Series 2
Box 1
Fl 5

BELLCOMM, INC.

TR-64-222-1

WSU S2B1F5

MANAGEMENT PROCEDURES IN COMPUTER PROGRAMMING FOR APOLLO - INTERIM REPORT

November 30, 1964

W. M. Keese - Bellcomm, Inc.
B. H. Liebowitz - Bellcomm, Inc.
W. J. Martin - Bellcomm, Inc.
I. D. Nehama - Bellcomm, Inc.
A. H. Scheinman - Bellcomm, Inc.
C. S. Sherrerd - Bellcomm, Inc.

W. C. Dennis - Computer Sciences Corporation
S. E. Fliege - Computer Sciences Corporation
D. A. Jackson - Computer Sciences Corporation
B. J. Thielen - Computer Sciences Corporation

Work Performed for Manned Space Flight, National Aeronautics and Space Administration, as part of Task 22 under Contract NASw-417.

BELLCOMM, INC.

TABLE OF CONTENTS

BELLCOMM, INC.

ABSTRACT

This report concludes that computer programming is a describable, orderly process having a determinable, predictable end product, It is susceptible to close management control at the NASA center level for the purpose of delivering usable programs on schedule within estimated costs,

A set of management functions and procedures is proposed, which are submitted at this time as tentative recommendations, and which will ensure control over procurement and production of computer programs for Apollo, It is recommended that:

1, A mechanism be established for the identification of computer programming end items, and the reporting of milestone information be made to specified levels of management.

2, The contract be used as the principal instrument of NASA control of procurement and production of computer programs; standard contractual clauses be developed to assure adherence to delivery costs, documentation and production requirements,

3. The function of quality control of computer programs be implemented at the NASA center level, Groups of information processing specialists be formed at the centers and headquarters to provide a source of programming expertise to management.

4. Managers at all levels of Project Apollo be made aware of the computer programming process so as to better understand the requirements and costs involved,

5. Pertinent statistics related to the production of computer programs be collected and processed to provide criteria for measuring contractor performance and estimating expected costs,,

These proposals will be examined and developed in depth, and final recommendations made in forthcoming reports,

BELLCOMM, INC.

MANAGEMENT PROCEDURES IN COMPUTER PROGRAMMING FOR APOLLO

INTERIM REPORT

1.0 INTRODUCTION

This is the first report on Task 22 undertaken by Bellcomm at the request of the Director of the Apollo Program for the purpose of studying, developing and recommending management procedures in computer programming for Apollo.

The conditions which gave rise to this task can be summarized as follows:

The utilization of computers in the Apollo Program -- considering all the technical uses, i.e., real-time and off-line, to which they will be put -- represents a twenty- to fifty-fold increase compared to the previous manned space flight programs that will have been undertaken by NASA. Experience from similar situations indicates, without exception, that the effects on costs, manpower and management resources associated with the production of the necessary computer programs will be greater by a factor of two to five compared to the corresponding effects associated with computer hardware.

One of the more serious problems created by the drastic increases in computer usage in the Apollo Program is the problem of technical management of computer programming. That NASA is not alone in facing this problem is attested by the vast amount of literature on the subject from sources in government and industry.* The underlying cause seems to be that, while the management capability available in these agencies and industrial organizations is derived from long experience with hardware systems, the experience with computer "software" has been relatively small.

* A bibliography of articles and books discussing problems associated with computer programming and programmers is given at the end of this report.

Several factors contribute in compounding the problems associated with computer programming in Apollo, One factor is the size .of the effort involved. It has been estimated that a total of more than five million machine language instructions will have to be produced, Another factor is that responsibility for computer programs in Apollo is spread over four NASA Space Flight Centers: GSFC, KSC, MSC and MSFC, Eighteen operational* programming systems associated with these Centers have been identified (see Table 1). These programs do not constitute complete systems by themselves, but are intended to perform certain functions in conjunction with other systems for the accomplishment of a mission. The specification of requirements for these programs, perhaps the most. important initial step in the process, is the responsibility o.f the "ultimate user" -- in general, not an information processing specialist -- in the Center.

It is the thesis of this report that the problems of technical management of computer programming can be solved, and that working, well-documented programs can be delivered on time within estimated costs; that management tools useful at the project level in the Centers can be developed and implemented to insure control over production of computer programs; and that information reporting schemes can be established so that management at all levels in NASA can be made aware of, and reac't to problems related to computer programming,

The report includes a broad discussion of the major ordered activities and elements of the computer programming process involved in a large-scale programming effort, such as will be required in Apollo,' From this broad treatment, a number of tentative conclusions are drawn concerning the type of management procedures needed, and means of implementing them in the NASA environment. The concept of quality control in programming is also introduced, Some important management functions are described in terms of a hypothetical organization structure for purposes of illustration only, and are not intended to be final recommendations at this time,

The tentative conclusions presented in this report will be examined in detail in the continuing effort on this task, and recommendations will be made in a series of forthcoming reports,

---

* For definitions of some of the special terms used in this report, see the GLOSSARY,

2.0 PROBLEMS IN DEVELOPING COMPUTER PROGRAMS

In this section, five basic problems of program development, typical of all large programming efforts, are discussed in re= lation to the symptoms 'by which they manifest themselves, Many symptoms can be caused by more than one of the basic problems; there is not necessarily a one-for-one correspondence between basic problems, symptoms, and solutions,

2.1 Lack of Definitive Requirements

A critical problem which has been found in the preparation of computer programs is' the lack of definitive requirements in the procurement of computer programming systems, This results in poor performance and high cost, and in general makes it very difficult to obtain or give satisfactory performance, Some symptoms that arise due to the lack of definitive requirements are: not knowing the desired form or content of the final deliverable product; the Inability of the contractor to main- tain and in many cases to develop a realistic physical and fiscal schedule; slipped schedules and overruns; the failure of the final system to function properly or even at all; the lack of control over the contractor by the contracting agency; the need to purchase additional equipment at the last minute due to an underestimation of the size of the final computer program; and a misinterpretation of poorly defined requirements resulting in a computer system that successfully solves the wrong problem,

2.2 Complex Interface of Elements

A critical problem, especially in real time systems, is the complex interface of the various elements making up the total system, These elements are personnel, computer programs, com- puters, storage devices, communications devices, output displays, and input devices, If these interfaces are not or cannot be adequately described, such problems arise as: an incomplete or Inoperable final system; the inability to develop or main- tain a realistic program implementation schedule; bick'ering between the contractors as to their responsibilities; diffi- culties in calculating economic tradeoffs between elements when changes become necessary; redundant effort among con- tractors; completely missing elements; and lack of complete or proper acceptance tests for checking the total system of integrated elements.

2.3 Inadequate .Quality

Another major problem that sometimes occurs, even if the requirements and element interfaces are well defined, is inadequate quality of the product, The quality of a computer program can be measured by its ability to: solve the problem in a reasonable time; be understood; be changed; and be maintained. This lack of quality manifests itself as: programs that fail to operate properly due to inadequate programming and checkout techniques; poor documentation; changes that are difficult and costly to make; deliveries that are unspecified and hence difficult to effect; and maintenance that is difficult.

2.4 Inadequate Understanding of the Programming Processes

One of the basic problems is the lack of understanding of the programming processes by the customer, contractor personnel, and the programming profession, This lack of' understanding results in: inadequate definition of programming tasks, resulting in missed schedules and excessive implementation costs; systems that do not make optimum use of the computer; computer programming activities that are on the schedule's most critical paths; and the design of inflexible programs which cannot benefit from a changing technology, resulting in the inability to take advantage of major potential cost savings.

2.5 Inadequate Understanding of the Cost Elements

Another general problem with respect to overall costs is the lack of understanding of the cost; elements in computer program development by customer, contractor personnel, and the programming profession. This results in: excessive costs, by not using desirable programming techniques; procurement problems in writing contracts; extended schedules; and underestimation of costs by not recognizing all the items that go into making up a completed, checked-out computer program,

3.0 MANAGEMENT TOOLS FOR COMPUTER PROGRAMMING

This section describes several management tools that can be used to control the programming process so as to avoid the problems discussed in section 2.0.

3.1 Description of the Computer Programming Process

To achieve successful implementation of computer programs, a thorough understanding of the computer programming process is essential. This does not necessarily have to be an understanding of the details of the process but rather one of its general nature and how all system elements interact with each other,, To help describe this process, a detailed Computer Program Production Activity Chart has been developed which shows the phases, major milestones, and activities of problem analysis, design, implementation, and test. The preliminary chart is shown in figure 1 and discussed in section 3.1.4, and can be used as a planning tool during the early phases of problem planning and analysis. Although major portions of this chart hold true for the design of any computer program, it is oriented toward the large process control activity such as found in Project Apollo. The remainder of this section deals with some general Ideas with respect to the computer programming process.

3.1.1 Understanding the Process

In order to understand the computer programming process it is necessary to dispel some basic misunderstandings about its nature, First of all, it is neither mysterious nor "artistic". Rather it is describable, orderly, and predictable and therefore susceptible to close management control, Second, it consists of elements which have a close correspondence to the processes involved in hardware development, although there arc several areas where there are marked differences which the manager must understand. Third, a program once written is never general purpose, and the computer, while under the direction of that program, is itself not general purpose, It is performing a detailed, thought out, and fully specified function. Neither is the program completely flexible once it is written; changes to the program may require as much time and expenditure as changes to hardware elements. Each change must be evaluated individually,, with a careful "engineering" analysis based on a thorough knowledge of the software and associated hardware systems in order to generate meaningful cost and' schedule estimates.

Of all the points concerned with computer programming, the limitations of flexibility are the most critical and least generally understood aspect of the process, The common practice is to treat programming as an adjunct to system development that can be called in from time to time, after a problem has developed, in hopes of a quick cure. The lead times required in such cases are often sufficiently large to render computer programming the pacing item. With the present day understanding of the programming process, computer programming should only rarely be on the critical path on any project, When it is, the most likely reason is the lack of participation of information processing personnel during the project definition phase or the isolation of information processing personnel from daily participation in overall project technical and management problem solving.

Finally, a number of important steps in the computer programming process should be accomplished prior to the writing of any code; an example is the generation of design specifications, or even the selection of the computer itself. In fact, there is much to be gained by this early participation, in that problems can be identified sooner and common programming requirements can be recognized and planned for, thus saving in overall time and cost,

3.1.2 The Elements of a Programming System

There are four basic, interrelated elements of every programming system: a utility system, a support system, the operational system, and a data base. Traditionally, the user of a computer has been interested in only the latter two, The operational system performs the job that the user wants done, as for example the generation of a payroll, the calculation of a trajectory, or the vectoring of an aircraft. The data base contains that static or dynamic information supplied to the program, such as individual salary data and income tax exemptions, launch site coordinates, or aircraft performance parameters, However, the utility and support systems may represent as great an investment as the operational system and should receive as much attention by the customer as the operational system, Figure 1 shows that the analysis, design, and production of the utility and support programs are separate and distinct processes from the operational programs, It also shows how testing is tied into each of the systems and how the three systems are interrelated during the implementation phase.

The utility system includes all those tools necessary to generate operational and support programs. These are compilers, assemblers, tape loading and maintenance routines, sub-program test and debugging tools, and program update tools, If, these can be standardized for use across many operational programs, smaller costs would acorue to each operational program, While certain changes may be required for each operational program, the' potential for significant dollar savings by standardization is extremely high, a fact which is often overlooked by inexperienced personnel.

Support programs include simulation, data recording, data reduction and analysis, and adaptation manipulation programs. These programs are required as aids in the production of the operational computer program whenever this program involves large volumes of inputs and outputs or where there is significant interaction be= tween personnel in the system, the computer, and other elements of the physical environment, One of the basic attributes of a computer program in a command and control system is its capability for handling large volumes of data which occur in a dynamically changing environment. While it is possible to hand-generate data for testing small pieces of the program system, it becomes impossible to hand-generate the large volumes of data which are necessary to check out the critical features of the operational program system, A data generation capability in the support program system is a technique by which a computer can be used to generate required checkout data, However, the checkout of operational programs is only one of the requirements for easily specifiable and producible volumes of data. Additional requirements are levied by the system integration engineer', who is responsible for effecting interface testing and the interconnection of physical components into an operating configuration. In testing an output communication line, for example, the capability will be needed for generating all the classes of information at the rate and frequency with which they must be transmitted, Quite often the only reasonable approaah is to use computer-generated data,

Another requirement for computer-generated simulation data is system exercising for training and evaluation purposes. In order for training and evaluation to be effective, the exact characteristics of the system inputs must be known so'they can be used for the definition of the conditions of system operation, both initially and during the complete time span of the system exercise. It must be possible to control these inputs so that certain aspects of system operation are stressed, either for the purpose of training or evaluation, In order to evaluate the system or to provide feedback on per- formance, certain aspects of system operation must be recorded, Recording requirements vary, depending upon the use of the data to be recorded. Different levels of detail, for example, are re= quired for program or equipment debugging and for performance evaluation or debriefing, Therefore, a flexible recording capa- bility must be provided which covers the total range of require- ments for recording,

Closely allied with.the recording function is the requirement for data reduction and analysis. Computers have the capability for performing operations in microseconds; they can generate vast quantities of data which must be formatted and aggregated in some fashion to be comprehensible, These data reduction and analysis requirements cover a number of areas in the system from the computer program test activity clear through the evaluation of problems or deficiencies in the operational system,

If all requirements for data generation, recording, reduction and analysis are clearly specified at the beginning of a project, a program system, which will satisfy the needs of all interested parties, can be produced by a single group. This will result in significant savings of cost over and above that if each individual party generated his own system. Furthermore, problems of control and interfaces with the operational program system and with the hardware will be reduced, which will significantly alleviate the problem of management control of computer programming,

The adaptation program of the support system manipulates input data supplied by the user into a form acceptable to an opera- tional system.. Quite often this program is composed of trans- formation and formatting routines which may be used in a number of different operational systems. If a library of these. routines is maintained, it may be possible to realize cost and schedule improvements by making these routines available to any interested user,

The data base element is the substantive information (data) and documentation necessary for system operation, The purpose of the data base is to assure an orderly and effective development of the system by providing an up-to-date body of authoritative informa- tion. The documentation encompasses all system requirements, performance specifications, and design documents, which can be maintained by configuration management procedures, The raw in- formation data is often prepared by the support system to produce inputs compatible with the data base structure. This structure is organized to facilitate interprogram and man/machine communications.

3.1.3 Types of Computer Programming Projects

Computer programs can be categorized in terms of: scientific, administrative, and process control types, Scientific computer programming is used to support scientific or engineering projects where some type of numerical analysis or other mathematics is required. Programs written for this purpose may be, needed for a single, one-time use or may be used repetitively, If they are for one-time use, the most direct programming approach should be used to minimize the time.of the programmer. If the program is to be used more than just a few times, the question of efficiency of operation should be raised because there is an economic break-even point in the tradeoff between, man hours to write the program and computer time needed to run it. A tightly written program designed for frequent use should be well documented and stored in a library where it can be maintained and easily retrieved for possible use, Traditionally, scientific programs tend to be mathematical in nature and require a utility system based on a language such as the Formula Translator (FORTRAN) or the International Algorithmic Language (ALGOL), which can easily express mathematical relations.

In administrative data processing, the focus is on clerical rather than mathematical functions. File searching and updating, inventory record keeping, and payroll computing are typical functions performed. Once again, the language required for such programming, and its associated utility system, must be amenable to expressing these clerical functions. An example of such a language is the Common Business Oriented Language (COBOL).

Process control computer programs are those which must sense and react to environmental stimuli. Such programs are sometimes called real-time programs because they must sense and respond to environmental changes as they occur. Command and control systems such as the Semi-Automatic Ground Environment System (SAGE), the proposed FAA National Aerospace System (NAS) and the Real-Time Computing Complex (RTCC) are examples of such systems. They are characterized by a combination of mathematical and administrative data processing under the control of an executive system which monitors time and occurrences in the environment and calls up the proper program for each set of environmental conditions. Of the three types of computer programs, process control programs are the most complex and costly to produce. Furthermore, the time required for an adequate system analysis is usually significant, while the lack of such analysis is usually costly if not disastrous to system implementation, The language necessary for process control programming must contain both mathematical and administrative capability as well as the ability for the types of logical expressions necessary for computer program decision making, An example of such a language is the Naval Electronic Laboratory International Algebraic Compiler (NELIAC),

3.1.4 The Phases of Problem Solutio?

Computer programming problem definition and implementation for large-scale real-time systems can generally be categorized into six phases, which are detailed graphically in the Computer Program Production Activity Chart (Figure 1, sheets 1 through 4):

1. Definition of System Requirements

During this phase, the general performance criteria are established; the system functions and subfunctions and those which are critical are identified; critical engineering components are identified; and the system requirement specifications are produced, Figure 1, sheet 1, depicts these several activities.

