AD-A049 869 SCIENCE APPLICATIONS INC MCLEAN VA F/G 9/2
SOFTWARE ENGINEERING. VOLUME I. ITS DEVELOPMENT AND STANDARDS.(U)
JAN 78 DACA88-76-C-0010
UNCLASSIFIED CERL-TR-E-126-VOL-1 NL

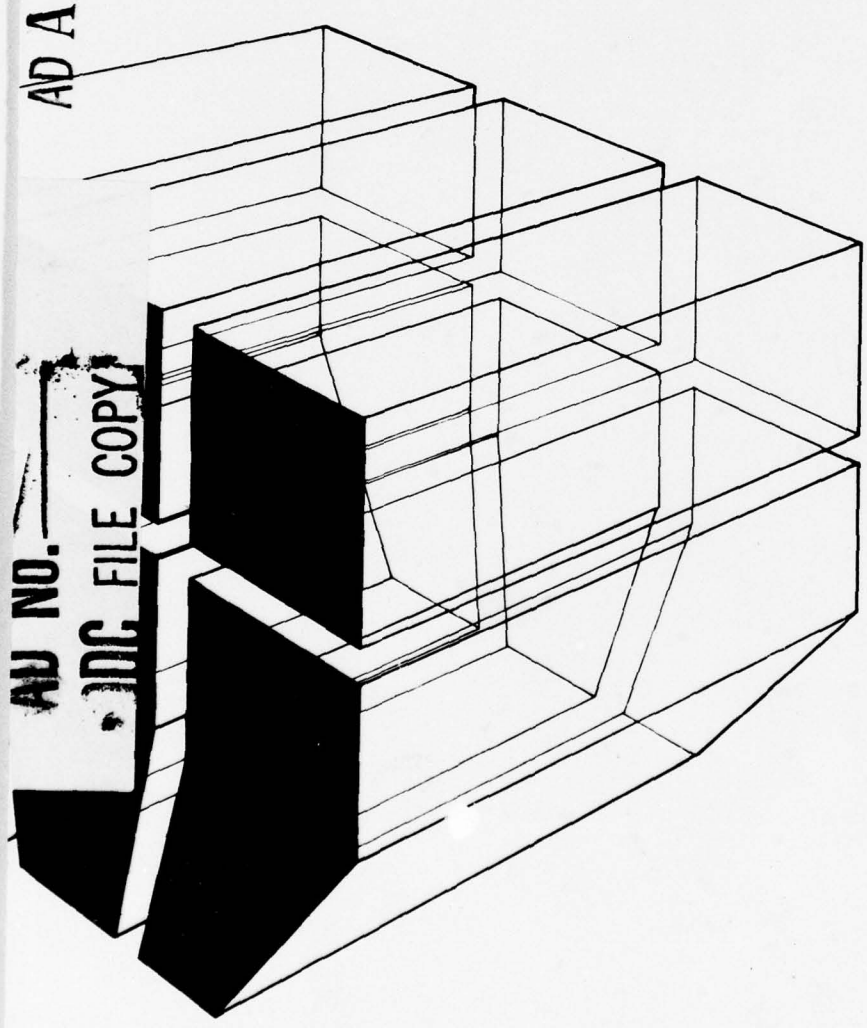
1 OF 2
AD
A049 869

construction
engineering
research
laboratory

12 p.s

TECHNICAL REPORT E-126
January 1978

SOFTWARE ENGINEERING
VOL I: ITS DEVELOPMENT AND STANDARDS



CERL

Approved for public release; distribution unlimited.

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official indorsement or approval of the use of such commercial products. The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED
DO NOT RETURN IT TO THE ORIGINATOR

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 **REPORT DOCUMENTATION PAGE**

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER (18) CERL-TR-E-126-VOL-1	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) (6) SOFTWARE ENGINEERING, VOLUME I, ITS DEVELOPMENT AND STANDARDS,	5. TYPE OF REPORT & PERIOD COVERED (9) FINAL rept.
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s) (15) DACA88-76-C-0010 New
9. PERFORMING ORGANIZATION NAME AND ADDRESS Science Applications, Inc. McLean, VA 22101	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (16) 4A762725AT11
11. CONTROLLING OFFICE NAME AND ADDRESS (11) CONSTRUCTION ENGINEERING RESEARCH LABORATORY P.O. Box 4005 Champaign, IL 61820	12. REPORT DATE Jan 78
	13. NUMBER OF PAGES 108 (12) 109p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

Copies are obtainable from National Technical Information Service
Springfield, VA 22151

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

software development
software engineering
standardization

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

This report presents the findings of a study to evaluate the maturity of the software development process and its accompanying standards, project future development in the field, and to identify areas needing standards and the means for achieving the required standardization. The report examines development as a generic process to provide a perspective for describing the evolution and present state of software engineering. The purpose and contributions of standards in development are defined and described. Analysis of

2 rept

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

408 404

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Page

Block 20 continued.

software engineering's evolution to date indicates that it is still in the active, experimental stages of its evolution, in which it is forming the elements of principle and practice on which its mature character will be built.

An analysis of the phases of the development cycle in terms of their required inputs and the availability and applicability of standards for these inputs indicated that most procedures and inputs to software engineering are not now standardized. Thirteen specific areas requiring standards were identified. Such standardization is essential to the further evolution and maturation of the discipline. Steps needed to attain standardization are outlined. These suggested steps are necessarily aimed at the top levels of a national standards formulating group and their general scope is at least national. As a consequence, unilateral follow-on action by the military participants supporting this study has not been recommended. Instead, this study attempts to present a systematic approach to software standards and to raise national awareness of the need for and benefits to be derived from improved software standards. In this way, it is hoped that the military will be but one of the beneficiaries of broader national action.

The report also provides a list of organizations working on computer and information sciences standards and lists of references on software standards.

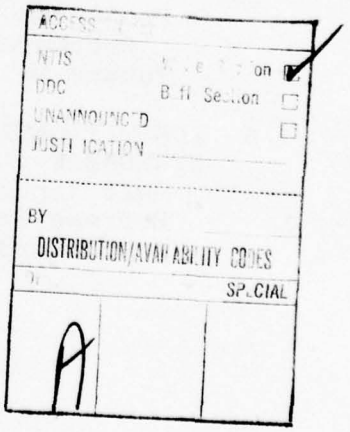
Volume II of this report provides a list of individuals who are participating in software engineering standardization.

FOREWORD

This study was performed for the Energy Branch (EPE), Energy and Power Division (EP), of the U.S. Army Construction Engineering Research Laboratory (CERL) by Science Applications, Inc., McLean, VA, under Contract No. DACA88-76-C-0010. Mr. D. C. Hittle was the CERL Principal Investigator.

The work was funded by the Directorate of Military Construction, Office of the Chief of Engineers (OCE), under Project 4A762725AT11, "Engineering Software Development." The OCE Technical Monitor is Mr. R. A. McMurrer.

COL J. E. Hays is Commander and Director of CERL and Dr. L. R. Shaffer is Technical Director. Dr. D. J. Leverenz is Chief of EPE and Mr. R. G. Donaghy is Chief of EP.



CONTENTS

CONTENTS (Cont'd)

SOFTWARE ENGINEERING: VOLUME I, ITS DEVELOPMENT AND STANDARDS

1. INTRODUCTION

Background

Software engineering is a relatively new discipline whose development has been shaped by the increasing demand for its product. Because of the pressures created by this demand, software engineering has evolved very rapidly; however, its development has been subjected to very little of the critical analysis required to indicate where future development and standardization are needed. This study therefore represents a first attempt at drawing together many of the general concepts of software engineering and related information as an initial step in the refinement of the discipline of software engineering.

Objective

The objectives of this study were (1) to evaluate the maturity of the software development process and its accompanying standards, (2) to project future developments in its evolution, and (3) to identify specific areas requiring standards and the means for achieving that standardization.

Approach

This study was conducted as a top-down analysis of the factors which define the development of the software engineering discipline. A common sequence of phases which can be used to classify the functions performed in all engineering disciplines was first defined (Chapter 2), and the role of standards in the development process of an engineering discipline was studied (Chapter 3). The evolution of software engineering and its present status

were then examined in light of the structures developed in the first two steps (Chapter 4) as a basis for evaluating its maturity and projecting probable future directions in software engineering (Chapter 5). Based on the preceding steps, each software engineering phase was dissected into its unit inputs, and the current availability of those inputs was determined; from this analysis, a list of specific areas requiring standards was developed (Chapter 6). Chapter 7 outlines the steps required to achieve the recommended standardization, and Chapter 8 summarizes the study conclusions.

A list of organizations participating in software engineering standardization and of relevant reference works was developed in performing the above steps. Appendix A lists participating organizations. The bibliography (Appendix B) presents a wide variety of works, including standards, discussions of methodology, and subject-oriented articles. Short abstracts are provided to identify the subjects discussed or standardized. Volume II of this report provides a list of individuals who are participating in software engineering standardization.

2. AN OVERVIEW OF DEVELOPMENT AS A PROCESS

Before a perspective for viewing the evolution of software development was formulated, the generic nature of the development process was examined. If similarities exist between the evolution of the software development discipline and other engineering disciplines, these similarities may provide insight into the strengths and weaknesses or the level of maturation of the software discipline.

The distinguishing feature of the human species is its ability to manipulate its environment through the use of tools and techniques for their use. Development, or the art of invention, is the process by which tools and techniques are derived to meet a recognized need.

It is hypothesized that every invention is realized through a series of developmental steps or phases: direction, requirements, design, fabrication, verification, and dispersion. Each phase has a goal which must be attained before any of the following phases can be successfully achieved, unless by accident. A step may appear to have been skipped, as when sudden insight occurs, but such occurrences actually imply almost instantaneous completion of intervening steps in invention. The phases may occur somewhat in parallel or in a disjointed progression involving going back and forth between steps until necessary preliminaries exist for subsequent actions. The following paragraphs describe each of the phases.

The direction phase is the stage in which a need is recognized. The eventual product's explicit functions of structure may not be understood, but a void in the environment's resources is recognized. Statements of

direction define the void and will later result in statements of requirements or design in an attempt to fill the void.

This very important phase, sometimes also called the problem or concept phase, is the one which has been given least formal recognition. If the void is not properly given dimension, there is little hope that subsequent requirements, designs, or products will adequately fill it except, again, by accident. An incomplete or incorrect definition of the void is a major contributing element in the failure of many attempts at development, since any omission or error in this phase affects every subsequent activity.

The requirements phase involves generation of statements of those functions which must be performed to fill the defined void. The aim is a close fit in which the requirements fill, but do not overflow, the functional space of the void. However, because statements of direction often specify voids for which functions may only be probabilistically defined, this phase also involves recognition that only those problems which can be practicably approached can be solved.

Requirements can only be verified to have been met through the performance of the eventual product, and must therefore be capable of explicit demonstration. If statements can only be shown to be satisfied circumstantially, but not for all cases, they are statements of direction--voids which can be shown to be partially filled. This again demonstrates that the purpose of the requirements phase is to define that portion of the directed void which will be filled.

The design phase identifies the physical structure of the product which will perform the functions defined by the requirements. While specifically attempting to satisfy requirements during the design phase, the designer must be fully aware of the direction upon which they were based, so that any necessary design excesses can be channeled toward these ultimate goals. Satisfaction of design statements can only be verified through examination of the product. Thus an implemented design can be seen or physically measured in some way which does not require that the product perform its required functions.

The fabrication phase produces the product to meet the design, thus providing the physical form and substance of the development product. Its common feature is the conversion of raw materials into a structurally or functionally different form to fill the defined void. This is the phase which is most significant to the engineering discipline supporting the development cycle.

The verification phase provides assurance that the translation conducted at each of the preceding four phases is true to its predecessors. This includes verification that the direction describes an actual void that fits within the dimensions of a real world. Verification is not a phase in the sense that the other steps are, since it must occur with and support every other preceding phase and is thus actually a continuous function.

The sixth phase, dispersion, involves the formal changing of the product from something under development to something in use. The level of formality of such a transition depends on the particular development. It may involve much training and transfer of responsibilities. This phase provides a point at which the five preceding

phases may be examined to determine whether their performance satisfied the rigor expected of an engineering discipline. Unfortunately, in most development processes, this is the only phase in which the product's eventual user participates.

These six phases can be seen as six dimensions in a simultaneous equation. Being concurrently aware of the status and impact of each dimension on all the others is quite often impossible. Rules for development, usually called standards, have therefore been developed to provide continuity between the dimensions. Chapter 3 defines the role of standards in the development process and describes how their development reflects the maturity of an engineering discipline.

3. THE ROLE OF STANDARDS IN THE DEVELOPMENT PROCESS

Standards as a Measure of an Engineering Discipline's Maturity

From the beginnings of engineering, man has continually expanded and divided its disciplines. Disciplines usually begin with the definition of unusual phenomena or the recognition of principles which are to some degree marginal to an existing discipline. Thus, statistics were born from a certain degree of frustration with the performance of other disciplines against the exacting demands of mathematics.

Each discipline is also continually undergoing evolution, which may involve subdivision into new disciplines, formalization, or redirection. In the early stages, a discipline has little organization or coherence. How long such early stages last depends on the incentive behind the organization, the recognition of basic principles, and the investment made in formalization. During the past century, disciplines have tended to mature rapidly because of the great pressures applied to exploit them. Recently, however, constraints on research financing have limited this "exploitive evolution."

From these early stages, a discipline moves toward the definition of principles and their implementation as standards. Standards allow the continued, more sophisticated evolution of a discipline to occur upon common bases. The evolutionary status of a discipline can therefore be measured by how directly the principles which are its foundation are translated into the standards which control its exploitation.

The following sections describe standards and their specific role in the development process.

Standards as Inputs to Development Phases

Although most studies of development phases have focused on the outputs from those phases, considering inputs offers significant advantages. It permits consideration of inputs provided by the engineering discipline involved, as well as the inputs derived from the output of other phases. While the inputs from other phases tend to be peculiar to the application or product being developed, the inputs provided by the engineering discipline are standard unit inputs to that phase of development. The standard unit inputs--or standards--include the building blocks from which items meeting application specifications will be selected, and standard procedures which have been demonstrated to be reliable and effective.

Application of standards provides implicit continuity and communication. Their use relieves experienced practitioners of an engineering discipline of a large part of the definition and documentation process, and helps the uninitiated to learn the basic building blocks of the discipline quickly. Since standards are used at comparable stages in every development, they provide the framework for evaluating the relative merits of activities and products.

The role of the standard as a unit input to the development process is therefore crucial to the efficiency and evaluation of that process. To understand the evolution and maturity of a process, what kind of requirements are met by standards and how they are met must be examined.

Requirements for Standards

Consistency is the basis of all requirements for standards. A generator of standards must balance the positive effects of consistency against its potential to inhibit creativity. Among the possible benefits of standardization are simplification of the growing variety of products and procedures in human life; communication; overall economy; safety, health, and protection of life; protection of consumer and community interests, and elimination of trade barriers.[1] Although these are only a few of the benefits, they do reflect elements of "quality" which is the normal target of the consistency imposed by standards.

Standards promote common bases for all instances of a defined context to assure compatibility with preceding or subsequent activities, or to provide a basis for measurement. Standards thus do not apply or exist in a uniquely occurring circumstance. The need for standards depends directly on the frequency with which a context occur, and the negative impact of a lack of consistency.

Standards may exist whenever repetitive processes are performed or repetitive circumstances exist. These prerequisites are met in most activities during developments within an engineering discipline. Standards support the transmitting of engineering technology, but can inhibit the inventiveness required in engineering developments if misapplied. Standards can also overshadow the peculiar requirements of an application or development.

---

[1] T.R.B. Sanders, *The Aims and Principles of Standardization* (International Organization for Standardization, October 1972) pp 3, 17.

The maturity of an engineering discipline is thus not only reflected in the existence of standards, but also in the degree to which those standards meet their own requirements without impeding the achievement of other goals in the development activity.

Characteristics of Standards

This section discusses two of the many characteristics of standards: mode of imposition and content of origin or formalization. Modes of imposition are generally characterized as voluntary or mandatory. Imposed military standards and specifications are mandatory, as are linear and columetric measures. If the recipient of a product or service requires that it meet a certain standard, that standard is mandatory. Standards organizations such as the American National Standards Institute (ANSI), the National Bureau of Standards (NBS), and the International Organization for Standardization (ISO) are generally concerned with standards which become mandatory through usage. For example, a building contractor expects a "2 by 4" to measure 1 1/2 in. by 3 1/2 in. (38.1 by 88.9 mm). This standard is not made mandatory by a law requiring that it be met, but by its assumption in modern construction design, and the contractor's refusal to accept materials of larger or smaller dimensions.

Voluntary standards are those which can be presumed if desired. A developer may voluntarily impose certain procedural and unit standards to encourage communication and enhance subsequent activities. Repetition of a function leads to development of voluntary standards by producing behavioral expectations on and for others. Although the requirements which such standard behavior supports may not be easily verbalized, the behavior does promote continuity, and is therefore a standard.

The context of origin or formlization of a standard is its recognition in explicit terms. Standards are usually either adopted from some other standard which has been applied, or specially produced in anticipation of a need. Most formal standards organizations deal exclusively with adoptive standards. While their product may not be identical with any specific existing standard, it is an amalgamation of demonstrated candidates. Anticipatory standards are imposed when a new situation requiring continuity is foreseen. Such foresight is rare, and normally only emanates from an engineering discipline which has long, mature experience with standards and has learned to respect their contributions. When a discipline evolves simultaneously from an established discipline and an application-oriented environment, conflicts rapidly arise over anticipatory standards, unless they have clear empirical support.

4 EVOLUTION OF SOFTWARE DEVELOPMENT AS AN ENGINEERING DISCIPLINE

This chapter investigates the history of software engineering, the forces influencing its evolution, and its current status in the context of the discussions in Chapters 2 and 3. This analysis is used as the basis for evaluating software engineering's maturity as an engineering discipline and projecting future developments (Chapter 5).

