference with project progress as is the time taken away from work and the irksomeness of preparing reports, but 10% of the project managers surveyed thought it did occur. For the data collection system, gathering research data is seen as over and above that necessary for the management of the project. Project resistance to incurring the additional expense in project interference and extra effort may be expected.

Two general actions may be taken to counter the costs of data collection: first, actions to minimize the costs, and second, directly defraying the costs via contract coverage. Automation, standardization and more granular (coarser) data are seen as ways to reduce costs. Direct coverage of data collection costs in the contract not only pays for the extra effort involved, but provides motivation for reporting and removes much of the reluctance to do so. Independent audit and data collection agencies and procedures may not only reduce the degree of interference with the technical work but may increase the objectivity of the measures. Some of the costs and frustrations encountered in data collection are engendered by lack of understanding of data collection goals, measures and procedures. A standard, well-understood, widely-used system would not only make reporting easier, but reduce frustration and data unreliability. Automation may yield a plethora of cheap data, but is itself expensive to develop and install in a diversity of projects. Further, most automated data collection is very fine grain and specific to a problem and not generally pertinent to project performance. However, increased mechanization of data collection through project monitor systems provides a sort of automation that may be expected to spread.

3.1.6 Systemic Problems

There are several other problems that are inherent in the normal behavior of systems. These include delayed responses, filtering effects, averaging and summation, forecasting efficiency, and stability.

Time delays occur between the occurrence of an event, the reporting of the event, and the reaction to it. Inappropriate response may then be made to situations that have already been corrected or grown worse. This delay can be minimized by maximizing automation.

Information that passes through several level can get distorted by averaging or interpreting. As much objectivity as possible should be sought and algorithms should be used for combining, filtering, and averaging data to remove personal bias.

Instability is caused by personnel turnover, requirements changes, and incorrect responses. This is a problem without an obvious solution. Interpretation of data must be done with the knowledge that the system measured is not static.

Forecasting is not an exact science. For best results, planning and forecasting should be iterative. Contracts of the Cost Plus Award Fee and Cost Plus Incentive Fee types allow cyclical reestimating and provide an incentive to complete the job within a reasonable time period.

3.1.7 Problem Summary

The degree of seriousness of the problems depends in part on the types of data collected. For instance, in the Survey (Volume/005), it was found that managers were most reluctant to release cost data and most willing to release software problem data, i.e., change and error statistics. Performance and productivity data, including software quality evaluations, were deemed the most useful measures for management, but were also deemed most subject to the distorting effects of optimism, bias and subjective processing during reduction for summary reporting.

Although reluctance to release data that might reflect negatively upon project performance and general resistance to managerial control are universal problems for which there is currently no apparent final solutions, the answers to other problems found in more objective measurement and standard practices also serve to alleviate resistance. A standard, well-established, well-understood data collection system - procedures, formats and measures - not only provides project comparability for research but also improves data integrity and reduces the costs of collection.

A detailed discussion of these factors and their impacts on software development data collection will be found in Volume 002 of this series, "An Analysis of Software Data Collection Problems and Current Capabilities."

3.2 CURRENT DATA COLLECTION TECHNIQUES

In general, military standards, regulations and directives provide for a full range of project performance and configuration management practices, but do not specify specific standards. Instead, many standards, procedures, and reporting forms to provide the desired level of control are established either by contractor, project management plans or program office directives. Consequently, a considerable range of software data collection practices exist over the full scope of military software projects.

In studying the implications of current data collection practices, SDC reviewed not only such military standards as MIL-STD-483 and AFR 800-14, but more detailed procedures from project management plans, plus instructions issued by software acquisition and maintenance agencies. The results of this investigation covering reporting practices for performance measures, configuration control, documentation practices, and product quality reviews are reported in Volume/002. Numerous examples of special reports and data collection forms are given there.

3.2.1 Environmental Characteristics

No explicit data items describing a project are specified for collection by most projects except estimates of system size and resource requirements. In the past, research into the impact of such environmental factors as customer relations, familiarity with the area of application and the customer, project organization and personnel mix, had to depend upon retrieval of this information from project plans, contracts and progress reports, or from the personal recollections of participants. Since many projects diverge from their original implementation plans and since after-the-fact estimates frequently reflect much strong feeling and bias, these data have been found to be highly unreliable, as well as answering no definitive questions concerning environmental impacts. So far as research support is concerned, this is a serious deficiency in current data collection practices.

3.2.2 Project Performance Characteristics

Military practices in the collection of project performance characteristics were found to be, in general, more well-rounded than most commercial practices. The basic approach is through Work Breakdown Structures (WBS) and Resource Utilization reports. The WBS uses both a product breakdown (configuration Items) at top levels and task (for function) breakdown at lower levels to help structure and organize performance reporting. Functional tasks are assigned to organizational elements and the account numbers used in reporting are tied to WBS element identification.

Unfortunately, not all projects use WBS to structure and organize accounts Reporting is often confined to the top level configuration items, entailing much summarization of data with attendent ambiguity of detail. As used, the WBS also tends to be somewhat awkward and inflexible (difficult to change). That is, it does not reflect the changing configurations in successive system representations. Also, if the initial WBS is very detailed it may represent premature design decisions that can hamper both progress and reporting.

At one time, PERT schedule planning and reporting was very strongly advocated for military projects and is still recommended, but a more broader range of performance reporting schemes are now advocated. Reporting frequency ranges from monthly to quarterly; granularity of information ranges from major tasks or functions to project phase and principle configuration items. Hence, so far as data integrity is concerned, reporting delays and summarization permit many systemic effects to operate.

Resource utilization, where used, is generally restricted to manpower utilization reports, but computer time utilization may also be included. Unless otherwise required, most resource reporting is given as dollar costs. Even where required, reports tend to give actual utilization for past performance and projected utilization for future estimates. Unless historical records are kept, it is difficult to compare actual to planned performance. Although some contrary examples were found, projects seldom collect resource expenditures against system modifications as reflected in engineering changes or error corrections.

3.2.3 Configuration Characteristics

Although several military agencies have specified configuration status reporting requirements, the full range of reporting is seldom actually required or enforced. Some agencies do practice strict configuration controls (the USAF Satellite Control Facility, for example), but the practice is not generally employed. Hence, although it is alleged that high change rates and "soft" baseline control are frequent causes of cost and schedule overrun, actual statistics are not widely available. For most systems, provision is not made to accumulate error reports into configuration status reports, and reliability studies are usually done outside the regular status reporting system - that is, they are special efforts.

3.2.4 Quality Control Data

Quality control for military software projects is exercised through a series of specified reviews, audits and tests. Reviews and tests normally establish the excellence of the product and baseline its configurational content and structure. For research purposes, it would be desirable to have records of all alleged discrepancies detected during a review and some history of their processing and disposition available. Some projects do publish review minutes and/or action items and many use software problem reports (error reports) during integration and system testing, but normally these are present only if the testing is done by an independent test agency.

Although test documentation tends to be fairly formal, providing a firm basis for issuing an error report (failure to meet a test), review criteria are not often formally stated. Hence, reviews tend to be quite variable and subjective, resulting in noncomparable data.

3.2.5 Summary of Current Military Practices

While it is true that there are a large number of military standards, regulations and instructions dealing with project administration and configuration accounting, actual utilization and enforcement of the recommended practices is often lax. There is a relatively low level of standardization in the data items, report formats, and reporting procedures. Approximately the same software development model is used by almost all regulatory agencies, but the many minor differences represent a major obstacle to basing a software data repository on current practices. In order to build a software data collection system around current manual procedures, additional data requirements would have to built into contracts, and standard report forms and collection procedures would be needed to derive a coherent set of data. Further, guides for both regulatory documentation and the proposed data collection system would be necessary to ensure standard interpretation of regulations and standardization of the information collected.

3.3 DATA COLLECTION MONITOR SYSTEMS

In recent years, a number of data collection systems have been proposed and/or developed. This study looked at several of the systems (both prepared or in use), including SDC's IMPACT and MITRE's SIMON, that employed data collection monitor systems. While it is somewhat difficult to compare systems that were developed with differing objectives and scopes of applicability, it is possible to make a comparative evaluation if current military practices and this study's derived data requirements are taken as the criteria for analysis. The detailed evaluation of these systems is contained in Volume/002, but some of the conclusions drawn from this study is summarized below.

3.3.1 IMPACT

IMPACT is a software data collection monitor designed specifically for software project management. Of the systems evaluated, IMPACT collects the most balanced set of project control, configuration control and quality control data. However, it collects only a minimal amount of environmental data. IMPACT is a very flexible and comprehensive tool, but using it to its full capacity demands a staggering volume of input. The system flexibility must be offset by detailed user manuals and examples. Automatic data acquisition consists of a log of operations and computer time accumulated from the program library executive.

3.3.2 SIMON

The SIMON system is geared to a somewhat more restricted model of software development. Schedule control is aimed at project phases rather than individual tasks; configuration accounting is aimed at individual program modules and ignores most of the provisions of military configuration status accounts. Although SIMON is geared to collect limited data for research into factors affecting software quality, the level of detail seems insufficient to support reliability research. However, SIMON does have an interface with software production and analytic tools, obtaining such information as system structure and set-used tables. Complexity measures of software programs will be an important data set when adequately developed and integrated into the operational system.

3.3.3 BMDATC Quantitative Data Base

The Army Ballistic Missile Divisions Advanced Technology Center's Quantitative Data Base has quite limited research objectives, but is intended to provide software development data from BMDATC associated contractors and the Advanced Research Center (ARC) contractor. The limited project performance data includes mandays and computer time spent by configuration item and project phase. It records changes and error reports and tracks resources expended on diagnosis, analysis and implementation of modifications. It also gathers module and test statistics, but does not produce configuration status reports per se. Although intended for research on programming methodology, such project environment data is recorded independently of the established data base and is consequently not available for automatic retrieval and analytic work.

3.3.4 IBM Management Data Collection and Reporting System

This proposed system is oriented toward the IBM structured programming software development model, and may not be applicable to investigation of other techniques without considerable expansion. The information collected is after-the-fact and not intended to support project management or configuration control. The idea of successive baselined system representations (configurations) is not obviously supported since milestones are not overtly identified. Personnel assignments and schedule maintenance are not specified, and not monitored by the system. No connection is made between products and project activities, resulting in a lack of data necessary to support productivity analyses. Error statistics, module statistics and modifications are collected, as in most monitor systems, but are not cast in the military configuration accounting mold used by most of the projects that will be supporting the RADC Software Data Repository.

3.3.5 TRW Software Reliability Study

This research study had the very limited objectives of a detailed study of software error types, techniques for detecting and diagnosising errors, and improvements in software reliability. Since the study is not concerned with collecting other software development data, little can be said about the overall adequacy of data collection with regards to the RADC repository. The TRW study does yield insights into the benefits and costs of collecting a detailed sample of error data. Not only did an extensive classification of errors result (perhaps more detailed than would be useful for most projects), but so did guidance on collection and analysis procedures and interpretation of error analyses. For reliability modeling, more extensive operational information of the software system and more information on the types and amounts of testing employed should be collected. As an indicator of the amount of time and effort that must go into detailed data collection, the study has real value. Data collection and analysis do have associated expenses. This study gives an indication of the amount of effort required to set up and administer a data collection system.

3.3.6 Evaluation of Software Monitors

While there is no one system yet available that provides all the capabilities that would be desirable to meet RADC's data collection requirements, the software monitor approach is a valuable asset to a collection system. IMPACT demonstrates the tremendous amount of data involved in the detailed planning and control of a software development project. SIMON illustrates acquisition of data through the integration of the project monitor with programming support tools, as does IBM's proposed system with the program support library. The TRW Reliability Study illustrates the amount and type of work that needs to be done to properly delineate the range and organization of each of the parameters involved in the study of software quality. Although there are major differences in the operating philosophies and structures of these monitors, there is a large commonality in the data items collected.

This study concludes that one of the most effective ways of enforcing standardization of data items, collection procedures and reporting formats is through a standard project monitor system. Much work remains to be done in terms of the details of data base structure and content, integration with program production library operations, and utilization of data obtained from instrumented programming support and program analysis tools, but the ultimate inclusion of project monitors in the software data collection system is strongly recommended.

3.4 EVALUATION OF REPORTING FORMATS AND DATA TYPES

In addition to the data forms and report formats utilized by the various project monitor systems and specified by the military standards and regulations, this study evaluated packages of reporting forms used by several internal SDC projects, some from the IBM Federal Systems Division, and by the structured programming test project at Vandenberg AFB. As with the project monitors, some of these reporting formats were intended for both research and project management. Where the forms were intended for research, more environmental data were collected. However, these data collections tended also to be more summary and after-the-fact than those intended primarily for immediate management objectives. Although the same classes of data were generally collected, the collection and report formats varied a great deal. If a viable software data repository is to be realized, standard reporting formats are necessary.

