COMPUTER SCIENCE & TECHNOLOGY:

CONVERSION OF FEDERAL ADP SYSTEMS: A TUTORIAL

NBS Special Publication 500-62

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities[2] — Radiation Research — Thermodynamics and Molecular Science — Analytical Chemistry — Materials Science.

THE NATIONAL ENGINEERING LABORATORY provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

Applied Mathematics — Electronics and Electrical Engineering[2] — Mechanical Engineering and Process Technology[2] — Building Technology — Fire Research — Consumer Product Technology — Field Methods.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

Programming Science and Technology — Computer Systems Engineering.

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Washington, DC 20234.
[2]Some divisions within the center are located at Boulder, CO 80303.

COMPUTER SCIENCE & TECHNOLOGY:

CONVERSION OF FEDERAL ADP SYSTEMS: A TUTORIAL

Joseph Collica
Mark Skall
Gloria Bolotsky

Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

t
Special publication

U.S. DEPARTMENT OF COMMERCE, Philip M. Klutznick Secretary

Luther H. Hodges, Jr., Deputy Secretary

Jordan J. Baruch, Assistant Secretary for Productivity, Technology and Innovation

U.S. NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

Issued August 1980

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publicaton.

National Bureau of Standards Special Publication 500-62
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-62, 73 pages (Aug. 1980)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 80-600106

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1980

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
Price $4.00
(Add 25 percent for other than U.S. mailing)

PREFACE

The average life of a Federal computer system is estimated to be between seven and eight years, which is about double the average tenure of a computer programmer/analyst at a given installation. Too often, an agency is faced with the prospect of a conversion with personnel who are inexperienced in conversion techniques, and also lack an intimate knowledge of the system requiring the conversion. With the original implementors long since gone, and documentation, if it exists at all, of very poor quality, the conversion task becomes a dreaded experience. Hence, the conversion of Federal computer systems software has become one of the most costly, difficult and time-consuming tasks associated with the use and maintenance of computer software.

The General Accounting Office, in a 1977 report entitled MILLIONS IN SAVINGS POSSIBLE IN CONVERTING PROGRAMS FROM ONE COMPUTER TO ANOTHER, estimates that over $450 million is spent annually on Federal computer software conversion. Of that amount, GAO estimates that more than $100 million could be saved by utilizing better conversion techniques. One of the reasons for which the National Bureau of Standards undertook this study was to understand better the process of conversion.

This study has shown that, often, the results of conversions have been less than satisfactory in the excessive costs, in the performance of the converted system, and in the system's usefulness. Too often, conversion is attempted without an adequate analysis and assessment of the existing system, which includes programs, input data, output data, documentation requirements, and the conversion technique(s) which could most efficiently and effectively handle the conversion problems. Often, time spent reevaluating and resolving problems can be reduced if the adequate conversion experience and technique(s) are employed.

Due to time constraints and the lack of in-house expertise, agencies are often compelled to contract out the conversion. Many times, poor performance of a converted system can be traced back to the RFP, written by Federal agencies to acquire conversion services. Federal agency personnel who have been involved in writing RFP's for conversion services are very often confused concerning the type of information that should be included within the RFP, the types of deliverables which need to be specified, and the ways in which to define completion. Many agencies fail to include any performance constraints in a conversion RFP, asking instead for functionally equivalent source code. This invariably leads to degradation in performance and may result in a converted system which does not meet the user's needs. Confusion in contracting out for conversion services is further evidenced by the fact that there does not presently exist an official procurement policy or regulation which adequately deals with the treatment of conversion costs in evaluating vendor proposals during computer system acquisition.

Conversion is difficult even under well-planned conditions and a direct correlation can be drawn between the high costs associated with conversion and the lack of adequate conversion guidelines, analysis, experience and training. Another factor which can have almost as much impact on software conversion as cost, can be the time required to convert from one software system to another.

The major intent of this study is to develop a comprehensive conversion program within the National Bureau of Standards which would assist in reducing the costs and difficulties of conversion in the Federal Government. Chapter 7 describes the guidelines and standards which NBS has included in the formulation of its conversion program.