History of Software Engineering

The *Modern Dictionary of Electronics* defines software as a program package available for work on general-purpose digital-computer hardware.[2] While several arguments can be made against this definition, it points out the computer as a primary prerequisite. The history of software engineering therefore begins around 1950, with the use of tube-technology, digital computers in several laboratories and for the national census. These machines worked on a direct encoding basis in which a limited set of explicitly implementable machine functions could be called upon by providing a stream of binary codes through manual, paper-tape or punched-card input. The concept of the stored program existed only in the rudimentary sense that an iteratively implemented binary control string could be temporarily stored while that iterative process looped.

The one-for-one instruction machine-action type of coding limited the amount of special-purpose processing that could be developed, but did not materially change the development cycle itself. The coding phase clearly occurred when a problem was first decomposed completely to prime logical operations with direct computer-operation equivalents. Thus, the development cycle for early

[2] *Modern Dictionary of Electronics*, 4th Ed. (Howard W. Sams and Co., 1972).

software engineering efforts can be clearly delineated into phases: the standards were those rigidly imposed by machine constraints. The main problems involved were the tedious tasks of programming and debugging machine-level codes.

This tedium is an important incentive for change in any engineering discipline. In software engineering, it first resulted in the scattered invention of "macros"--segments of machine code that implemented functions which were often needed and similarly applied in many different software applications. Macros evolved to fill the need to avoid repetitive programming and to reduce the need to debug the same or similar codes repeatedly.

In addition, around 1952, computers began to assume some of the work involved in preparing and placing machine codes into operational form.[3] Machines started providing simple storage allocations, mnemonic translations, and automatic incorporation of macros. Some early systems provided "assemblers" which performed permanent translation of a source code to an executable machine code, while others provided "interpreters" which translated source codes each time the code required execution. Assemblers, which were more efficient, are still used, while interpreters have evolved to support interactive programming.

Automated translation of codes to facilitate development has evolved rapidly since IBM introduced FORTRAN in 1957. These new levels, called languages, allowed users to write their codes in the much more natural (to the user) forms of formulae or English-like statements, and translated them to assembly codes which were again translated to machine codes. Languages soon proliferated to meet different demands (COBOL, LISP), and to overcome technical problems some people found with FORTRAN (ALGOL, MADCAP).

---

[3] Mark B. Wells, *Evolution of Computer Software* (Los Alamos Scientific Laboratory of the University of California, February 1971).

Languages have undergone minor and major overhauls to keep up with computer technology. Major advances have necessitated changes in names; FORTRAN and COBOL became PL/I, and ALGOL became PASCAL. However, these changes in languages have had little effect on the sequence of the development process, except to limit the applicability of software engineering standards.

Several concepts which have had a large impact on software engineering are flow charting, structuring, differentiation of the roles of programmer and analyst, verification and validation, and interactive programming. These concepts evolved from adaptations of methodology to software engineering needs, into accepted standards. Some have lost acceptance as their formalized modes became less responsive to the changing needs of the discipline.

Flow charting is a good example of this evolution. Conceived in several forms to provide visual understanding of complex functional and decision flows, flow charting symbols and semantics were standardized by industry, government, and other institutions. Unfortunately, flow charting has not been flexible enough to support representation of parallel or inter-leaved processing, to provide a strong control and understanding of pseudo-code levels higher than source codes, or to devolve into well-represented decision flows. While structuring allows flow charts to be reasonably portrayed, it also obviates the reason for flow charting, making codes approximately as readable, and providing more user information. While flow charting is still extensively used, its use in advanced applications is declining.

The concept of structuring was developed to restrict the number of decision constructs to a level which will

allow all necessary operations without overly impeding programming and will make programs as linearly direct as possible. Structured programming and structured design currently exist, but neither requirements nor testing has evolved enough formalism to require constraint. Structuring is an extremely healthy example of a discipline containing and constraining its implementation and design techniques to provide a more uniform and understandable product.

Early computer scientists were by necessity their own programmers. As computers became more common and their application stereotyped, the role of the software systems analyst emerged. This analyst acted as the design link between the customer's requirements and the programmer's implementation. As software applications have expanded, demand has continued to exceed supply for programmers, resulting in certain programmer substitutes, including interactive design and modularly built systems. Interactive programming has begun to impact implementation seriously, and should displace almost all other forms of program construction (barring an even better technique) within 10 years.

Finally, since about 1968, a new field--verification and validation (V&V)--has been defined. Absorbing the old concepts of configuration audits and testing, V&V has become a subdiscipline of software engineering with tools and techniques of its own. Although the software engineering discipline has, during this period, become aware that its tools and techniques are not keeping up with the evolution of hardware or the expansion of applications and users, V&V has not yet accomplished any meaningful changes in the actual methodology applied to the old problems. It

has simply consolidated activities, making them less redundant and expensive. Some attempts have been made to improve techniques by replacing man with automation or adopting techniques from other disciplines, but almost all have met with limited or no success.

Generic Forces in Software Development

Six of the major forces active in software engineering's evolution, both historically and currently, are demand, dependency, popularity, humanity, quality, and incorporeity.

Demand

Since the business and scientific worlds first recognized the fantastic potential in exploiting the capabilities offered by software engineering, their demand for software products has consistently grown. This demand has insured the industry's security as an on-going concern, but meeting the demand has resulted in impulsive and somewhat disjointed growth. The industry is now beginning to evaluate its past, present, and future in light of historical experience as a basis for formalizing its practices in universal, yet flexible, precepts that effectively serve its proponents.

Dependency

Research in any field is heavily subject to the dependence of the economic environment on it. Hence, software engineering was essentially born in the demands of military defense. However, when the business world discovered the economic savings and profit potential software engineering afforded, its investment in the data processing industry caused almost overnight expansion. Software engineering has also impacted national economies to the point where a kind of symbiotic relationship exists between the two.

Perhaps more significant, however, is the apparent independent security that software engineering enjoys apart from the overall economic condition in which it exists. This characteristic has tended to give it more continuity than many of the other engineering disciplines. The economy of the business world has become so entirely dependent upon software engineering that it is uneconomical for business not to exercise and exploit software engineering's potentials to the fullest.

Popularity

Because of its dynamic and evolving nature, software engineering has been significantly impacted by "fads." Publication of ideas, methodologies, and approaches without empirical justification or support has dramatically altered the course of the accepted and respected practices in the industry. Although some of these concepts have been tested and found to be worthwhile, others have lapsed into disuse when time failed to justify them.

Humanity

The ultimate purpose of software engineering intimately involves the human element. Although the specific and unique needs of man in a technological environment exert tremendous influence on all the engineering disciplines, human factors hold an especially significant place in the future of software engineering. With continued growth in the complexity of software systems, accommodations for human interface will become increasingly important. Standardization will be a means of simplifying this complexity at the human level.

Quality

The large quantity of products generated by the evolution and success of software engineering in the business

and scientific fields has presented fantastic maintenance demands requiring large financial commitments. Thus, economy has been the motivating factor giving rise to requirements for quality. Qualities such as reliability, transportability, and maintainability are receiving special emphasis in requirement and design specifications in reaction to the problems experienced historically in software maintenance and system development. However, these qualities have not yet been adequately defined in the objective, measurable manner which will provide the desired performance-monitoring concurrent with system development.

Incorporeity

Software, as executed, is a changing electronic state within a host computer system. It has no physical body or form and is thus incorporeal. Leading up to this truly incorporeal software are several levels of translations of symbolic representations of the desired electronic state changes. These take the form of magnetic codes, light codes, physical digital codes (cards and paper tape), and printed records of source or object codes. All are only representative of an electronic state transition sequence to be induced within a computer.

There is little physical limitation on the form of rendition which will induce the desired electronic transitions in a computer. New language compilers and other translators continue to be invented to meet requirements of specific applications. The incorporeal nature of software has allowed it to be extremely responsive to the needs of differing applications.

However, that same incorporeity has caused two problems which have greatly impeded the maturation of software engineering. First, because the media with which the engineer

must work physically impose almost no constraints on software engineering, the engineer is not limited by the tensile strength or friability of logic. Program sizing is rapidly disappearing as a problem in many applications. Programmers quite often deviate from certain coding practices and specified languages, implanting assembly code shortcuts, and taking advantage of peculiarities in a local computer environment. Second, the lack of structural constraint allows software to operate at a functional level of complexity beyond any mechanical equivalent. The combination of unprecedented complexity with arbitrariness of conventions and practices has aggravated many dormant problems in the development cycle.

Development Phases and Their Software Engineering Equivalents

This section describes contemporary software engineering in terms of the six development phases described in Chapter 2.

Direction

The direction phase in software engineering can be divided into three subphases: needs identification, conceptualization process, and problem definition.

Identification of a void in the environment initiates every software development process. The void, which is evidenced by some undesirable situation or circumstance, motivates researchers to seek some method that enables more effective or comfortable functioning within the environmental demands. Accurate identification of such a missing element is critical to the solution of any problem. Incorrect identification of a void will result in products which do not impact the real problem.

Avoiding propagation of faults from the identification subphase makes the conceptualization process critical. The insight needed to approach a problem appropriately depends largely on the educational and experimental background of the evaluator. This same education and experience, however, tends to bias and often inhibit the conceptualization process. Creative thinking is often suppressed, since its results are inherently untried and without empirical justification. Hence, the conceptualization process tends to follow traditional trends within the confines of the conceptualizer's experience.

The conceptualization process is also critical because of the gross impact it has on problem definition. Thomas B. Gildersleeve emphasizes that "a precisely stated problem is a long step toward solving the problem. After all, it's pretty hard to get someplace if you haven't spelled out where you're going."[4] Failure to properly identify the problem leads to the statement of what in actuality is not a problem at all but rather a solution, and eventually to the frustration of realizing the real problem has never been defined, much less solved.

Too often, surface problems are pinpointed, but the root causes are never identified. Much time, money, and effort may be wasted in dealing with surface causes, problems, and effects while the root cause, its associated problem and propagated effect are never addressed.

The direction phase of software engineering is thus crucial to the success or failure of satisfying user needs. It is primarily a user function and requires insight and foresight to avoid inadequacies that would propagate throughout the development life cycle. Adequate identification of needs, clearly defined conceptual processes, and

[4] Thomas R. Gildersleeve, "Insight and Creativity," *Datamation* (July 1976) p 91.

a proper problem definition are critical elements to giving any software engineering project a clear direction.

Requirements

In software engineering, the requirements definition phase of the development cycle specifies clearly and completely the elements of the product necessary to adequately meet the user's needs. Foresight is also needed in requirements definition, since any faults in this phase will be propagated throughout the resulting products. Measurability and change are key concepts in defining requirements. An objective standard of performance must be defined if progress, success, or failure are to be measured. Requirements should be specified in forms which can be clearly seen to be demonstrable through testing or examination at later development stages.

Appropriately defined measurable objectives can be used as guideposts to progress. Care in defining these objectives is essential, since these guideposts begin to define schedules, and schedule slippage historically has contributed significantly to the cost of software products. To avoid later schedule slippage, requirements which are likely to change over the software development life cycle must be identified early in the requirements definition phase. This is often difficult, since what changes in the user's needs will occur may not be entirely clear at this time. Nevertheless, if the potential for change is recognized and identified at this point, that change can often be accommodated in the design, resulting in greater flexibility in the final product.

Specifying requirements is still primarily a user function. The user should exercise keen foresight in

maintenance considerations and similar future involvement with the operational product. Maintenance concerns are often left out of software requirements; implementing them after product delivery results in additional work.

Design

The design process in software engineering is just that--a process, not a product. Although it is not practical to design dynamically, the design process must be sufficiently flexible to accommodate changes in user requirements. At some point, generally determined by the schedule instead of design preparedness, the product design must be "frozen" and only allowed to change when requirements arise which justifiably impact the product's response to critical user needs.

Standardization in design could lead to "building block" approaches in software products. This increased generalization is often associated with decreased efficiency. However, just as the "2 by 4" is an economically available component with sufficient versatility to merit widespread use in the construction industry, such building block design should also be possible, economic, and effective in software engineering.

Fabrication

The fabrication process in software engineering is where the actual product takes shape. It is the most thoroughly exercised and discussed phase in the software development cycle. Since this is the phase that initially seems to yield most readily to standardization, it is the area in which standardization was first attempted. Although a myriad of techniques, approaches, styles, and conventions exist, few methologies have any significant empirical support.

Verification

The verification phase in software engineering tests, evaluates, and insures that the process is moving toward the defined goal.

Verification can be facilitated by appropriate consideration and foresight in the requirements definition and design process phases. Unless provisions are made for test point interfaces in the design, verification may be difficult, or additional development work may be required. Unless requirements are clearly specified and design constraints clearly defined, verification tests may not be meaningful.

Definition of faults in the software development cycle is essential to the adequate definition of verification techniques for detecting development phase errors. Hence, the need for software qualities definition becomes acute when verification is attempted. An integrated methodology which considers verification throughout the previous software development phases is needed.

Dispersion

Dispersion in software engineering is the culmination of the development process. It represents the real test of the entire software development process--how effectively the product meets the real needs of the user. How well the requirements were expressed and whether standardization facilitated or limited success can be assessed. Communication in the developer-user direction is as essential as user-developer communication was earlier.

Summary

The direction phase includes the identification of

needs, and conceptualization and definition of the problem.

The requirement phase defines the software product in measurable terms.

The design phase, taking requirements as input, produces the "working drawings" and specifications (flowcharts or pseudo-language descriptions, perhaps; data structure specifications, overall program "architecture") from which the software is fabricated.

The fabrication phase is where software is coded or "built." Fabrication produces the product which is executed on a computer.

The verification phase includes first the verification of the software design, followed by testing of the software product.

In the dispersion phase, the software is made available to its intended users.

The Present State of Software Engineering Standards

A number of legislative bodies at various levels are currently impacting on software standards. Most of the legislative bodies concerned with software engineering standards operate through committees as consultants on their respective levels of government.

At the international level, the International Organization for Standardization (ISO) attempts to coordinate the national standardization organizations of each of its member countries to facilitate international exchange of products and to promote mutual economic and scientific cooperation. As an international nongovernmental organization, ISO functions as a consultant with the United Nations and other international consulting groups. ISO accepts draft proposals for international standards and assigns them to a technical committee for study. Its technical committee for computers and information processing (TC97) oversees multinational subcommittees responsible for both hardware and software standardization.[5] If and when a draft proposal is supported by a sufficient number of technical committee members, it becomes a draft international standard and is circulated among all members for approval. If approved by a sufficient number of members, the final standard is published as an international standard. Adherence to these standards is by consent of the participating member nations.[6]

---

[5] Marjorie F. Hill, *The World of EDP Standards*, Tech. Memo TM4 (Control Data Corporation, January 1973) pp iv-12 to iv-25.

[6] Hill, pp iv-21 to iv-22.

A second international organization with vested interests in software engineering standards is the International Federation for Information Processing (IFIP). IFIP, however, does not develop standards, but simply attempts to facilitate international communication in information processing through organizing symposia, sponsoring conferences, and establishing study committees.[7]

One of the national standards organizations in ISO/TC97 is the American National Standards Institute (ANSI). One of its 18 technical advisory boards--the Information Systems Technical Advisory Board (ISTAB)--seeks to establish software engineering standards. The ANSI X3 Technical Committee of ISTAB is the group concerned specifically with software. Its members include producers of hardware and software products, consumer associations, and societies with general interest in software.[8]

ANSI does not itself develop standards; rather, it provides the organization through which standards can be approved. The technical advisory boards review the technical content, a review board determines that consensus has been reached, and an approval board gives final approval.[9] Approved standards are implemented by publication and enforced by voluntary consensus.

At the government level in the United States, the National Bureau of Standards (NBS) is responsible for studying, establishing, and executing standardization. The Interagency Committee on Automatic Data Processing (IAC/ADP) and the Federal Information Processing Standards Coordinating and Advisory Committee (FIPSCAC) are two groups specifically concerned with the problems of Federal software

---

[7]Hill, pp iv-7 to iv-8.

[8]Hill, pp iv-1 to iv-25.

[9]Hill, p vi-9.

The Present State of Software Engineering Standards

A number of legislative bodies at various levels are currently impacting on software standards. Most of the legislative bodies concerned with software engineering standards operate through committees as consultants on their respective levels of government.

At the international level, the International Organization for Standardization (ISO) attempts to coordinate the national standardization organizations of each of its member countries to facilitate international exchange of products and to promote mutual economic and scientific cooperation. As an international nongovernmental organization, ISO functions as a consultant with the United Nations and other international consulting groups. ISO accepts draft proposals for international standards and assigns them to a technical committee for study. Its technical committee for computers and information processing (TC97) oversees multinational subcommittees responsible for both hardware and software standardization.[5] If and when a draft proposal is supported by a sufficient number of technical committee members, it becomes a draft international standard and is circulated among all members for approval. If approved by a sufficient number of members, the final standard is published as an international standard. Adherence to these standards is by consent of the participating member nations.[6]

---

[5] Marjorie F. Hill, *The World of EDP Standards*, Tech. Memo TM4 (Control Data Corporation, January 1973) pp iv-12 to iv-25.

[6] Hill, pp iv-21 to iv-22.

A second international organization with vested interests in software engineering standards is the International Federation for Information Processing (IFIP). IFIP, however, does not develop standards, but simply attempts to facilitate international communication in information processing through organizing symposia, sponsoring conferences, and establishing study committees.[7]

One of the national standards organizations in ISO/TC97 is the American National Standards Institute (ANSI). One of its 18 technical advisory boards--the Information Systems Technical Advisory Board (ISTAB)--seeks to establish software engineering standards. The ANSI X3 Technical Committee of ISTAB is the group concerned specifically with software. Its members include producers of hardware and software products, consumer associations, and societies with general interest in software.[8]

ANSI does not itself develop standards; rather, it provides the organization through which standards can be approved. The technical advisory boards review the technical content, a review board determines that consensus has been reached, and an approval board gives final approval.[9] Approved standards are implemented by publication and enforced by voluntary consensus.