2. ,Definition of Performance Specifications

During this phase, functional analysis and design is performed; equipment and facility capabilities, system timing requirements, and utility and support system requirements are established; and the performance specifications documents are produced. These activities are depicted in figure 1, sheet 2,

3. Development of Program Design

During this phase, the individual programs are designed to a level which will permit detailed flow-diagramming, and the program design documents are produced. Figure 1, sheet 3, shows the interrelationships of these activities.

4. Program Implementation

The implementation phase consists of the activities that make up the production and subsystem testing of computer programs, It Includes such steps as: detailed analysis and design of individual operational, utility and support computer programs; coding, assembly, test and documentation of individual computer programs; the production and test of operational position handbooks; the development of the data base; the checkout of' computer program subsystems; the simulated environment tests; and the acceptance tests of the utility, operational and support computer programs. Figure 1, sheet 3, illustrates how the several activities of this phase are interdependent and related to the outputs of the program design development phase,

5. System Integration

The system integration phase involves the integration of hardware, software, and operating personnel in the real operating environment and the checking out of the total system. During this phase, the software contractor will provide support and will be responsible for program corrections or modifications, maintenance of the baseline documentation, and the updating of all documents, flow-diagrams, etc. not included in the baseline. These activities are shown in figure 1, sheet 4.

6, Program Maintenance

This phase, sketched in figure 1, sheet 4, involves the maintenance of the computer program system after delivery to the customer.

Adequate time and effort must be spent in each phase so that the information and documents produced by that phase are complete. If this is not done, each succeeding phase will suffer from insuf" ficicnt information and poor planning, The major milestone documents which are the inputs and outputs of each phase are shown in figure 1, The Identification of additional milestones associated with each phase is being continued, This information will be included in the final report.

3.2 Policies, Procedures, and Standards

This section discusses some policies and procedures, derived from an awareness of the programming process, which could prevent or reduce the problems associated with computer programming as outlined in section 2.0. These policies and procedures cover the planning, production, and documentation of computer programs from both the customer and contractor points of view.

3.2.1 Planning

The following areas should receive careful attention during the planning phase of a computer programming task:

1. Role of the Information Processing Specialist in Planning

The participation of information processing specialists during the planning phase of a task leads to a good technical product when adequate emphasis is placed on the software at an early stage, thus helping to eliminate software as the pacing item. The information processing specialist should be brought into a project during the early phases of the definition of system requirements, Specific recommendations for his responsibilities will be defined in the final report,

The general responsibilities of the information processing specialists should include: the identification of software-hardware interfaces; the initial portions of the software system baseline documentation; the initial software cost estimate and delivery schedule; the investigation and recommendation of computer hardware requirements; the selection of software contract clauses; the identification of software-hardware and software-software contractor liaison; program language selection for the production of the software system; identification of the system simulation and data reduction requirements; selection of the procedures and standards for inclusion in the software contract; identification of the software system requirements; and integration of the software design into the overall task plan,

2, The Software System Baseline

The software system baseline is composed of documents which define the requirements levied upon the operational software system, Baselines may be established for all software systems, including the utility and support systems, The baseline documentation provides the customer as well as the contractor with the criteria for evaluating the progress of the software system during its various phases of development, As the software system progresses, additional and more detailed documents will be added to the baseline.

Usually the Initial software system baseline is composed of the preliminary software systems requirements specification. This document contains a general prose description of the software system; the various hardware to be employed; which major functions are to be implemented by hardware, which by software,, dnd which by manual activities; and any restrictions and limitations of the system.

From the systems requirements document the initial software performance specifications documents are derived, These documents represent a logical grouping of the functions to be performed by the system into those areas similar in nature or purpose, and is the basis for the program design specification documents. They are prepared by the information processing specialist responsible for the analysis of the operational software system,

The production and maintenance of the baseline documents cannot be over-emphasized since they will be reflected in the operational software system, Also they provide management with control over the programming process, Specific examples will be developed of documents that are to be produced in each phase of the problem solution along with an outline of the contents of these documents, These documents will constitute the baseline.

3. Planning, Scheduling, and Cost Estimating

Throughout the development of a software system, tools which assist in planning, scheduling, and cost estimating can be very useful and are of prime importance, Guidelines for the formulation of such tools are being developed, These tools include general program flow charts, checklists, and implementation procedures and standards, Sample checklists are included in the Appendix.

The cost estimation process can be improved by the identification of key cost elements in the programming process. Data derived from past programming efforts can serve as the basis for cost prediction in present or future projects. Guidelines for cost estimation should be made available to all appropriate NASA personnel and to contractors as needed. To develop these guidelines pertinent statistics related to the programming process will be identified and data will be collected. Preliminary investigation In this area indicates the following statistics would be useful in the cost estimation process': a distribution of dollars spent per working instruction produced; the cost per instruction changed for various. types of programming efforts; debug runs per program checkout; hours of computer time utilized for checkout per working instruction; total costs of' various programming efforts; pages of documentation per number of instructions; number of working instructions written per man month; total time debugging vs. time spent coding for different size programming efforts, Although the use of all statistics gathered would have to be tempered by the realities of the particular programming jobs from which they were derived, guidelines can be developed to help the program manager augment his intuition during the cost estimating process,

The complexities and dependencies of a large software system require an effective scheduling and monitoring tool. Several tools such as the Program Evaluation and Review Technique (PERT) and configuration management which have been used successfully in large hardware projects are being evaluated and certain of these will be recommended for all software development, As of now, PERT has shown several distinct advantages, specifically its ability to reflect the complex interactions that occur in the large systems. The implementation phase of the Computer Program Production Activity Chart in figure 1 is an example of the interactions that occur during the programming activity. PERT can, and has, been successfully used to plan and monitor this type of activity, Application of this technique to computer program design and implementation will be explored further,

3.2.2 Development of Program Production Procedures and Standards

It is extremely important that procedures and standards be developed for the implementation phase of a software project. This is the production phase of the project where code is produced and verified. It is important that the code be well documented, well checked out, and protected against uncontrolled change.

1. Programming Procedures and Standards

The application of procedures and standards during the entire programming phase assures the customer that the contract requirements arc being fulfilled. These procedures and standards provide a means to achieve uniformity within the system, facilitate status/progress reporting, increase scheduling accuracy, and allow easier interchange of' personnel. The types of procedures and standards that should be developed for the NASA environment arc being studied along with actual procedures and standards that have been used successfully in software development. Procedures that will be established to guide the programming during the various aspects of the work pertain,to:program design;,flow-diagramming3 coding; desk checking; program assembly and modifications; test data preparation, checkout, and testing; and documentation.

~~A. Program Design~~

Experience in computer program development consistently points to strong advantages of procedures concerning modularity in program design. This means that large computer programs are made up 'of small, functional modules or subprograms, each of which represents a distinct logic entity. Each subprogram should have a single, or at most two or three closely related entry points, There is no set upper limit on the size of such modules, although subprograms larger in size than two or three-hundred source program statements should be avoided unless specifically Justified by the logic structure of the program, The requirements on program modularity are that each subprogram must: represent a distinct functional entity in the program logic; *be* readily comprehensible as a unit to other coding personnel; have all interfaces with other subprograms minimal and explicitly identified; be compilable as a separate entity; be amendable to modification without necessitating associated changes in other subprograms unless it is the interface with another subprogram which is being modified; be separately documented; and permit sequential consolidation of groups of subprograms into working program packages, The advantages of program design modularity are that it: enables several programmers to work simultaneously in development of large programs, each subprogram being coded by one programmer; makes possible orderly consolidation testing of large programs; and eases modifications of large programs, particularly when management configuration control procedures are established in conjunction with program modularity,

Another area of program design, closely related to modularity but also intrinsic to program costing, in which procedures are appropriate, concerns an economic balance being achieved between computer operating time, program storage requirements (total program and data structures size), and program implementation (personnel) costs, Experience has indicated that except in rare cases, program implementation costs are the most important and computer operating time the least important of these three considerations. Only in cases of very often executed subprograms or unusual real-time situations is the program implementation cost of tricky, optimal (operating time) coding justified; in all other cases, such tricky coding is to be avoided and straightforward coding structures be used, And only in cases where many nearly identical computing systems are to be purchased, or where the total size and/or weight of the computer is critical, is the program implementation cost of tricky, optimal (program storage usage) coding justified; in other cases the hardware cost of additional program store capacity is usually less than the additional program implementation costs which would otherwise be required, If these considerations are ignored, savings in cost resulting from coding optimization is very apt to be much smaller than the increased coding costs of such optimization and hence not justifiable,

B. Flow Diagramming

The procedures for flow-diagramming should define the amount of detail within each level, segmentation of logic, and standards on a subprogram basis. Flow-diagramming standards include explanations of: level of detail, with examples, for the overall flow diagram which charts the major program areas and inter-subprogram relationships; level of detail, with examples, of the flow diagrams which expand each major program area into sufficient detail to allow coding; identification and control information; paper size, margins, and page numbering; rules for number of blocks per page, number of pages per program, and range for number of instructions per page; alphanumeric labeling scheme for each block/page/program; and rules for further elaboration of formulas and unusual logic.

C. Program Coding

Program coding consists of the translation of the flow-diagram(s) into a coding language. Coding procedures and standards should emphasize good coding techniques, listing formats, and documentation. In addition to the assembler/compiler restrictions, coding standards include: program and page identification schemes; machine setup explanations in the listing documentation; rules for subprogram calling sequences and nesting hierarchies; formats of buffers/tables/temporary storage; assignment of program and data item symbology; use of good programming techniques (i.e., never assuming the status of the required initial environment, resetting loops prior to iterations, and avoiding tricky coding); and providing adequate comments in the listing on lines of coding for each flow-diagram area, for calling sequences, and for any unusual communications required.

D. Program Desk Checking

Desk checking is a manual review of program logic with iterations through the program using data samples. Procedures and standards for program desk checking should include: checks for clerical errors, missing elements, and legibility; iterations of a minimum number of data samples through the program; and reviews by supervisor/programmer.

E. Program Assembly and Modification

Procedures for program assembly and modification supply rules which pertain to: program re-assembly (compilation) versus modification (patches) during various stages of checkout and testing; correction card format; and control of system materials,

Program assembly and modification standards include: rules to determine a re-assembly or patch utilization; specification of repository for current master listings, copies of correctors, tapes, etc.; delineation of correction card format; and machine setup and operation instructions,

F. Program Testing

Procedures and standards for program testing should include: delineation of the several stages of program testing and the requirements for each; data selection, testing, and re-testing techniques to uncover and correct program/logic errors; design and production of data inputs; test deck structure; machine setup and operating instructions; preparation of test plans and test results; and other documentation if required,

G. Documenting

Procedures and standards for software documentation are detailed in section 3.2.3.

2. Configuration Management

It is important to have a formal set of procedures by which a uniform system of configuration identification, control, and accounting is established and maintained, The following section defines some of the 'terms of configurations management and outlines techniques that have been used for the control of large software systems.

Configuration management is composed of three major functions:. configuration identification is the definition of the contract and functional end items which comprise the baseline for subsequent configuration control and accounting; configuration control is the process by which proposed changes to the established baseline are formally evaluated, coordinated, and approved or disapproved; and configuration accounting is the process of reporting and documenting changes made to contract end items in order to maintain a current status of the established baseline,

Configuration management procedures will maintain effective control over changes to software systems and their elements, These procedures will .also insure that all proposed changes to the software system will be reviewed to determine any interfaces or systems impact. Decisions concerning proposed changes will be disseminated to all agencies affected by each change in a manner to permit timely implementation, Subsequent to approval of a baseline change, the change will be incorporated in revised baseline documentation and the implementation status of the change will be monitored via configuration accounting procedures.

There are several possible reasons for originating computer program change proposals, such as system requirements changes, system hardware changes, program errors or design flaws uncovered during testing, and updating *of* documents which have become inaccurate or obsolete,

It is the responsibility of the contractor, in cooperation with associate contractors, to insure that every deliverable item of the contract for which the contractor has responsibility is accurately and completely described in the baseline documentation and further, that the status of each item be readily determined *at* any time, Other specific responsibilities in the areas of configuration identification, control, and accounting should be:

A, <u>Configuration Identification</u>

(1) Maintain configuration identification documentation to reflect the status of all contract end items,

(2) Deliver *periodically* to the customer a listing which reflects the approved configuration of all items for which a baseline has been established,

B. <u>Configuration Control</u>

(1) Prepare for customer consideration computer program change recommendations (CPCR) for any item on which a change is deemed necessary, The CPCR is used to recommend to the customer changes beyond the scope of the baseline specifications.

(2) Prepare for customer consideration computer program change proposal (CPCP) forms on all changes related to the computer program baseline documentation for which the contractor has responsibility, The CPCP contains more detailed information on the change than the CPCR and *can* result from a CPCR,

(3) Insure that prior to the submission of a CPCP the implications of the change have been explored with associate contractors to the fullest extent possible.

(4) Provide coordination and assistance in response to requests for review of engineering change proposals (ECP) contemplated by other agencies or contractors.

(5) Review. ECPs and submit comments if the proposed changes will have an impact upon the contractor's responsibilities,

(6) Prepare for customer consideration a CPCP for all CPCRs.

C. Configuration Accounting

(1) The contractor should maintain and deliver periodically to the customer a status report of computer program change recommendation and computer change proposals showing status including implementation of all CPCRs ,and CPCPs submitted by the contractor,

(2) The customer has the responsibility to furnish to the contractor the initial documentation which constitutes the baseline, The customer should establish and maintain a change control board (chaired by the customer) for evaluating CPCPs, CPCRs, and contractor configuration accounting reports,

3.2.3 Documentation Requirements

The documentation requirements of computer programming are closely related to the very nature of the process itself and are both varied and specific in objectives and type. To assure that the documentation requirements are met dynamically during the several phases of the programming process discussed in section 3.1.4, documentation procedures and standards are necessary, This section suggests documents to be covered by such procedures,

1. Documentation Related to Program Requirements

A. Definition of Software Requirements

In the first phase of computer program definition and implementation, several documents will be produced with the assistance of an information processing specialist, The preliminary software system requirements specification discussed in section 3.1.4 is the major milestone document of this phase,

B. Software Performance Specifications

The initial software performance specifications is the major milestone document of the second phase of program definition and implementation discussed in section 3.1.4.

C. Development of Program Design

Documents to be produced and added to the software baseline documentation during the software design phase include the following, as appropriate, for each of the utility, operational, and support program systems:

(1) Computer Program System Design Specifications

This document describes the proposed program system design, It should be subject to approval before a major portion of the program development process is begun, This proposed program design specification should contain sections on: operation requirements, environment requirements, program design modularity, the assembly system, data base, man-machine interfaces, operating procedures, automatic maintenance, the utility system, consolidation testing, adaptation procedures and acceptance and integration test responsibilities,

(2) Basic Programming Language Manual

In almost all computer systems, an assembler is needed to enable machine translation of coding from a symbolic form (called "source language") used by programmers into the absolute numeric form used by the machine, If such an assembler does not already exist for a computer to be used, one must be developed, prior to coding the operational program, as part of basic programming tools. Such an assembler should preferably involve a machine-language translator with macro capability and a data structures assembly feature. A basic programming language manual should then be prepared describing the assembler's Input language, including appropriate general purpose problem-oriented programming language statements as well as the basic machine order structure, If more than one computer be involved, say for the utility *system* and support programs, separate basic programming language manuals should be provided for each computer.