TABLE OF CONTENTS

CONVERSION OF FEDERAL ADP SYSTEMS: A TUTORIAL

Joseph C. Collica
Mark W. Skall
Gloria R. Bolotsky

This tutorial report was undertaken to provide a better understanding of conversion of Federal Government ADP Systems. Three sources were used for gathering the required information to prepare this tutorial: (1) interviews with commercial conversion experts; (2) interviews with Federal Government agency personnel who have recently experienced conversions; (3) current literature. The first three chapters comprise the tutorial. The next three chapters discuss the information gathered from the above three sources. The last chapter summarizes the authors' conclusions, while highlighting the major problem areas requiring guidance.

Key words: Conversion costs; conversion problems; conversion tools; database management; Federal agencies; language translators; maintenance; portability.

1. INTRODUCTION

Human beings have the ability to adapt to a changing environment. Adapting is seldom accomplished easily or quickly. Humans also have the tendency to resist change. The ability to adapt successfully to change versus the tendency to resist change characterizes the two extremes involved in converting Federal ADP systems.

The process and difficulty of converting Federal Government computer systems is not well understood. This study of Federal ADP system conversion by the Center for Programming Science and Technology provides information based upon three major sources: the experience of Federal Government agencies who have gone through conversion, interviews with vendors who perform conversion services and provide conversion tools, and available conversion literature.

There are seven chapters and two appendices in this study. The first three chapters comprise a tutorial. Chapter 1 defines conversion, describes the alternative techniques used in conversions and relates conversion to maintenance, new development and portability. Chapter 2 describes the typical conversion process and the various phases of conversion. Chapter 3 describes the ways in which database management considerations affect the conversion process. The next three chapters discuss the information gathered from the sources used in this study. Chapter 4 summarizes the information obtained from conversion specialists who perform conversions and provide conversion tools. Chapter 5 describes the experience of Federal Government agencies who have undergone conversion. Chapter 6 provides a review of significant conversion literature and Chapter 7 summarizes the authors' conclusions. Note: Bracketed numbers ,i.e. [N], refer to references in Appendix B.

There are many very unfortunate stories about the difficulties and costs of conversion. In a 1977 report [23] GAO estimated that $450 million are spent annually by the Federal Government on software conversion. In that report it was also estimated that $100 million of that amount could have been saved. The intent of this study is to bring into focus major problem areas in the conversion process. Having pin-pointed the problems, this study recommends conversion guidelines which will aid Federal Government users in avoiding the pitfalls and exorbitant costs of conversions that have been experienced to date.

This study was performed by representatives from the three Divisions of the Center for Programming Science and Technology: the Programming Science Division, the Application Systems Division, and the Operations Engineering Division. Each of the division representatives provided a different viewpoint concerning conversion: programming languages, database management systems, and computer operations (security, performance).

Federal Government personnel who participated in three large conversion efforts were interviewed along with four commercial conversion companies. One of the four commercial conversion companies visited NBS to present their view of conversion. The questions the participants were asked were very similiar to those in Appendix A. Some of these questions were:

* What was the reason for conversion?
* What can be done to minimize conversion problems?
* What types of automated tools are helpful?
* What are the human problems in conversion?

What Is Conversion?

Conversion can be defined as the process of transporting a computer system from one environment to a different environment while maintaining the functional requirements of the original system. From the user's viewpoint, the system of programs performs the same functions in the old and new environments. The existing environment is referred to as the source environment, and the new environment is referred to as the target environment.

Programs are designed for a particular hardware and software environment. When programs are designed, the best available techniques should be used for program development (modular programming, standardized languages with no vendor extensions, and no unusual tricks that would make programs difficult to convert). Even though these techniques may cause degradation of run-time efficiency, these techniques, along with many others, are currently recommended for new development since they will minimize the efforts of future conversions.

However, many existing programs in Federal Government agencies were designed for, and are operating on, computer systems that are fifteen years old and older, before on-line systems, random access and telecommunications were generally available. Many of these programs use techniques common to second generation computers, such as tape oriented batch systems, programming in assembly language or programming in very early versions of COBOL, FORTRAN, or other languages such as AUTOCODER.