At the government level in the United States, the National Bureau of Standards (NBS) is responsible for studying, establishing, and executing standardization. The Interagency Committee on Automatic Data Processing (IAC/ADP) and the Federal Information Processing Standards Coordinating and Advisory Committee (FIPSCAC) are two groups specifically concerned with the problems of Federal software

---

[7] Hill, pp iv-7 to iv-8.

[8] Hill, pp iv-1 to iv-25.

[9] Hill, p vi-9.

engineering. Under FIPSCAC are a number of Federal information processing standards task groups which work closely with NBS in particular areas of standards definition.[10]

The Center for Computer Sciences and Technology (CCST) within NBS recommends "uniform federal standards to improve compatibility in automatic data processing equipment procured by the government."[11] CCST is responsible for overseeing the work of FIPSCAC, which in turn executes the Federal Information Processing Standards (FIPS) program. The FIPS programs publishes the FIPS register, which lists the official documents specifying standards to which adherence is required throughout the Federal Government. These documents also specify the requirements private industry must meet when producing hardware and software products for government use.

While attempting to correlate with the FIPS program, the Department of Defense (DOD) is actively working toward establishing standards that serve the unique needs of its specialized environment. Because of its mission and organization, it is better able to exercise authority in implementing standardization. In the software engineering areas, the Department of Defense issues DOD directives, DOD instructions, military specifications, and military standards that impact all the military services (see the bibliography). Each of the services also defines and adminsters standards impacting areas of software engineering unique to their mission and environment (see the bibliography).

---

[10] *Federal Information Processing Standards Index*, FIPS Pub 12-2 (U.S. Department of Commerce, National Bureau of Standards, 1 December 1974) pp 76-79.

[11] Hill, p vi-36.

The annotated bibliography of existing ISO, ANSI, FIPS program, and DOD standards presents a cursory profile of the presently existing standards in software engineering. The present emphasis in software standardization is on programming languages, documentation, and to some degree, working vocabulary. By far, the weight of work impacts hardware. Only the DOD standards concern the prefabrication, test and evaluation, quality assurance, and configuration management areas of the software development cycle. Since the individual services are closest to one subset of the overall scope of Federal and DOD standardization, they can deal more effectively with specific elements in the software development cycle. Hence, it is at this level that definition in the prefabrication phases of the software development life cycle is first seen.

In general, the present profile of software engineering standards is skewed toward definition of common programming language elements and form in documentation. The marked void in definition of standard practice in problem definition, user requirements specification, design specification, test and evaluation, and quality assurance probably indicates the present lack of commonality in these areas. They are, therefore, the key facets of software engineering which presently merit standards study.

5 SOFTWARE DEVELOPMENT AS AN ENGINEERING DISCIPLINE: ITS PRESENT MATURITY AND FUTURE DEVELOPMENT

Software Development's Maturity as an Engineering Discipline

One of the first elements demonstrating maturity in the engineering disciplines is the historic base upon which they rest. In the length of time mechanical engineering has had to assimilate experience, empirical observation, and cognitive hindsight, it has built a firm foundation of principles found to be trustworthy through years of testing, evaluation, and reevaluation. By contrast, software engineering has had dynamic, but short and somewhat disjointed history. Demand for its product has established its place among the engineering disciplines, but has simultaneously eliminated the time necessary for its experience to evolve into principle.

Part of the reason for software engineering's inability to establish an accepted base is the dynamic evolution still taking place in the field. The evolutionary cycles of other engineering disciplines have peaked and stablized. Attaining this stability has allowed them to concentrate on simplification and flexibility. Software engineering, on the other hand, is still being shaped. As a result, software technology is still growing in complexity. It is still in the active, experimental stages in which it is forming the elements of principle and practice on which its mature character will be built.

Attaining an evolutionary plateau of methodological standardization has also enabled the engineering disciplines to establish continuity and commonality in their development. Definition of critical interfaces within and across

the engineering disciplines has been a key in facilitating fluid communication and speeding the process of consensus critical to establishing workable standards. Software engineering has yet to adequately identify its critical interfaces. Communication across its phases has historically been fragmented, resulting in fragmented development.

As stated in Chapter 3, a discipline's maturity is also indicated by its level of consensus as embodied in the standards currently available. Programming languages and documentation appear to be areas in software engineering in which the greatest degree of consensus or at least the greatest commonality has been achieved. Little investigation into standardizing the prefabrication phases of software development has been done--indeed, as previously stated, it is not yet clear or agreed upon just what these phases entail. Few standards exist in the requirements definition, design process, and problem definition direction phases, indicating the present lack of common practice in these areas. The national and international organizations charged with standards definition are still investigating standardization in the information-processing sciences. Based on the criteria of development of existing standards, software engineering is immature in comparison with other engineering disciplines.

This lack of standards is related and in fact is due in part to software engineering's short historic base. The paucity of historical data has led to hesitation in formulating standards because standards defined and executed without adequate empirical demonstration could jeopardize future efforts toward standardization by imposing rigidity in a rapidly evolving discipline.

Importance of Maturity in Software Engineering

Attaining a definable level of maturity in any engineering discipline helps to establish a frame of reference by which to evaluate new and potentially discordant applications. A given level of maturity tends to raise the initiation of creativity and research in the discipline. Repeated lower level efforts can be standardized to avoid the syndrome of "reinventing the wheel."

An established and defined level of maturity in software engineering would provide an intermediate evolutionary plateau from which development could spring, as well as a standard for evaluation. This comparability and continuity would allow measurement of the relative value of other processes, building up empirical data from which new standards could be developed. Such standards of evaluation are critical to establishing procedures and acceptance criteria and insuring progress.

Maturity in software engineering is essential to establishing an operating vocabulary, thus facilitating effective communication and assuring an accurate response to user's needs. In software engineering, this means objective definition of desirable qualities such as reliability, transportability, and maintainability.

Finally, maturity is needed to control the discipline's growth. Without controlled growth in the modern technological environment, many of the potential benefits that software affords will be lost or delayed.

Projected Continuity and Change in Software Engineering

In the immediate future of software development as an engineering discipline, the following trends can be expected

to continue: structured approaches to problems, the impact of popularity, the duality in the business versus the scientific facets of the field, and the lack of adequate and accepted definitions of software qualities.

There is an ever-increasing move toward constrained structure in management, design, and documentation, as well as in programming itself. This tendency toward constraint is due not only to the structured programming techniques initiated by Djikstra's[12] initial publication, but also to the need for organization in the fact of increasingly large and more complex systems.

Popular concepts will continue to significantly impact software engineering both in the academic area and as a marketing tool. Marketing expertise currently outweighs technical expertise. Funding will continue to be the driving force behind software research and development. The government will continue to be a major funding source for software engineering work as the demands of national defense, resource conservation, and the domestic economy grow. Because of its increasing involvement in software, the government's demands and directives for standardization will begin to have a more significant impact on the direction of the industry.

Computer science in the academic environment will continue to develop into a specialized major field. As it does, it will supply the empirical evidence necessary to validate or disprove the popular movements. The academic world will continue to attempt to formalize software engineering into scientific principles and in so doing may initiate and identify key areas for standardization. As hardware technology approaches its theoretical limits

---

[12] E. W. Djikstra, "Notes on Structured Programming," in *Structured Programming* by Dahl, O. J.; Djikstra, E. W.; and Hoare, C. A. R. (Academic Press, 1972).

in size, speed, and material, increased emphasis will be given to achieving parity between software engineering technology and hardware.

Perhaps the most challenging area in software engineering is that of human factors. The human element in the software development process will become increasingly critical as automation spreads in both hardware and software development. How the human in the human-machine interface effectively functions, what factors contribute to his success or failure, and what motivating factors are operative in his performance will be key subjects for psychological investigation. This research will identify problems in software design, user requirements specification, problem definition, and the human cognitive process that historically have received little attention.

As software engineering matures as an engineering discipline and the software development process is more adequately understood and formalized, a move toward specialization will occur in both the academic world and the working environment. Specialization in software development will produce "building block" elements fundamental to the assembly-line development approach. The Japanese have recently invested heavily in this idea.[13] Standardization in the software engineering industry may lead to the kind of tool and component building methology used in the construction industry.

The duality in the business versus scientific applications fields in software engineering will continue to develop independently as software engineering matures. Yet, as development and evaluation methodologies are formalized and confirmed, bridges may be built between these two

---

[13] "Miti-Directed Software Cooperation," *Datamation* (September 1976) p 97.

application areas. Appropriately defined standards will be universally applicable and will begin to dissolve the business/science duality.

As the complexity of software systems grows with the industry, adequate and clear terminology must be developed to insure accurate communication. Qualities definition will provide the foundation for defining performance and quality evaluation methodology and criteria. Communication will become increasingly critical as software engineering matures to the level of the other engineering disciplines and receives more extensive application in the engineering environment.

Future Role of Standardization

The great motivation for standardization is economy. Software costs are increasing dramatically as software development becomes more and more the critical element in system construction. Therefore, if software standards are to have a significant life span of application, they must satisfy the economic requirement of decreasing cost.

Faults implanted in the requirements specification and design phases have historically caused expensive development phase schedule slippage and follow-on maintenance requirements. Consequently, standards in the critical prefabrication phases of software development and a validation methodology for quality assurance will be key future needs.

Psychological research into the human factors mechanisms that determine success in execution of the prefabrication phases will also be critical. Standardization in human engineering will provide transportability of human experience, a time- and money-saving resource.

There is a continuing need for definition in software engineering terminology, both to avoid the ambiguity that now exists and to facilitate effective communication throughout the discipline's continuing evolution. Formalization of the software development process is also needed. Such formalization will be a milestone in software engineering standardization.

Controlled experimentation to empirically justify and document new conceptual approaches to the software development cycle can be directed by appropriate standardized methodologies that define criteria for acceptance or refusal. An integrated methodology for applying the fragmented tools and techniques that already exist or will be developed is a key need that standardization could impact.

In general, then, standardization has the potential of giving software engineering a universal, general management overlay which would provide an integrated structure for the phases and functions generic to the software development process.

6 IDENTIFICATION OF AREAS REQUIRING STANDARDS

As the discussion in Chapter 5 indicated, a number of general areas in software engineering lack standardization. To determine specific areas requiring standardization, the concept of unit inputs to the development cycle was expanded.

Unit Inputs to the Development Phases

Unit inputs are a subset of the unit commodities of any activity. The unit commodities of development activities are:

1. Instructions--the application-peculiar directions, requirements, or design statements input to a subsequent phase to drive its execution toward the desired products.

2. Procedures--organized sequences of activities pursued to result in the eventual creation of the desired products from other unit inputs.

3. Designations--assigned measures, values, or formats appropriate to the engineering discipline(s) being exercised to produce a product. Designations include such things as linear and volumetric measures, constants, documentation formats, and measurable qualities.

4. Ingredients and Facilities--Ingredients are the raw materials from which products are made, including obvious things such as sheets of paper and the applied graphite or ink, or less obvious things such as the space in which the product will fit, and the energy needed to drive facilities. Facilities are the equipment and environment required by the execution of the procedures. Facilities include items from the pencil containing the graphite, through the computer used to test the software, to the building housing other facilities and personnel.

5. Personnel--the people, by name or category, who perform manual parts of the procedures. Personnel may be seen as providing levels of skill, experience, and ingenuity against certain normalized expectations.

6. Products--the end items to which all preceding six types contribute, including not only the inputs to the next phase, but also feedback to preceding phases and information through reporting channels. Products are the only commodities which are not unit inputs to their own phase.

Representation of Unit Inputs by Standards

The following subsections examine the first five commodities in the context of each phase of software development to determine how they are represented by standards.

The Direction Phase

Instructions to the direction phase are the least defined commodities in the entire development cycle. Such instructions are the raw stimuli which emanate from and are the basis of "need." If they did not exist, there would be no incentive to define the environment's shortcomings and proceed to fill them. Unit incentives are usually quite difficult to define and of extremely diverse origin. However, if they are not explicitly recognized and accommodated in the statement of the problem, they are not likely to be accommodated by the end product, except by accident. Like so many other engineering disciplines, software engineering does not now explicitly define the incentives for development.

The term "procedures" cannot be appropriately used in the software engineering direction phase. There are no rigorous, recognized, consistently applied, or explicitly

taught procedures for defining direction for software development.

Designations usually evolve to support and reduce repetition of procedures. Since standard direction procedures are in effect non-existent, there are few designations peculiar to this phase. There are a few existing "problem statement formats" in the most advanced requirements specification methodologies, but they do not begin to support definition of direction--they only provide a format for instruction inputs to the requirements phase. Other designations, peculiar to an application's engineering context, or to software engineering itself, may be found to support needs for unit inputs to this phase. The most important omission of designations, however, is the almost total lack of dimensional definition and measurement needed to scope the voids--the defined basis of any problem.

Ingredients are not a problem in this phase, since it only requires that materials be supplied to support the documentation of definitions of problems. In the same sense, facilities are of little question. However, if experimental work is required to support definition of a problem, both ingredients and facilities can become major concerns.

Although personnel with skill and experience are available, their experience and skill are poorly defined, making them difficult to recognize, recruit, and assign.

In summary, essentially no standards exist for unit inputs peculiar to the direction phase. Since a solution cannot be expected to be any better than the statement of its problem, standards for this phase are crucial to the software development process.

The Requirements Phase

Instructions to this phase take the form of a defined problem--a void to be functionally filled. As noted in preceding chapters, a set of functional requirements is quite often erroneously specified as the problem. This clearly shows the lack of standards for content of problem statements as inputs to the requirements phase. Several companies and agencies (notably the Ballistic Missile Defense Advanced Technology Center) are currently working toward definition of standards, but the approach is an advanced-methodology-oriented one which is clearly not directed at immediate application or community consensus.

Considerable work has been done on procedures in this phase, including hierarchical requirements decomposition techniques and tools. Because of the extreme complexity of software missions and implementations, this area has recently received much attention and will continue to do so.

While some primitive standards exist for documentation of requirements, satisfactory designations, which traditionally trail procedures, are not readily available yet. However, designations follow directly upon evolution of procedures and will emerge as consensus is drawn around the newly evolved requirements-generation procedures.

Ingredients and facilities are again of minimal concern during this phase under most development circumstances. However, as procedures include requirements simulation for specification support, needs escalate rapidly and become of significant financial concern.

Personnel requirements are probably the most difficult problem to attack. New requirements-generation procedures

necessitate retraining and changing the attitudes of software engineers. Present software engineer certification programs do not address this problem. The various personnel specialties in software engineering must be recognized before the development cycle will be fully supported by available personnel resources.

The requirements phase, while not as poorly supported as the direction phase, has some major weaknesses in standards, the most serious of which is in problem definition content standards. Standards for defining requirements in forms which lead to clear measurement of the success or failure of the final product are lacking. While work is advanced on practices, that work is not being taken advantage of by training practitioners, even though standards imposition through education has been demonstrated in the past to be most effective.

The Design Phase

Instructions for design take the form of functional requirements specification. While many practitioners may call their specifications "requirements," few actually meet that goal. The reason for this is quite often that the only standard for a specification deals with its format--under designations--rather than content. Thus, the specification becomes a blend of statement of direction, requirements, and design. Development of base standards for content of requirements, as discussed above, is therefore crucial to the design phase and the remainder of the software development process.

Procedures for software design have recently received attention through the concept of "structured design." Although this concept may result in some improvements in procedures, it must be viewed in light of practitioners' adherence to the outward characteristics of procedures, without basic understanding of the driving principles or

intentions. Procedures must be carefully designed to implement their intentions despite lack of complete understanding by the practitioner. Therefore, structured design and other popular theories have a long way to evolve before they can be considered standard procedures.

Designations start to assume real consequence during the design phase. While the bit has been defined, failure with the byte signals that problems exist in the state of designations input to software design. While designers can now call for American Standard Code for Information Interchange (ASCII) codes and "standard" languages, they have little to match the modularity represented by buttons and thread, resistors and capacitors, test tubes and slides, and the myriad of other standardized items of other disciplines. Designations in a few areas (notably languages) exist, but work on the many parallel problem areas has not yet begun.

Ingredients are hard to define when designations for them do not exist. Little of modular form will exist until more work is done to support the designation definition. Facilities also cannot support what does not have form. Paper and pencil, desk, and chair may be supplied, but functional modules and automated design are far from standardized.

Personnel redeem the design phase today. The good software designer has had to overcome the lack of instructions and designations, determine how to do the job (each time to meet the total lack of comparable inputs), and provide the programmer with a specification that could be translated into executable code. Quite often the programmer must write a new code to avoid more immediate difficulties. The designer must be provided with working materials which can be depended upon to consistently meet adequate standards.

The design phase is faced with irregular instructions, few reliable designations, and the responsibility to compensate for the deficiencies of preceding phases. This in part reflects the impact of incorporeity (as discussed in Chapter 4).

The Fabrication Phase

Instructions for the fabrication phase take the form of design statements which define explicit sequences and parallel performance of state changes. These state changes as a whole are intended to implement the required control and computational functions when executed in a prescribed computer environment. The main difficulty with instructions is determining at what level design stops and programming (fabrication) begins. This problem is aggravated by the inconsistency of instructional-level functional complexity within and between software languages. The problem will not be alleviated to any degree until languages are standardized across machine boundaries at consistent functional levels. Beyond this problem, the content of design is generally transmitted from designer to programmer. When languages have matured, adequate empirical data on instruction methology should exist to allow standards organizations to adopt a suitable methodology.

Programming procedures have been subjected to a number of popular concepts. Currently, for example, structured programming is being retrofitted into circumstances and languages which sometimes cannot effectively accommodate the procedures. Unfortunately, no empirical data have been collected from the exercising of procedures. Until data can be collected and analyzed in real, working programming circumstances, little more than accidental progress will be made.