3.4.1 Environmental Data

Very little environmental or project description data are collected by either project monitors, military program offices or any of the projects that the study investigated, except the Vandenberg experiment. Hence, although these are the variables that many investigators and theorists believed most influencial in determining productivity, few specific environmental data types are readily available for application to the needs of the RADC data collection system.

Many of the data items associated with project environment descriptions tend to be subjective opinion and must be backed up by explanatory material to be understandable or comparable. In the area of customer relations, for instance, listing the type of contract is an objective parameter, but evaluation of the quality of requirements specifications, customer interaction, adequacy of personnel skills, organization effectiveness and other evaluations of risk and stress are quite subjective parameters.

Evaluations of the organization and constitution of the resources of the project - personnel, machines, and program tools - are more objective parameters. Skill levels, years of experience, and education are easily identified, as are the kinds and numbers of machines and support programs. Ratings of the actual adequacy of these resources in satisfying the needs of the project, however, are much more subjective and their reliability is questionable.

There is also some objective information that can be obtained about programming methodology, such as the type of technique, where the technique was applied in the development process, and the cost of acquisition and utilization. Because of the diversity of methodologies actually employed, a specific use of a technique should be further described or the researcher cannot determine the extent to which the technique or procedure was actually followed.

In most of the studies involving environmental factors as reported in the literature (see Volume/003), the data gathered was after-the-fact reports. To obtain a reasonable evaluation of environmental factors, one needs at least an original estimate of what the project thought its environment and resources would be and a final appraisal of the actual project attributes. Since environments are dynamic, it might also be profitable to obtain intermediate estimates for those modules of data that are affected whenever major changes occur. For current systems, very few of these environmental parameters are collected; where items are collected, there is almost no standardization.

3.4.2 Performance Data

Almost every system investigated collected performance data of some sort. While the information collected is similar - projected and actual schedule performance and resource expenditures - reporting formats vary greatly. For management reporting, the widespread use of PERT - type systems has fostered a report format generally called a "management summary". In a management summary, work activities are listed with scheduled start and completion dates, or durations, an indication of actual performance in relation to planned performance, and a positive or negative variance. Allocated and actual resource expenditures are also included with their variance. The schedule performance is frequently illustrated with a chart that depicts the schedule points. The activity data reported may or may not be associated with a particular product.

Obviously, a management summary represents summarized and processed data whether produced manually or by a project monitor. Raw data is usually collected from several sources, including task orders, cost logs, computer logs, and schedule reports. Task orders may contain prose descriptions of the jobs to be done, or they may refer to standard tasks or portions of a contract statement of work. For many projects, the tasks reported are of the semi-standard project phases.

Since performance data is used for management control purposes, it is normally reported frequently, such as weekly for internal control and monthly for external control. In some instances, direct evaluations of productivity in terms of "value earned" (production variance divided by resources expended) is calculated, usually translated into a common base of the dollar value of the product and resources. If the production units are lines of code or pages of documents, a reasonably objective value earned is pratical despite the likely variability in 'value' due to product complexity and difficulty.

3.4.3 Configuration Data

The relatively elaborate configuration control records and configuration status reporting procedures prescribed for large development projects are not often used by smaller projects. In general, even where formal engineering change and discrepancy reporting procedures are used, modifications are accounted for separately rather than associated with the products modified. IMPACT is the only automatic monitor system examined that offers a configuration accounting capability, either against specification documents or against identified configuration of functional, design and module representations.

The processing of proposed modifications (changes or error corrections) varies from being a very formal system-the problem report or modification request is submitted to a configuration control board clerk who assigns an account number; logs it in the system; distributes it for review or investigation; logs in replies; places it on the agenda of the configuration control board; records the decisions of the board; updates the official copy of the modification request for any changes as a result of the investigation; logs any change of status that may occur in the processing of the request; determines the product representation and version affected by the modification; provides change notice identifiers when the modifications to specifications, program modules, and other products are available; and logs the release of the modifications, including the version and mod identifiers of the particular products that the change affected. In practice, little of these activities appears in official reports except the request identifier, the product affected, the current status, and the change notice identifier.

If change control operates effectively, regular configuration status reports will be issued, usually at monthly intervals. If intermediate status is not kept, the only record is the request and notice of change. If a program library operates, library listings of the program mods belonging to each version and release of a system may be issued.

At a minimum, the problem report proposing a modification and the change notice giving the disposition of the request should be monitored. In a more elaborate system, control logs and status change (update) forms may be used. All this information may be reported in configuration status and change status reports, including the official processing steps that are scheduled and logged.

3.4.4 Quality Control Data

Little data is currently recorded for project reviews, for data reflecting the scheduling and passing of reviews. For research purposes, it is necessary to have an index of product quality, such as the number of deficiencies detected per product, and/or an index of the seriousness of the deficiencies, such as the time to resolve or revise the problems. While some projects do file problem reports, issue Action Items, or publish minutes and comments, the general mode of operation attempts to avoid this process, if possible.

Problem reports are quite widely utilized for product tests. Less commonly used are records reflecting test case/procedure failed and/or the function failed. In some instance as in IMPACT, provision is made for recording the number of passes made at a test, the test results (pass/fails), and the amounts of computer time expended.

Almost no instances were found in military projects or project monitors where run statistics were kept on systems in operational status. Neither failure rates, mean time between failures or operational evaluation results enter software data collection systems. Some systems do account for errors detected and corrected and some record cost information. However, the records kept are not adequate to support a wide range of reliability research, resulting in the need for every investigation of reliability to engage in a special data collection effort.

4. DATA REQUIREMENTS FOR PRODUCTIVITY AND RELIABILITY STUDIES

One of the central issues in this study has been to determine what data items to collect to evaluate project productivity, program reliability and software development costs. Obviously, it is impossible to foresee data that represents all aspects of the software development cycle. That is, at some time there may be a research requirement that encompasses a set of data items not considered important to the repository currently planned. It has been recommended that the repository remain flexible and extendable to encompass such data in the event it becomes important.

For the most part, selection of the data items has been based upon the utility of the parameters for the specific research objectives stated above. While some attention has been focused on the accessibility and costs of collecting specific data parameters, these have been secondary considerations in determining whether an item should be included in the collection process. In determining utility, SDC leaned heavily not only upon the literature, but upon priorities of data importance established by both RADC and SDC personnel.

A discussion of the justifications for considering sets of related data parameters are presented in Volume 003, Data Requirements for Productivity and Reliability Studies. The recommendations concerning specific data variables are presented in the Compendium of Procedures and Parameters, Volume 007 of the Data Collection Study. A brief summation of the investigations leading to the data requirements recommendations is presented below.

4.1 PRODUCTIVITY ANALYSES

The growing field of software development, which is involving all aspects of daily living and demanding ever more expenditures for the production of software, has focused wide attention on determining the factors that impact individual and joint productivity. Many studies in the past have been undertaken to identify factors relating to productivity with the result that a myriad of components are felt to be important contributors to productivity

and performance. For the most part, productivity involves human factors that are not only difficult to define but also to measure. Productivity itself is defined in various ways, further complicating attempts to measure it. The costs of developing software are currently estimated at $15-20 per line of code (with even higher rates for complex systems), while the reliability of the code produced is sometimes measured at the low rate of one error per 100 lines of code. For expenditures of that nature, both project productivity and software reliability are seen to be of prime importance to the overall software development process.

In the survey of the literature addressing project productivity, many factors were found to impact productivity by many different authors. Consensus of opinion is rare and it appears that human factors scientists have arrived at their individual opinions by direct observations rather than by use of instruments and/or systematic procedures that measure or assess productivity. On the other hand, while there is a definite requirement for assessment of this type, little is currently being done in the software industry to accumulate and analyze large volumes of data consistantly and methodically obtained that supports productivity studies directed towards improving the quantity of work produced. It is obvious that a data base of this nature requires a large financial as well as an intellectural commitment that may result in nothing more than a reiteration of the results of previous work. However, after examining the literautre and questioning a sample of software project managers, both by questionnaire and by symposium discussion, this study supports the concept of a software data repository to support research of this nature. The lack of a data base containing a large sample of diversified software development data that has been collected in a methodical, standardized format on a consistent and timely basis may be the element most responsible for the lack of conclusive results in past productivity studies, as well as other studies, such as reliability, cost estimate, and software quality analyses.

While it is possible to list key factors used in past productivity studies (see Volume 003 of this series), quantification and subsequent collection of the data remains a most difficult problem. For example, communication is considered by several authors to be a key factor in performance. Measurement of the degree, extent, and content of communication within a project is an obviously difficult, if not impossible, measure to obtain. Instead, analysis of the type, size and number of organizational groups and the managerial techniques used, such as chief programmer teams, can demonstrate the individual project's approach to the communication problem. In this manner, it may be possible to derive meaningful information that directly influences productivity and performance without the necessity of attempting to quantify human factor abstractions.

Productivity has in the past and continues to be measured by the number of work units produced per unit of time or unit of resource. A work unit may be a line of code, page of documentation, record of data, punched card, or any combination of these and other units. The general measure of productivity is expressed in lines of code. However, even this unit is ambiguous since the notation of a line of code does not indicate the language constraints inherent in the line of code notation. It is well recognized that a line of source code is less expensive to produce than a line of object code, without even considering the other software attributes of size, complexity, type of application, etc. Further, the amount of analysis contained in the design phase of software development may be the single most important activity and may contribute more to reliability, portability, maintainability and other abstract qualities than any other activity. The level and extent of the design analysis is conspicuously missing from productivity measurements. Therefore, the current productivity measurement conventions are felt to be somewhat deficient in their attention to extremely important and essential work contained in the development process due to the difficulty in quantification of abstract processess of this nature.

Perhaps reducing the productivity rate to number of source code statements per manyear of effort is not only administratively expeditious for arriving at cost figures, but also provides a fast and simple method for comparative

evaluations between software systems and/or system development companies. However, in order to support productivity analyses that may indeed provide useful insights into increasing productivity and performance, more data are required than total size of the delivered software, number of man-months required to produce it, and total resources spent during production.

One of the current trends in the analyses of project productivity involves using the traditional measuring algorithm while altering the methodologies employed by project personnel. In this manner, the need to collect large volumes of human factors, as well as production data, is circumvented as long as a log of the programming techniques employed by project personnel is adequately maintained. Unfortunately, this approach does not consider some important and well documented psychological effects inherent in a sample test group. If all other factors can be duplicated, including personnel, software, working environment, customer interactions and direction, etc., other factors such as the Hawthorne and Heisenberg effects must be accounted for in the final productivity measurements.

The conclusion that this study makes is that the field of productivity analyses is an extremely large area for study. An historical data base containing large volumes of data representing many aspects of software development appears to be essential to productivity studies. For example, personnel skill levels, training, and educational backgrounds form a set of conditions that obviously impacts performance. Although this particular area of study appears to be of little immediate interest to RADC, it is thought to have significant impact on performance by human factors scientists. In 1975, it was estimated that university trained information systems personnel comprised 35 percent of the analyst/programmer/manager work force, although surveys indicate that in state data processing agencies, the percentage of people receiving formal technical training is somewhat higher. Further, it was found that not only is the amount of training important, but so is the adequacy of that training, including such factors as instructors' qualifications, subject material, currency of programming technology, and technical standards. These areas of productivity analyses have been examined in the past without impacting the total software industry. Perhaps the lack of a

single, significant contribution to productivity in this field has had the effect of minimizing the emphasis of attention now being committed to the study of educational background, training, listing, etc. However, it is the conclusion of this study that these and other human factors data should be included in the repository. Again, the purpose of this study has been to determine data requirements necessary to support productivity studies. The exact productivity studies to be conducted at RADC have not been delineated and it has not been within the scope of this study to define such. Although the current areas of interest have been considered in determining data requirements, it is felt that a broader, longer range objective must necessarily be inherent in the formation of an historical data base. For these reasons, this study concludes that the data requirements necessary to analyze productivity include data from the following areas:

- Environmental Attributes - Those factors unique to the individual project that are mainly concerned with the people element, including the customer interactions and requirements, the man-machine interfaces, the organizational structuring of project personnel, the qualifications of project members, and the attributes of the software problem itself which impact individuals, such as complexity and size.
- Project Performance Data - Those real and measurable data reflecting the amount of work performed and the amount of resources allocated and expended, as constrained by time and deliverables.
- Product Quality Characteristics - Those data reflecting actual product behavior and structure as demonstrated by the use of specific project tools and aids.

In conclusion, collection of data supporting all productivity analyses can not possibly be attempted. Instead, this study recommends a data collection scheme that can be expanded to meet evolving RADC needs, while providing data sufficient to determine productivity measures while analyzing project and product dependent variables that may have impacted the productivity measures.