(3) Machine-Program Interfaces Specifications

In addition to the computer instruction repertoire, the basic information required for program development includes complete, detailed specifications of those groups of instructions, and all constraints thereon, necessary for program control of all peripheral equipment, input-output equipment, displays, keys (control push-buttons), etc,

(4) Man-Machine Interfaces Specification

A complete, detailed specification of all man-machine interfaces is necessary in order that all input and output processing functions of the operational program can be properly implemented, The man-machine interfaces specification should specify all manual control functions, input keying functions, type-ins, print-outs, displays, and external communications from machine-to-man, to be provided for by the operational program, Details concerning the coding of these functions and standard programming techniques to be followed shall be included in this document,

2, Documentation Related to Program Implementation

A, Individual Subprograms Coding Descriptions

The underlying philosophy of computer program documentation requirements is that a computer program is in essence not a deck of punched cards nor a magnetic tape but a set of specific documents, Specific manifestations of a program in machine-readable form (cards or tapes) follow as a natural consequence of this documentation, not vice versa,

Documentation is required of each subprogram for five purposes: to specify the end product (the subprogram itself); to provide a suitable basis for program consolidation, integration, and system acceptance testings; to provide a suitable basis for logic evaluation of the program; to provide a suitable basis for program maintenance and modifications; and provide instructions for proper use or operation of the subprogram,

The documentation required of each subprogram should be in three forms: coding specifications, flow charts, and subprogram listings. Procedures and standards coding specifications and flow charts will be described more completely in the final report,

B. Self-sufficiency of the Program Listing

The concept of self-sufficiency of the program listing discussed in this paragraph was developed primarily by Dr. William H, Wattenburg of the University of California (Berkeley) under the title "Single Document Method of Program Documentation", This technique has been used successfully at the University of California; its applicability to Project Apollo computer programming is under investigation, The compiler-produced listing of each program module (subprogram) is the most fundamental and important of all computer program documents, It is the most current and detailed description of the program as it actually exists, regardless of any errors or oversights which might be the case in the subprogram design. It represents the actual subprogram with an accuracy far more dependable than that of any human-produced documentation; its accuracy is limited only by the machine and compiler' reliability,

As a document, the compiler-produced listing is directed to coding-oriented personnel: the subprogram coders; consolidation, acceptance, and integration test teams personnel; and those responsible for program maintenance and modification, Experience has indicated that in spite of closely controlled program library procedures, the listing often becomes physically separated from the other forms of documentation (design specification, flow charts, and data base descriptions) of the subprograms, and that it Is often the only immediately available subprogram description from which test team and program maintenance personnel can work, For these reasons, it is essential that the compiler-produced listing of each subprogram be a self" sufficient document for coding-oriented personnel, and that it at all times be kept updated whenever program modifications or "patches" be made, When, at the end of program implementation and checkout activities, the program baseline documentation is updated, the computer-produced listings of each subprogram serves as the only reliable basis for validating such updating.

These considerations ,place the following requirements on generation and maintenance of the computer-produced listing of each subprogram:

(1) It must contain, in the form of a series of many comment cards, a complete description, in text form, of the system context and overall logic structure of the subprogram., such text specifying the identification and coder of the subprogram, the role of the subprogram in the overall program, the functions of the subprogram, the subprogram's operating environment (inputs, outputs, data structures, and interfaces with other subprograms and with hardware), the overall logic structure of the subprogram, and all design and coding Idiosyncrasies of the subprogram.

(2) Sufficient comments must be made throughout the subprogram coding to relate the coding to the overall logic structure.

(3) It must be constantly updated, both in coding and commentary, whenever changes or patches are made to the subprogram, This last requirement necessitates recompilation as the chief means of Subprogram modification, that temporary patches to the program be made only in emergency situations, and that such patches never be allowed to accumulate without corresppnding source language modifications and recompilations' to generate current, updated versions of the compiler-produced listing,

C. Subprogram Interrelationships Descriptions

In addition to the individual program coding description documents, several documents describing the subprogram Interrelationships and coding procedures are appropriate outputs of program implementation activities. These are a, specialized programming language manual, a. utility syatem specification, a support system specification, a program consolidation tests specification, a data base structures specification, acceptance and integration tests specifications, and a system user's manual, These documents are detailed expansions of' corresponding sections of the computer program system design specifications documents, but are written during program implementation and describe coding details rather than general design proposals.

D. Progress Reports

Progress reports are extremely important during the implementation of a project with many complex interfaces, Procedures and standards for the form and content of progress reports that have proven successful in large software projects 'are being studied. A recommendation will be made in the final report. In general, the progress report should contain a detailed outline of the,.. work accomplished since the previous report and'should also contain a prognosis of the work that is expected to be accomplished during the next report period. The progress report should include a description of any problem areas that have been detected as well as recommended solutions, The progress report should be prepared by the contractor and submitted to the customer on a periodic basis.

3.2.4 Software Testing

Several stages of software testing are involved in the computer programming process: subprogram unit testing; subprogram package (,consolidation) testing; program acceptance testing; software-hardware integration testing; simulated system environment testing; and live system environment..' testing. Procedures and standards for each of these testing stages will be discussed in the final report.

From the point of view of the customer-contractor relationship, the moot important of these testing stages is program acceptance testing, The acceptance test verifies to the customer that the program requirements have been satisfied and provides a test vehicle for the contractor. In preparation for the acceptance test, a test plan document is required which outlines the function to be tested along with the test content (inputs, expected outputs, etc,), and any special programs necessary for developing and performing the acceptance test, Acceptance testing can be divided into *a* number of tests which provide progress status at specified milestones and allows testing at the program, subsystem, and system levels, In any event, a final system test is necessary for formal, sign-off acceptance,

The detailed preparation of the tests requires the availability of these documents: preliminary software system requirements specifications, initial software performance specifications, program design specifications, acceptance test plan, and system user's manuals, Areas to be investigated include: determination if the customer, the software contractor, or some separate contractor or agency should prepare the tests for the support, utility, and operational programs; determination if the test preparation agency requires special simulation, data recording, and data reduction programs to generate and evaluate the test data results; determination of sufficient information to begin and complete preparation; determination of level and degree of customer/contractor concurrence; determination of the degree of testing required to demonstrate system capacity (overload), timing thresholds ,(resulting degradation), and the number and kinds of illegal user requests; and delegation of the analysis and evaluation to determine acceptance,

The acceptance test plan should be routed in draft form to customer/contractor for possible modifications and finally for a concurrence sign-off of the plan. Sections to be included in the test plan are: scope, objective of the test; interface with total acceptance testing; functions (from the performance specifications) to be demonstrated; delineation of inputs; expected outputs; checklist and actual outputs (possibly with tolerances) to provide a scoring (rating), and recommendations/conclusions to determine retesting or acceptance,

3.3. Writing of Contracts

The software contract should be used by NASA as the basic vehicle to specify and enforce policies, procedures and standards. Since the inputs, outputs and activities of the various phases of the program production process are identifiable, each phase can be contracted separately or a contract can be let for the entire process. Attention should be given to the development of checkpoints where contract termination can conveniently occur if unsatisfactory work is being done by the contractor.

3.3.1. Contract .Clauses

The selection of contract clauses will vary with the type of contract; cost plus fixed fee (CPFF), cost plus incentive. fee (CPIF), fixed price (FP), and time and materials (T&M). Areas of contract clauses under investigation include:

1, Customer Furnished Items

This clause should define all items other than documentation which the customer will furnish the contractor, Such items may include office space, office supplies, computer time, EAM support, etc.

2, Baseline Documentation

This clause should define by document number the documents which constitute the software system baseline, as produced by the customer.

3. Implementation Procedures and Standards

This clause should specify the procedures and standards, as discussed in section 3.2, to which the contractor will adhere during the implementation phase,

4. Planning, Scheduling and Progress Reporting

This clause should require that a detailed plan and schedule be delivered by the contractor to the customer at a specified date early in the contract,, Any nredetermined milestones and associated schedules required by the customer should be indicated, Any specific planning and scheduling tools required (such as PERT) should be indicated, A progress reporting technique along with a reporting period should be specified, Progress reports re= quirements and procedures are discussed in section 3.2.3. Techniques for modifying the master plan and schedule should also be shown.

5. Deliverable Items

This clause should specify the items which the contractor must deliver to the customer. Two levels of' deliverable items should be specified: those items associated with the operational, sup- port and utility systems which must conform to the standards sup- plied by the customer; and those programs that are incidental to development of the primary software system, and require a dif- ferent type and level of documentation,

6, Contract Reviews

The clause should specify that a review of the contractor's per- formance will occur at the end of each checkpoint of the Soft- ware system development and that continuation of the contract may depend on contractor performance.

Checkpoints for contract reviews are being investigated, and a recommendation will be made in the final report,

7. Acceptance Tests

This clause should specify the procedures and standards to be employed for the acceptance tests; the organization responsible for developing and programming the acceptance tests; the content of the acceptance tests; and the time the acceptance tests are to be run.

8. Milestone Dates

This clause should specify contract review dates, contract and item delivery dates, and other dates of significant importance.

9. Costs

This clause should specify the type of contract, the costs, and fees. The incentive formula should be included if a CPIF contract is negotiated.

10, Change Control

Contract clauses should be developed to cover the costing (in terms of schedule and dollars) and approval of the software system changes. The contract clauses will depend upon the type of contract. It is obvious that a change in scope in a fixed price contract could necessitate a renegotiation of the contract while a change in a CPFF contract could be much less formal.

3.3.2 Phases of the Programming Process and the Contract

The various phases of the programming process discussed in section 3.1.4 can be contracted separately or collectively. The type of contract and contract clauses, associated with each phase, is discussed:

1. Definition of System Requirements

In this phase, the basic activities are Identification of the problem to be solved, selection of a sound technical solution, and concise documentation of the solution.

Generally, this phase is most effective when contracted on a CPFF basis.

Contract clauses, as outlined in section 3.3.1, on planning, scheduling, progress reporting, deliverable Items, and documentation standards should be included in this phase.

2. Definition of Performance Specifications

The problems involved in this phase are integration of the equipment and facilities Into system design and the specifications of utility and support system requirements.

A CPIF or CPFF contract can be effective during this phase. Contract clauses that should be included are concerned with deliverable items, baseline documentation, and configuration control.

3. Development of Program Design

The most common problems encountered are involved with software-hardware interfaces, software-software interfaces, and definition of system environment.

The CPIF or FP contract can be effective during this phase. Contract clauses that should be included are configuration control, customer furnished items, baseline documentation, implementation procedures and standards, planning, scheduling and progress reporting, and deliverable items.

$3.3.1_3 - 3.3.2_1$

4, Program Implementation

The basic problems that occur during this phase are: poor documentation; poor programming practices; lack of adequate testing; incomplete acceptance tests; duplication; slipped schedules; and inefficient computer programs.

A fixed fee contract can be used very effectively and it should contain strong clauses on: documentation; implementation procedures and standards; acceptance testing; and change control.

5. Systems Integration

The problems encountered involve: inadequate live environmental test plans; some element of the system not being available; various elements of the system not interfacing properly; poor acceptance phase checkout; and Inadequate training of operating personnel,

A cost plus fixed fee or cost plus incentive fee contract would seem appropriate for this phase since it is difficult to estimate the number and complexity of the problems that might arise,

The contract clause that is most important Involves the updating and maintenance of baseline documentation,

6. Program Maintenance

Requirements for continued program maintenance usually cannot be pre-specified; hence a time and materials contract for computer programmers should fit this phase, The most important contract clauses involve baseline documentation, program and operating documentation, change control and acceptance testing,

3.4 MONITORING AND CONTROL BY MANAGEMENT

3.4.1 Introduction

Quite distinct from the supervision on contractors exercised by the Center organizations directly responsible for implementation of programming, the management at the Centers and Headquarters need to monitor the overall programming efforts expended within their authority, This section deals with a number of management-support functions required 'for this purpose. The effectiveness of the group of people performing such functions depends to a considerable extent on the position of the group in the organizational hierarchy, To illustrate this point, the management-support functions are described below in the framework of assumed organizations at the various NASA levels. The main emphasis is on the nature of the responsibilities of these groups rather than their authority, although this latter question must be resolved when it is decided to implement these functions,

3.4.2 Management-Support Functions and Organizations

1. ~~A Quality Assurance Group~~

Each programming contract which is let or programming job done should be monitored by members of a quality assurance group. Such a group would not only assure the customer that he does get the quality that the job requires, but would also make it easier for the programming contractor to supply that quality: it gives him a justification for taking the time and trouble to make the capital investment of building proper quality in the programs, whereas he might not recognize a mandate to do *so* if the quality of the programs were not being directly assessed.

A, The Need for Quality Assurance in Software Development

The quality of a computer program is measured by: the capability of the program to do the job for which it was intended; the understandability of the program; the changeability of the program as determined by the ease in which the program can be modified to meet changing requirements; and the maintainability as determined by the ease of upkeep of the program,

The weighting of the above attributes in attempting an assessment of any individual job should, of course, be a function of the objectives for that job., For example, the operational programs of Project Apollo can be reasonably expected to be under constant modification due to the changing ne'eds of a multi-mission research and development effort; hence the attributes of changeability and maintinability should be heavily weighted.

A great deal of latitude is possible in the amount of effort taken to insure that the computer program will be satisfactory, and it is difficult to decide just how much should be expected. In fact, the contractor who makes a special effort to do a really worthwhile job, which may save considerable amount of time and money in the long run, may at first appear to have done a slow and expensive job, Increased quality assurance activity during the course of programming development correspondingly increases the probability of satisfactory completion,

B. Responsibilities of the Group

(1) Monitoring

In order to achieve a quality program, it is necessary to closely adhere to standards representing good programming practices, One hundred percent confidence in programs cannot be obtained through testing alone. Monitoring throughout the programming process to insure that proper procedures and standards are followed is necessary and should be the main responsibility of the quality assurance group. The most important standards to be monitored are those concerned with documentation, modularity,. programming conventions, coding practices and languages used.

(2) Testing

The quality assurance group should administer suitable testing practices and insure the procurement of test and support programs (including simulators) necessary to certify that the programs adequately meet specifications, The group should also critique the acceptance test plans if they are prepared by the program development contractor,

C. Staffing

The technical staff of the Group should contain three types of people: systems analysts, programmers, and clerical personnel,

The systems analysts would be primarily responsible for deciding just what degree of effort must be expended on quality for the particular programming job and to assist in determining a concomitant set of standards to which the programming process should conform. They should be familiar with all levels of data processing and should play a direct role in the planning phases of the programming process, and their approval should be required on the final program system design,

There must be a testing staff consisting of personnel who are acquainted with the needs of the customer and with the entire programming effort, They should be experienced programmers familiar with the intent of the programs inspected and adept at perception of possible trouble areas,

Finally, the clerical staff would inspect all source programs, flow charts, etc., to make sure that they do adhere to the standards,, Ideally, this function should be automated.

D. Independence of the Group

In order for the quality assurance group to be ab'le to successfully fulfill its monitoring function, it must not be under the command of the implementation group. They should report high enough up in the organizational structure to be independent of all who are directly concerned with the implementation of the programs. In particular, if the programming is being done by a contractor then the quality assurance group should not work for that same contractor.

The group would derive its authority from the fact that no contract end item may be accepted without having its approval,

2. Center Programming Group

This is a staff of technically experienced personnel to advise and assist the director of each NASA center in the overall management of those aspects of Project Apollo related to computer programming.

The responsibilities of this group would include: coordination of' the various computer programming activities within a center; resolution of interface problems between the various efforts; running intracenter requirements reviews; attending and critiquing design reviews of the programming efforts; preparation and transmission of schedule, requirements, milestone, and status information to headquarters; participation in intercenter requirements review; technical consultation in the procurement of computer hardware and software within the center; gathering and disseminating of computer programming performance statistics both within the center and throughout Project Apollo; and disseminating information concerning the availability of support and utility programs.