Designations for programming including coding formats, data formats, interface rules, and the numerous other details which assure that the program will reach execution time and work as intended when there. Unfortunately, except for the bit, there is little one can depend on to be the same between operational (and consequently programming) environments. Progress is being made on languages and interfaces, but is sorely lacking in almost all other areas. The lack of common designations makes it impossible to compare data on the performance of programming procedures, and thus takes priority over the empirical data problem; at least an interim solution is required.

Ingredients and facilities in this phase include items peculiar to data processing, such as computer cards and their associated machinery, punched tape, magnetic media, terminals, compilers, assemblers, etc. While many physical media have been standardized (cards, tape, etc.), few software facilities have been. Operating systems and their supporting code production facilities, while conforming in gross ways to gross functional conventions, are still very unique and peculiar to their operating environments. Lack of standard facilities and their implications on standardization of languages and interfaces will continue to impede the evolution of software engineering until some standardization is achieved.

Personnel are the energy expended in the fabrication phase. While there have been predictions that the programmer will soon be replaced by automatons which translate design directly into code, this is not likely to occur immediately. Personnel must now be accredited not only as programmers, but specifically for machines, operating systems, and languages. Their thought processes are

adjusted to meet machine specifications, not vice versa. Standardization of facilities, languages, and interfaces must progress in concert with human factors engineering of the roles to be performed by man in this and the other phases.

The fabrication phase suffers from a diversity of facilities and procedures, widely dispersed and massively applied; performance data are not, however, returned for use as a basis for adoptive standardization. This information void must be filled if programming is to improve in a controlled and rational manner.

The Verification Phase

Instructions to the verification phase are all of the incentives, directions, requirements, design, and code of the preceding four phases. Inputs make verification of software extremely difficult, if not impossible.

Procedures are readily available. However, since everyone is working against a peculiar set of instructions and designations, procedures for verification do not travel well. The basic concept of auditing each level of instructions against itself and its predecessors is well understood, but its practice is improbable because the incentives are never defined.

Designations in effect do not exist for this phase. Qualities are not defined in measurable terms. Formats do not exist for tests or their plans, and there are no unit measures of performance.

Ingredients and facilities are the only items generally available; however, there is little which can be identified as standard enough to support comparison of results on this basis. The little that exists has been the basis for a

branch of testing called "benchmarking," a very coarse means of comparing performance in a given environment.

Personnel are again the commodity which partially redeems this phase. By constantly improvising testing, practitioners have imparted some small degree of confidence to software. However, this improvisational skill is not engineering; it is both rare and self-taught.

In addition to inheriting the ills of preceding phases, verification has its own faults. Testing and verification are implemented using poorly defined qualities; statistics are misused to allege a degree of confidence which is rarely if ever attained. The practicing software community's response to date has been to rationalize an approach to verification. A firm look at basic principles and procedures is needed to determine the scope of the voids that exist in verification.

The Dispersion Phase

Instructions to the dispersion phase include the product code, its documentation and verification results, and requirements for user documentation. This phase suffers problems caused by poor instructions from preceding phases; almost no standards exist for its instructions.

Procedures exist in many organizations to implement the delivery of software. However, few procedures pertaining to the preparation of user documentation exist or are standardized, reflecting the poor representation of human engineering in software development, as well as the almost total lack of user-involvement in the software development cycle. This lack of communication and involvement should be a target for work in establishing procedural standards for this phase and related activities in other phases.

Only some cursory designations exist for the identification of software products and their related documentation, even though sufficient experience exists from which to adopt standards. Designations of man-machine relationships are very poorly defined, and measures and units have yet to be conventionalized.

Ingredients and facilities commonly involve normal clerical supplies, although some interactive, computer-supported tools are emerging to support dispersion and user documentation.

Personnel with the cross-training and experience in software engineering and human factors of configuration management are not common, and are consequently difficult to find and fit into a development scheme. The dispersion phase usually depends on how the individuals involved deal with the situation; some controlling standards are therefore urgently needed.

The dispersion phase is often a point of confrontation between the developer and the user. Standards might reduce this conflict by providing control and limiting specifications.

Summary

Table 1 summarizes the specific areas in the software development cycle which require standardization.

Table 1

Areas Needing Study and Standardization

1.	Methods to define initial incentives for software development efforts (page 43, paragraph 4).
2.	Procedures for the definition and scoping of problems (page 44, paragraph 2).
3.	Standards for the content of problem statements (page 45, paragraph 1).
4.	Training and certification of software requirements engineers (page 45, last paragraph).
5.	Standards for the content of requirements (page 46, paragraph 3).
6.	Definitions of units to measure software performance and size (page 46, paragraph 2, see also page 27, paragraph 2).
7.	Designations for inputs to software design (page 47, paragraph 2).
8.	Collection of empirical data on development procedures' performance and effectiveness (page 48, last paragraph).
9.	Standards for operating systems and their support facilities (page 49, paragraph 2).
10.	Human engineering of the roles and functions of man in software engineering (page 49, paragraph 3).
11.	Definition of explicit inputs required for verification, to be imposed as requirements on preceding phases (page 51, paragraph 3).
12.	Procedures for designation of software configuration items (page 52, paragraph 1).
13.	Procedures for communication and translation between users, procurers, and software engineers (page 51, last paragraph).

7 STANDARDS DEVELOPMENT PLAN

This chapter presents a plan for development of standards for the areas identified in Table 1. Finding reasonable solutions for all these problem areas will require many years of work by many people. Consequently, the first step in the standards development plan is a priority ranking of the areas. Certain areas presume at least initial work in others. However, different segments of the software community may see some of these initial work areas as more important than others. Since standards are based upon consensus, the ranking of the areas must be performed so as to maximize consensus.

One way to do this is to survey the software community--or at a minimum, persons with interests in standards--to obtain a ranking of the expected value of the listed needs. Such a ranking would provide a firm basis for selection and pursuit of new standardization efforts by the several voluntary and government organizations and possibly industry and academic institutions. Such a survey, if cosponsored by ANSI, would reach and sample a broad spectrum of interests, which could be factored to identify correlations with standards needs. The tabulated results of the survey should be reported to the major standardization organizations and in the software engineering literature.

For the standards organizations to use such a tasking consensus effectively, the present procedures used to formulate draft standards must be revised. As this study has shown, the principles of division of the development cycle into six phases, and the marshaling and definition of unit inputs can be applied to standards engineering. Properly defining the problem to be solved, the requirements upon that solution, and the likely means for verification before

designing the standards will lead to the most effective use of the tasking consensus results.

The previously described survey must also be designed to produce a clear statement of the incentives for standardization of software engineering practices. Boards of the rank of ANSI X3 or FIPSCAC must convene panels to explicitly define the problems of each selected incentive area. These panels must report their findings so that technical committees can start defining requirements.

This 3- to 5-year effort represents only the beginning of the search for solutions to the software engineering problems.

At this point, work must be switched to an organization whose structure and operations will enhance the possibility of creating a standard--an "Underwriters' Laboratory" for software engineering, working through subscribers from the software users community, could evolve standards for measuring performance and assurance of software which would thus become attractive to the development community. Operating systems could be a natural target of such an organization.

Parallel with this effort must be an effort in the academic community to promote understanding of the role of standards. Emphasis should be shifted from languages and operating systems to include the full development cycle, including the direction, requirements, verification, and dispersion phases as well as maintenance. Such emphasis would allow the practitioner to understand and implement the standards which will be developed. Improvements in educational curricula can be made rapidly, with demonstrable effects on the profession obtainable in 4 years.

The government could promote such changes through the establishment of a "National Institute for Information Processing," with the expressed mission of promoting and maturing today's computer science curricula.

Thus, development of the standards crucial to software engineering's maturity will require cooperation at the highest policy levels of the Federal government, standards organizations, academic institutions, and industry.

8 CONCLUSIONS

This study of the evolution of software engineering indicated that although software engineering is actually progressing very rapidly compared to the rate at which other engineering disciplines progressed from their origins, this rapid progress has produced stresses that other disciplines have not had to deal with during their evolutions. Demand for software engineering's product has established its security as an engineering discipline while eliminating the time necessary for experience to evolve into principle (p. 39).

Analysis indicated that most procedures and inputs to software engineering are not now standardized (p. 40). This lack of an established base of principle and standards indicates that software engineering is relatively immature as an engineering discipline; it is still in the active, experimental stages of its evolution in which it is forming the elements of principle and practice on which its mature character will be built.

Development of standards is crucial to the maturing of software engineering. Standardization has the potential of giving software engineering a universal, general management overlay which would provide an integrated structure for the phases and functions generic to the software development process (p. 45). Existing standards are skewed toward definition of common programming language elements and form in documentation. The key areas requiring standard practices are problem definition, user requirements specification, design specification, testing and evaluation, and quality assurance (p. 45). Table 1 lists the specific areas identified as needing standards in an analysis of the unit inputs to each phase of the software development

cycle (Chapter 6). Because software engineering is still in a dynamic stage of development, standards must be sensitive to its developing needs to avoid imposing excessive rigidity and therefore impeding acceptance of standards (p. 40).

Many of the current trends in software engineering can be expected to continue in the future: structured approaches to problems, the impact of popularity, the duality in the business versus the scientific facets of the field, and the lack of definitions of software qualities. The importance of quality and the effort to define it will be one of the most significant forces shaping software engineering in the future. As software becomes more complex, the need for qualities definitions to serve as the foundation for defining performance and quality evaluation methodology and criteria will become critical (pp. 42-44).

As major sponsors of software development, the Army and Department of Defense have a vested interest in standards leading to quality software products (p. 37, Appendix A).

REFERENCES

E. W. Djikstra, "Notes on Structured Programming," in *Structured Programming* by Kahl, O. J., Djikstra, E. W. and Hoarc, C. A. R. (Academic Press 1972).

Federal Information Processing Standards Index, FIPS Pub 12-2 (U.S. Department of Commerce, National Bureau of Standards, 1 December 1974) pp 76-79.

Gildersleeve, Thomas R., "Insight and Creativity," *Datamation* (July 1976) p 91.

Hill, Marjorie F., *The World of EDP Standards*, Tech. Memo TM4 (Control Data Corporation, January 1973) pp iv-12 to iv-25.

"Miti-Directed Software Cooperation," *Datamation* (September 1976) p 97.

Modern Dictionary of Electronics, 4th Ed. (Howard Sams & Co., 1972).

Sanders, T.R.B., *The Aims and Principles of Standardization* (International Organization for Standardization, October 1972) pp 3, 17.

Wells, Mark B., *Evolution of Computer Software* (Los Alamos Scientific Laboratory of the University of California, February 1971).

January 1977

APPENDIX A

SOFTWARE STANDARDS INVENTORY: ORGANIZATIONS

This appendix lists organizations operating or represented in the United States, which are concerned with the development and codification of software standards. The second column lists officers and representatives to other organizations. Abbreviations used in the listing are defined on pp. A16 - A17.

Name & Address	Officers and Representatives
Acoustical Society of America American Institute of Physics 335 East 45th St. New York, NY 10017 Tel: 212/685-1940	
Air Transport Association 1709 New York Ave., NW Washington, DC 20036 Tel: 202/872-4296	X3 Rep: Frank C. White
American Bankers Association 1120 Connecticut Ave., NW Washington, DC 20036 Tel: 202/467-4296	X3 O: James T. Booth
American Gas Association 1515 Wilson Blvd. Arlington, VA 22209 Tel: 703/524-2000	X3 O: Robert J. Brunner
American Library Association 50 East Huron St. Chicago, IL 60611 Tel: 312/944-6780	X3 Rep: James R. Rizzolo
American National Standards Institute 1430 Broadway New York, NY 10018 Tel: 212/868-1220	X3 O: Marie Hogsett

ANSI X3 Committee (Computers and Information Processing) c/o CBEMA 1828 L St., NW Washington, DC 20036 Tel: 202/466-2288	Ch: John F. Auwaerter
ANSI X3 International Advisory Committee (IAC)	Ch: Thomas J. McNamara
ANSI X3 Standards Planning and Requirements Committee (SPARC)	Ch: William G. Madison
ANSI X3J1 (PL/I)	Ch: Lois C. Frampton
ANSI X3J2 (BASIC)	Ch: Thomas E. Kurtz
ANSI X3J3 (FORTRAN)	Ch: Frank Engel, Jr.
ANSI X3J4 (COBOL Standards)	Ch: Jitze Couperus
ANSI X3J41 (COBOL Audit Routines)	
ANSI X3J5 (COMPACT II/ACTION/SPLIT)	Ch: Robert F. Guise, Jr.
ANSI X3J7 (APT)	Ch: Elliot J. Brebner
ANSI X3J8 (ARGOL)	Ch: Marjorie L. Green
ANSI X3K1 (Project Documentation)	Ch: James Ridgell
ANSI X3K2 (Flow Charts)	Ch: David Mace
ANSI X3K5 (Terminology and Glossary)	Ch: Martin H. Welk, Jr.
ANSI X3K6 (Network-Oriented Project Management)	Ch: Kenneth A. Frey
ANSI X3K7 (Program Abstracts)	Ch: Margaret K. Butler
ANSI X3L2 (Character Sets & Codes)	Ch: Charles D. Card
ANSI X3L5 (Labels & File Structure)	Ch: Jean G. Smith
ANSI X3L8 (Representation of Data Elements)	Ch: Harry S. White, Jr.

ANSI X3L81 (Data Standardization Criteria)	
ANSI X3/SPARC COMPACT II/ACTION/ SPLIT (Comp) Study Group	Ch: Robert F. Guise, Jr.
ANSI X3L82 (Time Designations)	Ch: James W. Gillespie
ANSI X3L83 (Individual and Business Identifications)	Ch: Shiela Smythe
ANSI X3L84 (Geographic Units)	Ch: Walter L. Schlenker
ANSI X3L86 (Quantitative Expressions)	Ch: Durane J. Marquis
ANSI X3S3 (Data Communications)	Ch: Gerald C. Schutz
ANSI X3S31 (Communications Standards Planning)	Ch: Gerald C. Schutz
ANSI X3S32 (Data Communications Vocabulary)	Ch: George W. White
ANSI X3S33 (Data Communications Formats)	Ch: William F. Emmons
ANSI X3S34 (Data Communications Control Procedures)	Ch: David E. Carlson
ANSI X3S35 (System Performance)	Ch: G. J. McAllister
ANSI X3S36 (Digital Data Signalling Rates)	Ch: Harold J. Crowley
ANSI X3/SPARC COMPACT II/ACTION/ SPLIT (Comp) Study Group	Ch: Robert F. Guise, Jr.
ANSI X3/SPARC Data Base Management Systems (DBMS) Study Group	Ch: Thomas B. Steel, Jr.
ANSI X3/SPARC Long Range Planning for Programming Language Standards (LRPL) Study Group	Ch: Maurice Halstead
ANSI X3/SPARC Operating Systems Control Language (OSCL) Study Group	Ch: Edgard H. Sibley

ANSI X3/SPARC Programming
Language for Text Processing
(PIPT) Study Group

ANSI X3T9 (I/O Interface
Standards) — Ch: Delbert L. Shoemaker

ANSI X4 Committee
(Office Machines)
c/o CBEMA
1828 L St., NW
Washington, DC 20036
Tel: 202/466-2288

ANSI Z39 Committee — Ch: Jerold Orne
(Library Work, Documentation and
Related Publishing Practices)

American Newspaper Publishers
Association — X3 O: William D. Rinehart
11600 Sunrise Valley Dr.
Reston, VA 22070
Tel: 703/620-9500

American Nuclear Society — X3 Rep: Mel Couchman
244 East Ogden Ave.
Hinsdale, IL 60521
Tel: 312/325-1991

American Society for Testing
and Materials
1916 Race St.
Philadelphia, PA 19103
Tel: 215/569-4290

American Society for Information
Sciences — X3 Rep: Michael C. Kepplinger
1155 16th St., NW
Washington, DC 20036

American Society of Agricultural
Engineers — X3 O: Kenneth A. Jordan
2950 Niles Rd.
St. Joseph, MI 49085
Tel: 616/983-6521

Association for Computing Machinery
1133 Avenue of the Americas
New York, NY 10036
Tel: 212/263-6300

X3 Rep: Pat Skelly

Association for Computing Machinery
Joint Users Group (JUG)

Ch: Richard J. McQuillan
X3 Rep: Theodore E. Wiese

Association for Data Processing
Service Organizations
210 Summit Ave.
Montvale, NJ 07645
Tel: 201/391-0870

X3 Rep: John B. Christiansen
(ADAPSO CMTE on
Industry Standards)

Association for Educational Data
Systems
1201 16th St., NW
Washington, DC 20036

X3 Rep: A. Kenneth Swanson

Association for Library Automation
California State University &
Colleges
5670 Wilshire Blvd.
(Chancellors Office)
Los Angeles, CA 90036
Tel: 213/938-2981 x 412

Association for Systems Management
24587 Bagley Rd.
Cleveland, OH 44138
Tel: 216/643-6900

X3 O: Richard Irwin
Executive Director

Association Francaise de
Normalisation (AFNOR)
Tour Europe
Cedex 7
92 Paris-La Defense

Association of American Railroads
1920 L St., NW
Washington, DC 20036
Tel: 202/293-4000

X3 Rep: R. A. Petrash
Executive Director

Association of Computer
Programmers and Analysts
P.O. Box 2349
Chicago, IL 60690

X3 Rep: Lawrence A. Ruh
Ch: Martin A. Morris, Jr.