It is clear that project performance data must be collected in order to evaluate the amount of work accomplished - information that is basic to productivity analyses. This study recommends that a standardized method for obtaining progress information on all products for which work was performed during the reporting cycle be initiated. By using a reporting technique applicable to the definition and collection of all work data, productivity measurements may be obtained that are not necessarily tied to any one productivity notation, i.e., lines of code, pages of documentation. At a minimum, a reporting system wherein the software products are defined, work progress made during the reporting period for each product is submitted, and all resources allocated and expended are accounted for, will provide productivity measurements. The results of this information coupled with project environment data, such as project progress and summary reports, computer utilization data, project manpower summaries, etc., should meet the basic requirements of productivity studies. Further, the collection of the product's structual and operational characteristics may also contribute to productivity analyses by providing insights into significant product factors that impact the production process.

4.2 RELIABILITY ANALYSES

Software reliability has been under scrutiny since the first development of related programs. Unfortunately, the problems existing then still exist today and perhaps the industry is not much closer to producing error-free programs than it was decades ago. This study has examined past and present attempts to improve software reliability, resulting in the following conclusion.

No one set of data parameters collected for research purposes will significantly support a wide range of reliability analyses. Because of the diversity of studies of this nature and the specialized parameters needed, a data collection effort geared to the acquisition of software development data sufficient to form an historical data base with the intention of supporting current and future research needs in the general field of reliability is a prohibitively large and costly task. Consequently, this study recommends the

collection of specific sets of reliability-related data sufficient to support existing and current research needs, while developing tools and techniques that can be integrated into a flexible and expandable data collection system leading to the support of future research requirements for reliability studies as they become apparent.

The field of reliability analyses includes studies covering every aspect of software development from the requirements specifications to design methodologies and languages to testing principles and development of test tools to measuring software failure rates for reliability predictions. The amount of data generated by and for any one of these investigations may negate the possibility of concurrently collecting data for any other study. Therefore, although it is apparent that detailed error data and failure rates are necessary data to collect for current RADC requirements, their utility may be short term and applicable to a limited field of research. While storage of this type of data may consist of both data files for machine storage and hard copy library storage, methods for systematically accessing large volumes of detailed data are necessary if the data is to be useful, either presently or in the future.

Some of the current RADC work in reliability consists of modeling and software error analysis. Reliability modeling is directed at predicting a measure of software reliability by accumulating operational data of a given software system, the number and type of errors detected, and subsequently, producing a function which predicts failure rates. There are numerous methods currently being used in reliability modeling, most of them requiring similar input data. The basic problem of this type of reliability analysis may be exactly that fact. A statistical analysis of similar software development data obtained from entirely dissimilar software projects, including application, complexity, operating systems, computers, methodologies, personnel, etc., is being used to mathematically predict the reliability of that software. Further, the data and measurements needed for the different modeling techniques are not exactly the same, resulting in the necessity of a myriad of collection schemes; the modeling predictions have little or no

applicability to large development projects in an evolutionary state, and the problem of introducing new errors in software systems while correcting previous failures is not within the scope of current modeling technology. Consequently, a large data collection effort geared to support a small interest group of this nature is not recommended.

Research work on the nature and density of software errors, on the other hand, appears to be a meaningful endeavor that may affect all phases of the software development life cycle. Collection of error data must be initiated with continual evaluation given to the operational methods employed in the collection scheme, categorization of error data, and uses to which the data are put. Collection of error data implies large volumes of detailed data at high costs, a process which must be methodically and carefully examined.

In summary, this study recommends collection of three types of data to support current research requirements in software reliability, including:

- Environmental Attributes - Those factors unique to the individual software project that impact the software product, both directly and indirectly. These factors include size and complexity of the software application; familiarity of project personnel with the specific application and development environment; tools, techniques and concepts employed during the development process; and the time period allocated to each phase, as well as total elapsed time, given to perform the job.
- Project Performance Data - Those factors reflecting the actual amount of work performed in a measurable time period; the management provisions, both planned and actual, taken to monitor work progress and product quality as demonstrated by milestones, reviews, deliveries and/or other formal or informal examinations; and the reporting vehicles by which management monitors the discrepancies, errors, changes and other related data made to the developing product during the testing period.

- Product Quality Characteristics - Those factors revealing product structure and operational behavior that may directly impact the reliability of the product.

4.3 SOFTWARE DEVELOPMENT COST STUDIES

In addition to the examination of data requirements for productivity and reliability studies, attention was also focused on the factors contributing to the ever-increasing costs of software development. It is generally agreed that one of the greatest services that a data repository could provide would be a data base of detailed and reliable cost data from which the salient cost factors for software development could be isolated.

Complicating the process of cost analyses is the lack of uniformity and visibility of the methodology used to define, plan and distribute the total work package as it relates to available resources and deliverable products. In this regard, collection of software costs data reported against a uniform work breakdown structure defining work tasks and associated products should initiate a standardized approach for collection of costs data and the evaluation of cost estimating techniques.

In the area of costs, the evidence seems to indicate that it is the gross inaccuracies in the original estimation of costs rather than inefficiencies in controlling costs that create the aura of financial irresponsibility surrounding software development. The process of estimating the different components of a given software job, deriving the proper weight or relative importance of a task for the particular project, estimating the resources required to perform the tasks, and achieving an optimal allocation of available resources is complex and difficult. For instance, the relative proportionate cost of analysis, programming and test shift dramatically with system size. That is, the programming cost for a small system comprise 60% of the total costs as compared to 15% of a large system. In a like manner, testing costs for a small system are about 15% compared to 45% for the large; design and analysis costs are 15% for the small, 40% for the large. Those tasks requiring a great deal of coordination, communication and interaction grow much faster with system complexity than those that can be performed

relatively independently. In actual practice, since analysis, design and testing are less well defined than programming and more likely to be performed inadequately, the true cost ratios may be even more disparate. For accurate estimates, the software development project must be broken into much smaller components than those three project phases, and much more accurate costing formulae applied since the ones available yield such poor results. Data must be obtained at many strategic points in time and location in the development cycle to permit proper evaluation of the contributing factors.

Not only is there a current lack of understanding of the components of software development and of refinement in the estimating process, but the data that are collected appear badly contaminated through subjective estimates and/or inclusion of the costs of irrelevant or seim-irrelevant tasks, such as training and administrative duties. To derive valid estimates, cost data must be systematically collected in a reporting period of short enough duration and a task breakdown small enough to avoid the perturbations of subjective estimates and lack of detailed precision. This situation is further complicated by the sensitivity accorded cost data by project managers (see Volume 005 of these reports). It appears that managers are more willing to report manpower and machine utilization than dollar costs. As resources, both of these are much more complex and less defined than dollar costs. That is, personnel classifications are far from standard in the industry, a condition that professional and governmental agencies are seeking to remedy. Obviously, a thorough description of the machines is required for proper evaluation of machine-time expenditures. In designing a data collection system, it is recommended that considerable flexibility be provided in the data collection forms to describe the tasks, products, and resources involved in cost data collection. At some later date, perhaps more standard personnel classifications and task descriptions may be established, but at the present time it is deemed infeasible to force a standardization scheme of this nature.

In many ways, cost analyses are a counterpart of productivity analyses and the factors that are involved in one are involved in the other. Hence, for understanding of the cost factors, much environmental data needs to be collected and analyzed just as for productivity.

In summary, in addition to the project's performance data collected to support productivity and reliability studies, cost data relating to the project's work breakdown structure, although sensitive to public scrutiny, may be collected with some additional effort. These data will form an historical data base sufficient to analyze the allocation and expenditure of resources by the project, as well as providing data for related studies in the definition and allocation of the work package and the resource estimation process. The data defining project environment and product quality together form a data base from which comparative cost studies may be realized.

5. DATA COLLECTION AUTOMATION REQUIREMENTS

The nature of the interactions of the data collection system with the central software data repository will be influenced by the type of data structure used, the data management functions supported, the configuration of data entry and data storage devices employed, and the manner in which the data collection and repository system is managed. The detailed investigation of these concerns is reported in Volume/004. A short summary of this report follows.

5.1 DATA STRUCTURES

In determining the relative advantage and disadvantages of various data base organizations, this study evaluated the degree of centralization/decentralization of the data base, the physical and logical structures available, and the access methods that might be employed. Evaluation criteria included the relative costs of storage, flexibility in meeting users' needs and changing conditions, security, and ease of implementation and use.

5.1.1 Degree of Centralization

The issue of whether the data base should be located in a single, centralized repository or distributed among several regional or local data bases depends upon the utilization of the data base. The data base may be used for long range research only, for short term study of specific issues, and/or for project management.

(Since short term studies of specific issues are not the principal objective for which the repository is to function, a discussion of the advantages and disadvantages of centralization to this type of study will be omitted. However, it appears that there will exist the need to collect additional and/or differing sets of data for such studies, requiring the data collection system and repository to take this into consideration in its formation. It is exactly this requirement that has lead to the data classification scheme proposed by this study, which will be discussed in Section 6.4.)

If only basic, long term research is supported by the repository, a

completely centralized data base offers many advantages. A centralized data base is easier and cheaper to implement and operate than a decentralized system, but poses some secondary problems for the data collection system. A centralized system may be most adaptable to non-standard data collection practices at local projects, but only at the expense of extensive data editing at the centralized source.

The most serious problems and disadvantages of a centralized data base operation are those concerned with the sheer volume of input, such as processing the large number of input errors that may be expected, especially until user familiarity is established. Further, data suppliers will probably receive little benefit from a centralized repository and may resist supplying data. Even a small software system may expect to submit a large volume of data. For a moderately large system, the load and cost of key entry and data analysis may be expected to be quite large; and if erroneous and missing data are returned to suppliers for correction, the edit cycle will be long and unreliable. Past experience indicates that data repositories based on long-range research applications have been unsuccessful with such problems as not being able to process the input load from diverse projects, little immediate return to justify the costs of the data base, and a low level of acceptance and use. (See Volume/002 for a more detailed discussion.)

These problems are somewhat ameliorated by standard reporting requirements, limiting automatic data collection to source code analysis and summarized run data, restricting the number of items collected, and holding to a reporting cycle not more frequent than monthly. Source data automation - keying and editing the data at the source of acquisition - and accepting summarized and machine-readible input also reduce the impacts of high data entry and analysis at the centralized sites.

If direct management support to either program management offices or to individual projects is to be provided, a distributed or local data base has numerious advantages, including:

1) Automatic collection points can be built into the supplied software.

2) Fine, detailed data may be kept at the project or regional site and only filtered, summarized data need be sent to the central data base.

3) Direct benefits are provided to data suppliers in the form of management support.

4) Data entry may be done interactively at the project site, permitting immediate feedback and correction of errors. Even if data entry is not done interactively, a shortened error correction cycle is realized.

5) Project management may feel that their data base is more secure if it resides at the project site.

The chief disadvantages of a decentralized system are the problems and costs of providing standard software to a variety of project computers and/or of providing standard hardware to many projects. Maintaining standard software and a standard data base for potentially conflicting needs of many locations is invariably a problem for multisite systems. Certainly, creating many versions of the software will require an extended time period and is not a feasible solution for an initial capability. Gradual adoption of a standard project monitor or distributed data base system is feasible on a long term basis, but places the recognizable requirements of portability, maintainability and adaptability on the central system.

The problem of accessing distributed data bases on a variety of local computers must be recognized. Direct access from the central location is possible, but involves considerable transmission costs and user resistance may arise if the privacy of project-sensitive data is threatened. If central access is not used, an extract program can be run to sample, filter or summarize the data before transmission to the central computer, either directly or via a machine-readible or hard copy media. This gives the local project manager greater control of his data, but reduces the responsiveness of the system for research purposes.

Regional or command centered data bases on a specialized basis - i.e., as

for a Reliability Center, Productivity Center, or Maintainability Center - patterned after the specialized research facilities served by the ARPA were also examined. It appears that probable research activity seems insufficient to support such specialization.

5.1.2 Data Base Structure

There are many issues in data base design that are still being addressed rigorously by data base theorists. At issue are the physical organization of the data base, the method of access and the logical structure of the data base apparent to the user. Query languages are based on logical views of the data structure, and a data management system maps the logical structure onto the physical structure for retrieval. The data access method exists at the interface between the logical and physical views of the data and largely determines the nature of the mapping that is done.

5.1.2.1 Physical Structure

The advantages of serial, linked list, direct and inverted files were investigated. Serial organization has advantages in relative independence and ease of implementation. It uses space efficiently, grows readily and accommodates several access methods. It has disadvantages in flexibility of record length and order, and maintenance is expensive if many changes and reorderings must occur.

Linked lists have the advantage of keeping records in a file in order without sorting or moving about, but are space consuming and have slow access times. Although complex networks of relationships can be expressed to yield a powerful query capability, this freezes a logical view of the data base into the physical structure. System malfunctions are difficult to recover from and the maintenance costs of reforming complex relationships may be high.