In many cases, these programs perform vital functions for agencies such as providing payments to retirees, students, or agency personnel. Converting these operational programs from the source environment to the target environment can be a difficult, costly, and time-consuming task. Conversion of these systems must, therefore, be justified by existing constraints, such as:

* Obsolete hardware or software - The hardware or software is no longer supported by its manufacturer - for example, parts are unavailable.

* System saturation - There are not enough hours in a day to run all the application programs - i.e., an outgrowth of the existing computer hardware/software environment.

The decision to convert should be based upon sound economic philosophy. Conversion should be more cost-effective than the various alternatives to conversion which are available. Some of these alternatives are:

* An additional hardware system like the existing one can be obtained and run in parallel.

* A more powerful and compatible version of the existing hardware/software system can replace the existing system.

* Compatible timesharing services which would augment the present computer system can be obtained.

* The existing computer system can be augmented with more main memory, newer or more modern peripherals, etc.

* Performance measurement techniques can be used to fine tune the system and circumvent bottlenecks.

Procuring a more powerful computer system which is not compatible with the existing system, or new software such as a database management system or operating system which is not compatible with the existing system, will probably necessitate a conversion.

Techniques In The Conversion Process

Once the decision to convert has been made, there are a number of techniques which can be used in conversion: recoding, reprogramming, and redesign. The conversion literature uses inconsistent terminology to describe conversion techniques. The following definitions were developed for this report and are used consistently throughout:

* Recoding - Each line of code is translated to an equivalent line(s) of code on the new computer system. This translation can be accomplished manually through visual inspection of the code, automatically through specialized software, or by a combination of the two techniques. From the user's and designer's point of view, the system remains unchanged (with the exception of processing time,

which may either have increased or decreased.) One line of code may get translated to many lines of code or many lines of code may get translated to one line of code. The type of transformation that occurs depends upon the source and target languages. Examples are COBOL to COBOL, assembly language to COBOL, and FORTRAN to FORTRAN. If the source and target languages are similar, much of the translation will be one-to-one.

* Reprogramming or functional translation - Some or all new code is produced. All the code is not translated on a line-for-line basis, but the same functions from a designer's and user's viewpoint remain, along with the same algorithms. Different logic is used to program the same functions, resulting in greater efficiency.

* Redesign - A new system specification is required. The entire design of the system may change, resulting in different algorithms to accomplish the same functions. Different program structure and different logic may be used. New techniques such as database management may be incorporated, and the structure of program files may change drastically. The new system specification is related to the previous system because the same functions, from the viewpoint of the user, are produced. Changes in the system are transparent to the user.

Note: All three of these techniques are considered conversion because the functional requirements remain the same. Once the functional requirements change, a new development will be required.

A particular conversion may use any one, or a combination, of recoding, reprogramming, and redesign. Few additional specifications are required for recoding since the existing programs and data constitute the specifications.

There are inherent pitfalls, however, in the choice of recoding as the single conversion method. There is a greater probability that state-of-the-art techniques for improving program efficiency and maintainability will not be incorporated in the target system, ultimately reducing its expected life. Additionally, this method may result in substantial increases in maintenance costs.

The reprogramming method of conversion is used to produce a more efficient system. Many of the techniques described above, such as modular programming and standardized languages can be incorporated into the new reprogrammed system. In order to perform reprogramming, documentation of the functions of the system as well as program code may be required. This documentation may not always be available. For a reprogramming conversion, the combination of functional specifications together with the original programs comprises the new system specifications.

In redesign, system specifications of the new system are required. When this option is chosen, the new system is more likely to be implemented in-house since in-depth knowledge of the source system is needed.

Maintenance And Conversion

Maintenance can be defined as any change made to a system after completion of the initial development, excluding conversion. Both maintenance and conversion involve changes to programs. When does conversion end and maintenance begin, or when does maintenance end and conversion begin? Conversion certainly occurs when programs are being moved to a new operating environment (target). However, what about the changes that occur from the following? These conditions cannot be considered conversion:

* The user requirements have changed, thus leading to enhancements which must be incorporated into the program.

* The performance of the program needs to be improved.

* The program needs to be structured better for easier maintenance.

* Errors which need to be corrected exist in the program.