3. NASA Headquarters Programming Group

A staff organization at NASA headquarters would assist the Director of the Apollo Program in areas involving computer programming,

A. Administrative and Monitoring Functions

Specific functions include monitoring computer programming action items resulting from computer programming status and requirement reviews; monitoring computer programming milestones*; eliminating intercenter duplication of computer programming effort; determining possible widespread use of support programs which have been written during the course of development at a center; serving as an advisory body to centers in programming areas and setting up ad hoc committees to examine problem areas as required; investigating the advisability of NASA-wide standards; establishing policy with regard to computer programming procurement and control; maintaining liaison with the NASA Automatic Data Processing (ADP) committee; preparing periodic reports on the status of the computer programming effort for the Director; and collecting and processing cost and performance statistics related to completed computer programming efforts. These statistics will be used in the development of performance criteria for specifying and measuring contractor performance and estimating expected coots (Section 3.2.1, paragraph 3, discusses examples of the types of statistics that would be collected),

B. Computer Program Status and Requirements Reviews

In order to maintain control and direction of the programming effort, management should be continuously informed on the programming status, Periodic reviews should be scheduled, organized, and coordinated by the headquarters programming group,

---

* This function would include the setting of standards for the definition of milestones and the inclusion of these milestones in the presently existing NASA management control structure, i.e., Schedules and Resources Procedures (SARP) charts.

given by the various centers, and should be attended by the key headquarters and center personnel who are able to identify and evaluate problems, provide guidance and direction, and initiate appropriate corrective action when required, The reviews should be held at the discretion of the NASA headquarters programming group in advance of the occurrence of major mission milestones, with a final review in advance of the mission. This would, hopefully, enable any errors of omission or commission to be detected and corrective action initiated far enough in advance so that schedules will be minimally affected, The programming review should accomplish the following objectives: enable management attention to be directed to important problem areas; spotlight system interfaces and possible integration problems; provide a summary of programming development status; evaluate the development status with respect to schedule milestones; review major expenditures; evaluate the adequacy of coverage of programming developments (i.e., that all necessary programs are under development, particularly where programs are developed and tested on machines other than those on which they will ultimately run); assess any program criticality to determine if special techniques or possibly parallel developments may be required to insure the satisfactory completion of especially critical programs; improve understanding of system objectives and improve communication among participants; evaluate merit and effectiveness of the overall operation; and provide training and information for project participants whose involvement lies in other areas, but who may be affected by the programming effort.

3.4.3 Example of an Integrated Management Structure

Descriptions of organization procedures and functions related to the problem of managing computer programming have been given, In this section, an example of how these methods can be applied to the Apollo project is presented, The example depicts the organizations required for the monitoring and control of computer programming, the functions of these organizations, the reporting requirements, and the interrelations among these groups, This example is a model on which further study can be based; it is not meant to imply a definitive recommendation at this time,

The setting up of new organizations and personnel within existing organizations should not obscure the basic premise that computer programming can best be managed from the local level and that direct, well-defined customer-contractor communications is the major administrative requirement in the production of working computer programs.

Figure 2 indicates an assumed organizational setup in a NASA Center and the relationship between that Center and NASA Headquarters. The example is modeled along the lines of MSC with no implication that MSC either needs or is more amenable to this type of organizational structure than any other center. The solid lines indicate organizations already in existence; the dotted lines indicate additional organizations and lines of communication,

For clarity, the example depicts a situation in which two programming subcontractors are being monitored by the quality assurance group, In actuality, the quality assurance group would monitor all computer programming efforts within the Center that are designated for this degree of control,

The terms Directorate, Division and Programming Branch describe various levels of organization within the NASA Center, Programming Branch is the organization that sets requirements for and monitors the outputs of the programming subcontractor, This type of organization is not always called out formally as a separate entity, but for the purpose of this illustration this will be considered the case. The prime responsibility for computer programming management and control presently lies in the programming branch, This sftuation would not be changed by the depicted management structure. The programming branch would be assisted by the quality assurance group as a monitor on computer programming quality, and by the computer programming office in the task of relating the particular computing job in hand to the Apollo system. The Center project office would insure that there is no duplication of effort in the Center and would provide a path of communication of information from the Center to NASA Headquarters, All contact with the contractor would be handled through the programming branch,

An outline of management and computer program implementation responsibilities for the organizations depicted in figure 2 is given in figure 3. The Organizations described are: the programming branch; the contractor; the quality assurance group; the centerwide computer programming office; and the headquarters programming group.

4.0 CONCLUSIONS

Computer programs are not unlike hardware in that they represent the deliverable end items of a process which can be well defined. It follows that programs can and should be subjected to the same degree of management control as are hardware items.

The problems of effective management control of computer programming in the Apollo Project are intensified for two reasons: a) compared to previous NASA projects computer utilization in Apollo will increase drastically, possibly by a factor of fifty; and b) the management experience built-up within NASA is derived mostly from hardware projects.

What tools are necessary for solving the problems of management of computer programming is known. The majority of them have been used successfully on hardware, and it is primarily a matter of adapting them to the programming situation in Apollo. Only very few require to be developed especially. In both cases, the tasks can be accomplished in a reasonable time, because the nature of the programming process is understood.

A number of specific conclusions can be drawn at this time:

1. The contract should be the principal instrument of control over procurement and development of computer programs. Standard contractual clauses should be developed to cover **requirements** for: proper documentation; adherence to program standards in the area of coding, flow charting, program verification; costs, schedules and requirements; methods of payment; and acceptance testing.

2. An explanation of the programming process should be made to managers at all levels of the Apollo Project so that they can better understand the need for planning, requirement identification, Configuration management, and costs as they pertain to computer programming.

3. Criteria for measuring contractor performance and evaluating proposals must be developed. Essential to this purpose is the collection of pertinent statistics on all programming efforts.

5.0 FUTURE EFFORT

The main effort in the Bellcomm study will be directed towards the development in depth of means of implementation of the basic concepts set forth in this interim report, The results will be incorporated in detailed recommendations covering each major area of the programming process.

To insure that these recommendations: a) meet immediate needs over the range of application of computer programming in Project Apollo; b) can b.e put into effect in the shortest possible time consistent with basic NASA policy and objectives; it is intended that:

1. Close contact will bo maintained with personnel at the NASA centers who are concerned with computer programming for a broad exchange of ideas and information,

2. Recommendations most suitable for immediate implementation will be chosen and presented for consideration. Reports describing specific recommendations will be published as completed.

BELLCOMM, INC.

A BIBLIOGRAPHY

Of Books and Articles Discussing Problems and Programs Associated with Computer Programming

BOOKS

Brandon, Dick H., Management Standards for Data Processing

ARTICLES

Blumenthal, S., An Approach to On-Line Processing, Datamation, June 1961

Brandon, Dick H., Performance Standards for Programming, Journal of Machine Accounting, September 1963

Brandon, Dick H., Standards for Computer Programming, Computers and Automation, April 1964

Brandon, Dick H., Systems Analysis Standards, Computers and Automation, December 1963

Brandon, Dick H. and Kirch, Frederick, Performance Standards, Computers and Automation, July 1964

Compise, J. A., Advanced Management in Data Processing, Journal of Data Management, June 1963

Crnkovich, J. E. and Stewart, W. E., Program Change Procedures, Datamation, June 1964

Daniels, A. E., Some Problems Associated With Large Programming Efforts, American Federation of Information Processing Societies Spring Joint Computer Conference Proceedings, Vol. 25 (1964), pp. 231-238.

Farr, L. and Nanus, B., Some Cost Contributors to Large-Scale Programs, American Federation of Information Processing Societies Spring Joint Computer Conference Proceedings, Vol. 25 (1964), pp. 239-248.

Flores, Ivan, Program Debugging, Computers and Data Processing, May 1964

Frank, W. L., Gardner, W. H., and Stack, G. L., Programming On Line Systems, Datamation, May and June 1963

Halpern, Mark, Computers and False Economics, Datamation, April 1964, pp 26-28

Head, Robert W., Testing Real-Time Systems, Datamation, July and August 1964

Holdiman, T., Management Techniques for Real Time Computer Programming, Journal of the Association for Computing Machinery, July 1962

Hosier, W. A., Pitfalls and Safeguards in Real Time Digital Systems with Emphasis on Programming, IRE Transactions on Engineering Management, June 1961

Maitland, D., The Retrospective Review in Data Processing, Computer Bulletin, March 1963

Martino, R. L., CPM and PERT - Misused, Misunderstood - But Powerful Tools for Management, Data Processing Yearbook, 1963-64

Mast, Larry T., Automatic Test and Checkout in Missile and Space Systems, Astronautics and Aerospace Engineering, March 1963

Patrick, Robert L., Measuring Performance, Datamation, July 1964

Pope, Donald L., Implementing Changes in a Large Scale System, Proceedings of the Eleventh Annual Meeting of the Institute of Management Sciences, March 1964

Ream, N. J., On Line Management Information, Datamation March and April 1964

Schroeder, R., Input Data Source Limitations for Real-Time Operation of Digital Computer, Journal of the Association for Computing Machinery, April 1964

Statland, N., Methods of Evaluating Computer Systems Performance, Computers and Automation, February 1964

A GLOSSARY

of Definitions of Some Terms and Symbols, Pertaining to Project Apollo Computer Programming, Used in This Interim Report:

Configuration: The complete technical description required to fabricate, test, accept, operate, maintain and logistically support systems, equipment, and/or software.

Configuration Accounting: Act of reporting and documenting changes made to systems/equipment/software subsequent to the establishment of a baseline configuration in order to maintain a configuration status.

Configuration Control: Systematic evaluation, coordination, and approval or disapproval of proposed changes to any baseline.

Configuration Identification: The technical documentation defining the approved configuration of systems/equipment/software under design, implementation, test, and maintenance.

Configuration Management (General Definition): The formal set of procedural concepts by which a uniform system of configuration identification, control, and accounting is established and maintained for all NASA systems/equipment/software and components thereof.

Contract End Item: A deliverable item resulting from the programming process; an arbitrary designation for the portions of a system/equipment/software identification as a result of a formal functional analysis: it is a functional entity physically related and selected for the purpose of system development, procurement, and logistics.

Contractor: For the purpose of this interim report, "contractor" is defined to be an organization performing services under contract to NASA.

Customer: For the purpose of this interim report, "customer" means the NASA agency immediately responsible for the programming task.

On-line: Descriptive of a system and of the peripheral equipment or devices in a system in which the operation of such equipment is under control of the central processing unit, and in which information reflecting current environment is introduced directly into the main flow of the data processing,

Operational Program: The primary computer program of a particular system or mission, exclusive of utility and support computer software, which operates in real time and/or in direct support of the system functions or mission and in direct relationship to the system or mission environment.

PERT: Program Evaluation and Review Technique,

Practices: Official agreements on the procedures and standards which shall apply in a given project or activity.

Procedure: A formal instruction, carrying management approval, .governing and prescribing the means by which personnel are to operate to accomplish an objective or programming process ‹

Real-Time Programs: Computer programs which operate without; significant delay, and in *so* close a relation to the source of input data as to make possible the immediate and direct use of results for process control and other simultaneous monitoring purposes.

SARP: Schedule and Resources Procedures.

Software: The totality *of* computer programs and routines used to extend the capabilities of computers•

Standard: Uniform engineering and technical criteria to which items, materials, processes, methods, design, and engineering practices shall conform,

APPENDIX

1.0 CHECKLIST FOR SOFTWARE-HARDWARE INTERFACES

This checklist will contain a list of such potential interface areas as:

1.1 Computer Peripherals

What peripheral devices are associated with the computer to be used? Are there any special or unusual devices for this application? Is it a multi-computer complex? Is there data transfer between computers?

1.2 Communication Equipment

Are there any display devices, keyboard consoles, radar sets, electronic or manual switching devices, associated with the computer? What are the data formats and rates of transfer?

2.0 CHECKLIST FOR PROGRAM LANGUAGE SELECTION

This checklist will cover such areas as:

2.1 Program Size Limitations

Will the program language (compiler or assembler) process programs of the size estimated for this application?

2.2 Computer Storage

Is the computer's primary (core memory) and secondary (disk, tapes, drums, etc.) storage sufficient for effective operation of the compiler or assembler?

2.3 Programmer Training

Is programmer re-training necessary? If so, how long will it take?

2.4 Programming Applications

What is the application? Scientific, business, command and control, or a combination?

2.5 Delivery Schedule

Is the software the pacing item? Will the programming language shorten the delivery schedule?

2.6 Documentation Requirements

Will the language facilitate flow diagramming or program documentation?

2.7 Program Complexity

Are the programs to be produced of a complex nature? Will the language provide for this complexity?

2.8 Compiler or Assembler Availability and Reliability

Is it being used on the object computer? How long? How frequently are errors detected?

2.9 Program Library

Is there a program library available? Docs it contain the type of library routines required in this task? How well are these routines documented? How long has the library been used?

3.0 PROGRAM DESIGN CHECKLIST

This checklist, will indicate the areas which should be considered during the design ,phase, For example:

Have all program hardware interfaces been resolved? Have all interfaces with communication devices been resolved? Have all Interfaces with other programs been resolved? Does the program satisfy all of' its functional requirements? Have the economic tradeoffs been assessed?

4.0 PROGRAM IMPLEMENTATION CHECKLIST

This checklist will contain suggested guidelines for program implementation, For example:

Review the program design for completeness; produce the program transfer document; flow diagram the program; perform program codes with liberal comments; perform desk checking of produced code; analysis of program code for completeness; generate program test plan; perform program testing; produce final program documentation; verify program's correct operation; and integrate program into the system for system tests,

TABLE 1
OPERATIONAL PROGRAMS FOR APOLLO
MANNED SPACECRAFT CENTER

Name of Computer Program	Computer	Purpose	NASA Office	Programming Contractor	Remarks
Operational Computer Program	IBM 7094 (One of two duplexed machines)	To perform those computations required to assist the flight controllers in the MSCC during an operational mission	Lynwood Dunseith; Flight Operations Directorate (FOD) Mission Planning & Analysis Division, Real Time Program Development Branch	IBM Houston Operation 140 programmers	Estimated Computer instructions 700K
Communications Processor	Univac 490 (one of two duplexed machines)	Message checking and data routing within the MSCC	Flight Support Division Dave Myles; Head, Operational Ground Communication Group	Univac	Univac
Data Reduction.	To be aetermined	Off-line recording and data reduction for Apollo mission	Mission Analysis Division Computations and Analysis Division		
Spacecraft Guidance Control	MIT Apollo Guidance Computer	Apollo Spacecraft; Guidance	Guidance and Control Division Robert C. Duncan Jack Funk	MIT	
ACE - S/C Checkout	CDC 160G (10 installations 2 computers each)	To provide automatic checkout of Spacecraft, LEM, G&N Computer by computer controlled test equipment	Rolf Lanzkron ACE Spacecraft Project Office Operational Programs to be run at KSC, R&D effort (programs) at Houston. Programming at GE Daytona	General Electric Daytona Beach	Each computer installation consists of one Uplink and one Downlink computer. Installation as follows 4-Kennedy 3-N.American 1-Houston 2-Grumman

Sheet 1 of 6

TABLE 1

OPERATIONAL PROGRAMS FOR APOLLO

GEORGE C. MARSHALL SPACE-FLIGHT CENTER

Name of' Computer .Program	Computer	Purpose	NASA Office	Programming Contractor	Remarks
Saturn Guidance Programs	Saturn Guidance Computer and Advanced S.G.C (IBM)	G & N of Launch Vehicle	kstrionics Labs. - Guidance and Control Systems Div. V. Mack	IBM G. Hudson T. Woodsel	
Saturn Vehicle Systens Checkout	(2) RCA 110A	Control cf Prelaunch Automatic: Checkout of Saturn Launch Vehicle; Monitor Countdown	Astrionics Labs. - Guidance and Control Systems Div. C. N. Swearingen J. Turner	IBM J. Meadlock	Communicating Computers, one at LCC Controlling, one at LUT To be run at KSC
Huntsville Operations Support Center	B 5000 (Burroughs)	Monitor Countdown and Launch for Technical backup of RSC Launch Csntrol	Conputer Lab. - Data Systems Engineering Office J. T. Felder		Includes Launch Info exchange Facility Burroughs computer normally used for offline date processing

TABLE 1

OPERATIONAL PROGRAMS FOR APOLLO

JOHN.F. KENNEDY SPACE CENTER

Name of Computer Program	Computer	Purpose	NASA Office	Programming Contractor	Remarks
Realtime Data Reduction and Display Programs (CIF)	Contract in Procurement	Receive and store. data from launch vehicle and spacecraft via telemetry. Retrieve data and convert from telemetry measurements to appropriate engineering units for display, on request.	Dr. Rudolf Bruns Chief of Data Acquisition and. Data Analysis Dvision	Not yet Determined	
ACE S/C Checkout	CDC 160 G	Checkout of Spacecraft will run at KSC; developed at MSC See MSC chart for further information.			
Saturn Vehicle System Checkout	RCA 110 A	Launch vehicle Checkout at VAB and on pad. Developed at MSFC. See MSFC chart for further Information.			