Direct, hash-addressed files have advantages in the speed of retrieval, but sorting or retrieving on secondary keys may destroy much of this advantage. There are almost always some overflow problems, e.g., more than one key hashed to a single address bucket, and space utilization greater than 80% is undesirable because of this.

Fully inverted files are very fast and powerful in answering very complex queries, but are expensive to maintain. Alteration of item attributes and deletion of fields have far reaching impacts on the data base. Hence, for a long-life, relatively high maintenance rate system without a very demanding retrieval level, as the repository may be expected to be, fully inverted files are not a good choice.

In summary, it appears that a serial data structure with current records in direct access storage and historical records on tape will be the most economic and satisfactory system.

5.1.2.2 Access Methods

An access method mediates between the logical view of the system and the physical structure. To a degree, the physical structure of a system helps determine the access method, but some access methods may be used for more than one structure, (see Figure 5). A data management system may support files of diverse structures and several access methods. An evaluation of the

Organization	Sequential	Chained	Algorithmic	Indexed
Serial	X			X
Linked List	P	X		P
Direct	P		X	P
Fully Inverted				X

X = generally used

P = possible in some cases

Figure 5. Access Method Matrix

sequential, chained, algorithmic and indexed access methods was made in terms of the needs of the repository and the interactions with the data collection system.

The sequential access method depends upon the physical structure of the data base being serially arranged in some way, but partitioning the data base permits random access to subsets of the file. Sequential access may be the only access to a storage device (e.g., tape) and it is efficient when the access rate per pass is high. It has disadvantages in reduced security and report sorting requirements where the order of presentation differs and is more complex than the physical structure.

Chained access applies to linked list structures and is very effective where the information satisfying a query is known in advance so that the linking is established to answer the query. Where the physical structure does not reflect the logical query, access may be slow. A query causing retrieval to skip around in the data file is also expensive. In short, linked lists and chained access are only efficient when the nature of queries against the data base can be anticipated and built into the data structure. This is not likely to be the case for the repository, but might readily be the case for a standardized project monitor satisfying standard report requirements.

Algorithmic access is used with direct file organization to reconstruct hashed keys. Algorithmic access is useful for fast access to large files with records being retrieved by a prime key. Security provisions are high. Access using multiple and secondary keys is quite slow, a factor that makes it rather unsuitable for the repository.

Indexing is the access method used by most data management systems and may be applied to each of the physical storage structures. It has definite advantages in ease of data base maintenance and multiple keys may be used for retrieval, but there is a cost associated with the creation and maintenance of index files or catalogs. Since an index is a file like any other, any of the basic physical organizations may be used for index file structures. Real advantage may accrue from searching and sorting relatively small index files

rather than massive data records. Also, additional searching methods are possible that result in improved efficiency over other methods.

In view of the flexibility and power of indexed access, this method appears most appropriate to the repository despite the cost of index computation and maintenance.

5.1.2.3 Logical Structures

A logical structure of a data base is the view that a user has of the data and may have only an incidental relationship to the physical structure. The logical definition of the data base may vary from user to user; a particular user's view is called a subschema and may encompass only part of the data. The classes of logical views investigated by the study include the network, the hierarchical and the relational logical structures.

A network view of the data base is very powerful for defining data that are interrelated in multiple ways and is usually associated with linked lists and chained access. Unless separated from the high cost of maintenance of linked lists, the network view is not appropriate for a data base with a high volume of update activity or unknown queries.

A hierarchical view of a data base is typified as a tree structure and is often associated with multi-level indexed access methods. Multiple views of the data base may exist. The different access methods may all be addressed by a hierarchical view. Although most implementations of hierarchical views preserve some reflection in the physical structure of the data base, great independence can exist. For ease of use and data base maintenance, the hierarchical view provides adequate flexibility without too great a cost.

The relational view of a data base is that of a table or an array. Unlike network and hierarchical views, which usually retain some connection to the physical structure of the data base, the relational view is a logical view only. While the complete independence of the logical and physical structures permits great flexibility in maintaining and modifying either structure,

there may be considerable costs in mapping one onto the other through the access method. Hence, while the relational view is a good choice for the repository, its feasibility depends upon discovering an efficient implementation scheme.

Since the data base structure must support growth and change, multiple keys, and unknown queries, it would appear that hierarchical views of the data base structure offer the most flexible structure realizable within a reasonable effort.

5.2 DATA MANAGEMENT FUNCTIONS

The data management functions investigated include data definition languages, data base creation and maintenance, information retrieval, ensuring the integrity of the data base in case of system failure, security, accounting for usage, restructuring the data base, managing core, generating reports, and performing administrative maintenance tasks. The additional considerations of scheduling and conversational dialogues were addressed for interactive systems. Criteria of efficiency, ease of use, ease of implementation, and ease of modification were considered in evaluations.

5.2.1 Data Definition Language

The alternatives for data definition languages range from the minimal declaration of coding type and item length to editing rules, access authorization, and usage rules. The data management system may use the definitions to bind data at either compile or execution time. Data definitions may permit aliases and alternative definitions for the same items or sets of items. Coding types and data structures may be limited to a few standard classes or be extended to a large variety of classes. Extensible data definitions are special features of some advanced programming language.

While advanced features in a data definition language are desirable, their implementation takes time and increases the complexity of the languages in use. None of the parameters defined for the repository are at all unusual and the data description requirements are straightforward. It is desirable

to permit multiple logical views to enable users to incorporate definitions to their particular interests, but this feature is not essential for an initial capability. Compiler-bound variables are adequate for current data, but execution-bound variables are desirable for historical data whose definitions may change with time.

5.2.2 Data Base Generation and Maintenance

The data base maintenance alternatives investigated include fixed data definitions versus directory - sensitive routines, batch versus transaction orientation, levels of editing and degree of accounting. Interactive data base maintenance always raises a few problems in contention, and complexity.

Since it is anticipated that the input level will exceed the retrieval load and that there will not be a high demand for instant currency, a deferred batch operation is probably preferable to the immediate input of transactions, thus avoiding some contention problems. Symbolic editing is desirable. If multiple logical views are permitted, there is a need for mapping between the logical and physical structures.

5.2.3 Data Base Query and Information Retrieval

A very wide range of retrieval capabilities are available in various systems ranging from set, programmed reports to powerful query languages. Since the research questions that may be asked of the repository are not known to this study, it is assumed that a flexible query language will be needed for the data management system.

5.2.4 Data Base Integrity

Restart and recovery in case of hardware failures and program halts may or may not present a problem, depending upon the degree of integrity demanded of the system. In an interactive system, most operations tend to be small, consisting of a single or a few transactions, so that resubmitting the total job is not a great penalty when a fault occurs. In a batch update, however, not only is considerable time lost, but an abort may leave many faulty records in storage. In a statistical data base, absolute accuracy may not be required

- i.e., a few missing or inaccurate records may not make a noticable difference in results - but accuracy is psychologically appealing. It is common in long update runs to save periodic dumps as restart points to minimize rerun times. Also, duplicate copies of a data base on tape are kept to reload the database in case of a irretrievable abort during an update. Selective dumping of changed records is more efficient than a total data base dump if activity rates are not high for all records.

In addition to providing recovery checkpoints, some means of verifying the accuracy of the modified data is necessary. Some checks can be made automatically, such as record counts and chain verification, but often visual checks, benchmark tests, and data samples are necessary to insure that the system is operating correctly.

Restart provisions can be expensive, but it appears that only moderate recovery capability seems required of the repository data management system. There will be little need to update existing records; most of the data will be new records, additional software to insure proper storage, proper linkage and selective dumping of updated records seems desirable.

5.2.5 Security

Although data privacy and computer security are subjects undergoing close scrutiny and moderate development, elementary security measures appear adequate, given that the data is desensitized before inclusion in the repository. The data management system should require that security keys be provided before data base access is allowed. Further, users of the data base should not be permitted access to data items outside their logical views of the data base. If direct access is limited to analysts associated with the repository facility, the analysts may screen requests for sensitive material.

Security on a local level where desensitization would impair the usefulness of the data presents a different problem. Observance of security keys and procedures should be more stringent. Access to local data bases is a subject that will have to be solved by future negotiation. Although project managers are generally quite willing to share data (See Volume/005), some information

is considered sensitive to the point where direct access to the local data base may be refused.

5.2.6 Accounting

Detailed accounting for computer resource utilization is normally needed in any shared facility to distribute costs fairly. It is also desirable as a means of evaluating system efficiency and bottlenecks. In the case of the repository, in a wholly owned facility, data collection and data base update activities are facility specific costs. Providing other services to research projects and other users may be justifiable charges to those projects via a preestablished cost algorithm. Unnecessary and excessive usage should be discouraged to avoid overloads. Even if RADC shares operational costs, a fee should be employed to avoid abuse.

5.2.7 Data Base Restructuring

Restructuring serves two purposes for a data base. First, it cleans up "garbage" and puts the data base in an efficiently usable condition, and secondly, it allows reorganization to accommodate changes. Independence of logical views and physical structure permits one to be changed without affecting the other. The repository system will not be static and a reasonable amount of logical and physical independence should be achieved. Some amount of restructuring might be necessary, but the extent will depend upon operational experience and the cost of restructuring.

5.2.8 Core Management

Core management can be done either by the data management system or by the operating system. Most existing data management systems manage their own allocation of core, but some are bound to operating systems. New virtual storage management techniques bring in special hardware features, altering core management requirements. The techniques of memory management chosen for the repository will depend upon the capabilities available in the operating system and the hardware configuration under which the data management system must operate.

5.2.9 Reporting

Reporting system status (as apposed to user report generation) is an essential support service for the data management system to perform. Its exact nature depends upon the data structure chosen so that load factors and system quality are accurately reflected. It is necessary to keep activity ratios for use as a basis for relegating records to history files or for purging, requiring a data base monitor, but several basic system design decisions are required before these types of requirements can be explored adequately.

5.2.10 Scheduling

Scheduling is not a serious problem until multi-user interactive systems operations are entered. The chief problem is the avoidance of deadlock. If a wholly batch system is used, scheduling problems will be minimized; if an interactive system is chosen, operations to control the shared use of the data will be necessary.

5.2.11 Coaching

In interactive systems only, an extensive dialogue may be carried on between the user and the data management system; first, to prompt the user in providing inputs, and second, in requesting clarification and immediate error corrections. In addition, instructional material may be included in the system to teach or help the user in system operations.

Constructing an efficient, effective, thorough dialogue is a highly technical task and capable of almost limitless expansion. At least, minimum prompting and error message interaction must be provided if an interactive system is chosen.

5.2.12 Data Management System Capabilities

It is recommended that a data management system be developed whose physical data base structure is transparent to the user; i.e., whose structure is viewed at a logical level only. The system should:

- Provide a data definition language capable of defining

multiple logical views of the data base by building a table to be used by access routines to map the logical views to the physical view.

- Provide sufficient independence of the logical from the physical view so that new data items and record relationships can be added without affecting the existing logical views.
- Include security keys that must be supplied before data base access is allowed.
- Prevent access to data items outside the logical view of the user.
- Defer batch updates to periods of minimal activity to avoid problems of deadlock and restart.
- Base costs on research access or project management access but not on data entry.

5.3 SYSTEM HARDWARE CONFIGURATION

Three classes of system hardware were reviewed: data storage, data entry and data processors. Again, the basic criteria used in the investigation were flexibility, security, cost and growth potential.

5.3.1 Data Base Storage Media

The media that offer adequate storage for a large data base are serial access devices, direct access devices, and mass storage devices.

The cost of storage on tape (serial access devices) is low; it can be physically protected by locking; and it can grow indefinitely; however, it is slow and inflexible. It is an excellent back-up medium for historical files and/or other files with a low activity rate.

Direct access devices, which consist of a wide variety of disks, drums, and data cells, vary greatly in cost depending upon the speed, channels, and

capacity of the device. Non-removable, fixed head devices offer very fast access, but disk packs offer opportunity for physical security and unlimited storage (at the expense of mounting and dismounting packs). For huge volumes of infrequently accessed data, direct access devices are relatively expensive as compared to tape and mass storage devices, but are economical for frequently accessed data. Disks are relatively flexible and data may be rearranged quite easily.

Mass storage devices have huge storage capacity and some direct access capability. Costs are lower for mass storage than for direct access storage and speeds can be quite fast using parallelism and look-ahead buffering. It is flexible, expandible and capable of containing both high access and historical data files. The tapes and cassettes used for block storage are removable and some may be locked out. Unfortunately, mass storage devices are largely untested and those currently available seem to be insufficiently reliable to support a data base operation with adequate efficiency and dependability.

It is recommended, then, that direct access storage be used for current and highly active files and that serial access storage (tape) be used for back-up, historical and/or low activity rate files.