These four conditions do not appropriately fit under the category of conversion and should be classified as maintenance activities. However, any or all of the above conditions may exist at the same time that a conversion is undertaken, and may need to be addressed to make the source programs work successfully in the target environment. These conditions are sometimes erroneously thought of as being part of the conversion process. Experienced conversion companies tend to make these changes before the conversion begins or after the conversion has been completed in order to prevent the compounding of errors.

New Development Versus Conversion

New development, maintenance, and conversion are software-producing activities which are inter-dependent. The quality of the original design specifications directly influences maintenance and conversion costs. If new systems are designed for ease of maintenance and conversion, maintenance and conversion costs may be significantly reduced.

In the New Software Economics [9], Werner Frank states that maintenance costs surpass the original development effort by an impressive factor of 3 to 5 times, assuming at least a five year operating life for the software. The maintenance costs must be an important consideration in the conversion decision. One factor that affects future maintenance is the quality of the conversion.

Before conversion is attempted, all of the techniques (recoding, reprogramming, and redesign) should be assessed. Before this assessment can be done, information must be gathered concerning the types of programs in the inventory, their future life expectancy, and the functions that will be required in the future. If the future functional requirements match the present system's capabilities, then conversion can be a good option. If the system design is relatively modern, it is more likely that an effective conversion will result. On the other hand, during the NBS conversion study, the members of the conversion team have seen cases of conversion of assembly language programs to COBOL in which the running times are many times greater on the new hardware than on the old hardware. This increased running time resulted from using the old design on the new system.

The decision may be that new system requirements would be better met by a redesign or new development rather than a recoding or reprogramming. Redesign or new development could incorporate more efficient use of hardware and software resources and at the same time incorporate those aspects which make the new system easier to maintain and eventually convert.

Portability And Conversion

Portability is the term used to describe the properties of a program which enable it to be more easily transferred to another environment. Portability should be one of the major concerns in new system design and development. Conversion will be easier and less costly to perform if potential conversion problems are anticipated and planned for during the design of the system.

Conversion is easier when no unusual tricks are added to the program, when standard programming languages like Federal Standard COBOL and FORTRAN are used, and when no vendor extensions are added to the standard languages. Writing programs in the lowest possible level of FIPS 21-1 COBOL is a good beginning in making a system easier to convert. Note: Ensuring portability in this manner may require foregoing the use of more powerful features of COBOL.

In the next chapter we will address the detailed phases involved in the conversion process.

2. THE CONVERSION PROCESS

In order to have a smoothly running conversion, technical and management problems must be overcome. Many of the conversion companies that were interviewed consider the management of the conversion process as the most difficult problem in conversion.

The following discussion concerning the process of conversion is actually descriptive of recoding. When reprogramming is the technique used to accomplish the conversion, and significant changes do not have to be made to the programming logic, then, generally, recoding is done first, followed by the changes. This technique is used to minimize the introduction of errors at critical points in the conversion which can be very difficult to remove.

The process of converting operational programs from one machine to another challenges, educates, and frustrates. Conversion challenges and educates because technical problems arise which require solutions. Problems occur, in large part, because of the singularity of the conversion experience which only occurs, for an installation, once or twice in a decade. In many cases, no training has been given to inexperienced personnel for solving these problems. Conversion experiences can be unique, and some of the techniques developed to solve the problems may be used only once.

Conversion frustrates because of a misunderstanding of the conversion process, the resources and time required, and the uncertain priority of conversion in an organization's data processing environment. Many programmers view conversion as a relatively undesirable activity, perhaps even worse than maintenance. Certainly, conversion does not appear as challenging or rewarding as new design and implementation.

The conversion process can be described in six phases: planning, data preparation, translation, unit testing, system testing, and parallel testing. The basic ideas of the phases of conversion and some of the figures describing each phase were obtained from publications of Rand Information Systems [19].

Planning Phase

Planning is the process of identifying and documenting computer systems, programs, and data files that operate on the source computer system, and making a determination as to their future status on the new computer system. In addition, planning includes the decision as to how the transition to the new computer system should be made.

A detailed and accurate picture of the existing computer resources is a necessity. Each system, each program within each system, and each data file for each program must be documented. Each system documentation should include all data flows into and out of each system. Each program documentation should include the programming languages used and the number of lines of code of each programming language.