T[illegible]E 1

OPERATIONAL PROGRAMS FOR APOLLO GODDARD SPACE FLIGHT CENTER

Name of Computer Program	Computer	Purpose	NASA Office	Programming Contractor	Remarks
Network Status Testing (CADFISS)	IBM 7094	Perform automated network status tests, in realtime, of all major subsystems of the manned spaceflight network tracking sites to determine their ability and readiness to support Apollo missions. Also used to assist in installation and acceptance phases of new equipment or modification to existing equipment.	*W. Adams	IBM - Goddard Real-time Programming Division Ross Peavy, Bethesda, Maryland	Approx. 100 K Instructions for GT 3
Network Unified S-Band Qualification Program	7094	Computer oriented tests and computations programs required for orbital man-rated qualification of unified S-Band tracking network.	*J.J.Donegan	IBM	pprox. A 80 K Inst. used for early Apollo flights only
Orbital Mission Control Program	7094	Provide realtime flight control data during all-phases of Apollo unmanned missions and simulations. Includes programs to monitor launch, evaluate abort alternatives, orbit prediction/correction, etc.	*C. Packard	IBM	Approx. 300 K Inst. for Apollo

TABLE 1

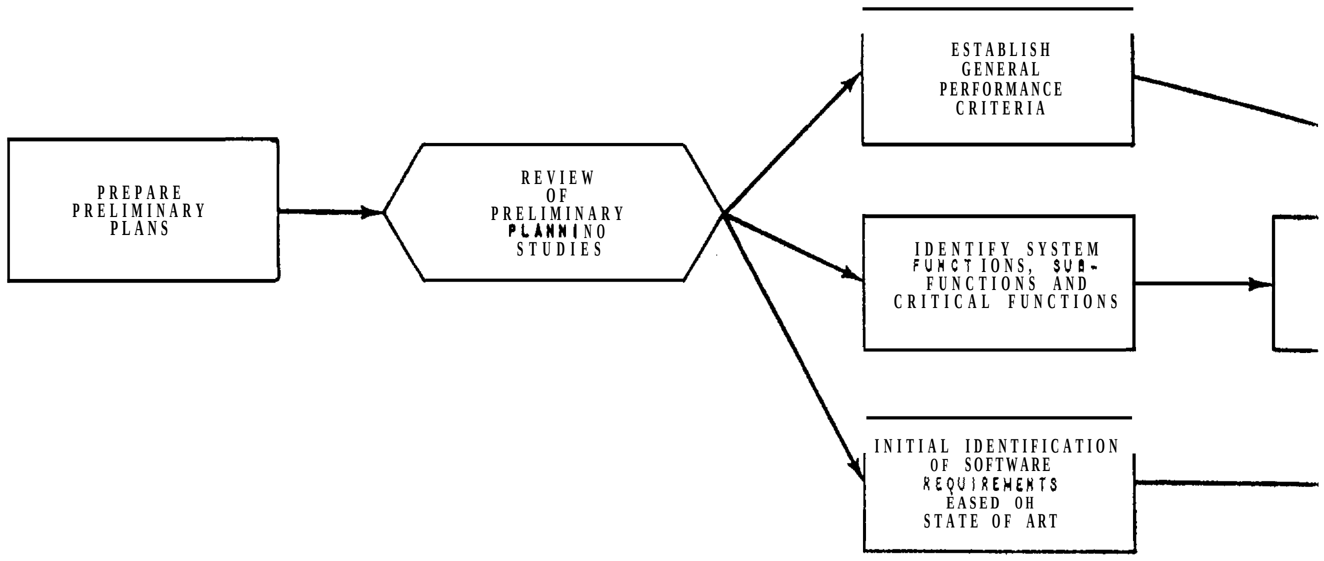
OPERATIONAL PROGRAMS FOR APOLLO
GODDARD SPACE FLIGHT CENTER

Name of Computer Program	Computer	Purpose	NASA Office	Programming Contractor	Remarks:
Real-time Network: Performance Monitor Program	7094	To give continuing evaluation of the performance of the tracking equipments and maintain mission integrity of the manned spaceflight network during all phases of manned and unmanned Apollo Missions	*C. Packard	IBM - Goddard Real-time Programming Division Ross Peavy, Bethesda, Maryland	Approx 80K Inst. for GT 3
Post Flight Reporter Program	7094	Non-realtime program to provide post flight analysis and evaluation of mission data for Apollo unmanned orbital missions	*L. Woldorff	IBM	Approx 23K Inst. for GT 3
Remote Site Computer Programs	Present sites, UNIVAC 1218 All others under Procurement	Generate computer programs to enable remote site computers to receive the telemetry Segments of the unified S-Band transmission from spacecraft, to decode, reformat, and transmit this data to the central computers	*P. Pashby	IBM	
Communications Processing Programs	UN 490	Message checking and automatic data switching	V. Stelter Division Chief, Communications Division	Programs Complete 16 GSFC Programmers 1 Univac. Programmer.	Approx. 13K Inst.

TABLE 1
OPERATIONAL PROGRAMS FOR APOLLO
GODDARD SPACE FLIGHT CENTER

Name of Computer Program	Computer		NASA Office	Programming Contractor	Remarks
Miscel-laneous	DDP-24	Although used principally for simulations, it may be used in Apollo for trans-lating and reformating acquisition data in real-time.	*J. J. Donegan	IBM Ross Peavy	

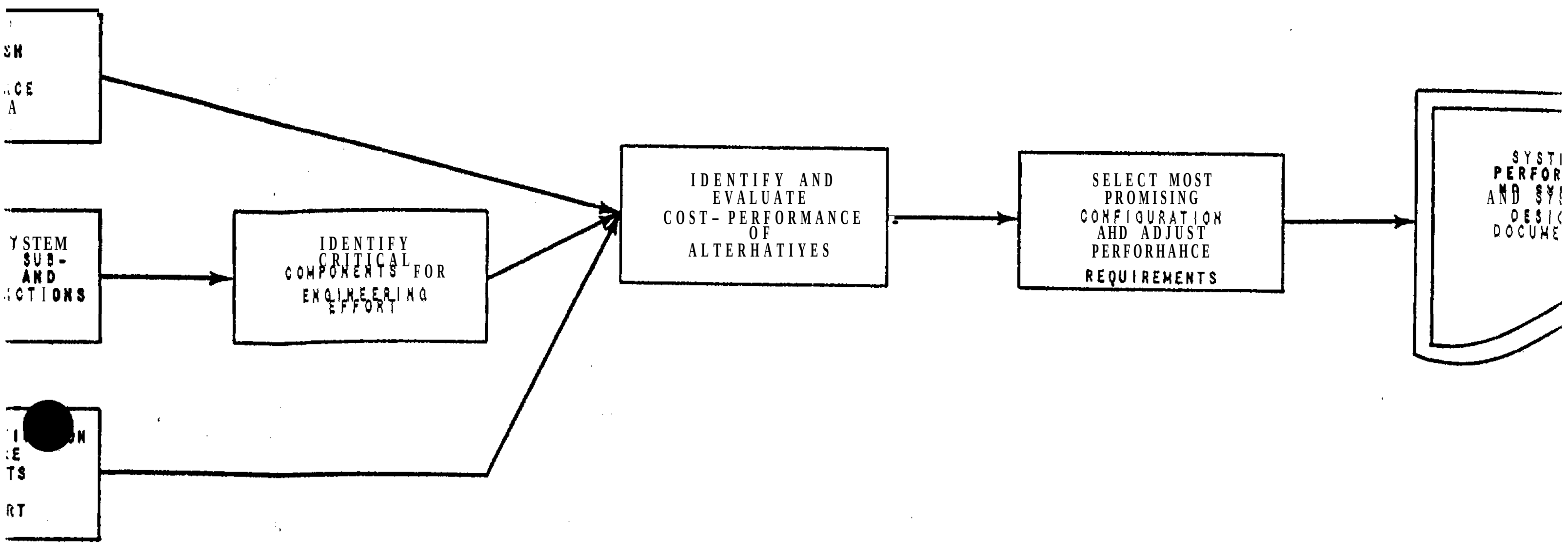
*These programs are under the cognizance of the Tracking and Data Systems Directorate, Manned Flight Operations Division, Data Operations Branch. J. J. Donegan is chief of Data Operations Branch. The responsible individual is shown.

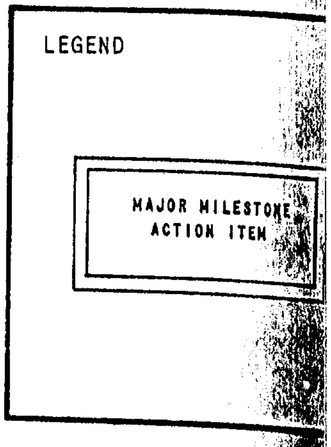

PREPARE PRELIMINARY PLANS
REVIEW OF PRELIMINARY PLANNING STUDIES
ESTABLISH GENERAL PERFORMANCE CRITERIA
IDENTIFY SYSTEM FUNCTIONS, SUB-FUNCTIONS AND CRITICAL FUNCTIONS
INITIAL IDENTIFICATION OF SOFTWARE REQUIREMENTS BASED ON STATE OF ART

PHASE 1; DEFINIT

PHASE 101

MAJOR MI1

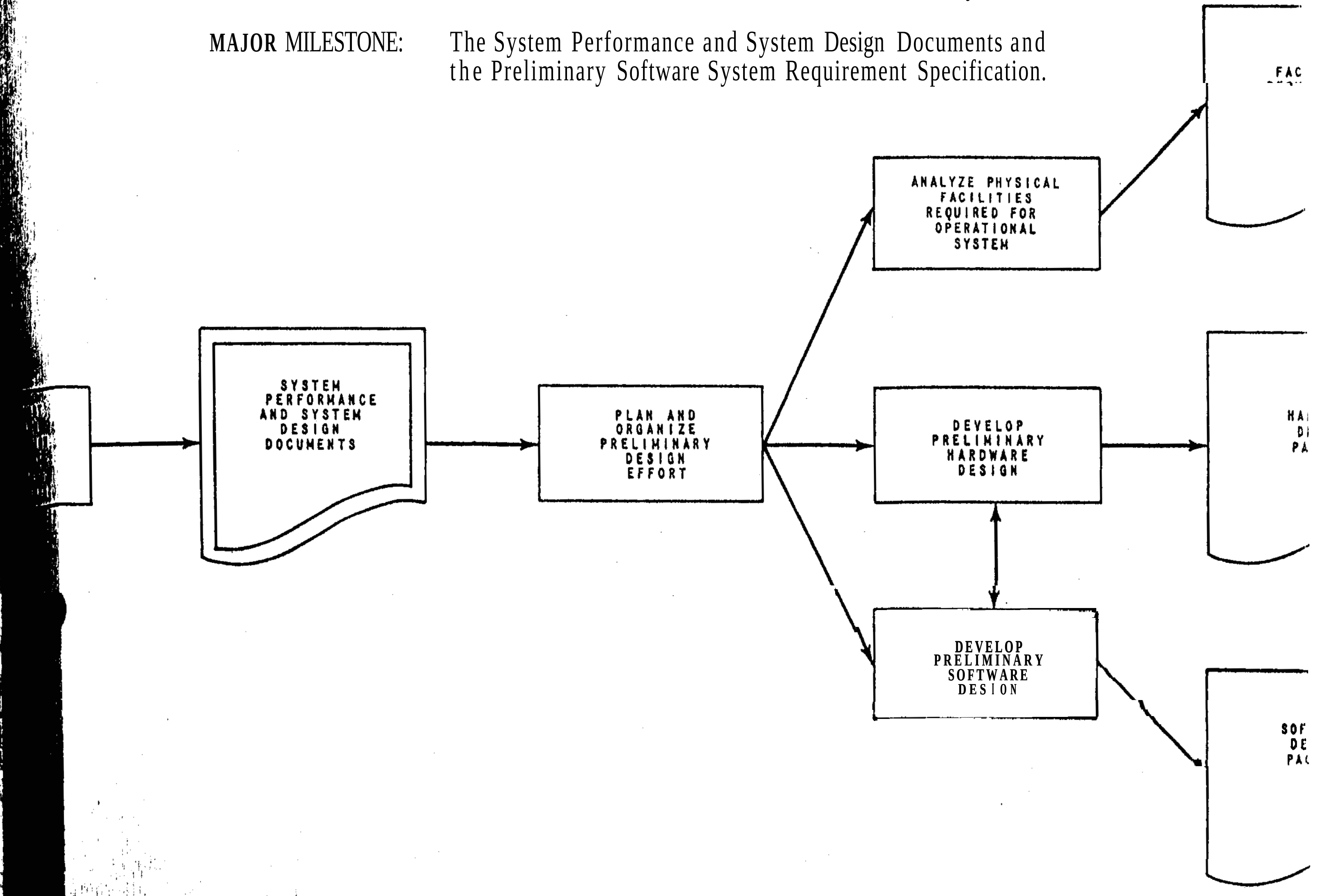

SH
ANCE
A
YSTEM
SUB-
AND
CTIONS
IDENTIFY CRITICAL COMPONENTS FOR ENGINEERING EFFORT
IDENTIFY AND EVALUATE COST-PERFORMANCE OF ALTERNATIVES
SELECT MOST PROMISING CONFIGURATION AND ADJUST PERFORMANCE REQUIREMENTS
SYST
PERFOR
AND SYS
DESIG
DOCUME
TS
RT