5.3.2 Data Entry Methods

There is several data entry devices and methods available, even omitting special data capture devices such as recording time clocks and plastic card readers. Key to card, tape and disk devices range from simple card punches to complex intelligent terminals.

Evaluating candidates for a data entry system involves such a great number of options and alternatives that the task appears insurmountable without a great deal of effort. The options chosen are strongly influenced by the relative centralization of the data base and whether the data acquisition mode is automatic or manual. In making a final evaluation of data entry methods, three candidate configurations were formed, including a relatively small, completely centralized system with a limited data management system; a medium sized system with a more powerful, interactive data management system, with a

potential capability for supporting distributed data base; and a large scale, high traffic system. This study evaluated the impact of data input load as it relates to efficient and effective error correction, and the data entry system as it relates to effective servicing of repository users.

For the small research-only system (as the initial pilot facility at RADC might be), it is economically feasible to acquire all data on either manual input forms or hardcopy reports, transmit the data by mail, preprocess it manually by data analysts to detect and correct errors, prepare the data for input, and key it for entry. Card or cassette buffering is adequate. Some profit might be expected from a terminal with some intelligence to aid error detection, but it is not required.

For the medium-sized system, the input load may stress the completely centralized facility's capabilities and alternative input modes need to be inspected. Keypunch is outmoded, being relatively slow in preparation and involving several processing steps. Key-to-tape and key-to-disk without input editing capability are of medium cost, but have rather inadequate editing and error detection capabilities. Optical character readers provide an acceptable volume of input, but many readers require special type elements, most are unreliable and "cranky" in operation and an inordinate amount of time is entailed in error diagnosis and correction.

The most reassurable resolution of the data entry load problem at the completely centralized facility is to submit the bulk of the data on some machine-readible medium. Ruling out cards as too bulky and OCR as too unreliable, leaves tape (either punched or magnetic), magnetic cards, and direct transmission. Key-to-store over telecommunication lines for any substantial distance consumes much transmission time, both for initial input and to correct transmission and keying errors, and is relatively expensive. However, key-to-store at regional or local data bases is a reasonable expense if the urgency of management control data is added to research requirements. Key-to-tape or local data base-to-tape, either reel or cassette, is an entirely satisfactory medium (except for the turn around time on error correction, a

consideration true for all manually transported media). Records may be in either card or report format, with or without blocked records, depending upon the power of the local tape operation. Direct transmission from a local data base or temporary store is possible, if currency of data is a factor. Otherwise, the protocols involved in interfacing with many computer mediated data sources may require extensive programming. (It must also be noted that maintaining compatibility of tapes produced by many devices, especially cassettes, is not a trivial task.)

Data entry at the point of acquisition has other advantages, such as the speed and ease of correction of keying and data errors. Editing of inputs does require some intelligence, either in the form of an intelligent terminal or an input editor in a local computer. Even with a project monitor system mediating input to the repository, there is some argument for an intelligent terminal with a CRT display. Editing is greatly facilitated and immediate entry encouraged.

While manual input modes are still feasible for a medium sized facility, dispersion of the key entry load to local facilities and the utilization of machine-readable inputs may reduce both the data congestion and error correction problems.

For a large system, some means of semi-automatic data capture (e.g., intelligent terminals and project monitors), some preliminary reduction of data, and machine-readable, if not automatically transmitted data, are desirable. Although the software data collection system could grow this large, the final facility would be several years in fruition. It implies a sophisticated data management and information retrieval system, huge amounts of storage, perhaps some distribution of the data base, and considerable development work on project monitor programs and data editors for intelligent terminals. Without some degree of automation, a really large facility is not deemed highly practical.

It is recommended that the data repository move to attain machine-readable input at the central repository, considering the following factors:

- Data entry should be at or near the point of acquisition.
- Data entering the data base should be as "pure" and purged of error as is possible.
- The data entry system must be flexible in adapting to input changes in individual item values, additional inputs, and forms variation.
- Data entry should match the highly structured, modularized, and interrelated data collection forms with corresponding alternative and replaceable software modules.

To meet these goals, it is further recommended that the preferable data entry device is an intelligent terminal with moderately powerful editing logic, storage capacity and CRT display capability. An intelligent terminal has the following advantages:

- A superior editing capability.
- Interactive input and error prompting.
- Storage and review prior to final entry.
- Flexibility in adapting to changes.
- Entry routines easily reflect modularized and structured input forms.
- Machine-readable and edited output wherever the entry device is located.

It is also recommended that further investigation be initiated to determine design and user requirements for an interactive editor for the intelligent terminal based on the data requirements defined herein, as modified by changing repository requirements.

5.3.3 Processors

Only a cursory inspection of central processing units was made. Many processors are available that are equal to the tasks suggested. However, it was understood that the host computer for the pilot facility would be the

Honeywell 6180 computer. This computer is quite adequate to handle either a small or medium system even on a shared basis.

6. SOFTWARE DATA COLLECTION SYSTEM SPECIFICATIONS

The Data Collection Study has derived requirements for the RADC Software Data Repository, considering all areas of investigations made to date. It must be noted that because there are numerous alternatives existing in the formation process, the specifications herein presented should be viewed as general requirements, dependent upon RADC's final selection of alternatives.

6.1 PURPOSE

These specifications summarize the recommendations derived by this project for the creation of a software data collection and reporting system. They will address the basic assumptions for the system, the general functional requirements of the system, the data types required for the study of productivity, reliability and costs, data acquisition, transmission, data entry methods, and the interface of the data collection system with the data management system governing the central repository. In making these recommendations tne general influences of cost, ensuring data integrity, and overcoming such data collection problems as contractor reluctance to release data will be discussed.

6.2 ASSUMPTIONS AND LIMITATIONS

The basic constraints on a software data collection system are, first, that it meets the standards imposed by the existing military regulations, practices and concepts; second, that it satisfies the concepts of operation imposed by the research objectives of the RADC Software Data Repository; third, that it addresses, if not adequately solves, problems of data integrity, contractor reluctance and high costs of data collection.

Any long-range data collection effort for software methodological research must be integrated with data collection requirements for project monitoring and management. Military standard practices assume a reasonably standard model of the software development process and define standard products and standard quality assurance procedures. They also specify a way of organizing work packages (a Work Breakdown Structure) and schedule and resource utilization accounting. Finally, standards for configuration management

(configuration identification, change control, and configuration accounting) are specified. In actual practice, there is considerable deviation in interpretation of these project management requirements and many exceptions are implicitly (or explicitly) granted. If research goals are to be achieved, it would appear that more stringent standards are desirable and should be so stated in the requirement specifications of the RADC Software Data Repository.

The concept of operation for the repository includes, first, basic support for software methodological research, and second, an evolutionary implementation of the data collection and data management system. Support for methodological research involves the collection of much more information about project environmental conditions, including the use of specific tools, techniques and methods, than is normally required for project management. Further, although current research interest is largely focused on the productivity, cost effectiveness and product reliability of software projects, the data collection system must be readily adaptable or expandable to encompass other research goals.

The concept of evolutionary implementation implies that the repository specifications must provide for (a) an initial capability or pilot facility utilizing an existing computer, operating system and data management system supporting a select set of research problems, and (b) expanding the initial capability to handle a much broader range of projects, data types, data volumes and research objectives than the pilot facility, providing that the trial operation establishes the practical feasibility of the more sophisticated and larger system.

Finally, it is assumed that at least a partial solution can be found for the problems that plague current data collection efforts and impair the comparability of data from project to project. Much of the unreliability of data that inhibits current research efforts lies in a lack of standardization of terminology, measurements and data collection procedures. Further, data reliability is decreased by the bias and subjectivity inherent in current collection procedures, in the natural resistance of projects to management

controls, and in the reluctance to release sensitive data - data that may reflect on project competence or reveal proprietary technology.

It must also be recognized that efforts to collect a large volume of high quality data will entail high costs. First, the benefits to be expected from a wide representation of project specific data must be weighed against these costs, and second, mechanisms for establishing data delivery requirements and for reimbursing collection costs to projects must be considered in these specifications.

The objectives of the software data collection system specifications are to:

- Recommend procedures to acquire reliable, objective and standard data to overcome the problems facing current data collection efforts.
- Define the specific data parameters to be collected, the instruments to collect them, and specify minimal data collection points and frequencies.
- Determine data entry methods and configurations.
- Define the interface between the data collection and reporting system and the data base and data management system.
- Define the possible repository's operations management alternatives.

6.3 DATA ACQUISITION PROCEDURES

The data acquisition procedures represent the interface with participating software development projects and the RADC Software Data Repository, and as such must be adaptable to a broad range of conditions and numerous problems. A wide variety of data parameters which may change and expand over time and which represent a large volume of data at considerable cost over all projects must be acquired. An examination of project conditions has been presented in Volume 003; evolution changes in Volume 004; and data collection problems in Volume 002 of the Data Collection Study series. Standardized procedures need to be developed to ensure that the data collected are acquired under comparable conditions and will provide valid data for experimental studies.

The data acquisition procedures must be adaptable to projects that:

- Vary in size from very large to quite small.
- Vary in difficulty from very complex and innovative to quite simple.
- Entail any portion, or combination of portions, or all of the life-cycle phases.
- Have objectives that vary from pure research to automating basic operations.
- Are located throughout the world.
- Employ different computers, operation systems, languages, and methodologies.

In order to evaluate the impacts of project attributes upon project performance, to account for project differences, and/or to ensure that projects under study are comparable, it is recommended that a variety of environmental parameters be collected. Specific recommendations for environmental parameters are presented in Section 6.4, Data Requirements.

Data acquisition procedures must be constructed so as to eliminate or minimize such problems as:

- Lack of standardization of terminology, measures and instrumentation of the software development process.
- Subjectivity and bias in the measures taken.
- Reluctance to release information that is sensitive or reflects negetively on project efficiency.
- Resistance to managerial controls.

The standards for collection that are developed must be enforceable. Countermeasures for subjectivity and bias must be developed that deal not only with the unreliability of introspective measures (e.g., subjective estimates, ratings and recollections), but also of personal involvement, summarizing and averaging over time periods, and differing organizational reporting levels and configurational hierarchies.

It is recommended that the software development project models presented in Section 2.1 be adopted as the standard models for the definition and design of the data collection system. The measurements and collection procedures described in Section 6.4 should be selectively established with contract provisions made for their execution, including specified penalties for non-observance of the prescribed collection conditions.

To reduce the subjectivity of the measures taken, it is recommended that objectivity and data integrity be increased by:

- Acquiring data at the point of occurrence.
- Introducing automated and mechanical recording methodology.
- Employing independent observers to audit performance, review and test products, and record data.
- Providing specific standards and review criteria to guide judgmental data.

- Avoiding reporting techniques that interfere with ongoing technical performance, irritate or distract technical personnel, or appear threatening and coercive.
- Integrating data requirements with technical products and performance as part of the technical work.
- Acquiring data in small increments at short, regular intervals to minimize interference of large data collection efforts.
- Indoctrinating project personnel thoroughly with the goals, procedures, requirements and definitions for the data gathered.
- Developing objective algorithms for the summation, averaging and filtering of data.

The above measures will also help to minimize reluctance and resistance to releasing information. Also required as an integral part of the Data Collection System, are procedures and handbooks for:

- Project monitors to assist in software development management procedures, integrate existing military practices, and promote effective project monitoring practices.
- Project personnel to indoctrinate project personnel into the objectives, specific data requirements and reporting procedures established by RADC.

To provide the necessary flexibility to adapt to differing environmental conditions and to changing data requirements, it is recommended that data collection forms and procedures be designed and used in a modular manner. To alleviate some of the impacts of change, procedures and mechanisms should be established for the prompt and timely maintenance of the user procedures and documentation recommended above. In so far as possible, user cooperation should be solicited in defining and agreeing to the procedures for collection of data for experimental purposes.

6.4 DATA REQUIREMENTS

At a minimum, software development data must be collected that will support project productivity, program reliability and cost studies. Information is also desired that will lead to further insight and understanding of the software development process. Further, expansion of research goals to cover other indices of software quality, such as program efficiency, modifiability and maintainability, convertability and transportability, and ease of use and understanding, must be anticipated in the future. The data parameters collected for the repository will be in part identical to or derivable from those currently collected and/or observed by project management; both data for management and data for research must be collected using the same procedures. On the basis of the study reported in Volume 003 of these reports, it is recommended that three classes of data parameters be collected.[1]

- Project environment data
- Project performance data
- Product quality characteristics

Project environment data consists of those project attributes that are likely to influence project performance, including:

- Project definition and related data, specifically:
  - Project identifier
  - Title
  - Description
  - Start date
  - End date
  - Control Authority
  - Number of subcontractors
  - Total manpower
  - Total pages of documentation
  - Total number program modules
  - Total number subsystems
  - Total number operational source statements

[1] All data parameters are fully defined in the Compendium of Procedures and and Parameters, Volume/007.