The data files should be similarly analyzed by documenting each file, the number of data records per file and the number of characters per record. In addition, documentation of the data record layouts for each file should be included.

The frequency of use of each system, program, and data file should be captured and documented. Programs and data files can then be assigned priority numbers for the conversion or be eliminated altogether. The techniques that can best accomplish the conversion (recoding, reprogramming, redesigning), and the tools that could aid, should be identified.

Estimating the manpower, computer resources, and time necessary to accomplish the conversion is not an easy task. Therefore, a prototype conversion is recommended, in which a representative sample of the programs to be converted is chosen along with the people who will perform the conversion. Once the sample conversion is successfully completed, rough estimates can be made of the manpower and computer resources needed for the total conversion. Reference [17] may be helpful in making these estimates.

Two factors will necessitate adjustments to the parameters derived from the prototype. One is the initial learning phase which will undoubtedly cause the time estimate to be high. The other factor is the interrelationship problems which must be solved when systems are highly interconnected. The prototype may not be representative of system interrelationships, and consequently, the time estimates derived from the prototype would be low. Probably the most critical factors affecting the parameters are the abilities and experience of the people who will perform the conversion. Conversion is likely to be more successful, and accomplished in fewer man-hours, when performed by people experienced in conversion. Experienced conversion people are difficult to find.

After these parameters are obtained from the prototype, a conversion plan can be laid out with time and personnel estimates. The conversion can be separated into groups and scheduled according to priority. Decisions can then be made as to whether the conversion resources are available utilizing in-house personnel or whether the conversion would be better accomplished with outside assistance.

Although the conversion schedule can be compressed to some degree, it is generally inappropriate to compress the schedule during the early stages, when the techniques are being developed. Only when the conversion is progressing well and the techniques are fully developed will additional manpower resources be effective. Additional manpower applied when the project is having difficulty with time schedules may be counterproductive and destroy the progress that has been made.

Data Preparation Phase

Data preparation is the process of gathering all the materials that will be used in the conversion. It is important that the data preparation team have well-defined instructions about the types and sequence of information to gather, and the ways in which to prepare the conversion materials. The information gathered in the planning phase is used to assure that all the inputs to the conversion process (program source listings, input files, record layouts, and documentation) are ready for the conversion. Test data must also be generated to assure that adequate testing and validation will be performed on the converted system. See Figure 1.

Test data, that can exercise at least 70% of the program code, should be generated on the source system. Although the study of testing is currently still in the research stage, the 70% figure is regarded by conversion specialists to be a reasonable rule of thumb. It is desirable to keep the input test data files as small as possible, while still providing adequate testing. A 150,000 character test file is generally considered small enough to yield acceptable run times for the converted program, and still be representative of data that can test 70% of the code. Tools which insert code in a source program for automatically gathering run-time statistics are available. One example of such a tool can measure the percentage of lines of code which have been executed on both the source and target systems.