Association of Time Sharing Users 75 Manhattan Dr., Ste 204 Boulder, CO 80303	President: Hillel Segal X3 Rep: William G. Madison
British Standards Institute 2 Park St. London, WIA 2BS	
Canadian Government Specifications Board 88 Metcalfe St. Ottawa, Canada	
Canadian Standards Association Executive Office Suite 2100, Tower A Place de Ville Ottawa, 4, Ontario	
Computer and Business Equipment Manufacturers Association (CBEMA) 1828 L St., NW Washington, DC 20036 Tel: 202/466-2288	
Computer and Communications Industry Association 1911 Ft. Meyer Dr. Suite 801 Arlington, VA 22209 Tel: 703/524-1360	Executive Director: A. G. W. Biddle X3 Rep: Normal J. Ream
Conference on Data Systems Languages (CODASYL) P.O. Box 124 Monroeville, PA 15146	X3 O: William B. Rinehuls
CODASYL Data Base Language Task Group	Ch: M. O'Connell
CODASYL Data Description Language Committee	Ch: Earl Broadwin
CODASYL Decision Table Task Group	Ch: Paul Jorgensen
CODASYL End User Facilities Committee	Ch: Henry C. Lefkovitz

CODASYL Executive Committee	Ch: John L. Jones
CODASYL File Processing Task Group	Ch: Donald G. McCrimmon
CODASYL FORTRAN Data Manipulation Language Committee	Ch: Chester M. Smith
CODASYL Operating Systems Control Control Language Task Group	Ch: Hasan Sayani
CODASYL Proposal Editing	Ch: Richard E. Blasius
CODASYL Stored Data Definition and Translation Task Group (CODASYL/SDDT)	Ch: Robert Taylor
CODASYL Systems Committee	Ch: William H. Stieger
Data Processing Management Association DPMA International Hq 505 Busse Hwy Park Ridge, IL 60068 Tel: 312/825-8124	X3 Rep: Ardyn E. Dubnow
Department of Defense (DOD) Washington, DC 20301	X3 Rep: Wharton L. McGrier
DOD ADP Policy Committee Assistant Sceretary of Defense (Comptroller) Washington, DC 20301	
DOD Standardization Area Computer Aided Design and Numerical Control Naval Ship Engineering Center Carderock, MD	
DOD Standardization Program for Information Processing Standards for Computers (IPSC) Directorate of Data Automation (AF/KRAX) Headquarters, USAF Washington, DC 20330	Contact: William Rinehuls

DOD Working Group on Computer-Generated Military Symbology (DOD/Display)
Ch: MAJ Joseph Box

DOD Working Group on DOD Computer Documentation Standards (DOD/DOCN)
Ch: Donald Wagus

Department of Health, Education, and Welfare
Independence Ave., SW
Washington, DC 20201
Tel: 202/245-6541
Ch: Wallace R. McPherson, Jr.

Digital Equipment Computer Users Society (DECUS)
Digital Equipment Corp.
Maynard, MA 01754
X3 Rep: Patricia Caroom

Edison Electric Institute
90 Park Ave.
New York, NY 10016
Tel: 212/986-4100
X3 Rep: S. P. Shrivastava

Electrical Testing Laboratories, Inc.
2 East Ave.
New York, NY 10021

Electronic Industries Association
2001 Eye St., NW
Washington, DC 20006
Tel: 202/457-4900

European Computer Manufacturers Association
Rue du Rhone 114
1204 Geneva, Switzerland
X3 O: Dara Hekimi
Secretary General

Federal COBOL Interpretations Committee (FCIC)
Ch: Mabel V. Vickers

Federal Information Processing Standards Coordinating and Advisory Committee (FIPSCAC)
National Bureau of Standards
Washington, DC 20234
Tel: 301/921-1000
Ch: Harry S. White, Jr.

FIPS Task Group 1 (Objectives and Requirements for Standards)		
FIPS Task Group 4 (Subsections on Standards for Use in Requests for Proposals)		
FIPS Task Group 5 (Federal Information Processing Vocabulary)	Ch:	Josephine L. Walkowicz
FIPS Task Group 8 (Guidelines for Describing Data Interchange Formats)		
FIPS Task Group 9 (COBOL Standards)	Ch:	G. Stanley Doore
FIPS Task Group 13 (Workload Definition Benchmarking)	Ch:	D. W. Lambert
FIPS Task Group 14 (Documentation for Information Processing Systems	Ch:	James W. Gillespie
FIPS Task Group 16 (Basic Standard Programming Language)	Ch:	Trotter Hardy
FIPS Task Group 17 (Data Element Directories)	Ch:	Tim Bergen
FIPS Task Group 19 (Automatically Programmed Tool Group)	Ch:	John M. Evans
FIPS Task Group 20 (User-terminal Protocols)	Ch:	Albrecht Newman
FIPS Task Group 21 (HDP Systems Interface Standards)	Ch:	W. Bruce Ramsey
Federal Telecommunications Program Standards Committee (FTPSC) National Communications System NCS-TS Washington, DC 20305 Tel: 202/692-2131	Ch:	Dennis Bodson

Federal Telecommunications Standards Committee (FTSC) National Communications System NCS-TS Washington, DC 20305 Tel: 202/692-2124	Ch: Marshall Cain
Federation of NCR Users	X3 O: Richard H. Vandenburg
Forum of Control Data Users (FOCUS) International	
FOCUS Standards Committee	X3 O: William B. Stelwagon
General Services Administration Washington, DC 20405	X3 Rep: Delbert L. Shoemaker
Guide International	X3 Rep: Theodore E. Wiese
IFIP Administrative Data Processing Group 6, Stadhouderskade Amsterdam, 1013, Holland	
IFIP Administrative Secretary 32, Rue de L'Athenee 1206 Geneva, Switzerland	
Institute of Electrical and Electronic Engineers (IEEE) 345 East 47th St. New York, NY 10017	X3 Rep: R. L. Curtis

Insurance Accounting & Statistical Association
406 West 34th St.
Kansas City, MO 64111
Tel: 816/756-3443

X3 O: Louis E. Schoemer

Interagency Committee on ADP (IAC/ADP)

Ch: Israel Feldman

International Electro-Technical Commission (IEC)
Centraol Office, 1, Rue de Varembe
1211 Geneva 20, Switzerland

International Organization for Standards (ISO)
Central Secretariat
1, Rue De Varembe, 1211
Geneva, Switzerland

ISO/IC97 Subcommittee 1
(Vocabulary)

ISO/IC97 Subcommittee 2
(Character Sets and Coding)

ISO/IC97 Subcommittee 5
(Programming Languages)

ISO/IC97 Subcommittee 6
(Data Communications)

ISO/IC97 Subcommittee 9
(Programming Languages for Numerical Control)

ISO/IC97 Subcommittee 14
(Representation of Data Elements)

ISO/IC97 Subcommittee 15
(Labelling and File Structure)

ISO Technical Committee 37
(Terminology)

ISO Technical Committee 46
(Documentation)

ISO Technical Committee 68
(Banking Procedures)

SO Technical Committee 97
(Computers and Information
Processing)

International Telegraph and
Telephone Consultative
Committee (CCITT)
Place Des Nations
CH-1211 Geneva 20, Switzerland

Interuniversity Communications
Council (EDUCOM)
P. O. Box 364, Rosedale Rd.
Princeton, NJ 08540
Tel: 609/921-7575

Japanese Standards Association
1-24 Alasaka 4 Chome
Minato-KU, Tokyo, 107 Japan

Japanese Standards Association
16, Chemin De La Vote-Greuse
1202 Geneva, Switzerland

Life Office Management Association
100 Park Ave.
New York, NY 10017
Tel: 212/725-1300

Manager: James F. Foley, Jr.
Systems Research

X3 Rep: Richard E. Ricketts

National Association of State
Information Systems
Iron Works Pike
Lexington, KY 40511
Tel: 606/252-2291

X3 Rep: George H. Roehm

National Bureau of Standards
Center for Computer Sciences
& Technology
Washington, DC 20234

X3 Rep: Harry S. White, Jr.

National Communications NCS-TS
8th & South Courthouse Rd.
Arlington, VA 22204
Tel: 202/692-2124

X3 Rep: Marshall L. Cain

National Electrical Manufacturers Association
155 East 44th St.
New York, NY 10017
Tel: 212/682-1500

National Fire Protection Association
60 Batterymarch St.
Boston, MA 02110
Tel: 617/482-8755

National Machine Tool Builders Association
7901 Westpark Dr.
McLean, VA 22101
Tel: 703/893-2900

X3 Rep: O. A. Rodrigues

National Retail Merchants Association
100 West 31st St.
New York, NY 10001
Tel: 212/244-8780

X3 O: Laurence Abzug

National Technical Information Service
Department of Commerce
5285 Port Royal Rd.
Springfield, VA 22151

OCR Users Association

X3 Rep: Herbert F. Schantz
Director

Printing Industries of America, Inc.
1730 North Lynn St.
Arlington, VA 22209
Tel: 703/527-6000

X3 Rep: Norman Scharpf

Privacy Protection Study Commission
2120 L St., NW, Suite 424
Washington, DC 20506

Ch: David F. Linowes

Scientific Apparatus Makers Association
1140 Connecticut Ave., NW
Washington, DC 20036
Tel: 202/223-1360

X3 Rep: Abraham Savitsky

Share Inc.
1 Illinois Center
111 E. Wacker Dr.
Chicago, IL 60601
Tel: 313/822-0932

President: John Hogan
X3 Rep: Thomas B. Steel, Jr.

Society of Actuaries
208 South LaSalle St.
Chicago, IL 60604
Tel: 312/236-3833

X3 O: John Kirkman

Society of Automotive Engineers
485 Lexington Ave.
New York, NY 10017

Society of Certified Data Processors
38 Main St.
Hudson, MA 01749
Tel: 617/562-9319

X3 Rep: Thomas M. Kurihara

Society for Wang Applications and Programs (SWAP)
Wang Laboratories, Inc.
836 North St.
Tewksbury, MA 01876

Executive Director: Jason R. Taylor

Standards Engineers Society
P.O. Box 7505
Philadelphia, PA 19101

Technical Association of the Pulp and Paper Industry
360 Lexington Ave.
New York, NY 10017

Telephone Group X3 Rep: V. N. Vaughan, Jr.

Underwriters' Laboratories, Inc.
Corporate Headquarters
207 East Ohio St.
Chicago, IL 60611
Tel: 312/642-6969

Union Internationale Des
Telecommunications
CCITT
Place Des Nations
1211 Geneva, Switzerland

U.S. Army Computer Systems
Command
Ft Belvoir, VA 22060

United States National
Committee of IEC
1430 Broadway
New York, NY 10018

VIM, Inc. X3 Rep: Sam W. White
Secretary: Albert Siegel

LIST OF ABBREVIATIONS

A	Alternate Member
ADAPSO	Association for Data Processing Service Operations
ADPESO	Automated Data Processing Equipment Selection Office
AFNOR	Association Francaise de Normalisation
ANSI	American National Standards Institute
CBEMA	Computer and Business Equipment Manufacturers Association
CCITT	International Telegraph and Telephone Consultative Committee
CCST	NBS Center for Computer Sciences and Technology
CDP	Certified Data Processor
Ch	Chairperson
CODASYL	Conference on Data Systems Languages
CODASYL/ATG	CODASYL/PLC's Asynchronous Task Group
CODASYL/DBLTG	CODASYL Data Base Language Task Group
CODASYL/DDLC	CODASYL Data Description Language Committee
CODASYL/DTTG	CODASYL Decision Table Task Group
CODASYL/EC	CODASYL Executive Committee
CODASYL/EUFC	CODASYL End User Facilities Committee
CODASYL/FDMLC	CODASYL FORTRAN Data Manipulation Language Committee
CODASYL/FPTG	CODASYL File Processing Task Group
CODASYL/OSCL	CODASYL Operating Systems Control Language Task Group
CODASYL/PETG	CODASYL Proposal Editing Task Group
CODASYL/PLC	CODASYL Programming Language Committee
CODASYL/SC	CODASYL Systems Committee
DECUS	Digital Equipment Computer Users Society
DOD	Department of Defense
DOD/DISPLAY	DOD Working Group on Computer-Generated Military Symbology
DOD/DOCN	DOD Working Group on DOD Computer Documentation Standards
DPMA	Data Processing Management Association
EDUCOM	Interuniversity Communications Council
FCIC	Federal COBOL Interpretation Committee
FIPS	Federal Information Processing Standards
FIPSCAC	FIPS Coordinating and Advisory Committee
FOCUS	Forum of Control Data Users
FTPSC	Federal Telecommunications Program Standards Committee
FTPSC/DEC	FTPSC for Data Elements and Codes
FTSC	Federal Telecommunications Standards Committee
IAC	ANSI X3 International Advisory Committee
IAC/ADP	Interagency Committee on Automatic Data Processing
IEC	International Electrotechnical Commission

LIST OF ABBREVIATIONS (CONT'D)

IEEE	Institute for Electrical and Electronic Engineers
IFIP	International Federation for Information Processing
IPSC	DOD Standardization Program for Information Processing Standards for Computers
ISO	International Organization for Standards
ISTAB	ANSI Information Systems Technical Advisory Board
JUG	Association for Computing Machinery Joint Users Group
NASIS	National Association for State Information Systems
NBS	National Bureau of Standards
O	Observer, ex officio member
P	Primary member
P(VC)	Vice chairperson
R	Reference Author
Rep	Representative
S	Secretary
SAI	Science Applications, Inc.
SPARC	ANSI X3 Standards Planning and Requirements Committee
SPARC/COMP	ANSI X3/SPARC COMPACT II/ACTION/SPLIT Study Group
SPARC/DBMS	ANSI X3/SPARC Data Base Management Systems Study Group
SPARC/LRPL	ANSI X3/SPARC Long Range Planning for Programming Language Standards Study Group
SPARC/OSCL	ANSI X3/SPARC Operating Systems Control Language Study Group
SPARC/TEXT	ANSI X3/SPARC Programming Language for Text Processing Study Group
SWAP	Society for Wang Applications and Programs
V&V	Verification and Validation
X3	ANSI X3 Committee
Z39	ANSI Z39 Committee

APPENDIX B:

BIBLIOGRAPHY

Military Publications

"Air Force Manuals and Regulations"

AFM 300-6, with AFSC and ESD supplements, Automatic Data Processing (ADP) Resource Management (1 June 1974).

This manual details all aspects of buying, operating, and disposing of ADP equipment. A revision of AFM 171-9, this manual covers general aspects, budgeting for data systems automation programs (DSAP), installation managing, evaluation and assistance, contractual matters, ADP equipment installation and operations, maintenance, ADP sharing, supplies, inventory and accountability, and reutilization and disposition.

AFM 300-12 with AFSC and ESD supplements, Procedures for Managing ADPS's (10 December 1971).

This manual prescribes additional procedures for managing automatic data processing systems (ADPS). The manual provides procedural guidance to implement ADPS management policy established by AFR 300-2. The procedures describe a comprehensive method for managing ADPS and ADPS elements throughout the ADPS life cycle. The scope of each project and the value of the resources committed will indicate the level of documentation, reports, review, and certification needed. The acquisition of software and related services, such as documentation, maintenance, and training, are also discussed.

AFR 57-1 with ESD supplement, Policies Responsibilities and Procedures for Obtaining New and Improved Operational Capabilities (17 August 1971).

This regulation establishes procedures, assigns responsibilities, and outlines documentation by which needs for new or improved operational capabilities are identified and advocated. This process includes recognition and statement of the needs and directive documentation. Instructions are given for preparing the statements of required operational capabilities which initiate the process and the program management directives which start the acquisition of the capabilities in response. Combat-required operational capabilities are treated as a special case.

AFR 65-3, Configuration Management (1 July 1974).

This regulation presents general policies and guidance for configuration management by the Air Force. This includes identifying, controlling, accounting for, and auditing the functional and physical characteristics of systems under procurement. Considerable attention is paid to the change process. An overview of the procurement process shows the conceptual, development/validation, full-scale development, and production/deployment phases marked by functional, allocated, and product baselines.

AFR 73-1 with AFSC supplement, Defense Standardization Program (DSP) (16 March 1967).

This regulation describes Air Force activities, responsibilites, channels of communication, and reporting for participation in the defense standardization program. Where industry standards are preferable, they may be used. Standards for new designs will be established for future use, with variety of items to be minimized. Locations for the departmental standardization office and supporting command standardization offices are established.

AFR 80-14 with AFSC supplement, Test and Evaluation (12 May 1972).

This regulation presents Air Force test and evaluation activity policies. All testing -- from basic research to system employment/deployment -- is covered, with special emphasis on operational test and evaluation. All Air Force organizations and activities are affected, and the test cycle is described in an attachment. Responsibilites for various Air Force organizational elements are defined.

AFR 102-5, USAF Management Policies Governing Development, Acquisition and Operation of Command and Control Systems (3 May 1972).

This regulation presents Air Force management policy and assigns responsibility in the development, acquisition, and operation of command and control systems. Command and control systems are to stress compatibility, commonality, and operational continuity. The Air Force will provide a central point of technology. Standards for languages, computer programs, data elements, etc. are stressed. Responsibilities for various Air Force organizational elements are explained.

AFR 300-1, ADP Program Management (10 June 1971).

This regulation prescribes policies and responsibilities for the selection, development, acquisition, management, and use of ADPSs, and for the design and development of automated data systems (ADS), except for ADP equipment excluded by paragraph I.A.2, enclosure 1, DOD Directive 5100.40. Management of ADPS/ADS, other than those used in combat weapon systems, will be decided on a project-by-project basis by Air Force Headquarters (HQ USAF) following review of the requirements document; HQ USAF decisions will be reflected in the applicable program management directive. This regulation stresses use of existing facilities and Air Force in-house capabilities.

AFR 300-2, Management of Automatic Data Processing Systems (12 November 1971).

This regulation prescribes policies and responsibilities for managing ADP systems of the ADP program. It applies to all Air Force activities with responsibilities for planning, authorizing, designing, developing, selecting, acquiring, using, maintaining, or managing (ADPSs) under AFR 300-1. This regulation provides a framework for ADPS management which permits early identification of a requirement or concept, analysis of the requirement or concept, and preliminary approval before there is extensive commitment of ADP resources.