Total number support source statements
Total number operational object instructions
Total number support object instructions
Total number bytes in data base
Overall project complexity
Application software complexity
Control/operating system complexity
Support system/tools/aids complexity
Data base structure complexity
Quality of requirement specifications
Quality of design specifications
Schedule adequacy
Overall project management effectiveness
Overall project personnel qualifications
Computer resources adequacy
Customer supplied information
Timeliness of review actions
Funding adequacy
Project software type

- Customer/contract definition and related data, specifically:
  - Contract type
  - Number of coordination points
  - Frequency of customer contact
  - Customer experience with data processing
  - Customer experience with application
  - Customer experience with target computer
  - Customer experience with contractor
  - Stringency of review procedures
  - Reasonableness of negotiations
  - Penalties for non-performance
  - Technical risk
  - Redirection rate
  - Contract and work compatibility

Contract renegotiability
Customer turnover
Customer rapport
Project location
Quality of physical facility

- Subcontract identification and related data, specifically:

  Subcontractor identifier
  Subcontractor type
  Responsibilities
  Experience in data processing
  Experience with subcontractor
  Experience with application
  Frequency of contact
  Quality of subcontractor supplied information
  Subcontractor rapport

- Software installation data, specifically:

  Location of target computer
  Installation technique
  Number of personnel in installation team
  Average experience of installation team
  On-site training
  Software adaptation
  Resource requirements for installation
  Installation difficulty rating
  Problem(s) description

- Project organizational data, specifically:

  Organization identifier
  Organization type
  Organization responsibilities
  Identifiers of reporting organizations
  Work identifier(s)

Initiation date
Completion date
Personnel skill level(s)
Manning number
Managerial techniques identifier(s)
Managerial turnover rate
Key personnel turnover rate
Project member turnover rate
Identifiers of higher level organizations
Identifiers of organizations on this level

Project employee information, specifically:

Employee identifier
Employee skill level
Employee job title
Organization identifier
Experience in data processing
Experience with project programming language
Experience in application area
Experience in management
Experience with target computer
Education level
Personnel's work identifier(s)

- Computer equipment and support facilities identification and capability data, specifically:

Device identifier
Memory size
Unit of measure
Number of CPU's
Number of I/O channels
Memory cycle time and unit of measure
Device type

Number of sequential access devices
Number of random access devices
Major input device type
Product identifiers
Location of facility
Mode of operation
Turnaround time
Computer availability
Quality of equipment and/or related services
Quality of operating system and support software
Quality of operating system and support software documentation

- Identification of project's utilization of programming methodologies, tools and techniques and related data, including utilization of a program production library. The specific parameters include:

  Technique identifier
    Technique class(es)
    Technique type
    Acquisition cost
    Training effort
    Independence rating
  PPL identifier
  Mode of operation
  Manpower resource allocation
  Skill level
  Cost of establishing PPL
  PPL operation and maintenance cost
  PPL computer utilization costs
  PPL effectivity rating

- Projects programming language identification and related data, specifically:

  Source/object language identifier
  Language acquisition costs
  Language training costs
  Compiler/assembler reliability

Quality of language documentation
Language efficiency
Language relevance to project goals
Language support tools

Project performance characteristics should measure both the productivity and the quality of work, including:

- Work package (work breakdown structure) identification data with scheduled performance of tasks, specifically including:

  Work level
  Work identifier
  Work description
  Identifiers of reporting work elements
  Initiation date
  Completion date
  Terminator
  Resource utilization data:
    Resource identifier
    Resource unit
    Resource allocated
    Resource expended
    Resource variance
    Work/product identifiers
  Identifiers of higher work elements
  Identifiers of work elements on this level
  Product identifiers

- Product identification data and methodologies used in its production. These data specifically include:

  Product identifier
  Product mod number
  Product version number
  Product type
  Reporting level
  Product description

Identifiers of product components

Work unit

Work size

Cost

Programming language identifier

Language relevance to product

Language efficiency for product

Product complexity

Product familiarity

Product stability

Programming techniques related to product data; including
- Technique identifier
- Relevance to product
- Integration rating

- Project performance, including resource expenditures, data current to the reporting period, specifically:

  Organization identifier

  Production data:
  - Work identifier
  - Product identifier
  - Resource identifier
  - Resource units expended
  - Work units produced
  - Product status

- Project quality assurance provisions including records of all proposed changes to a baselined product configuration, whether due to external forces, implementation difficulties or product faults/errors, or informal examination. Three classes of data are included:

  - Software Problem Report (SPR) data, specifically including:

    SPR identifier

    SPR type

    Employee identifier

    Date of problem discovery

    Time of day

Work identifier in progress
Status
Product(s) impacted by problem
Data base identifier
Test case identifier
Test tool identifier
Problem description
Date received
Employee assigned

- Software Modification Transmittal (SMT) data, specifically, including:

  SMT identifier
  Date of correction
  Time of day of correction
  Employee identifier
  SPR's resolved
  Production identifier(s)
  Old mod
  New mod
  Unit of change
  Amount of change
  Difficulty rating
  Work identifier
  Type of software termination
  Work identifier in progress when error generated
  Manpower resource required
  CPU time
  Error description
  Error type
  Date received

- Milestone event data, specifically including:

  Milestone identifier
  Work identifier
  Date of event
  Milestone type
  Milestone status
  Milestone description
  Milestone subevents; including:
    Work identifier(s)
    Product identifier
    Product status
  Product identifier(s) baselined
  SPR(s) established
  Authentication

Product quality measurements should consist of those characteristics of the final product structure and operation that can be measured most adequately using automated analysis and recording tools. The manually collected data supporting execution of the software using automated tools and aids includes:

- Information related to software operations, test case, job and total run time, and errors, specifically:

  Job identifier
  Employee identifier
  Test case identifier
  Product identifier(s)
  Technique/tool identifier(s)
  Description
  Device identifier
  Log date
  Job identifier
  Computer job acceptance time

Computer operations

Completion code

CPU time

- Information related to program structure, language construct usage, operational behavior, data structure and usage, complexity analyses data, etc. (The exact parameter types must be obtained from the specifications of the automatic tools used).

Examination of the data requirements recommended by this study will reveal the relative importance of various factors in influencing productivity and product quality, while also giving priority to current research. For this reason, the entire set of data parameters defined in Volume 007 have been divided into three groups, corresponding to three levels of importance. Group 1 consists of data items that should be gathered by all projects, and is believed to be the minimum set required for support of productivity, program reliability and cost studies. Group 1 items should be collected from all projects for the pilot facility. Group 2 consists of data of second priority and includes many more environmental factors that may influence project performance, and should be collected either selectively to support specific research projects or as a long range investment for future research. Group 3 represents an expansion in scope and detail of Group 1 and 2 parameters. Group 3 items should be collected selectively to support specific research or upon the development of a large capacity system. Each successive group of items represents an expansion in data volume and an increased collection costs. Since Group 3 encompasses most product quality parameters, it potentially represents very large volumes of data and manual storage techniques should be considered for items not involved in a specific research program.

To meet requirements for flexibility, modifiability, and expandability, the data collection forms and procedures should form a modular, easily modified collection system. Such forms have been defined in Volume 007, Compendium of Procedures and Parameters.

The specific compostion of the forms designed for the data collection system include:

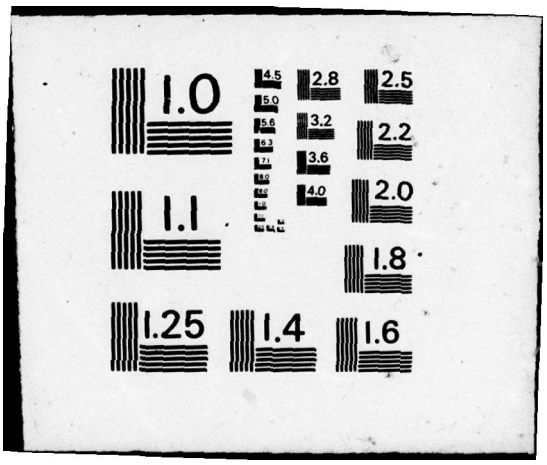
- Project Environment Information - A form for collecting project specific characteristics, to be completed and submitted at project initiation and completion.
- Contract/Customer Information - A form for collecting data about project unique customer and contract conditions, to be completed and submitted at project initiation and completion.
- Software Installation Information - A form to collect data on techniques, personnel, resources and problems of a software installation, to be completed and submitted at project initiation and completion.
- Subcontractor Information - A form to collect data on each subcontractor's responsibilities and interfaces, to be completed at project initiation and completion.
- Organization Information - A form to collect data on each organizational structure within the project, to be completed and submitted at project initiation and completion.
- Employee Information - A form to collect data on the experience levels for each project employee, to be completed at assignment of an employee to the project.
- Computer Equipment Information - A form to collect data for each computing device used by the project for software development.
- Computer Support Facility - A form to collect data on the quality and type of computer support facility used by the project.
- Program Methodology Information - A form to collect data on all concepts, tools, and techniques used in the development of software, to be completed and submitted at completion of the work definition phase and at the completion of the project.

AD-A036 115 SYSTEM DEVELOPMENT CORP SANTA MONICA CALIF F/G 9/2
SOFTWARE DATA COLLECTION STUDY. VOLUME I. SUMMARY AND CONCLUSIO--ETC(U)
DEC 76 N E WILLMORTH, M C FINFER F30602-75-C-0248
UNCLASSIFIED SDC-TM-5542/001/01 RADC-TR-76-329-VOL-1 NL

2 OF 2
AD A036115

END

DATE
FILMED
3 - 77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

- Programming Language Information - A form to collect data on each programming language used by project personnel, to be completed and submitted at project initiation and completion.

- Program Production Library Information - A form to collect data on the resource expenditures and the effectivity of the program libray, to be completed and submitted at project initiation and completion.

- Work Definition Information - A form to collect data on the work breakdown structure in an hierarchical manner, showing the relationship of all work elements and the chain of authority for each work element the software development project has defined. The form is to be completed and submitted at project initiation, work plan formation, or when the work package allocation is changed. A Work Definition Information Form is to be completed for each element for which resource allocation and expenditures are made.

- Product Identification Information - A form to collect data on each configuration item, its hierarchical structure, complexity, and resources allocated; this form is to be completed and submitted at completion of the work or product definition phase and at completion of the software development project for each product being developed.

- Project Performance Information - A form to collect data on resource expenditures, productivity, and work and product status. This form is to be completed and submitted at the end of each reporting period for the work progress and resource expenditures for that reporting period.

- Software Problem Report - A form to be used by project personnel when a problem is discovered in the software or other associated product items. The SPR is submitted to the RADC repository at the established reporting period. This form provides information on discrepancies found in any configuration item for both the project and the repository. (For the purposes

of this form, a problem is any condition that arises that would, if approved, cause a modification to be made to a software product, as a requirements change, a design or implementation dilemma or a program error.)

- Software Modification Report - A form to be used by project personnel upon successful modification to the configuration item. The SMT is submitted to the RADC repository at the established reporting period. This form provides information on and the final disposition of software modifications for use by both the project and the repository.
- Software Operations Log Information - A form to collect computer operations data manually by a project librarian or data collection clerk. This form is to be completed daily and submitted to the repository at the established reporting period.
- Job Identification Information - A form to define the relationships between test cases, products and test runs, to be completed and submitted upon completion of test case definition.
- Milestone Identification - A form to collect information on the product and/or work status, criteria for successful completion of a milestone, and error reports generated as a result of a milestone event. This form is to be completed and submitted at completion of the work definition phase and at completion of the milestone event for each milestone, or equivalent, in the software project.
- Computer Operations Definition - A form to define for RADC the computer operations and associated completion codes for each computing device used by the project. This form is to be completed and submited by the project librarian, or equivalent, prior to computer usage by project personnel. (This form is an example of one of many, definition forms that RADC may need. It is recommended that this type of information be stored as hard copy rather than input in the data base).

An example of some of the data collection forms and procedures contained in Volume 007 follows. These three example forms are applicable to all projects, and contain data from Group 1 exclusively.

To provide basic understanding and precise guidance necessary to help alleviate misunderstandings, subjectivity, bias and reluctance of the data suppliers, it is recommended that the data collection forms be accompanied by additional information that contributes to their understanding, use and maintenance, including:

- Statements of objectives and benefits.
- Roles and responsibilities of Air Force and contractor agencies.
- Control boards and committees and control and decision mechanisms for initiating and operating and maintaining the data collection system.
- For each form, precise definition and specifying procedures for each entry on the forms keyed to the entry blanks.
- For complexly derived entries, such as classifications of errors and programming methodology definitions, replaceable addendum specifying required information entries.
- For maintenance, data definition catalogs and control procedures.