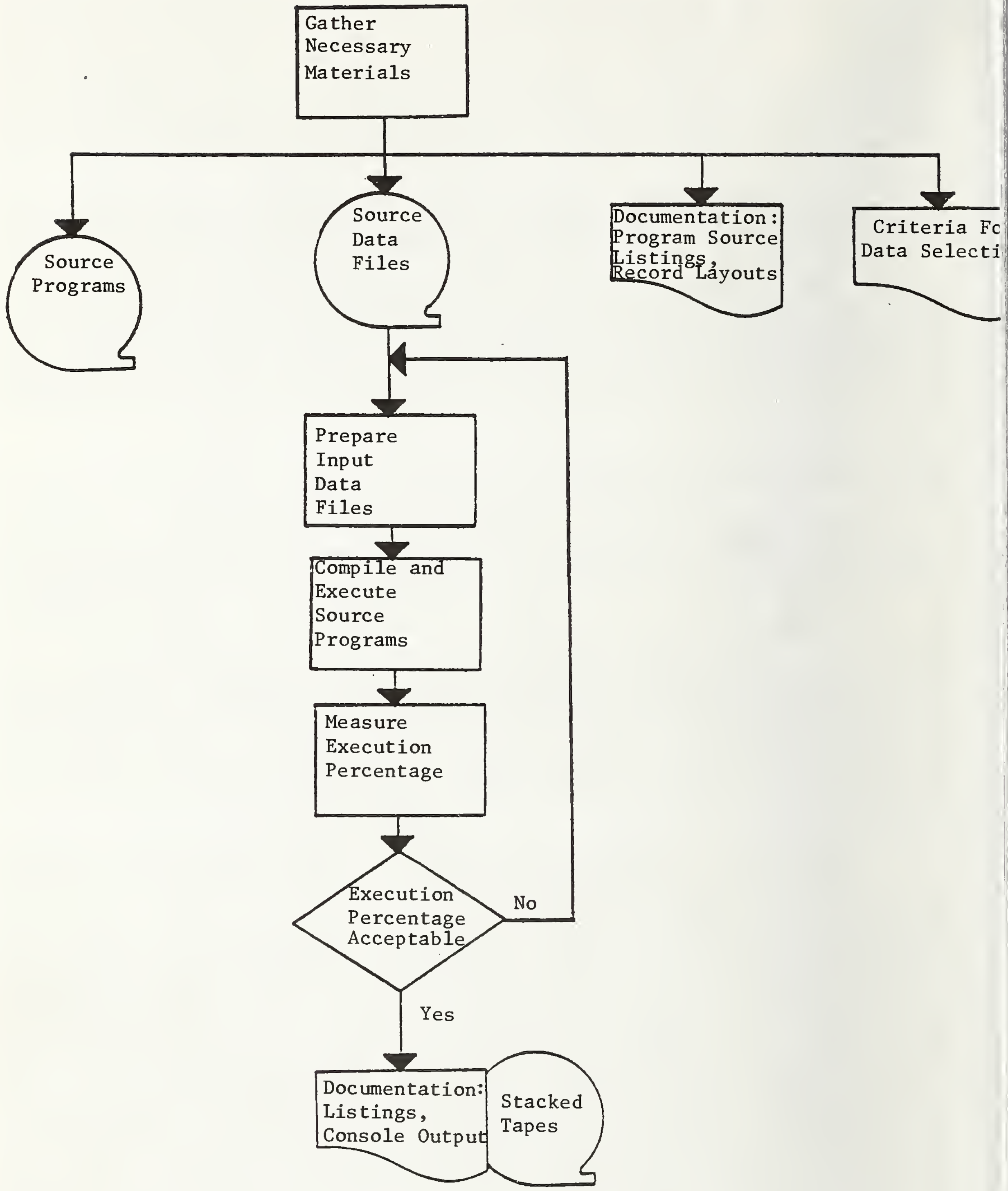
Gather Necessary Materials
Source Programs
Source Data Files
Documentation: Program Source Listings, Record Layouts
Criteria Fo
Data Selecti
Prepare Input Data Files
Compile and Execute Source Programs
Measure Execution Percentage
Execution Percentage Acceptable
No
Yes
Documentation: Listings, Console Output
Stacked Tapes

Figure 1: Data Preparation Phase

The alternatives to using a tool that can measure percentage of code execution are either estimating the percentage of code execution or using the existing production files for test data. Existing files may be quite large. Neither alternative appears to be as effective as a reasonably sized test data file that is known to test 70% of the code, or more.

Listings and test data files are needed for all inputs and outputs for each program. Later in the testing phase, the outputs of each program on the target system will be matched against the outputs of the source system to test and validate the conversion.

After assuring that the test files test an acceptable amount of the code, each source program must be recompiled and run with the test files. All the information pertinent to this execution (i.e. source program listing, all input, output, and intermediate files) must be captured on tape. It is best to write all the information for each program together on a tape. Keeping all the information for one program together provides good control of the conversion materials. These "stacked tapes", together with documentation such as listings and console messages, comprise the data preparation input package.

Translation Phase

The translation phase is what many people consider the essence of conversion. The materials prepared in the data preparation phase are now used along with the conversion plans from the planning phase to perform the translation of the programs, data, and documentation to the new computer system. See Figure 2.