AFR 310-1, Management of Contractor Data (30 June 1969).

This regulation defines procedures for managing data acquired under contract from industry. It implements DOD Instruction 5010.12. Responsibilities are defined for various Air Force organizational elements. Detailed guidance on the preparation, quality evaluation, and management of data standards is provided. Data includes all documentation and information associated with a contract.

AFR 800-2 with AFSC and ESD supplement, Program Management, 16 March 1972).

This regulation establishes policy, responsibilities, and reporting requirements for Air Force acquisition programs that are directed to be managed by this regulation. The regulation implements and includes all of the provisions of DOD Directive 5000.1. The regulation delegates maximum authority and responsibility for each program to the implementing command and the program manager to plan, organize, and conduct the acquisition, within the Air Force approved limits of system performance, schedule, and funding. The program manager is the technical and administrative focal point for all program activities, including the participation of all other organizations.

AFR 800-3, Engineering of Defense Systems (30 August 1973).

This regulation establishes policy and principles for management of a single totally integrated engineering effort for all AFR 800-2 managed programs. The regulation defines 11 engineering tasks and discusses those that are generally applied to each of the phases of acquisition management.

AFR 800-4, System/Equipment Turnover and Management Transition (19 November 1971).

This regulation states policy and assigns responsibilities for accomplishing system/equipment turnover and management transition. It applies to all programs under the acquisition policy of AFR 800-2. Turnover and transition agreements will normally be completed prior to the beginning of the production phase or at a more appropriate point in time by mutual agreement of the involved commands. Planning criteria and schedules for turnover and transition will be established by the program manager in conjunction with representatives of participating organizations early in the full-scale development phase. Schedules will be established as program milestones and reflected in the management documentation. Revisions must reflect the latest program direction and required actions will be monitored until completed. The transition of program management responsibilities will include consideration of all logistics functional elements. These elements will be transitioned concurrently if practical. A working group may be established by the program manager to insure integrated logistics support planning, schedule turnover, and trarsition events, and to insure that events are completed as scheduled.

AFR 800-6 with AFSC supplement, Program Control - Financial (14 July 1972).

This regulation assigns responsibility and provides guidance for using management control techniques to fulfill reporting requirements and to collect financial and other management data for use in financial analysis and program control. The application of specific techniques will be specified in the project management directives: the cost/schedule control criteria, WBS, cost performance report, contract funds status report, and contractor cost data reports. The latter three reports, when required, contain all the financial information required by the Air Force in a single contract. Contractors' only requirement for the cost/schedule control criteria is to satisfy the procedures, or organization upon the contractor. The description criteria for cost/schedule control and cost report are attached to the regulation.

AFR 800-8, Integrated Logistics Support (ILS) Program for Systems and Equipment (27 July 1972).

This regulation establishes policy and states criteria for the application of ILS throughout the entire life cycle for Air Force Systems. The regulation requires:

1. Conducting trade-offs among support ligistic alternatives and system design alternatives
2. Performing system and cost effectiveness analysis
3. Applying logistic support evaluation during contract source selection.

A deputy program manager for logistics is assigned to a program office to participate in ILS planning and to implement ILS considerations in engineering, design, and production efforts.

AFR 800-9, Production Management in the Acquisition Life Cycle (25 April 1973).

This regulation states policy and responsibilities for production management during the acquisition life cycle. The regulation is primarily concerned with influencing system engineering and design for efficient and economical quantity production. Some production considerations can apply to software development, e.g., maintainability, reliability, and logistic support availability and cost.

AFR 800-10, Management of Multi-Service Systems, Programs and Projects (12 September 1973).

This regulation authorizes the commanders of the Air Force Logistics and Systems Commands (AFLC and AFSC) to enter into agreements with Army and Navy counterparts to develop arrangements concerning the management of multi-service systems. Unless altered by mutual agreement, the policies that are applied will be those of the service that is designated as executive agent for a program.

AFR 800-11, Life Cycle Costing (LCC) (3 August 1973).

This regulation outlines policy and usage of life-cycle cost to estimate total cost of an item or system over its full life, including development, acquisition, operation, maintenance, support, and disposal. Life-cycle costing must be implemented early in the acquisition process, thereby influencing requirements, design, and production alternatives. LCC-3 is a DOD document referenced to provide guidance on the use of life-cycle costing in defense systems and subsystems.

AFR 800-12, For the Acquisition of Support Equipment (20 May 1974).

This regulation establishes policies and principles for acquisition of support equipment, i.e., equipment and computer programs not part of mission equipment and not required to perform mission operational functions. The regulation emphasizes cost-effectiveness of support equipment and standardization within a given system and among systems. Support computer programs must be planned, identified, and controlled using MIL-STD-483 as a guide. Testing (AFR 80-14) should demonstrate compatibility with mission equipment, and computer programs should be relatively free of errors.

AFR 800-14 with HFSC supplement, Management of Computer Resources in Systems, (10 May 1974).

This regulation establishes policy for the acquisition and support of computer equipment and computer programs employed as dedicated elements, subsystems, or components of systems developed or acquired under the program management concept established in AFR 800-2. This regulation applies to all Air Force activities responsible for planning, developing, acquiring, supporting, and using systems managed or acquired under AFR 800-2. The objective of this regulation is to insure that computer resources in systems are planned, developed, acquired, employed, and supported to effectively, efficiently, and economically accomplish Air Force-assigned missions. According to Air Force policy, computer resources in systems are managed as elements of major subsystems during conceptual, validation, full-scale development, production, employment, operation, and support phases. System performance requirements are allocated to these subsystems using in-depth trade-off studies and cost-effectiveness analyses.

AFR 800-15, Human Factors Engineering and Management (1 October 1974).

This regulation establishes policies and responsibilities to incorporated human factors engineering into the engineering and management efforts of all acquisition programs. Some human factors elements can be applied to computer programs, e.g., manning and training considerations, efficient human usage, and determining whether Air Force personnel with training can in fact operate, maintain, and support the system in its intended operational environment.

AFSC Design Handbook 4-2, Electronic Systems Test & Evaluation (10 April 1971).

This handbook has been prepared for use with the technological and engineering disciplines involved in test and evaluation of Air Force command, communications, control, surveillance, and warning systems. The data contained in this handbook are intended for use with Air Force systems programs being conducted under the management concepts established in other regulations. The handbook is directed toward the testing and evaluation processes as set forth in AFR 80-14. Planning, test tools and different types of computer program testing are covered.

AR 70-10, Test and Evaluation During Development and Acquisition of Material (15 September 1971).

This regulation prescribes the objectives, concepts, responsibilities, policies, and major tests which apply to the testing and evaluation leading to type classification of Army material. It covers the life cycle activities starting with the initial preparation of the coordinated test program (CTP) in the concept formulation phase and culminating with the successful completion of the production validation process.

AR 71-3, User Field Tests, Experiments and Evaluations (19 March 1968).

This regulation outlines objectives, policies, responsibilities, and procedures for conducting user field tests, experiments, and evaluations, including troop tests, confirmatory tests, field experiments, field evaluations, and combat evaluations. These tests establish the performance capabilities of selected items of Army equipment in the hands of the user, and the workability and effectiveness of organizational concepts, doctrine, tactics and techniques, and tables of organization duty positions.

AR 1000-1, Basic Policies for Systems Acquisition by the Department of the Army (5 November 1974).

This regulation establishes Army policies to minimize costs in acquiring material systems meeting operation requirements. Preference is to be given to systems with inherent performance growth potential. The technical and operational feasibility of a requirement must be demonstrated before it is formalized. Directions for high-level decision making, priority in testing, shortened development time, application of integrated logistic support, funding for priority programs, handling the cost versus quantity balance, and controlling program costs are also included.

"Army Regulations"

AR 70-10, Test and Evaluation During Development and Acquisition of Material (15 September 1971).

This regulation prescribes the objectives, concepts, responsilities, policies, and major tests which apply to the testing and evaluation leading to type classification of Army material. It covers the life cycle activities starting with the initial preparation of the coordinated test program (CTP) in the concept formulation phase and culminating with the successful completion of the production validation process.

AR 71-3, User Field Tests, Experiments and Evaluations (19 March 1968).

This regulation outlines objectives, policies, responsibilites, and procedures for conducting user field tests, experiments, and evaluations, including troop tests, confirmatory tests, field experiments, field evaluations, and combat evaluations. These tests establish the performance capabilities of selected items of Army equipment in the hands of the user, and the workability and effectiveness of organizational concepts, doctrine, tactics and techniques, and tables of organization duty positions.

AR 1000-1, Basic Policies for Systems Acquisition by the Department of the Army (5 November 1974).

This regulation establishes Army policies to minimize costs in acquiring material systems meeting operational requirements. Preference is to be given to systems with inherent performance growth potential. The technical and operational feasibility of a requirement must be demonstrated before it is formalized. Directions for high-level decision making, priority in testing, shortened development time, application of integrated logistic support, funding for priority programs, handling the cost versus quantity balance, and controlling program costs are also included.

DOD Directive 5000-1, Acquisition of Major Defense Systems (22 December 1975).

This directive establishes policy for all major defense systems that (1) exceed a stated estimated cost, (2) are urgent, and (3) are recommended for inclusion by DOD component commanders. This policy is for decentralized management of individual programs by a single individual (i.e., a program manager) with sufficient authority to accomplish all program objectives. Initial program commitments and any increase in commitments are to be decided by the Secretary of Defense.

DOD Directive 5000.3, Test and Evaluation (19 January 1973).

This directive establishes policy for the conduct of test and evaluation of major programs (as defined in DOD Directive 5000.1). Its principles also apply to the acquisition of other defense systems. Test and evaluation shall be scheduled in the acquisition phase, and shall begin early and continue throughout the acquisition, in an effort to identify and reduce technical risk. Development and operational test and evaluation are required for all systems. Special provisions are made for one-of-a-kind systems.

DOD Directive 5010.19, Configuration Management (17 July 1968).

This directive establishes a policy for configuration management for all military departments, DOD components at all echelons, and all defense/industry interfaces. Configuration management must be applied to all configuration items procured, or obtained by agreement between in-house activities. The directive describes responsibilities for initiation, planning, documentation, and audits of configuration management, and describes the processes of functional and allocated identification, control, and status accounting.

DOD Directive 4105.55, Selection and Acquisition of Automatic Data Processing Resources (19 May 1972).

This directive applies to activities of the Office of the Secretary of Defense, Joint Chiefs of Staff (JCS), Military departments, and defense agencies, but not to those of government contractors. This directive implements guidance from the Office of Management and Budget and from the General Services Administration, and supplements the DOD Directive 5100.50. The policies and guidance defined by this directive cover all areas of ADP resource acquisition (original, upgrade, and replacement), but at a very high level. It stresses the need for competitive selection, planning prior to acquisition, and review of all alternatives before making a selection.

DOD Instruction 5010.12, Management of Technical Data (5 December 1968).

This instruction implements a program for the management and administration of technical data developed or contractually acquired by DOD or any component. The instruction identifies planning and continuous monitoring based on current need for all technical data. The instruction also establishes procurement data packages, data item description, and the contract data requirements list. Attachments give useful guidance on the need and usage of data items, preparation of data requirements (data call), and information on technical data standardization.

DOD Instruction 5010.21, Configuration Management Implementation Guidance (6 August 1968).

This instruction provides guidance for the implementation of DOD policies on configuation management established in DOD Directive 5010.19. It directs configuration identification, configuration control, configuration status accounting, configuration audits, procurement aspects, logistic support aspects, implementation, and definition of the application of configuration management to all DOD systems, equipments, and other designated material items.

DOD Instruction 7041.3, Economic Analysis of Proposed Department of Defense Investments (26 February 1969).

This instruction establishes policy and procedures for consistent application of economic analysis in order to:

1. Systematically identify the benefits and costs so that useful comparisons of alternative methods for accomplishing a task or mission can be made
2. Highlight the key variables and assumptions on which investment decisions are based, and allow evaluation of these assumptions
3. Evaluate alternative methods of financing investments
4. Compare the relative merits of various alternatives as an aid in selecting the best alternative.

Enclosures to this publication provide references, definition, instructions and forms for preparing summaries of project costs and project benefits, and a discussion of discounting techniques.

DOD Manual 4120.17M, Automated Data System Documentation Standards Manual (December 1972).

This publication provides a general discussion of ADS documentation, its purposes, authorship requirements, life-cycle applicability, and the needs of its audience. An attempt is made to correlate project complexity with documentation needs. The documentation types outlined and described in this publication are:

1. Functional Descriptions (ED)
2. Data Requirements Document (RD)
3. System/Subsystem Specification (SS)
4. Program Specification (PS)
5. Data Base Specifications (DS)

6. Users Manual (UM)
7. Computer Operation Manual (OM)
8. Program Maintenance Manual (MM)
9. Test and Implementation Plan (TP)
10. Test Analysis Report (TR).

These documents are intended to form the complete set of ADS documentation which could be added to standard system planning documentation requirements.

"Military Standards and Specifications"

MIL-H-468558, Human Engineering and Requirements for Military Systems, Equipment and Facilities (2 May 1972).

This specification defines requirements for applying principles and criteria (MIL-STD-1472A) of human engineering to the development and acquisition of systems. The specification requires a plan and tasks of analysis, design, development, and test and evaluation.

MIL-HDBK-217B, Reliability Prediction of Electronic Equipment (20 September 1974).

Oriented toward reliability prediction of military electronic equipment, this handbook provides a common basis for predicting and comparing predictions on military contracts and proposals. It provides two methods of reliability prediction: parts stress analysis, and parts count. Mathematical models for parts failure rates for use in computer programming and tables for determining base failure rates are also provided.

MIL-HDBK-472, Maintainability Prediction: Military Standardization Handbook (24 May 1966).

The purpose of this handbook is to familiarize project managers and design engineers with current maintainability prediction procedures for electronic systems and equipment, and general systems and equipments. Four procedures for maintainability prediction in electronic systems and equipment and in general systems and equipments are presented. Use of this handbook facilitates the design, development, and production of equipment and systems requiring a high order of maintainability.

MIL-Q-9858A, Quality Program Requirements (16 December 1963).

This specification applies to all supplies or services referenced in a configuration item specification or contract. It requires a contractor to plan, establish, and document a quality program for procedures, processes, and products, including purchased data, to make objective evidence of quality conformance available to the government representative.

MIL-S-52779 (AD), Software Quality Assurance Program Requirements (5 May 1974).

This specification establishes the requirements for a definitive, visible, contractor software quality assurance (SQA) program and its associated planning documents. This documentation shall include methods to aid in the identification of software development work packages, to track work progress, to assure configuration management practices are being maintained, to assess the success of software testing efforts, to detect and correct software deficiencies, and to establish software library controls. In addition, the SQA program is to address design evaluation, documentation reviews, technical review and audit schedule, and SQA tools and techniques.

MIL-S83490, Specifications, Types and Forms (30 October 1968).

This specification identifies types of permitted specifications. With reservation for technical society, industry assocation, and contractor standards and normal practices, all specifications must conform to MIL-STD-490 or Defense Standardization Manual 3120.3-M in format and content. To a large extent, MIL-STD-490 forms a part of this specification by specific reference. This specification includes provisions for quality assurance of specifications (not of the item described by the specification).

MIL-STD-100A, Engineering Drawing Practices (1 March 1965).

This standard prescribes procedures and format authorized for the preparation of form 1 engineering drawings and associated lists prepared by or for the departments and agencies of DOD, as prescribed by MIL-D-1000. MIL-D-1000 specifies the extent to which contractors are required to use MIL-STD-100A for form 2 engineering drawings.

MIL-STD-109, Inspection Terms and Definitions (24 June 1955).

This standard designates the names and associated terminology for examination and testing of supplies and services, and provides a basis for general understanding of such terms. It defines inspection and its categories, specifies inspection classes, outlines types of inspection , details amounts of inspection, discusses inspection lots, samples, classes of defects, and includes definitions of general inspection terms.

MIL-STD-109B, Quality Assurance Terms and Definitions (4 April 1969).

This document provides a standardized interpretation of quality assurance terms and definitions for use in specifications, standards, drawings, technical manuals, contracts, quality control inspection and related documents, and engineering evaluation reports.

MIL-STD-470, Maintainability Program Requirements for Systems and Equipments (24 March 1966).

The purpose of this standard is to establish maintainability programs through standard program requirements for DOD procurements. It directs the preparation of a maintainability program plan, a maintainability analysis, and maintainability integration in system design, engineering, and requirements development.

MIL-STD-480, Configuration Control - Engineering Changes, Deviations and Waivers (30 October 1968).

Both MIL-STD-480 and MIL-STD-481A deal with configuration control. MIL-STD-480 covers a broader area and requires a more complete analysis of the impact if the proposed engineering change were implemented. It requires that the package submitted with an engineering change proposal contain a description of all known interface effects and information concerning changes required in the functional/allocated/product configuration identification. MIL-STD-480, therefore, is imposed on prime contractors participating in the development, specification, and preparation of engineering change proposals impacting systems or high level configuration items. This standard also specifies requirements for submittal of deviations, waivers and notices of revision.

MIL-STD-481A, Configuration Control - Engineering Changes, Deviations and Waivers (Short Form) (30 October 1968).

This standard is intended for use in contracts involving the procurement of multi-application items or items for which the prescribed detailed design was not developed by the contractor. It prescribes requirements for the preparation and submission of proposed engineering changes and requests for deviations and waivers. It charges the procuring activity with the major responsibility for determination of possible effects of the engineering change on higher level or association items. When a more complete description of engineering changes is desired, MIL-STD-480 should be specified.

MIL-STD-482, Configuration Status Accounting Data Elements and Related Features (19 September 1968).

This standard establishes data items and aggregates to be used for configuration status accounting. Provisions are made for the addition of new programs or contractor data items. The standard includes item names, legal coded values, and an explanation of the item meaning.