P1 PROJECT ENVIRONMENT INFORMATION FORM

DATE OF SUBMITTAL: (1) _____ PROJECT IDENTIFIER: (2) _____

TITLE: (3) _____

DESCRIPTION: (4) _____

_____

_____

START DATE: (5) _____ END DATE: (6) _____

CONTROL AUTHORITY: (7) _____ NUMBER OF CONTRACTORS: (8) _____

PROJECT SIZE ESTIMATES

Total Manpower: (9) _____

Total Pages Documentation: (10) _____

Total Number Program Modules: (11) _____

Total Number Subsystems: (12) _____

Total Number Source Statements in Operational Software: (13) _____

Total Number Source Statements in Support Software: (14) _____

Total Number Object Statements in Operational Software: (15) _____

Total Number Object Statements in Support Software: (16) _____

Total Number Bytes in Data Base: (17) _____

PROJECT COMPLEXITY ESTIMATES

Overall Project: (18) _____ Application Software: (19) _____

Control/Operating System: (20) _____ Support System/Tools/Aids: (21) _____

Data Base Structure: (22) _____

PROJECT EVALUATION RATINGS

Quality of Requirements Specifications: (23) _____

Quality of Design Specifications: (24) _____

Schedule Adequacy: (25) _____

Overall Project Management Effectiveness: (26) _____

Overall Project Personnel Qualifications: (27) _____

Computer Resources Adequacy: (28) _____

Quality of Customer Supplied Information: (29) _____

Timeliness of Review Actions: (30) _____

Funding Adequacy: (31) _____

PROJECT SOFTWARE TYPE: (32) BUSINESS ☐ SCIENTIFIC ☐ SYSTEMS ☐
MAINTENANCE ☐ OTHER ☐

P1 PROJECT ATTRIBUTES INFORMATION FORM

To be completed and submitted at initiation and completion of the software development project. This form defines the reporting project, the project size, software complexity, and adequacy of other project specific parameters.

Parameter	Key	Format	E/A	Description
Date of Submittal	①	F-6		Current date, either project initiation or completion date, in the format yymmdd.
Project Identifier	②	F-8		An acronym, number or other identifier that uniquely specifies a project and identifies all data collection forms for the project.
Title	③	F-16		A short name or descriptive title for the project.
Description	④	V-256		A brief narrative description of the software development project, covering its objectives, scope and approach.
Start Date	⑤	F-6		Date project is initiated, yymmdd.
End Date	⑥	F-6	X	Date project is to terminate, yymmdd.
Control Authority	⑦	F-10		The name or phrase characterizing the customer's configuration control agency, e.g., SPO, CCB, project monitor, etc.
Number of Sub-contractors	⑧	F-4	X	Total number of subcontractors participating in software development project.
Total Manpower	⑨	F-8	X	The number of manyears required for the software project.
Total Pages of Documentation	⑩	F-8	X	The total number pages of documentation to be produced during the performance of the project.
Total Number Program Modules	⑪	F-8	X	The total number of modules to be produced during the performance of the project.
Total Number Subsystems	⑫	F-8	X	The total number of subsystems to be produced during the performance of the project.
Total Number Operational Source Statements	⑬	F-8	X	The number of deliverable POL statements in the operational system.

P1 PROJECT ATTRIBUTES INFORMATION FORM (cont'd)

Parameter	Key	Format	E/A	Description
Total Number Support Source Statements	(14)	F-8	X	The number of deliverable POL statements in the support software.
Total Number Operational Object Instructions	(15)	F-8	X	The number of deliverable MOL statements in the operational system.
Total Number Support Object Instructions	(16)	F-8	X	The number deliverable MOL statements in the support software.
Total Number Bytes in Data Base	(17)	F-8	X	The number of bytes of storage required for data storage.
Overall Project Complexity	(18)	F-2	X	An evaluation of the degree of complexity of the project, independent from the complexity of the software produced. (This evaluation should consider such factors as the number of coordination points, number of subcontractors, number of agencies per product, number of internal coordination points, number disciplines involved, number and variety of products produced, number and variety of information sources.) Rating scale is 1-10, where 1 = easy, 10 = most difficult.
Application Software Complexity	(19)	F-2	X	A complexity rating for the known characteristics of the solution algorithm software being developed. Rating scale is 1-10, where 1 = easy, 10 = most difficult.
Control/Operating System Complexity	(20)	F-2	X	A complexity rating of the control software or operating system, either being developed or used by the software development project. Rating scale is 1-10, where 1 = easy, 10 = most difficult.
Support System/ Tools/ Aids Complexity	(21)	F-2	X	A rating of the complexity in the use, interactions and/or documentation of the support software. Rating scale is 1-10, where 1 = easy, 10 = most difficulty.

P1 PROJECT ATTRIBUTES INFORMATION FORM (cont'd)

Parameter	Key	Format	E/A	Description
Data Base Structure Complexity	(22)	F-2	X	A rating of the complexity of the data base, size and structure. Rating scale is 1-10, where 1=easy, 10=most difficult
Quality of Requirement Specifications	(23)	F-2		An evaluation of the clarity, completeness, implementability and verifiability of the project requirement specifications. Rating scale is 1-10, where 1 = very high quality, 10 = poor quality.
Quality of Design Specifications	(24)	F-2	X	An evaluation of the design specifications for their completeness, clarity, and detail. Rating scale is 1-10, where 1=high quality, 10=poor quality.
Schedule Adequacy	(25)	F-2	X	An evaluation of the tightness of project scheduling in view of total project. Rating scale is 1-10, where 1= adequate, 10 = inadequate.
Overall Project Management Effectiveness	(26)	F-2	X	An evaluation of the management control of the project based on the stringency of administrative plans, configuration control procedures, technical direction given, etc. Rating scale is 1-10, where 1 = effective management, 10 = ineffective management.
Overall Project Personnel Qualifications	(27)	F-2	X	An evaluation of the project personnel, including management, technical and administrative support people, in meeting the projects performance goals. Rating scale is 1-10, where 1 = highly qualified, 10 = poorly qualified.
Computer Resources Adequacy	(28)	F-2	X	An evaluation of the computer resources and services to meet the requirements of the project. Rating scale is 1-10, where 1 = most adequate, 10 = highly inadequate.
Customer Supplied Information	(29)	F-2	X	An evaluation of the customer supplied data and/or equipment based on the completeness, timeliness and accuracy (freedom from errors and deficiencies). Rating scale is 1-10, where 1 = high quality, 10= poor quality.

P1 PROJECT ATTRIBUTES INFORMATION FORM (cont'd)

Parameter	Key	Format	E/A	Description
Timeliness of Review Actions	(30)	F-2	X	An evaluation of the length of time it takes to process an item through the review and approval cycle, especially in terms of exceeding scheduled review periods and priority, importance or immediacy of the need for a decision. Rating scale is 1-10, where 1 = most expendient, 10 = most time consuming.
Funding Adequacy	(31)	F-2	X	An evaluation of the adequacy of project funds to meet the software deliverable end items. Rating scale is 1-10, where 1 = adequate, 10 = most inadequate.
Project Software Type	(32)	F-2		Indicate the gross classification of programming choices include: BUSINESS SCIENTIFIC SYSTEMS MAINTENANCE OTHER

P7 COMPUTER EQUIPMENT INFORMATION FORM

DATE OF SUBMITTAL: (1) _____ PROJECT IDENTIFIER: (2) _____

DEVICE IDENTIFIER: (3) _____

DEVICE DESCRIPTION STATISTICS:

Memory Size: (4) _____ Unit of Measure: (5) _____

Number CPU's: (6) _____ Number I/O Channels: (7) _____

Memory Cycle Time: (8) _____

Unit of Time Measure: (9) Nano ☐ Micro ☐ Sec ☐

DEVICE TYPE: (10) Mini ☐ Micro ☐ Midi ☐ Maxi ☐ Special Purpose ☐

SECONDARY STORAGE STATISTICS:

Number Tape Drives: (11) _____

Number Random Access Devices: (12) _____

MAJOR INPUT DEVICE TYPE: (13) Card ☐ Paper Tape ☐ Terminal ☐

PRODUCT IDENTIFIERS: (14) _____

_____

P7 COMPUTER EQUIPMENT INFORMATION FORM

To be completed and submitted at project initiation for <u>each</u> computer configuration used by the project for software development. This form identifies the computer equipment capabilities.

Parameter	Key	Format	E/A	Description
Date of Submittal	①	F-6		Current date, either project initiation or completion date, in the format yymmdd.
Project Identifier	②	F-8		An acronym, number or other identifier that uniquely specifies a project, and identifies all data collection forms for the project.
Device Identifier	③	F-24		The name of the computing device employed, including the manufacturer of the equipment, the series number, and the model number.
Memory Size	④	F-10		The amount of information the computer memory can store and base, e.g., 64K.
Unit of Measure	⑤	F-10		The unit by which the storage capacity is measured, e.g., bit, byte, word.
Number of CPU's	⑥	F-4		The total number of central processing units associated with the identified computer.
Number of I/O Channels	⑦	F-4		The number of hardware devices that connects the CPU and main storage with the I/O control units.
Memory Cycle Time	⑧	F-8		CPU cycle or access time.
Unit of Measure	⑨	F-5		The unit by which cycle time is measured, e.g., nanoseconds, micro-seconds, seconds.
Device Type	⑩	F-7		The general classification of the computer equipment according to size. Mini - A computer with a portable mainframe. Micro - A computer that is micro-programmable and is also portable. Midi - A medium size computer, e.g., PDP10, IBM 360/20-370/158. Maxi - A large scale computer capable of multiprocessing, e.g., CDC 7600, IBM 370/191.

P7 COMPUTER EQUIPMENT INFORMATION FORM (cont'd)

Parameter	Key	Format	E/A	Description
Device Type (cont)				Special - Computer built to specific specifications for a particular application.
Number Sequential Access Devices	(11)	F-6		Number of devices providing secondary storage of sequential access type, e.g., tape drives.
Number Random Access Devices	(12)	F-6		Number of devices providing secondary storage of random access type, e.g., discs, drums.
Major Input Device Type	(13)	F-10		Type of input device that provides the major percentage of input data Choices include: Card - Punched cards Paper tape - Punched tape Terminal - Remote site input
Product Identifiers	(14)	F-8		Identifiers of product elements using this hardware device. This field may be left blank when a single computer device is used for the development of all products. In the event that more than one device is identified for software development, specify the highest level Product Identifier using the device, e.g., the subsystem name.

W1 WORK DEFINITION INFORMATION

DATE OF SUBMITTAL: (1) _____ PROJECT IDENTIFIER: (2) _____ _____

WORK LEVEL: (3) Project ☐ Phase ☐ Task ☐ Activity ☐

WORK IDENTIFIER: (4) _____

WORK DESCRIPTION: (5) _____

IDENTIFIERS OF REPORTING WORK ELEMENTS: (6) _____

INITIATION DATE: (7) _____ COMPLETION DATE: (8) _____

TERMINATOR: (9) _____

RESOURCE UTILIZATION DATA

RESOURCE ID	UNIT	ALLOCATED	EXPENDED
(10)	(11)	(12)	(13)

PRODUCT IDENTIFIERS: (14) _____

W1 WORK DEFINITION INFORMATION FORM

The Work Definition Information Form provides data on the work breakdown structure in an hierarchical manner, showing the relationship of all work elements and the chain of authority for each work element the software development project has defined. The form is to be completed and submitted at project initiation, work plan formation, or when the work package allocation is changed. A Work Definition Information Form is to be completed for each element for which resource allocation is made.

Parameter	Key	Format	Description
Date of Submittal	①	F-6	Current date, either project initiation, work plan formation or change in work plan definition, in the format yymmdd.
Project Identifier	②	F-8	An acronym, number of other identifier that uniquely specifies the project, and identifies all data collection forms.
Work Level	③	F-8	Indication of the work breakdown level. Choices are project, phase, task, activity.
Work Identifier	④	F-8	A name or number uniquely identifying this particular work element, and for which manpower, computer, etc., resources will be allocated.
Work Description	⑤	V-256	A brief narrative description of the work to be performed including the purpose, scope and method for this element.
Identifiers of Reporting Work Elements	⑥	F-8	Identification of all of the elements into which this work element is subdivided.
Initiation Date	⑦	F-6	The calendar date for starting the work element, in the form yymmdd.
Completion Date	⑧	F-6	The calendar date for completing the work element, in the form yymmdd.
Terminator	⑨	F-12	The action taken that completes the work element being defined. This may be a milestone, identified on the W8 form, an informal review, or a delivery.