In the translation phase each source program must be translated to run on the target machine. The translation phase is usually done on the new computer system or one very close to it. This translation can be accomplished in part by automated tools and in part by manual corrections. One automated tool, called a translator, uses the old source programs as input and produces new programs whose source code is compatible with the target computer. Generally, automated tools will not make all the changes required. Manual changes are also needed, and in most cases take up most of the translation phase time.

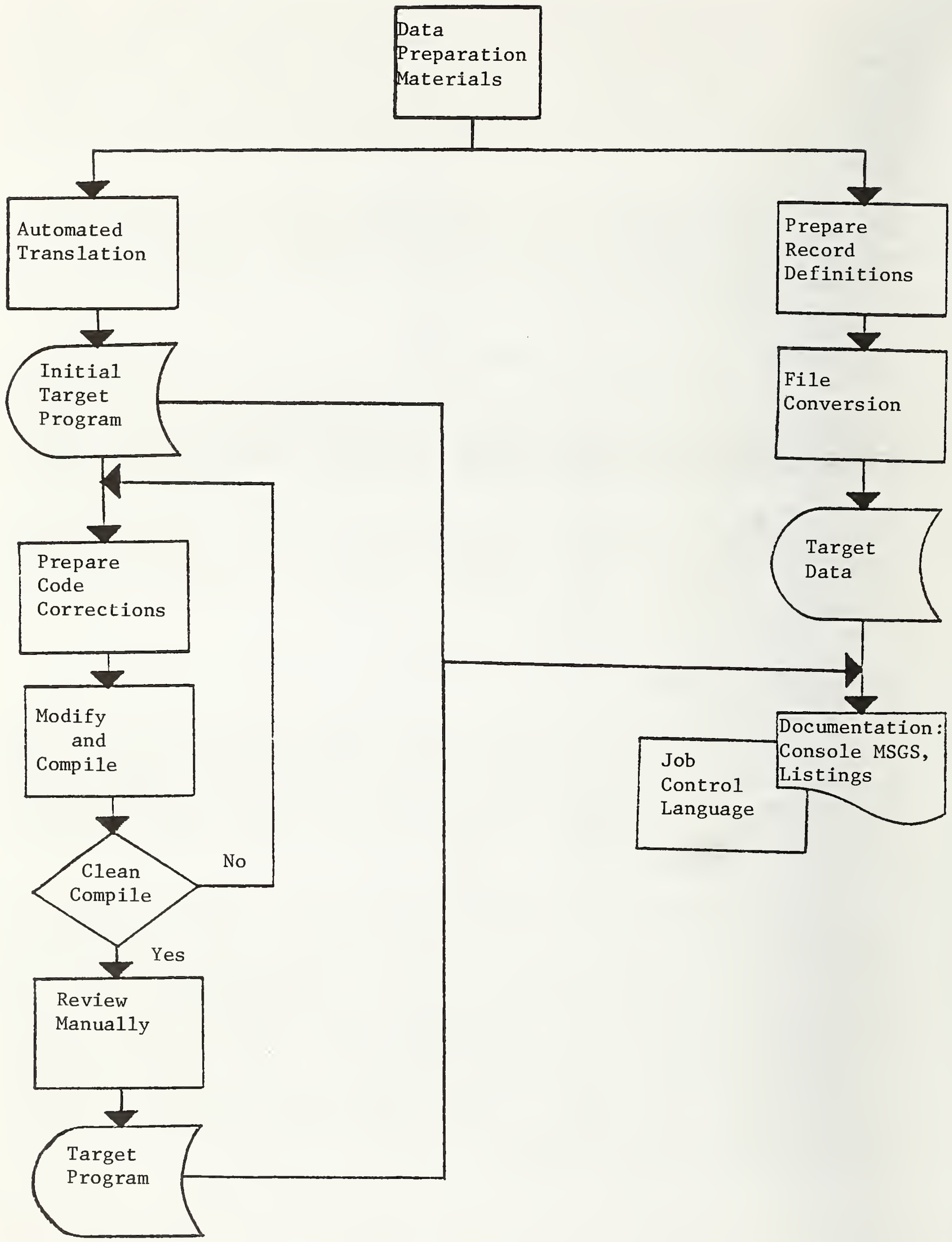

Figure 2: Translation Phase

The data files consisting of the input test files and output files must be converted to the target machine. Changes in character code and file format are done here. File conversion tools are very helpful in performing this conversion. Job control language commands are needed to run any automatic translators and file converters. These commands must be produced in this phase, along with the job control language commands needed for each program. Program and file listings are also produced. One set of documentation is produced after the automatic translation and file conversion, and an updated set is produced after manual corrections are made.