MIL-STD-483 (USAF), Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs (31 December 1970).

This standard establishes uniform configuration management practices that can be tailored to all USAF systems and configuration items, including computer programs. It supplements and adds requirements not contained in MIL-STD-480, 481, 482, and 490. The standard describes the general requirements of configuration management and outlines the content of 15 documents pertinent to identifying, establishing, and controlling system and configuration item baselines.

MIL-STD-490, Specification Practices (30 October 1968).

This publication describes the purposes, format, and technical content of various types of specifications. The applicable sets of specifications relating to the development of computer programs are the type A (system specification), type B5 (computer program development specification), and type C5 (computer program product specification). This material should be used with MIL-STD-483 to establish configuration baselines. The format and content of specifications in this standard and in MIL-STD-483 are somewhat different.

MIL-STD-499A, Engineering Management (1 May 1974).

This standard contains a set of criteria to serve as a guide for contractors to plan, conduct, and manage a system engineering effort. The standard emphasizes the tailoring of engineering tasks to each particular program and the need for all engineering tasks to be performed as a single total integrated engineering effort. The appendix includes task statements which can be tailored to particular program and become specific contractual requirements.

MIL-STD-756A, Reliability Prediction (15 May 1963).

This standard establishes uniform procedures for predicting the quantitative reliability of aircraft, missles, satellites, electronic equipment, and subdivisions of them throughout the development phases to reveal design weaknesses and to form a basis for apportionment of reliability requirements to the various subdivisions of the product. Feasibility prediction and design prediction are the two classes of reliability prediction procedures discussed.

MIL-STD-757, Reliability Evaluation From Demonstration Data (19 June 1964).

This document establishes uniform technical procedures for evaluating achieved reliability, the minimum input information necessary for this purpose, and the criteria under which this minimum information is gathered. The acceptable reliability level which must be achieved and the test procedures and conditions which apply to and become a part of this procedure are those stated in the detailed equipment specification.

MIL-STD-881A, Work Breakdown Structures for Defense Material Items (25 April 1975).

This standard describes a framework for planning and assigning technical responsibilities and tasks, providing uniform control over and reporting of progress and status of contractor efforts, and uniformity of definitions to improve interpretation and reconcilitation of reports. The standard describes the preparation and use of the WBS and sublevels of the structure that apply to summaries and to individual contracts. The highest three levels of the WBS, i.e., the summary WBS, are specified for seven different categories of systems, including electronic systems which encompass ADP and computer programs. The standard is used by both contractors and DOD components.

MIL-STD-1472A, Human Engineering Design Criteria for Military Systems, Equipment and Facilities (9 February 1968).

This standard specifies human engineering criteria for human/equipment interfaces. All of the criteria refer to physical limits. There are no explicit references to qualities of computer programs and their interface with humans.

MIL-STD-1521 (USAF), Technical Reviews and Audits for Systems, Equipment and Computer Programs (1 September 1972).

This standard describes the detailed requirements for conducting the following system milestone events:

1. System Requirements Review (SRR)
2. System Design Review (SDR)
3. Preliminary Design Review (PDR)
4. Critical Design Review (CDR)
5. Functional Configuration Audit (FCA)
6. Physical Configuration Review (PCA)
7. Formal Qualification Review (FQR)

The reviews and audits are conducted only when specified in the contract. The standard identifies contractor and government responsibilities in the conduct of the review or audit, and outlines the minimum content of information to be presented. The reviews and audits may be used individually or as a set, and may be applied to each item of an entire system, or to individual equipment or computer programs.

MIL-STD-1528 (USAF), Production Management (1 August 1972).

This standard prescribes the production management objectives and requirements which must be met by the contractor's production management system on any contract against which this standard is levied. This standard is applicable to contracts which involve the engineering development or production of military systems, subsystems, equipment, and components.

MIL-T-38804 (USAF), Preparation of Time Compliance Technical Orders (31 July 1972).

This specification identifies the requirements for preparing time compliance technical orders (TCTO) including format. TCTOs are used to impose or direct usage restrictions, retrofit changes, or special one-time inspection or replacement of components or systems. TCTO's are a means of changing and ordering the retesting of computer programs.

"Other Military Publications"

Advanced Ballistic Missile Defense Agency (ABMDA) Research and Development Software Standards, vol. 1, edition 2, (System Development Corporation, 8 September 1972).

This document presents the basic set of advanced ballistic missile defense agency (ARMDA) research and development soft-deliverables of a computer software research, design, and development project as documentation, source programs, and data. These standards are applicable to the deliverables of all associate contractors performing ABMDA research and development involving software deliverables. The scope of their applicability is to be determined by the individual contracts; deviations are not permitted except by prior approval of the ABMDA Data Processing Division.

Bligh, Alan B., Doris E. Gossett and Janet P. Mason, Research Computation Center Program Publication Guide (Naval Research Laboratory (NRL), February 1969).

This guide contains the research computation center (R66) standards for documenting, revising, classifying, submitting, identifying, obtaining, and evaluating computer programs at NRL. It is anticipated that the use of such standards will promote efficient organization of an RCC program library and will provide for maximum effective usage of this program library throughout NRL.

Cobol Compiler Validation System (Automatic Data Processing Selection Office (Navy), 26 January 1976).

This document is one of 14 volumes comprising the preliminary documentation for the 1974 U.S. Navy COBOL Compiler Validation System (CCVS). The 1974 CCVS wil consist of audit routines, their related data, and an execution routine (VP-Routine) which prepares the audit routine for compilation. Each audit routine is a COBOL program which includes many tests and supporting procedures indicating the results of the tests. The audit routines collectively contain all the features of American National Standard Programs Langauge COBOL - X3.23-1974 (except for the enter statement of the nucleus module) as specified in Federal Information Processing Standard (FIPS) 21-1.

Berning, Paul T., A Semanol (73) Implementation Standard for Jovial (J73) (Tactical Reconnaissance Wing Systems Group, 30 June 1975).

The formal definition of the programming language JOVIAL (J73) was produced by the metalanguage, SEMANOL. The process of definition resulted in the successful identification of many ambiguities and conflicts in the JOVIAL language which were reported to the language definition Committee. SEMANOL (73) is understandable by laymen processable by the SEMANOL interpreter computer program. The interpreter program was completed and debugged during the contract period. JOVIAL (J73), as processed by the SEMANOL interpreter, has been tested to the extent that (1) the JOVIAL (J73) level one subset grammer is well debugged, (2) the formal definition is syntactically correct, and (3) the simpler semantics are tested to yield correct answers. The results of this effort coupled with previous and concurrent efforts show SEMANOL as a highly valuable standarization tool for failure use in DOD language controls.

AD-A049 869 SCIENCE APPLICATIONS INC MCLEAN VA F/G 9/2
SOFTWARE ENGINEERING. VOLUME I. ITS DEVELOPMENT AND STANDARDS.(U)
JAN 78 DACA88-76-C-0010
UNCLASSIFIED CERL-TR-E-126-VOL-1 NL

2 OF 2
AD
A049 869

END
DATE
FILMED
3-78
DDC

Defense Standardization Program Area Information Processing Standards for Computers (IPSC), Project IPSC-0012 (IPSC Program Analysis, FY 77-81) (Department of the Air Force, 12 August 1976).

This program analysis describes the FIPS program and the Federal Government Program for management of automatic data processing in general, of which FIPS is a part. In addition to describing the national and international standardization activities in this area, it also describes other DOD standardization activities and the relationships of the IPSC area to all of these programs. Projects underway and assignments of responsibilities are defined. IPSC is charged with the assignment of standardization responsibility in the following areas: terminology, methods of problem description, programming language, communication characteristics, input-output media and format, character codes, and character recognition.

DOD Weapon Systems Software Management Study, (Applied Physics Laboratory, John Hopkins University, Draft, May 1975).

This report attempts to identify and define:

1. The nature of the critical software problems facing DOD
2. The principal factors contributing to problems
3. The high payoff areas and alternatives available
4. The management instruments and policies that are needed to define and bound the functions, responsibilities, and mission areas of weapon systems software management.

To achieve these objectives, the study includes a review and analysis of 10 recent major DOD-sponsored studies, a review of the software system design and management in 10 Navy and 2 Army weapon systems, and discussions with service and industry organizations involved in weapon system software acquisition, development, and maintenance.

HYPO-COBOL Language Specifications -- A Proper Subset of the Low-Intermediate Level of FIPS Pub 21-1 COBOL (Automated Data Processing Equipment Selection Office (NAVY), 29 December 1975).

This publication discusses HYPO-COBOL as a proper subset of the full American National Standard Programming Language COBOL as defined in ANSI X3.23-1974. HYPO-COBOL is oriented toward a compiling system which need not place heavy demands on its environment in terms of time and space, and provides functional capability beyond the minimum definition of COBOL. Most of the elements selected for inclusion in HYPO-COBOL are required in any implementation of a low-level COBOL compiler, as defined in FIPS Pub 21-1. However, HYPO-COBOL is not a proper subset of low-level COBOL because it does not contain all elements of low-level COBOL, but does contain selected elements from higher levels of COBOL.

Information Processing/Data Automation Implications of Air Force Command and Control Requirements in the 1980's, CCIP-85, vol 1, (Tactical Reconnaissance Wing, April 1972).

This study projects trends in Air Force command and control requirements and their information processing implications, infers the information processing technology requirements to satisfy those trends, determines world environment information processing technology trends impacting Air Force command and control, identifies information processing research and development programs to handle these impacting items, and integrates these programs into a systems design approach for Air Force research and development in command and control.

Kiselev, B. B., Standardization of Computer Technology Facilities (Foreign Technology Division, Wright-Patterson AFB, 26 November 1968).

This report points out that each computer-manufacturing organization has in the past had its own system of specifications, and the result has been a great variation in parameters of such computer equipment as analog and digital processors, magnetic tape, drum and core memories, I/O units and verifiers, sorters, and collaters. Various reasons for the desirability of standardizing basic characteristics to make computer equipment compatible and interchangeable are listed, and a program for realizing this goal in the next 2 years is presented. It is expected that this action will also promote the establishment of procedures, technical requirements, and methods for testing computers that will result in their increased speed, capacity, and reliability, as well as in compatibility of instruction repertoires and software, development of modern I/O units, and centralization and specialization of manufacturing facilities for computers and their components.

Mitchell, Wallace E., and Joseph L. Pokorney, A Systems Approach to Computer Programs (Electronic Systems Division, L. G. Hanscom AFB, Technical Requirements and Standards Office, February 1967).

This paper describes an Electronic Systems Division (ESD) approach to adapting existing AFSC system management techniques to computer programs. Procedures for insuring system compatibility, design integrity, and technical control are discussed, and a method for achieving design verification and qualification is presented. Particular emphasis is placed on the relationship of these techniques to computer programs as elements of large computer-based systems. The application of these techniques is illustrated through selected examples taken from current ESD system procurements.

Piligian, M. S. and Joseph L. Pokorney, Air Force Concepts for the Technical Control and Design Verification of Computer Programs (Electronic Systems Division, L. G. Hanscom AFB, Technical Requirements and Standards Office, April 1967).

This paper presents Air Force-developed concepts for technical control and design verification of computer programs. Starting with the definition of a computer as a deliverable contract end item requiring a design and development effort, management procedures for controlling the design and development process are explained. Technical control of computer program design through periodic design reviews is outlined, and test concepts for verification of computer program performance are presented. The techniques discussed are based on an exchange of technical information between the contractor and the procuring agency at a series of discrete milestones throughout the design and development process. The milestones, including design reviews, qualifications testing, etc., are described and their relationship to the design and development of a computer-based system is illustrated. The techniques are directly applicable to any large computer-based system, military or commercial, and can be easily tailored to fit a small computer system.

Weiss, David M., The Mudd Report: A Case Study of Navy Software Development Practices, NRL Report 7909 (Naval Research Laboratory, 21 May 1975).

The Mudd Report is a study of Navy software development practices which is based on a series of interviews with those responsible for the development of Navy systems. The study chronicles the development of a fictional system with requirements typical of Navy tactical systems currently operational or under development. A history of the decisions made during the development of the system is first given. Following the history is an analysis of the impact of each decision on the software developed for the system, and on the life cycle of the software. Finally, a set of recommendations for avoiding the pitfalls described in the report is given. The recommendations are designed to assist Navy program managers responsible for software development.

"Non-military Government Publications"

American National Standard COBOL as the Federal Standard COBOL, FIPS Pub 21 (National Bureau of Standards,).

This publication announces the adoption of the American National Standard COBOL (X3.23-1968) as the Federal Standard COBOL. The American National Standard defines the elements of the COBOL programming language and the rules for their use. It is used by implementors as the reference authority in developing compilers and by users for

writing programs in COBOL. The primary purpose of the standard is to promote a high degree of interchangeability of programs for use on a variety of automatic data processing systems.

Chapman, R., D. Klinglesmith, and G. Mason, Standards Guide for Space and Earth Sciences Computer Software (National Aeronautics and Space Administration, January 1972).

Guidelines for the preparation of systems analysis and programming work statements are presented. The data are geared toward the efficient administration of available monetary and equipment resources. Language standards and the application of good management techniques to software development are emphasized.

CODASYL Data Description Language Committee, CODASYL Data Description Language, Handbook 113 (National Bureau of Standards June 1973).

This is the first journal of development of the CODASYL Data Description Language Committee. The report specifies the syntactic and semantic rules of a data description language designed to "permit the description of the structure and contents of a data base in a language independent of, but common to many other high level languages." A background and history of the Committee and the major concepts of the language are also included.

Code for Information Exchange, FIPS Pub 1 (National Bureau of Standards, 1 November 1968).

This document provides administrative policy and general guidance information relative to the implementation and utilization of the standard code for information exchange. The technical specifications define a code and character set for use in FIPS communications systems and associated equipments.

Davis, Ruth M., "Computer Auditing Increasingly a Necessity," Dimensions (National Bureau of Standards, July 1976), pp 7-10.

This article dicusses the existing lack of auditing, accounting, and fidelity of computer systems. The articles calls for the development of fidelity standards by the computer user community, and for research and development of measuring and testing techniques as a basis for standards.

Federal Information Processing Standards Index, FIPS Pub 12-2 (National Bureau of Standards, 1 December 1974).

This publication provides material concerning standardization activities in the area of information processing at the Federal, National, and International levels. Also included are related policy and procedural guideline documents. A list of federal government participants involved in the development of federal information processing standards is provided.

Flowchart Symbols and Their Usage in Information Processing, FIPS Pub 24 (National Bureau of Standards, 30 June 1973).

This publication establishes standard flowchart symbols and specifies their use in the preparation of flowcharts in documenting information processing systems. This standard applies to any federal information processing operation in which use of symbolic representation is desirable to document the sequence of operations and the flow of data and paperwork.

Grooms, David W., Computer Software Standards: A Bibliography with Abstracts (National Technical Information Service, May 1976).

This document provides a listing of standards for the design and development of computer software, including compiler standards, programming language standards, and program development standards.

Grooms, David W., Programming Language Design: A Bibliography with Abstracts (National Technical Information Service, August 1975).

This bibliography presents research on the design, development and implementation of programming languages. The research includes specifications and applications for the programming languages in systems development and their use in specific cases, such as interactive graphic systems, univac computers, and others. The report also includes research on language compilers, syntax, semantics, and logic modules.

Guidelines for Describing Information Interchange Formats, FIPS Pub 20 (National Bureau of Standards, 1 March 1972).

This publication provides guidelines which identify and describe the various characteristics of formatted information that should be considered whenever formatted information is interchanged. The objective is to clarify and improve the documentation necessary to effectively provide, process, or use the information involved. The guidelines provided are to be used throughout the federal government as a checklist for preparing effective documentation of formatted information interchange.

Guidelines for Documentation of Computer Programs and Automated Data Systems, FIPS Pub 38 (National Bureau of Standards, 15 February 1976).

These guidelines provide a basis for determining the content and extent of documentation for computer programs and automated data systems. Software development phases and related document types are identified, several examples of documentation options are given, and content guidelines for the following document types are produced: Functional Requirements Document, Data Requirements Document, System/Subsystem Specification, Program Specification, Data Base Specification, Users Manual, Operations Manual, Program Maintenance Manual, Test Plan, Test Analysis Report.

Marron, Beatrice, Guidelines for Documentation of Computer Programs and Automated Data Systems (National Bureau of Standards, 15 February 1976).

These guidelines provide a basis for determining the content and extent of documentation for comptuer programs and automated data systems. Software development phases and related document types are identified, several examples of documentation options are given, and content guidelines for 10 document types are provided. The 10 document types are: Functional Requirements Document, Data Requirements Document, System/Subsystem, Specification, Program Specification, Data Base Specification, Users Manual, Operations Manual, Program Maintenance Manual, Test Plan, Test Analysis Report. The guidelines are intended to be a basic reference and a checklist for general use throughout the Federal Government to plan and evaluate documentation practices.

Neoterics, Inc., NASIS Data Base Management System - IBM 360/370 OS MVT Implementation: Vol 1: Installation Standards (National Aeronautics and Space Administration, September 1973).

The installation standards for the NASIS Data Base Management System are presented. The standard approach to preparing systems documentation and the program design and coding rules and conventions are outlined. Included are instructions for preparing all major specifications and suggestions for improving the quality and efficiency of the programming task.

Objectives and Requirements of the Federal Information Processing Standards Program, FIPS Pub 23 (National Bureau of Standards, 15 February 1973).

The FIPS Program was established in response to Public Law 89-306 (The Brooks Legislation), which provides for the economic and efficient purchase, lease, maintenance, operation, and utilization of ADP

equipment by Federal departments and agencies. This document outlines the objectives of the FIPS Program and identifies requirements for specific standards necessary to accomplish these objectives.

Software Summary for Describing Computer Programs and Automated Data Systems, FIPS Pub 30 (National Bureau of Standards, 30 June 1974).