W1 WORK DEFINITION INFORMATION FORM (cont'd)

Parameter	Key	Format	Description
Resource Identifier	(10)	F-12	A short name identifying the specific kind of resource to be utilized, e.g., personnel classification, machine type, travel type, computer time, storage, etc.
Resource Unit	(11)	F-10	The basic unit of expenditure of the resource, as manhours, mandays, hours, minutes, etc.
Resource Allocated	(12)	F-8	The total amount of the resource unit allocated or budgeted for the total work element.
Resource Expended	(13)	F-8	The amount of allocated resource expended to the reporting date for this work element.(Generally, this field is blank since the form is submitted at initiation of the work element, prior to resource expenditures. However, in the event that resources have been expended, include all expenditures for this work element to date.) Calculated from monthly status reports after initial input.
Product Identifier(s)	(14)	F-12	The unique identifier of the specific product, or service, whose production is evaluated. (The combination of characters uniquely identifying the work element and associated products together form the key by which all products within all work elements can be identified. Resource expenditures and productivity data are periodically collected via these identifiers.) All products identified must be described on the W2 Product Identification Form.

6.5 DATA ENTRY CONFIGURATION

The data entry system must meet the following requirements:

- Transmit data from the point of acquisition to the central repository.
- Convert the data to a machine readible mode.
- Detect and correct input errors.
- Be flexible in adapting to input changes in individual item values, additional parameters and variation in forms.
- Match the highly structured, modularized and interrelated data collection forms that have been devised and recommended by these reports.
- Operate as rapidly as required to support data collection and repository requirements in a cost-effective manner.

The data entry configuration must conform to the concept of operation adopted for the data repository in the following respects:

a. If the data repository is to operate in an off-line, statistical report research mode, stressing historical data over an extensive period of time without immediate support for project management or monitoring, it is recommended that data entry be oriented to a largely manual collection, batch entry mode.

b. If the data repository is to operate in an interactive mode, stressing immediacy of response and currency of data with a potential for immediate support of project monitoring and management, it is recommended that point-of-collection data entry and digital data transmission techniques be used with a largely transaction oriented entry mode.

For the pilot facility, collecting data from a small number of projects and supporting a limited amount of research in a trial basis, transmission of manual data collection forms to the central facility for key board entry is

deemed the most economical procedure, and it should prove satisfactory for the trial period. For an expanded facility, collecting data from a great many projects and supporting a large amount of research, it is strongly recommended that machine-readable input only be accepted at the central repository. A manual back-up capability must be preserved in case of failure, exceptional cases and error correction, but machine-readable input is necessary to avoid overload of data analysts and keyboard operators at the central facility.

In fact, data entry at or near the point of acquisition is highly desirable, not just to provide machine-readable input to the repository but to avoid much of the subjectivity and bias associated with subsequent manual processing to summarize, average and filter the data. The preferred data entry device is an intelligent terminal with moderately powerful editing logic, moderate storage capacity, and a CRT display capability for use in verification and editing of entries. Alternatively, the CRT terminal may be driven by a local computer. Either option has the following advantages:

- A superior editing capability
- Interactive input and error prompting
- Storage and review prior to final entry
- Flexibility in adapting to changes
- Entry routines easily reflect modularized and structured input forms
- Machine-readable and edited output wherever the entry device is located

It is also recommended that a standard project monitor system be defined and developed. The monitor should operate in conjunction with the intelligent terminal capability. A project monitor has great advantage in fostering and enforcing data collection standards by incorporating standard, objective and automatic algorithms for filtering, averaging and summarizing data in the monitor. Further, a project monitor represents an immediate payoff to the data suppliers in terms of improved project manage-

ment. Two approaches to a standard project monitor are available, either one is quite costly. Standard hardware and software (a very powerful "intelligent terminal") may be provided to all projects. This alternative does not appear tenable, considering the number and variety of projects. It also introduces some interface complexities in integrating the project monitor with program library operations in the production computer. The alternative of converting a standard software monitor to operate on many computers is also expensive. Despite the expense, this study believes that the degree of standardization and objectivity necessary to produce reliable and comparable data for valid research results can only be obtained by automating data collection at close proximity to the source.

No direct recommendations is made for the transmission of data from intelligent terminals and project monitors to the repository. If the data collection system is used to support research only and research results are not used to adjust the operation of the software development process, manual transmission of machine-readable media is quite adequate. If the data collection system is used beyond the local data base to support project management, or if research results are to influence software development, a more rapid data transfer, such as digital data communication lines, is indicated.

6.6 DATA BASE AND DATA BASE MANAGEMENT

The interface between the data collection and the data base management systems is determined by a large number of factors including:

- The degree of data base centralization
- The structure of the data base
- The capabilities of the data management system
- The nature of the data retrieval capability

If only basic, long-term research is supported using derived data, a completely centralized data base is indicated. If immediate project management is supported as well as research, localized data stores and project monitor systems are recommended to buffer input to the central store and to interact with the centralized data management system.

A centralized system is easier and cheaper to implement and operate than a decentralized system, but participating projects receive little or no benefit from the repository and may resist supplying data. Unless machine-readable data is delivered, key entry at the central repository will be at a high volume; error correction from a central location entails substantial delays without on-line access to distant data sources.

A decentralized system is more costly and difficult to implement and operate than a centralized system, but can provide direct benefit to data suppliers. Project management may also feel that their data base is more secure if it resides at the project site and data may be desensitized before transmittal to the central store. That is, fine detail may be collected at the project site, but only filtered, summarized data need be sent to the central data base. Data entry at the local data base permits immediate notification and correction of errors. Local data bases may also be more amenable to the employment of automatic collection capabilities since direct interfaces with program library operations, program production tools and computer operating systems are possible.

The data base structure employed must be flexible to permit growth over time and to allow new logical views of the data to be added as old ones are deleted. Serial records that are related through indices offer the most flexible structure and it is recommended that:

- Records be written serially on a device.
- Direct access storage be used for current records and tape for historical records.
- Records be indexed on one or more keys.
- Indexes be kept on direct access storage.
- Pointers in one record type be allowed to point to records of another record type or to an index containing pointers to records of another record type.

Flexibility is also needed in the data management system to provide for change in the data base and to adjust to the unforeseen needs of research users. It is recommended that a data management system be developed whose physical data base structure is transparent to the user. That is, the user should view the data base in a logical structure that is optimum to his employment and not be concerned with the mapping of his view onto the physical structure of the data base. To implement this recommendation, the system should:

- Provide a data definition language capable of defining multiple logical views of the data base by building a table to be used by access routines to map the logical views to the physical view.
- Provide sufficient independence of the logical from the physical view so that new data items and record relationships can be added without affecting the existing logical views.
- Include security keys that must be supplied before data base access is allowed.
- Prevent access to data items outside the logical view of the user.

- Defer batch updates to periods of minimal activity to avoid problems of deadlock and restart.

The data retrieval procedures do not have a major impact on the data collection system, but are intricately involved with the data base structure and the data management system unless the data retrieval requests are handled over the same communication channels as data entry. Data retrieval will range from standard reports to analytic requests to random queries. Extraction of data subsets for particular users is also a possibility. If the data collection and repository agency generates all reports, a less sophisticated security system is necessary than if access is semi-public. If the central repository is used to support management on a local or regional basis, the interaction frequency will be greatly increased. However, if local or distributed data bases are used, the level of activity at the central repository will be lessened. Simulation tools, software development models and project management tools may be developed for use with the data collection and data retrieval system. Only minimal direct impact should be expected on data collection procedures but the data base might need to change to support these tools.

6.7 REPOSITORY OPERATIONS MANAGEMENT

The relative merits of having the data collection and data repository facility managed by Air Force, Civil Service or civilian contractor were evaluated. The evaluation criteria used were the degree of flexiblity a particular agency would have in adjusting to changing requirements, the responsiveness of the agency to contract directives, the productivity of the agency, and the cost. It is also possible to assign specific functional operations to one or the other kinds of agencies. For instance, operation of the computer facility and the document library facility might well be assigned to one agency or be a time-shared facility with other operations, while data entry, system support and research analysis are done by another agency.

The results of the considerations are summarized in Figure 6. The rankings shown in the figure reflect fractional differences; in almost every case, a 'best case' argument for a less favored agency is better than a 'worst case' for others since individual persons and agencies vary widely in performance.

While Air Force agencies are normally flexible in organization and operation, they are often plagued by problems of personnel availability and turnover. Civil Service has better access to trained personnel, but manning restrictions and red tape impair its ability to adjust. Civilian contractors are normally quite flexible in meeting changed contract requirements and usually have access to trained personnel. Using Air Force or Civil Service personnel for stable positions and contract personnel for positions subject to rapid change might provide the greatest flexibility, but overall, a plus was given to the contract personnel.

Although Air Force and Civil Service personnel operate under the direct command of RADC personnel, penalties and rewards for performance are difficult to apply. The civilian contractor is the most effectively controlled, given the proper provisions for incentives in the contract. Controlling a mixed group is probably most difficult because of the diversity of authority and complexity of relationships.

MANAGEMENT OPTION	Flexibility	Controllability	Productivity	Cost	Rank
Civilian Contractor	+	+	+	0	1
Civil Service	-	-	0	0	4
Air Force	0	0	-	+	3
Mixed Group	0	0	0	0	2

Key

+ = positive

0 = neutral

- = negative

Figure 6. Decision Matrix for the Selection of an Operations Management Agency.

The productivity of Air Force personnel is inhibited by performance of other responsibilities, frequent turnover and inadequate training. Civil Service job standards ensure that adequately trained persons are assigned; but while stability is high, liberal vacation and sick leave policies may cut into productivity. Although levels of experience and training are not likely to be as rigidly observed as in civil service, contractor personnel are more subject to incentives and actual removal from the job if performance is not satisfactory, keeping productivity high.

Although cost is a critical factor in the evaluation of a potential operations management agency, there is little basis upon which to choose an agency. Although the Air Force may have a slight advantage, large salary variations with skill and experience obscure the differences among the agencies and choice should probably be made on other grounds.

This study recommends that the data collection, entry and retrieval system be managed by a single civilian agency under contract to RADC. As compared to either Civil Service or Air Force management, operations management by a private contractor is judged to be:

- More flexible in meeting changed demands.
- More easily controlled due to contractual control over financial incentives and more easily criticized than "in-house" agencies.
- More productive in that incentives and punishment are more readily applied.

On a body for body basis, there seems no clear cost advantage for any of the potential management agencies. Air Force personnel tend to have lower salaries, offset by the amount of time that may be devoted to other duties, high (albeit hidden) overhead costs, and high turnover rate. Civil Service tends to get equal salaries with private industry, plus generous fringe benefits.

One potential drawback to civilian management lies in the reluctance of other corporations to release information to a potential competitor, a real but surmountable difficulty. Hence, in view of the greater responsiveness of the contract agency, a civilian contract agency appears to be the better choice. It is true that inadequate contract provisions can lead to poor performance through both lack of guidance and inefficient application of penalties and incentives, but an opportunity to correct these arises each contract year. Operations management or support contracts are currently quite popular in a fiercely competitive market. The prospect of acquiring a competent, efficient operation is quite good.

7. IMPLICATIONS FOR FURTHER RESEARCH

Creating a viable data collection and software data repository will require a great deal of additional investigation and system development effort. While numerous recommendations have been set forth here, many of them are contingent upon the design decisions that are made for the total system. Assuming that an overall operational concept and system configuration has been selected and the data collection and data management system recommendations set forth by SDC and IITRI have been adopted, system specifications should be produced to set forth the requirements for:

- Acquisition, development and/or integration of software development tools into the data collection system.
- Determination of the effectiveness of the data collection procedures and parameters in meeting the needs of research in terms of data volume, data base structure, and data management system requirements.
- Selection of the mode of operation and appropriate hardware/software requirements.
- Development of an extensive set of user oriented manuals, including:
  - Data and terminology standards.
  - Procurement (contract) provisions to support the data system.
  - Mechanisms for data collection system maintenance and improvement.
  - System examples to illustrate usage.
  - Classification appendices to support complex data category values.
- Institution of repository studies to develop more objective indices, especially in the project environment attributes.

- Examination and incorporation of more comprehensive configuration management controls with the data collection procedures. (Only minimal requirements are incorporated in the suggested data collection forms).
- Investigation and specification of data sampling, filtering and summarizing algorithms to provide more objective data reduction to avoid the distorting effects of subjective procedures.
- Performance of trade-off studies of project monitors and intelligent terminals capable of local data entry editing and preparation of machine-readable reports for the central repository. (It is felt that submitting a high volume of detailed manual data forms for keying at the central repository may prove infeasible.)

It is felt that the following software capabilities need to be developed:

- An input editor or error analysis program to validate the data submitted to the repository while automatically obtaining error statistics related to the completion and submission of the collection forms.
- A data management system sufficient to meet the recommendations set forth in this study.
- A report generation system sufficient to meet the needs of RADC research and user requests for analyses.
- Statistical analysis programs to perform sampling, filtering summarizations and other data reduction tasks.
- Semi-automatic management systems or project monitor to satisfy the management needs of the project and the research needs of the repository as a data entry buffer system.

☆U.S. GOVERNMENT PRINTING OFFICE: 1977-714-025/83

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.