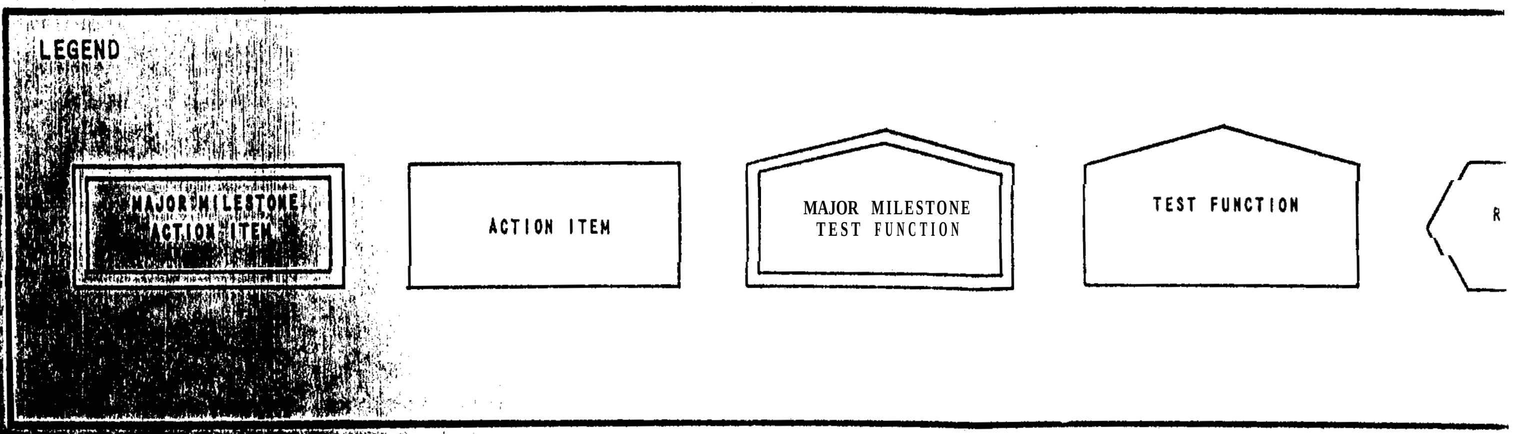
ASE 1; DEFINITION OF SYSTEM SOFTWARE REQUIREMENTS

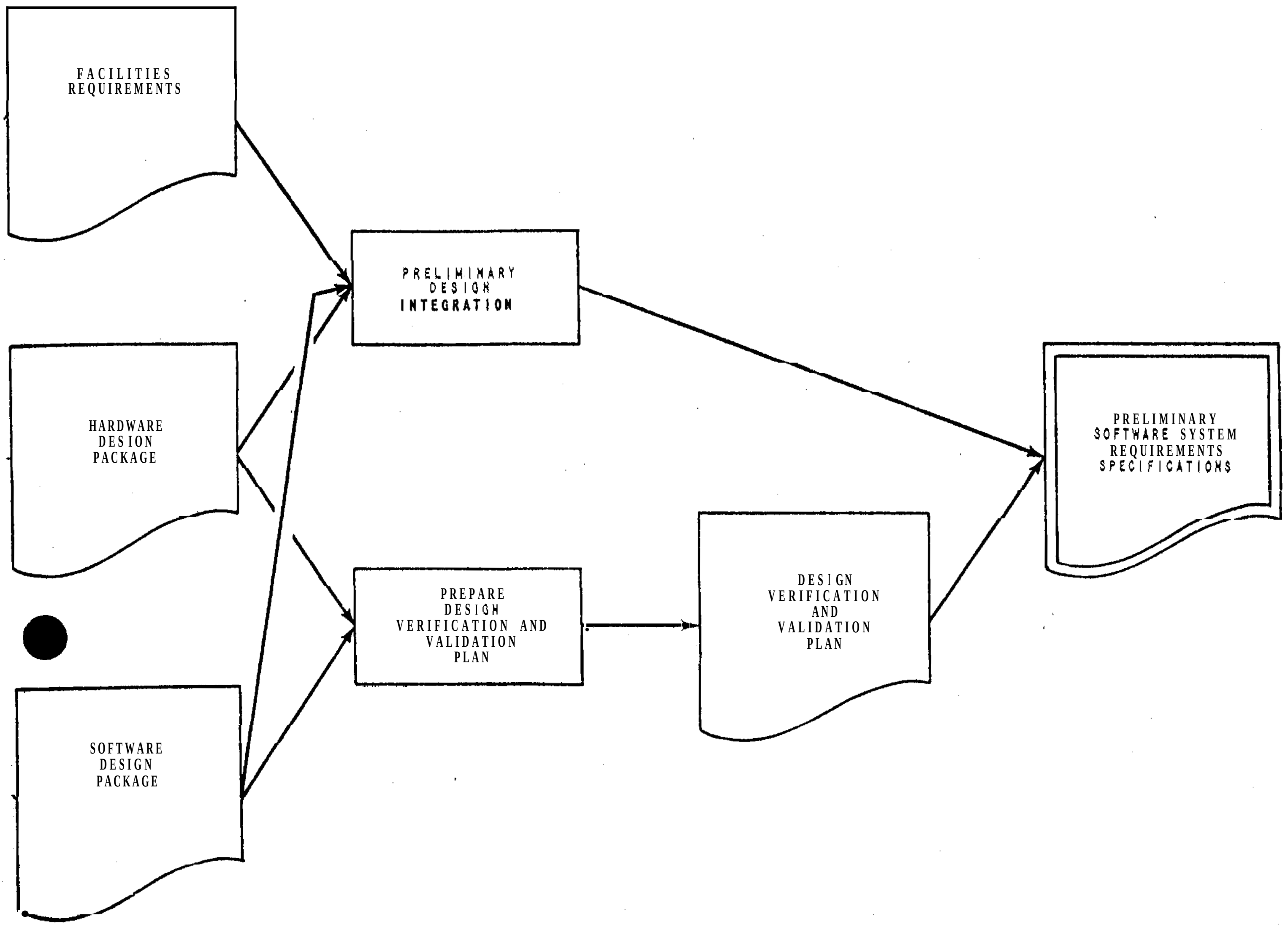
PHASE 1 OBJECTIVE: A Study Phase Leading to Definitions of System Functions and a Distinction Between Which Functions Shall be Implemented in Hardware and Which in Software and Which Manually.

MAJOR MILESTONE: The System Performance and System Design Documents and the Preliminary Software System Requirement Specification.



SYSTEM PERFORMANCE AND SYSTEM DESIGN DOCUMENTS
PLAN AND ORGANIZE PRELIMINARY DESIGN EFFORT
ANALYZE PHYSICAL FACILITIES REQUIRED FOR OPERATIONAL SYSTEM
DEVELOP PRELIMINARY HARDWARE DESIGN
DEVELOP PRELIMINARY SOFTWARE DESION
FAC
HA D PA
SOF DE PAC




FACILITIES REQUIREMENTS
HARDWARE DESION PACKAGE
SOFTWARE DESIGN PACKAGE
PRELIMINARY DESIGN INTEGRATION
PREPARE DESIGN VERIFICATION AND VALIDATION PLAN
DESIGN VERIFICATION AND VALIDATION PLAN
PRELIMINARY SOFTWARE SYSTEM REQUIREMENTS SPECIFICATIONS

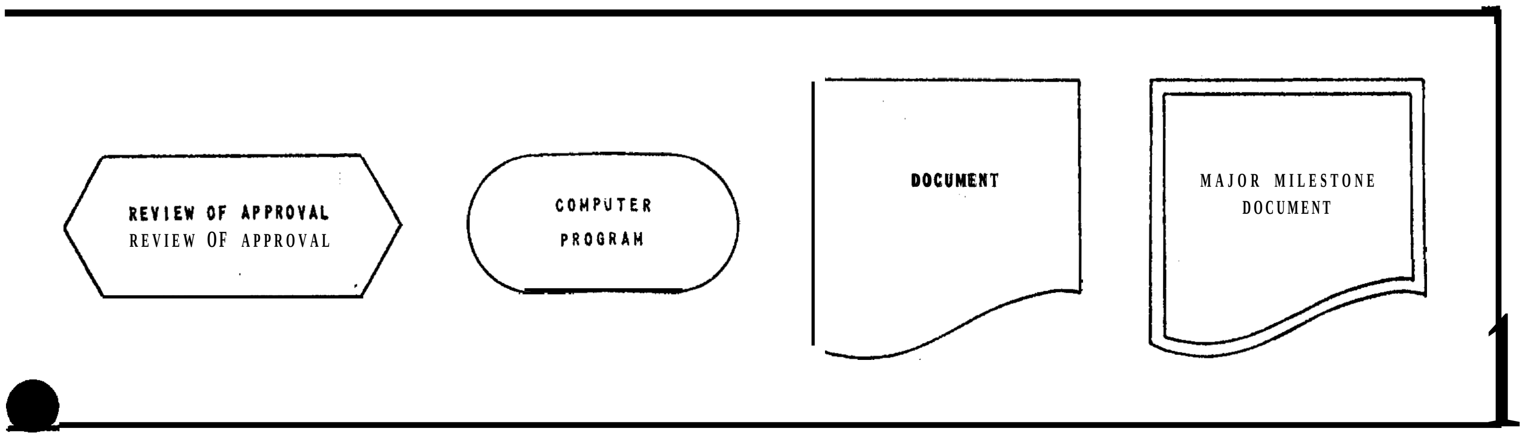
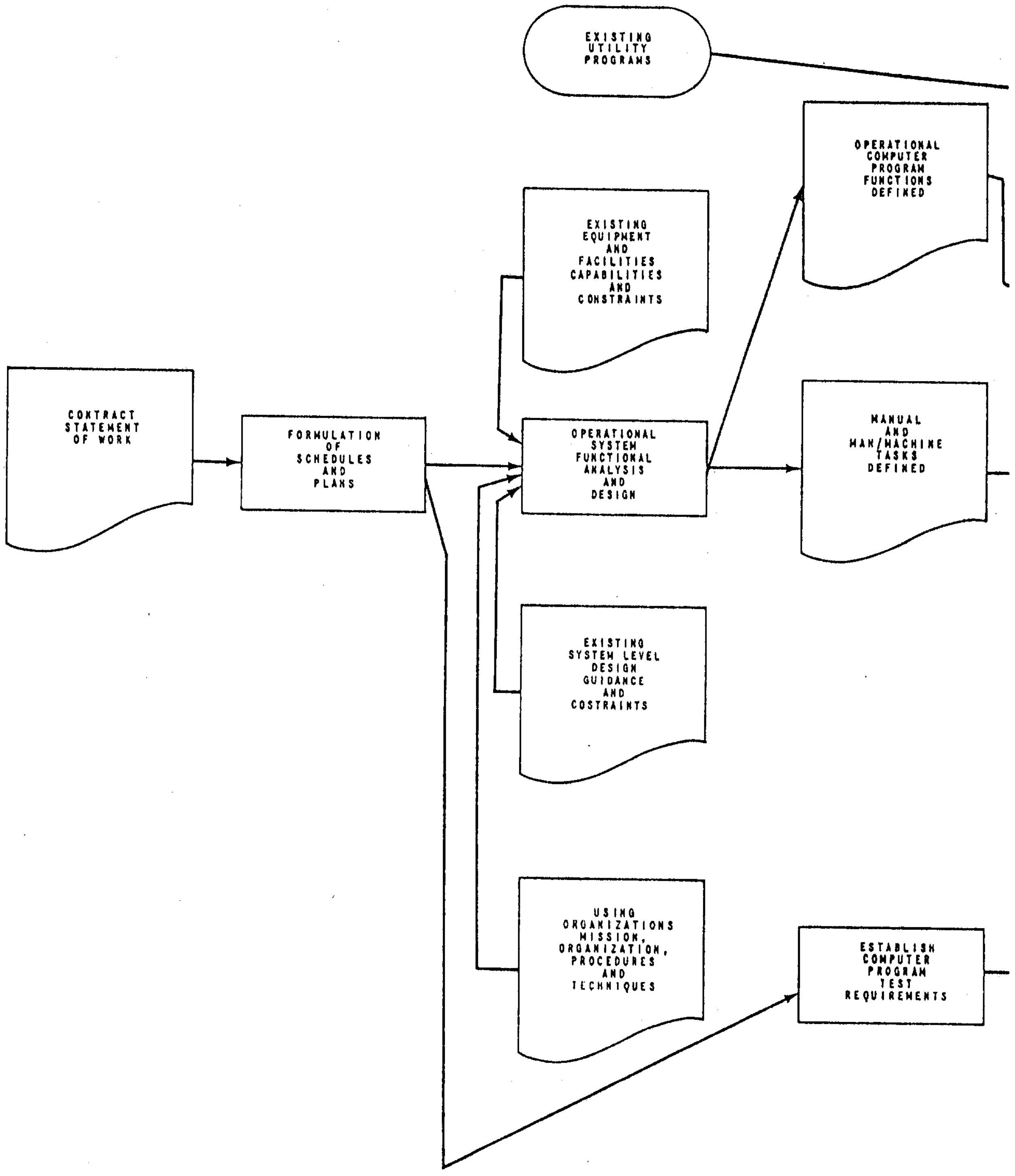
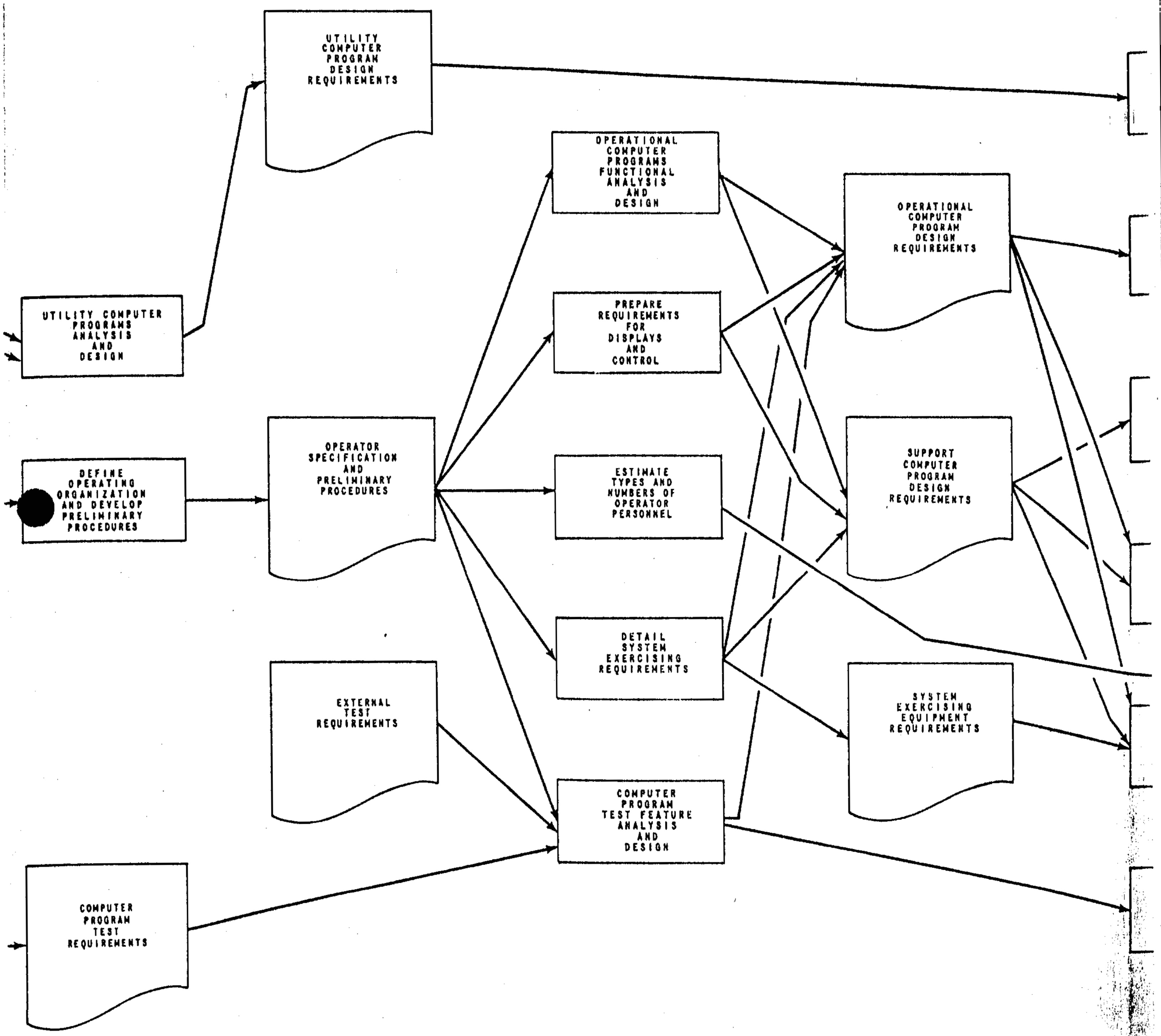

REVIEW OF APPROVAL
REVIEW OF APPROVAL
COMPUTER PROGRAM
DOCUMENT
MAJOR MILESTONE DOCUMENT

FIGURE 1


EXISTING UTILITY PROGRAMS
OPERATIONAL COMPUTER PROGRAM FUNCTIONS DEFINED
EXISTING EQUIPMENT AND FACILITIES CAPABILITIES AND CONSTRAINTS
CONTRACT STATEMENT OF WORK
FORMULATION OF SCHEDULES AND PLANS
OPERATIONAL SYSTEM FUNCTIONAL ANALYSIS AND DESIGN
MANUAL AND MAN/MACHINE TASKS DEFINED
EXISTING SYSTEM LEVEL DESIGN GUIDANCE AND COSTRAINTS
USING ORGANIZATIONS MISSION, ORGANIZATION, PROCEDURES AND TECHNIQUES
ESTABLISH COMPUTER PROGRAM TEST REQUIREMENTS

PHASE 2; DEFINITION OF SOFTWARE PERFORMANCE SP

PHASE 2 OBJECTIVE:	A Systems Analysis Phase Defining The Interrelations Between Software-Hardwa and The Interdependencies Between Software-Software.
MAJOR MILESTONE:	The Initial Software Performance Speci



UTILITY COMPUTER PROGRAM DESIGN REQUIREMENTS
OPERATIONAL COMPUTER PROGRAMS FUNCTIONAL ANALYSIS AND DESIGN
OPERATIONAL COMPUTER PROGRAM DESIGN REQUIREMENTS
UTILITY COMPUTER PROGRAMS ANALYSIS AND DESIGN
PREPARE REQUIREMENTS FOR DISPLAYS AND CONTROL
DEFINE OPERATING ORGANIZATION AND DEVELOP PRELIMINARY PROCEDURES
OPERATOR SPECIFICATION AND PRELIMINARY PROCEDURES
ESTIMATE TYPES AND NUMBERS OF OPERATOR PERSONNEL
SUPPORT COMPUTER PROGRAM DESIGN REQUIREMENTS
DETAIL SYSTEM EXERCISING REQUIREMENTS
EXTERNAL TEST REQUIREMENTS
SYSTEM EXERCISING EQUIPMENT REQUIREMENTS
COMPUTER PROGRAM TEST FEATURE ANALYSIS AND DESIGN
COMPUTER PROGRAM TEST REQUIREMENTS

E SPECIFICATIONS

ing The
-Hardware
een

e Specifications.