This publication provides a standard software summary form and instructions for describing computer programs or automated data systems for identification, reference, and dissementation purposes.

Standardization of Data Elements and Representations, FIPS Pub 28 (National Bureau of Standards, 5 December 1973).

This publication provides policy and identifies responsibilities of executive branch departments and independent agencies for a government-wide program for the standardization of data elements and representations used in Federal automated data systems.

Vocabulary for Information Processing, FIPS Pub 11 (National Bureau of Standards, 1 December 1970).

This publication provides an alphabetic listing of approximately 1200 terms and definitions for use in information processing activities such as the description, representation, communication, interpretation, and processing of data by human or automatic means. Multiple-word terms are listed in their natural order: terms with identical last words appear as "see references" under the common word. Other reference symbols indicate synonyms, preferred terms, contrasting meanings, and unabbreviated forms for defined terms which are acronyms or abbreviations.

Voluntary Industrial Standards in the United States: An Overview of Their Evolution and Significance For the Congress (Congressional Research Service, Library of Congress, July 1974).

This report is an overview of the standardization process and the national and international organizations impacting standardization in both the private and public sectors in the United States. It discusses the standardization process and the extent to which it is and has affected the American consumer. Current legislation for voluntary international standards is also discussed. A history of standardization and its industrial and consumer impact, including the development of NBS and ANSI an national standards organizations and their activities in voluntary standardization, provides a background for discussing trends towards an increasing government role in standardization. Standardization on the international scene and its impact on U.S. business are identified.

Vickers, Mabel V., Aids for COBOL Program Conversion (National Bureau of Standards, 1 December 1975), An Analysis.

Since COBOL is a living language, in the sense that it is under constant development and clarification, the Federal community which relies heavily on COBOL to satisfy programming needs has a large degree of assurance that COBOL will continue to meet their needs as future generation systems are introduced. However, along with the advantage of having more sophisticated and better COBOL tools to meet new systems requirements, there is a short term disadvantage. As clarifications and new facilities are added, they interact with language specifications already standardized, and this interaction sometimes requires changes in source programs. An analysis in the form of narrative descriptions and syntax comparisons, it aids the transition of COBOL program from use with compilers developed in accordance with the 1968 COBOL standard (FIPS Pub 21) to compilers developed in accordance with the 1974 COBOL standard (FIPS Pub 21-1).

Wood, John F., Guidelines for Benchmarking ADP Systems in the Competitive Procurement Environment (National Bureau of Standards, 15 December 1975).

This publication provides general guidelines to be used by Federal agencies as best practices of benchmark mix demonstrations for validating hardware and software performance in context with processing the users' expected actual workload. It treats selection of the benchmark mix, sanitation of the benchmark mix, and planning for and conducting the benchmark mix demonstration.

Brooks, Frederick P., Jr., The Mythical Man-Month Essays on Software Engineering (Addison-Wesley Publishing Co., 1975).

This publication contends that large programming projects suffer management problems different in kind from small ones due to the division of labor. It identifies the critical need as conceptual integrity of the product itself, and discusses the difficulties of achieving this unity and methods for achieving it.

Callender, E. D., M. Feliciano, and L. D. Jennings, SAMSO Computer Language and Software Development Environment Requirements (Aerospace Corp., 15 December 1975).

This report identifies the higher order language requirements and software development environment requirements for SAMSO applications. This document has three major sections: (1) Functional requirements for SAMSO computer programs, (2) Higher order language constructs and software development environment constructs necessary to support the functional requirements, and (3) recommendations. This report concludes that it is technically feasible for

most computer programming for new SAMSO projects to be done in one higher order language. If this technical feasibility is coupled with the increasing cost of software development and maintenance, standardization on a single higher order language becomes highly desirable. If cost savings in software development and maintenance are to be realized, a standard software development environment must be created and a program for the specifications, development, test, and maintenance of this single higher order language and its associated software development environment must be established.

Connolly, J. T., Software Acquisition Management Guidebook: Regulations, Specifications and Standards (Mitre Corp., October 1975).

Regulations, specifications, and standards pertinent to Air Force software acquisition are identified and categorized by management and software development tasks required during a system acquisition. The place of software development in a system life cycle and the two principal Air Force management regulations impacting on software acquisition are discussed. Brief summaries of regulations, specifications, and standards and keyword cross-reference lists for selected documents are included.

Englander, William R., "How Standard Are COBOL Compilers?", Business Automation, vol. 17, no 6 (June 1970), pp 65-70.

This document reports on COBOL compiler validation system (CCVS) test failures without identifying which compiler. It suggests using CCVS to select both a compiler and those features to be avoided as not compatible.

Fleiss, Joel E., Guy M. Philips, Andrew Edwards, and Larry Rieder, Programming for Transferability (International Computer Systems, Inc., September 1972).

This document presents the results of an investigation of design and documentation techniques used in computer programming in order to develop recommendations and guidelines for easing the transfer of computer programs written for one computing environment to another.

The first part of this study is concerned with a general transferability analysis and techniques to be utilized independent of any particular programming language. Emphasis is placed on the importance of transferability considerations during design and coding of the problem program.

The second section of the study deals with higher level and assembly/macro languages. Specific suggestions for improvement of FORTRAN, JOVIAL, and COBOL program design are included.

Fujii, Atsushi, "Software in Japan: Supported Growth," Datamation, Vol 17, No. 4, February 1971), pp 26-29.

This article reports that the Japanese government is encouraging and supporting software development in several ways. A new organization will buy software packages from developers and lease them to users. The government thus assumes the majority of the risk. The government gives examinations for information processing engineers. The Japanese government has launched a 5-year project (large, industrial engineering development) to upgrade the technology of the Japanese information processing industry. One program is for the development of common, hardware independent, software.

Gildersleeve, Thomas R., "Insight and Creativity," Datamation, Vol 22, No. 7 (July 1976) p. 91.

This article discusses the need for and application of both insight and creativity in the definition, design and construction of data processing systems. It identifies and discusses, with real-world examples, five types of activities in the inductive process:

1. Stating the problem
2. Collecting the facts
3. Organizing the facts
4. Developing ideas
5. Testing the ideas.

The article also illustrates the need for an adequately and precisely defined problem, the manner in which problems should be recognized and defined, the process of developing solution ideas, and the concurrent testing process necessary to assure pragmatic solutions to real problems.

Gill, Gerald W., and Alton P. Jensen, Management of Computer Programming, Part I: Practices and Problems (Georgia Institute of Technology, 1969).

This study investigates the management of computer centers, with emphasis on managing the programming effort. Problems and objectives of programming management are examined and techniques used in selected business and governmental organizations are presented. The data were collected by the case study method by surveying seven organizations. The particular aspects of the problem discussed in the report emphasize programming objectives and standards.

Hicks, Harry T., Jr., "ANSI COBOL," Datamation, Vol 16, No. 14 (January 1970), pp 32-36.

This article discussed the levels of ANSI COBOL, compiler validation, and the transition from prestandard COBOL. The arrival of the ANSI COBOL compilers will require many COBOL users to convert their existing programs. The organization can adopt a standard dialect to facilitate programmer training and make available a richer choice of future computers and compilers.

Hill, Majorie F., The World of EDP Standards, Control Data Tech. Memo TM4 (August 1972).

From official yearbooks, operating procedures, directives, and bylaws of the national and international standardization organizations, this report identifies the organizations comprising the standards environment, the processes by which standards are developed, and the operating procedures of many of the national and international organizations. Brief histories, membership qualifications, relationships to other organizations, and technical functions of international, European, national, and other nation's standardization institutions are discussed.

Hopper, Grace Murray, "Standarization of High-level Languages," Proceedings AFIPS (1969), pp 605-612.

This paper contends that the common element essential to mobility is the establishment of standards for programmers, programs, and documentation.

Jones, R., D. Lytle, H. McGeehan, and B. E. Scott, Survey of Computer-Program Documentation Practices at Seven Federal Government Agencies (Computing Technology, Inc., 31 March 1967).

This article argues that guidelines for establishing documentations requirements on a sound, rational basis would be of distinct help to EDP management. Programmers, when properly motivated, recognize the need to document their work, but generally lack an essential ingredient necessary for high quality documentation - the ability to write for the needs of others. The article states that standards for program documentation prevent anarchy in the documentation picture, but must not be too rigid. Documentation problems are inherently fewer in EDP organizations where the design, implementation, and maintenance functions are performed by a single task force.

Liskov, B. H., Guidelines for the Design and Implementation of Reliable Software Systems (Mitre Corp., February 1973).

This document proposes experimental programming and management guidelines governing the production of reliable software systems. These guidelines are to demonstrate the effectiveness of the "constructive" approach to software reliability which seeks to eliminate the sources of errors by making a concern for reliability an integral part of the development process. The programming guidelines are intended to enable programmers to cope with a complex system effectively. The management guidelines describe an organization of personnel intended to enhance the effect of the programming guidelines.

Manual for Data Processing Standards (East-West Gateway Coordinating Council, March 1969).

This manual establishes a set of standard procedures to be used by data processing contractors in the performance of work for the East-West Gateway Coordinating Council. The manual establishes the responsibilities of the council and its contractors, the standards which are to be met, the general system requirements, the techniques of data standardization, and the requirements for documentation of each computer program. The manual is to be used in conjunction with all work performed by data processing contractors.

Marshall, N. H., American National Standards Institute's FORTRAN Standard: Effect Upon Computer Program Exchange (Aerojet Nuclear Co., 1975).

Several of the extensions to FORTRAN included in the X3J3 proposed standard are discussed. Included are the addition of character data type, extensions to the DO loop, and extensions to FORTRAN input/output.

Mathis, N. S. and N. E. Willmorth, Software Milestone Measurement Study (System Development Corp., 7 November 1973).

This study seeks to determine how software development progress and product quality may be measured and what impact variances from required product quality and established schedules and budgets have on project success or failure. Accordingly, a list of candidate milestone products is produced, potential indicators of satisfactory or unsatisfactory progress and quality are identified, the impact of variances on future project performance is predicted, and the criticality of the indicators is estimated. Finally, the ability of currently specified software documentation standards to meet configuration control requirements is evaluated, as is the impact of granting waivers and exceptions to the specified standards.

McQuillin, Richard J., Computer Programs Directory 1974 (Macmillan Publishing Co., 1973).

This is a directory of standardized programs made available to members of computer manufacturer users' groups.

"Miti-Directed Software Cooperation," Datamation, Vol 22, No. 9 (September 1976) p 97.

This article briefly discusses a cooperative effort in Japan between Japanese software companies and the Government "to make it easier, faster, and cheaper to produce applications programs." Over the past 3 years, Japan's software firms joined forces in five groups covering business data processing, management information, scientific and engineering, operations research, and automatic control, to define software modules from previously written applications programs. This year, the Government and industry alliance formed the joint system development corporation to determine how to integrate those modules and to develop a common programming language called CPL.

Ortega, Louis H., Structured Programming Series, Vol VII, Documentation Standards (IBM Federal Systems Division, 21 September 1974).

This final report contains the full study findings for sow task 4.1.7. Included are proposed changes to DOD documentation standards necessary to realize the benefits of structured programming technology as related to documentation. The recommended changes to MIL-STD-483 and DOD 4120.17M constitute the initial step in improving software documentation.

Parnas, D. L., A Technique for Software Module Specification, CACM Vol 15, No. 5 (May 1972).

This paper presents an approach to writing specifications for parts of software systems. The main goal is to provide specifications sufficiently precise and complete that other pieces of software can be written to interact with the piece specified without additional information. The secondary goal is to include in the specification no more information than necessary to meet the first goal. The technique is illustrated by means of a variety of examples from a tutorial system.

Parnas, D. L. and G. Handzel, More on Specification Techniques for Software Modules (Fachbereich Informatik, 1975).

This paper presents modification and extensions of the technique for software module specification reported in the above paper by Parnas. The techniques involve the use of "bags" and "sets" for module specification. It is stated that the use of these techniques avoid over-specification, and avoid the use of "hidden" functions in module specification. Examples are included.

Perstein, Millard H., Some Techniques for Describing Programming Languages (System Development Corp., 4 January 1968).

This paper examines several techniques for describing a programming language, points out desirable qualities in programming language description, and notes some incompatibilities among these qualities. Misconceptions with regard to the appropriate role of compact SYNTAX Metalanguages are pointed out. Reasons are adduced for producing a single definitive document to specify a given programming language for the edification of all programmers skilled in the art. A lean mix of compate SYNTAX meta-languages with natural language is recommended for writing such a document.

Proposed Revision of American Standard COBOL (American National Standards Institute, January 1974).

This document is for review purposes only in anticipation of its becoming an American National Standard and, subsequently, a federal information processing standard. The American National Standard COBOL defines the elements of the COBOL programming language and the rules for their use. The standard is used by implementors writing programs in COBOL. The primary purpose of the standard is to promote a high degree of interchangeability of programs for use on a variety of ADP systems.

Shnders, T. R. B., Ed., The Aims and Principles of Standardization (International Organization for Standardization, October 1972).

This book is a condensation of a number of studies performed by STACO, the ISO standing committee charged with the study of the principles of standardization. Showing the development of standardization from a desirable activity to one that is vital in our society, the book identifies three major factors that have generated a national and international need to standardize: (1) the increasing influence of multi-national companies which sell their products and services on a world-wide market, (2) developing countries, trying to build domestic industry that is both independent and competitive in the world market, and (3) the national and international consumer organizations which demand the best possible value for their money.

Tinanoff, Nathan, Structured Programming Series: Vol II, Precompiler Specifications (IBM Federal Systems Center, 9 May 1975).

This report contains a description of and program specifications for implementation of the structured programming standards in volume I of the structured programming series. Specifications are presented for precompiler for macro processing programs for the ANS COBOL, ANS FORTRAN, JOVIAL J3, JOVIAL J73, and TACPOL compilers and the AN/GYK-12 and IBM S/360 assemblers. The precompiler or macro processing programs are designed to process four structured programming figures (if then else, dowhile, dountil, and case). The include capability, which facilitates top-down programming, is also addressed.

Walsh, Dorothy A., A Guide for Software Documentation (McGraw-Hill, 1969).

A series of outlines or models is given for the standard manual and references written to describe software. The outlines give a basic minimum content list and provide a basic minimum features checklist for the software itself. The text gives directions for adapting an outline for a general document type to the specific software item to be described.

Wells, Mark B., Evolution of Computer Software (Los Alamos Scientific Laboratory of the University of California, February 1971).

The history of programming languages and operating systems for digital computers is traced from about 1950 to the present. The significant contributions of languages such as FORTRAN, ALGOL, and APL are discussed. Operating system development is presented in the light of the growth of important concepts such as batch processing, multiprogramming, and paging. Interactions between software and hardware evolution are pointed out. For instance, the hardware introduction of input-output channels and associated buffering and the software concept of a resident supervisor go hand-in-hand. Finally, an attempt is made to pinpoint current trends of software evolution. Notable in this respect are more automated data management, an attempt to understand the functioning of complex operating systems, and the growing use of interactive and remote computing facilities.

CERL DISTRIBUTION

Chief of Engineers
ATTN: DAEN-ASI-L (2)
ATTN: DAEN-DSE/R. A. McMurrer
ATTN: DAEN-FEB
ATTN: DAEN-FEP
ATTN: DAEN-FEU
ATTN: DAEN-FEZ-A
ATTN: DAEN-MCZ-S
ATTN: DAEN-RDL
Dept of the Army
WASH DC 20314

Chief of Engineers
ATTN: DAEN-PMS
Dept of the Army
WASH DC 20314
for forwarding to:
International Organization
for Standards
Central Secretariat
1, Rue de Varembe, 1211
Geneva, Switzerland

US Army Materiel Development
and Readiness Command
ATTN: DRCDE-DK (3)
5001 Eisenhower Ave
Alexandria, VA 22333

Commander
US Army Electronics Command
ATTN: DRSEL-TL-M/Mr. Tenzer
Fort Monmouth, NJ 07703

DOD ADP Policy Committee
Assistant Secretary of Defense
(Comptroller)
WASH DC 20301

DOD Standardization Area
Computer Aided Design and
Numerical Control
Naval Ship Engineering Center
Hyattsville, MD 20782

DOD Standardization Program
for Information Processing
Standards for Computers (IPSC)
Directorate of Data Automation
(AF/KRAX)
HQ, USAF
WASH DC 20330

American National Standards
Institute
1430 Broadway
New York, NY 10018

ANSI X3 Committee
C/O CBEMA
1828 L Street, NW
WASH DC 20036

DOD Working Group on Computer
Documentation Standards
DOD/DOCN
The Pentagon
WASH DC 20301

DOD Working Groun on Computer-
Generated Military Symbology
DOD/DISPLAY
The Pentagon
WASH DC 20301

Federal Information Processing
Standards Coordinating and
Advisory Committee
Dept of Commerce
WASH DC 20234

Library of Congress
Exchange and Gift Division
ATTN: Federal Documents Section
WASH DC 20540

Interagency Committee on Automatic
Data Processing
National Bureau of Standards
WASH DC 20234

Defense Documentation Center
ATTN: TCA (12)
Cameron Station
Alexandria, VA 22314

Appendix A Organizations (159)

National Bureau of Standards
Institute for Computer Sciences
and Technology
WASH DC 20234

Science Applications
Software engineering / by Science Applications. -- Champaign, Ill. : Construction Engineering Research Laboratory ; Springfield, Va : for sale by National Technical Information Service , 1978.
2v. ; 27 cm. -- (Technical report - Construction Engineering Research Laboratory ; E-126)

Contents: v.1. Its development and standards. -- v.2. Software standards personnel inventory.

1. Computer programs. 2. Computer programs - standards. I. U.S. Construction Engineering Research Laboratory. II. Title. III. Series: U.S. Construction Engineering Research Laboratory. Technical report ; E-126.