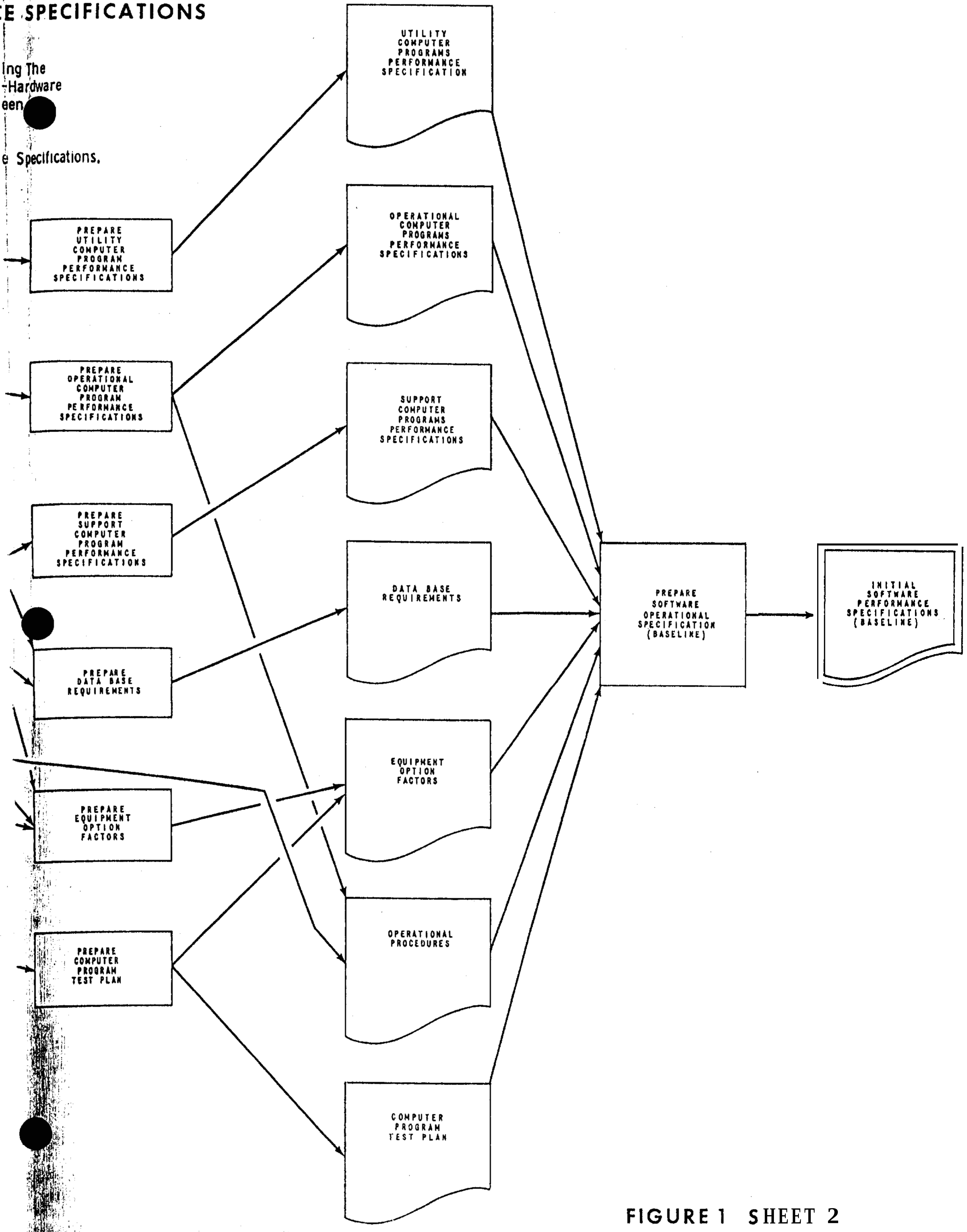
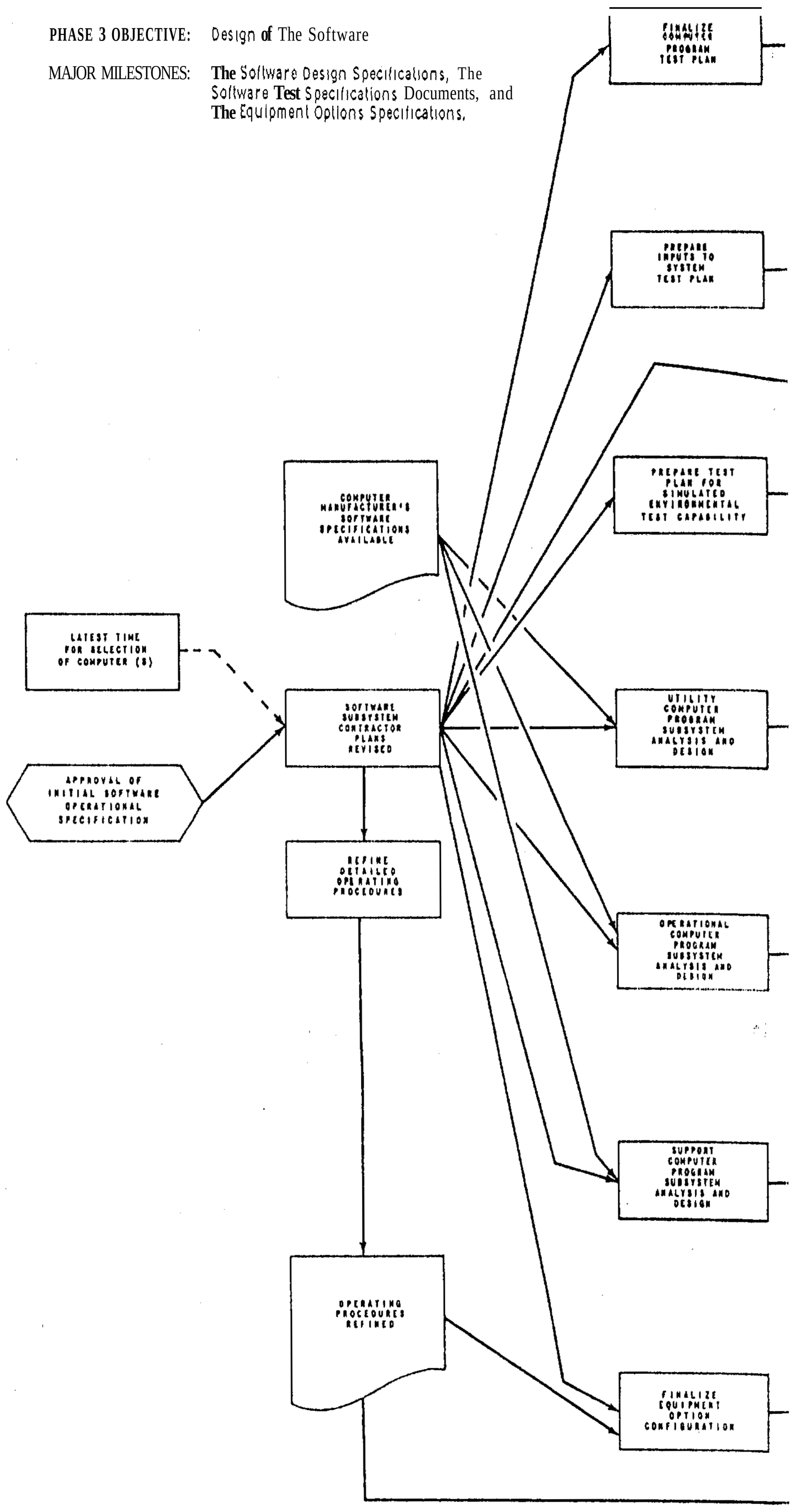

PREPARE UTILITY COMPUTER PROGRAM PERFORMANCE SPECIFICATIONS
PREPARE OPERATIONAL COMPUTER PROGRAM PERFORMANCE SPECIFICATIONS
PREPARE SUPPORT COMPUTER PROGRAM PERFORMANCE SPECIFICATIONS
PREPARE DATA BASE REQUIREMENTS
PREPARE EQUIPMENT OPTION FACTORS
PREPARE COMPUTER PROGRAM TEST PLAN
UTILITY COMPUTER PROGRAMS PERFORMANCE SPECIFICATION
OPERATIONAL COMPUTER PROGRAMS PERFORMANCE SPECIFICATIONS
SUPPORT COMPUTER PROGRAMS PERFORMANCE SPECIFICATIONS
DATA BASE REQUIREMENTS
EQUIPMENT OPTION FACTORS
OPERATIONAL PROCEDURES
COMPUTER PROGRAM TEST PLAN
PREPARE SOFTWARE OPERATIONAL SPECIFICATION (BASELINE)
INITIAL SOFTWARE PERFORMANCE SPECIFICATIONS (BASELINE)

FIGURE 1 SHEET 2

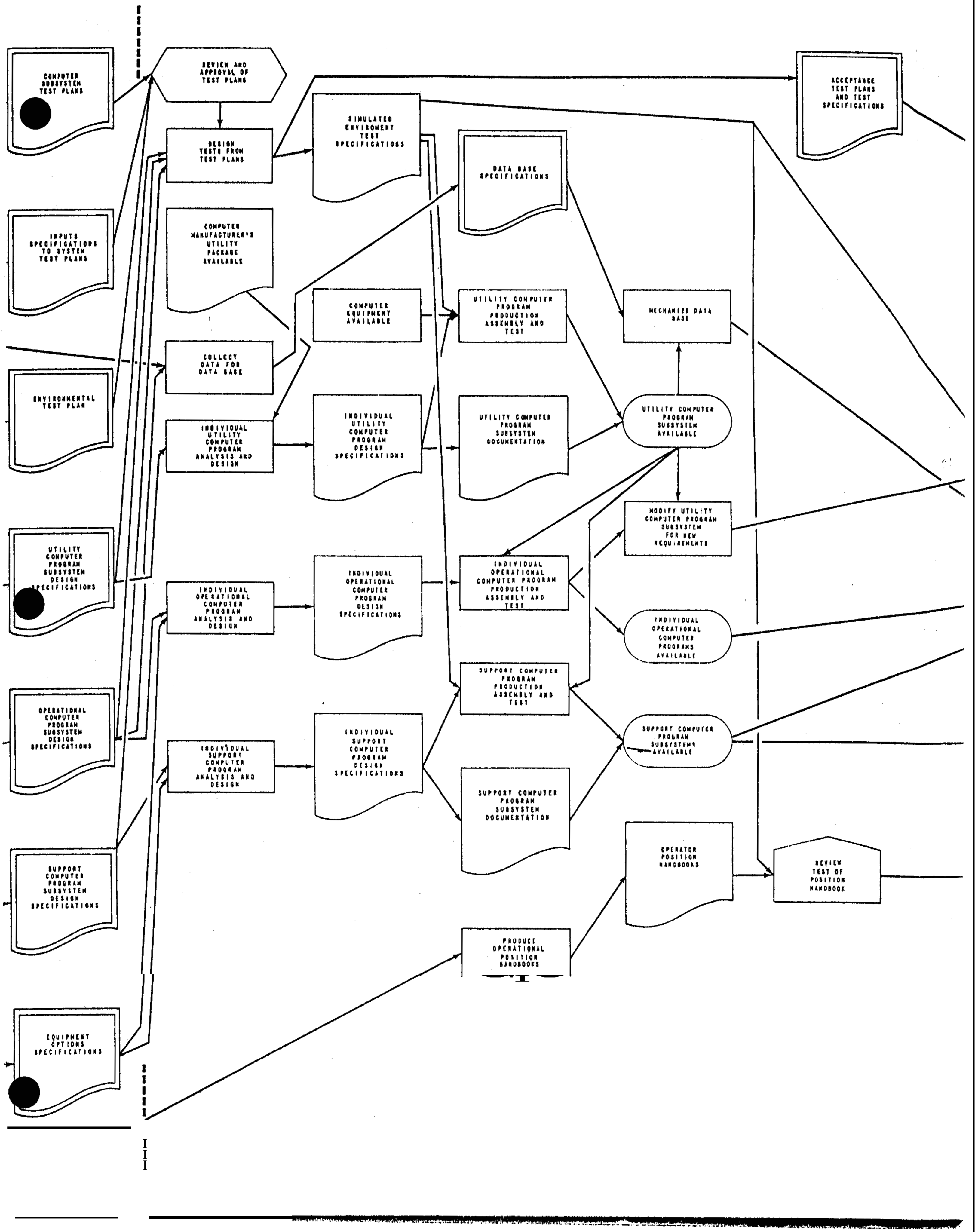
PHASE 3; DEVELOPMENT OF PROGRAM DESIGN

PHASE 3 OBJECTIVE: Design of The Software

MAJOR MILESTONES: The Software Design Specifications, The Software Test Specifications Documents, and The Equipment Options Specifications.



FINALIZE COMPUTER PROGRAM TEST PLAN
PREPARE INPUTS TO SYSTEM TEST PLAN
PREPARE TEST PLAN FOR SIMULATED ENVIRONMENTAL TEST CAPABILITY
COMPUTER MANUFACTURER'S SOFTWARE SPECIFICATIONS AVAILABLE
LATEST TIME FOR SELECTION OF COMPUTER (S)
SOFTWARE SUBSYSTEM CONTRACTOR PLANS REVISED
UTILITY COMPUTER PROGRAM SUBSYSTEM ANALYSIS AND DESIGN
APPROVAL OF INITIAL SOFTWARE OPERATIONAL SPECIFICATION
REFINE DETAILED OPERATING PROCEDURES
OPERATIONAL COMPUTER PROGRAM SUBSYSTEM ANALYSIS AND DESIGN
SUPPORT COMPUTER PROGRAM SUBSYSTEM ANALYSIS AND DESIGN
OPERATING PROCEDURES REFINED
FINALIZE EQUIPMENT OPTION CONFIGURATION


COMPUTER SUBSYSTEM TEST PLANS
INPUTS SPECIFICATIONS TO SYSTEM TEST PLANS
ENVIRONMENTAL TEST PLAN
UTILITY COMPUTER PROGRAM SUBSYSTEM DESIGN SPECIFICATIONS
OPERATIONAL COMPUTER PROGRAM SUBSYSTEM DESIGN SPECIFICATIONS
SUPPORT COMPUTER PROGRAM SUBSYSTEM DESIGN SPECIFICATIONS
EQUIPMENT OPTIONS SPECIFICATIONS
REVIEW AND APPROVAL OF TEST PLANS
DESIGN TESTS FROM TEST PLANS
COMPUTER MANUFACTURER'S UTILITY PACKAGE AVAILABLE
COLLECT DATA FOR DATA BASE
INDIVIDUAL UTILITY COMPUTER PROGRAM ANALYSIS AND DESIGN
INDIVIDUAL OPERATIONAL COMPUTER PROGRAM ANALYSIS AND DESIGN
INDIVIDUAL SUPPORT COMPUTER PROGRAM ANALYSIS AND DESIGN
SIMULATED ENVIRONMENT TEST SPECIFICATIONS
COMPUTER EQUIPMENT AVAILABLE
INDIVIDUAL UTILITY COMPUTER PROGRAM DESIGN SPECIFICATIONS
INDIVIDUAL OPERATIONAL COMPUTER PROGRAM DESIGN SPECIFICATIONS
INDIVIDUAL SUPPORT COMPUTER PROGRAM DESIGN SPECIFICATIONS
DATA BASE SPECIFICATIONS
UTILITY COMPUTER PROGRAM PRODUCTION ASSEMBLY AND TEST
UTILITY COMPUTER PROGRAM SUBSYSTEM DOCUMENTATION
INDIVIDUAL OPERATIONAL COMPUTER PROGRAM PRODUCTION ASSEMBLY AND TEST
SUPPORT COMPUTER PROGRAM PRODUCTION ASSEMBLY AND TEST
SUPPORT COMPUTER PROGRAM SUBSYSTEM DOCUMENTATION
PRODUCE OPERATIONAL POSITION HANDBOOKS
MECHANIZE DATA BASE
UTILITY COMPUTER PROGRAM SUBSYSTEM AVAILABLE
MODIFY UTILITY COMPUTER PROGRAM SUBSYSTEM FOR NEW REQUIREMENTS
INDIVIDUAL OPERATIONAL COMPUTER PROGRAMS AVAILABLE
SUPPORT COMPUTER PROGRAM SUBSYSTEMS AVAILABLE
OPERATOR POSITION HANDBOOKS
ACCEPTANCE TEST PLANS AND TEST SPECIFICATIONS
REVIEW TEST OF POSITION HANDBOOK
III

PHASE 4; PROGRAM IMPLEMENTATION

PHASE 4 OBJECTIVE: Implementation of The Software.

MAJOR MILESTONES: Data Base Specifications; Acceptance Test Specifications; Operator Position Handbooks; Completion of Acceptance Tests of Tne Operational, Utility, and Support Programs; Modification of Computer Programs Per Test Results.

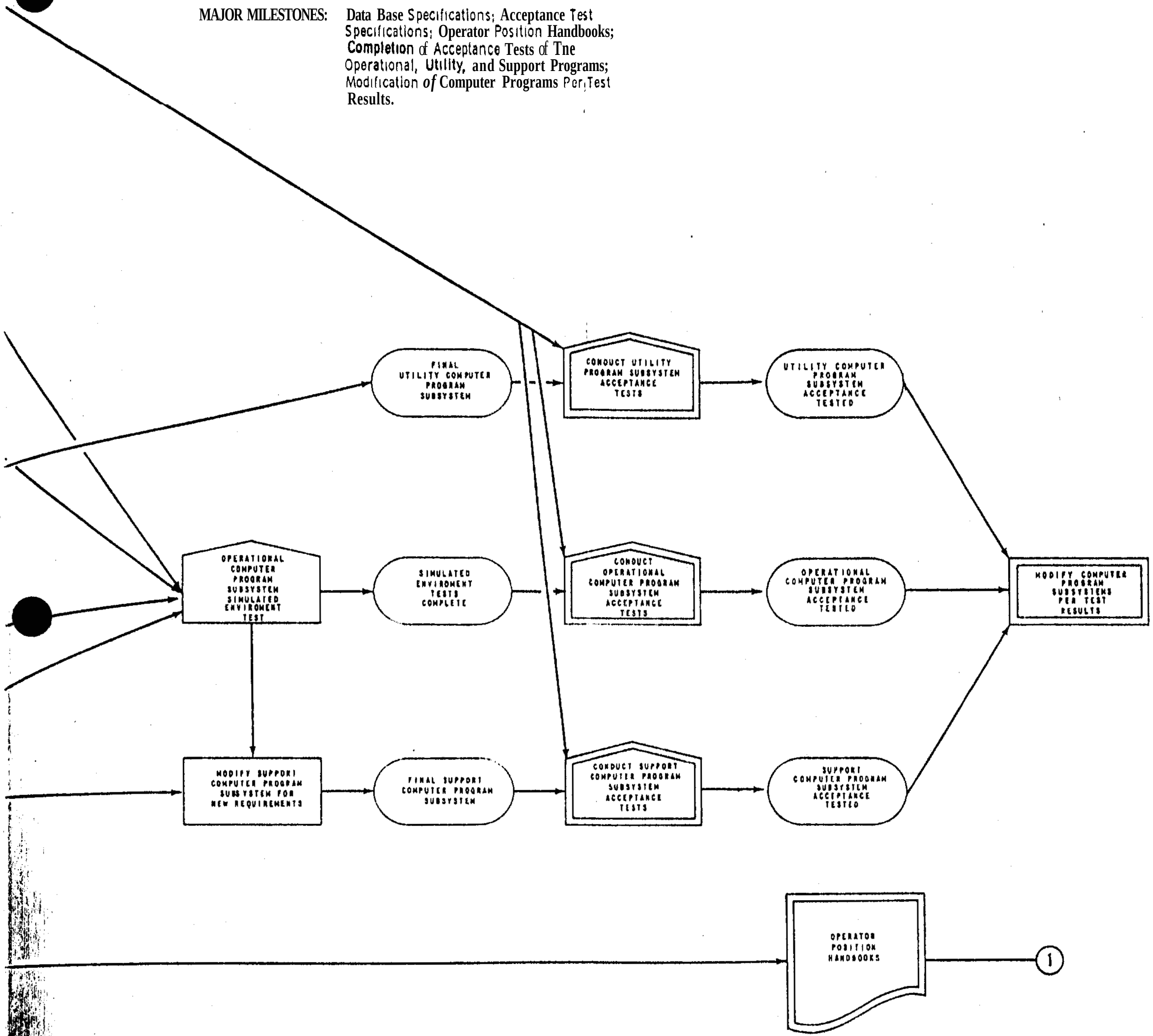
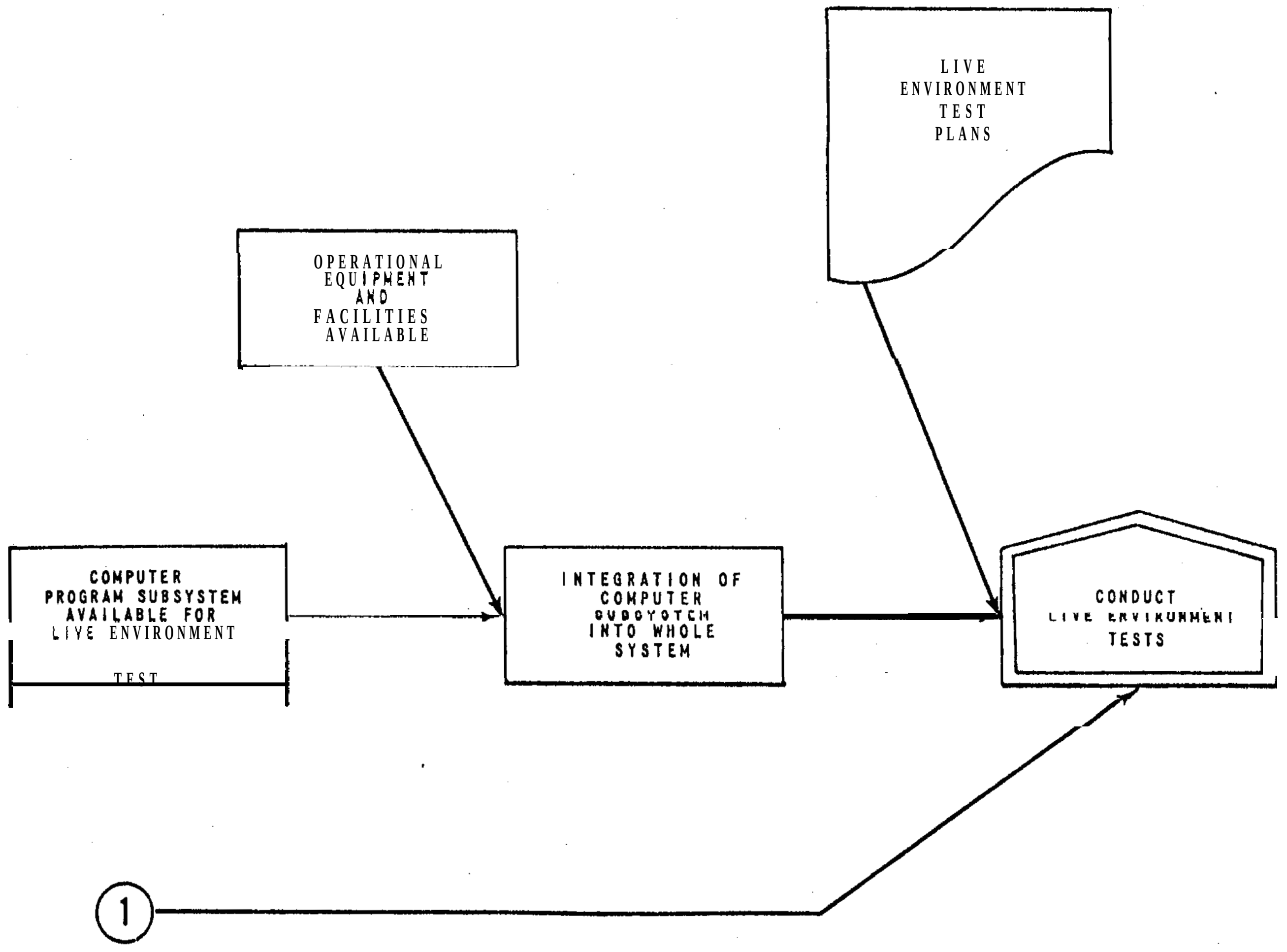

FINAL UTILITY COMPUTER PROGRAM SUBSYSTEM
CONDUCT UTILITY PROGRAM SUBSYSTEM ACCEPTANCE TESTS
UTILITY COMPUTER PROGRAM SUBSYSTEM ACCEPTANCE TESTED
OPERATIONAL COMPUTER PROGRAM SUBSYSTEM SIMULATED ENVIROMENT TEST
SIMULATED ENVIROMENT TESTS COMPLETE
CONDUCT OPERATIONAL COMPUTER PROGRAM SUBSYSTEM ACCEPTANCE TESTS
OPERATIONAL COMPUTER PROGRAM SUBSYSTEM ACCEPTANCE TESTED
MODIFY COMPUTER PROGRAM SUBSYSTEMS PER TEST RESULTS
MODIFY SUPPORT COMPUTER PROGRAM SUBSYSTEM FOR NEW REQUIREMENTS
FINAL SUPPORT COMPUTER PROGRAM SUBSYSTEM
CONDUCT SUPPORT COMPUTER PROGRAM SUBSYSTEM ACCEPTANCE TESTS
SUPPORT COMPUTER PROGRAM SUBSYSTEM ACCEPTANCE TESTED
OPERATOR POSITION HANDBOOKS
1

FIGURE 1 SHEET 3

PHASE 5; SYSTEM INTEGRATION

PHASE 5 OBJECTIVE: Integration of The Software, Hardware, and Personnel Into a Working System.

MAJOR MILESTONE: Successful Satisfaction of Line Environment Tests.


LIVE ENVIRONMENT TEST PLANS
OPERATIONAL EQUIPMENT AND FACILITIES AVAILABLE
COMPUTER PROGRAM SUBSYSTEM AVAILABLE FOR LIVE ENVIRONMENT TEST
INTEGRATION OF COMPUTER SUBSYSTEM INTO WHOLE SYSTEM
CONDUCT LIVE ENVIRONMENT TESTS
1

PHASE *6;* PROGRAM MAINTENANCE

PHASE 6 OBJECTIVE: Program Maintenance to Meet Changing User Requirements.

MAJOR MILESTONE: Turn Over Tested System To Users.

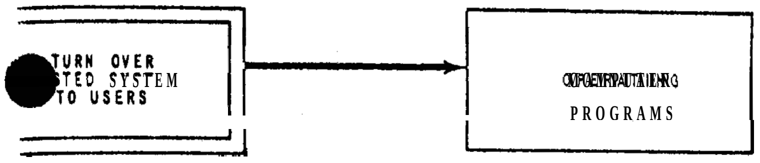


FIGURE 1 SHEET 4

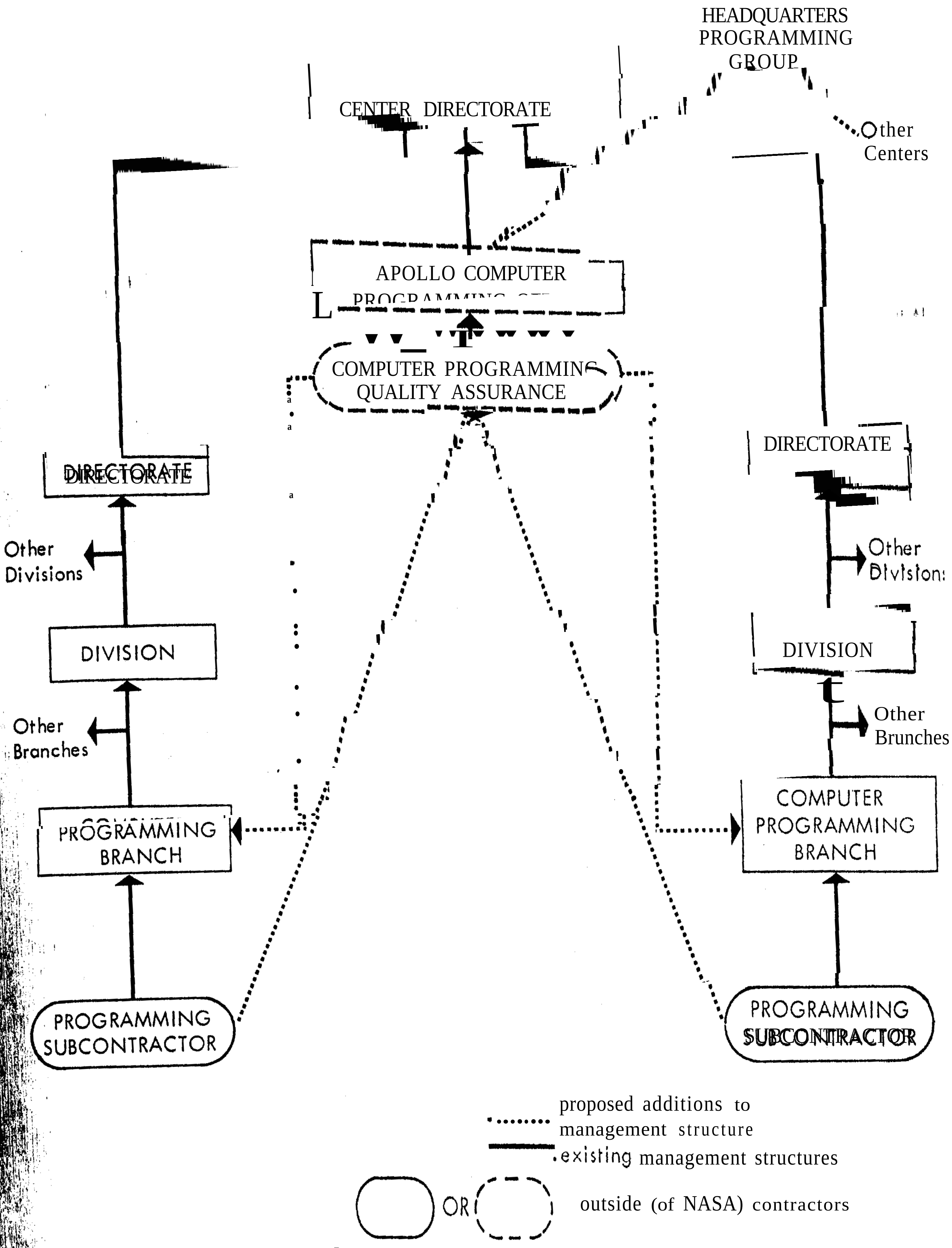

HEADQUARTERS PROGRAMMING GROUP
CENTER DIRECTORATE
Other Centers
APOLLO COMPUTER PROGRAMMING
COMPUTER PROGRAMMING QUALITY ASSURANCE
DIRECTORATE
Other Divisions
DIVISION
Other Branches
PROGRAMMING BRANCH
PROGRAMMING SUBCONTRACTOR
DIRECTORATE
Other Divisions
DIVISION
Other Brunches
COMPUTER PROGRAMMING BRANCH
PROGRAMMING SUBCONTRACTOR
proposed additions to management structure
existing management structures
OR
outside (of NASA) contractors

FIGURE 2 EXAMPLE OF MANAGEMENT CONTROL STRUCTURE