The programs are compiled, manually corrected, and recompiled until a clean compilation results. In many cases, a change in a version of a compiler or operating system can affect the compilation or execution of the translated program. Next, a desk review is performed with two people, one on the original source and one on the translated result, to assure that the changes are correct. The target program and the target format data (input and output test files) along with the the job control language commands are passed to the next phase - the unit testing phase.

Unit Testing Phase

The objective of the unit testing phase is to compile, execute and test each target program with its test data and to match the resultant execution output files with those produced on the source computer. See Figure 3. The test data (including input, intermediate, and output test files) were converted to the target format in the last phase. The intermediate and output files of each program are matched against those equivalent files created on the source computer. File comparison tools are very helpful here. Code corrections are applied where needed, and the program is recompiled and executed until the output data matches the source system output test data.

Every converted program is tested and corrected in this manner. The programs are monitored to assure that 70% of the code is executed. If 70% of the code is not executed, additional test data may be generated. The remaining code that is not tested is desk checked.

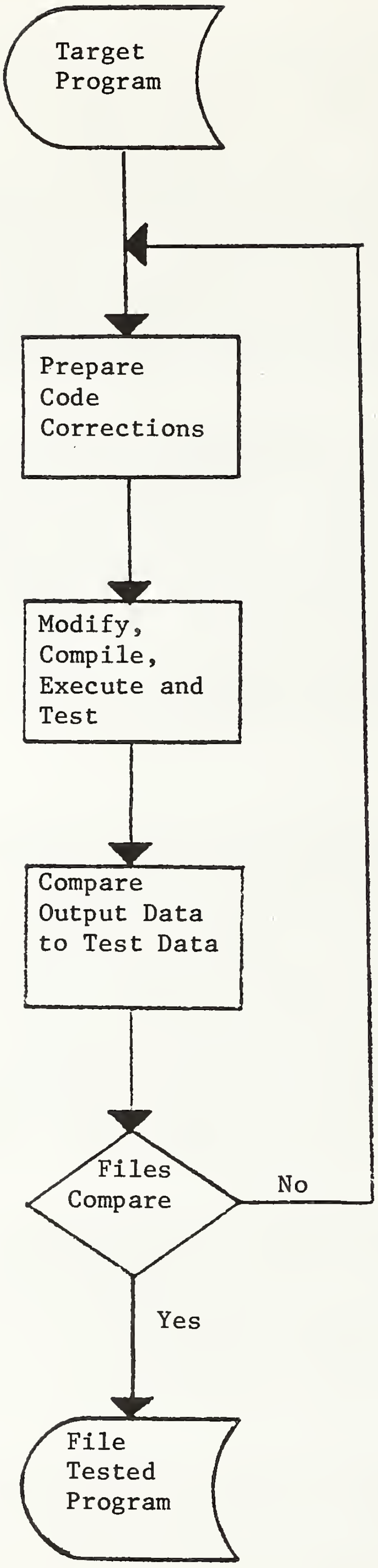

Figure 3: Unit Testing Phase

System Testing Phase

The system testing phase continues the testing beyond the unit test. See Figure 4. The interactions of the various programs which form the system are now tested. In the system test, the job control language commands necessary to execute each system are developed. The programs are compiled and executed as a system. The outputs are compared to the test outputs of the source system. If errors are detected, appropriate corrections are made to the code, data, or job control language commands. The programs are recompiled and re-executed until a perfect execution and comparison is accomplished. If required, the system may be sent back to the unit testing phase.

Parallel Testing Phase

After system testing is completed, parallel testing begins. See Figure 5. The production job control language commands are prepared and the production files are converted. The system is updated to incorporate production maintenance changes made during the conversion. The programs are compiled and executed and the outputs are compared to the existing production system. The process of code correction, compilation, and execution is continued until the system has a perfect execution comparison. The new system continues to run in parallel with the source system until production cutover.

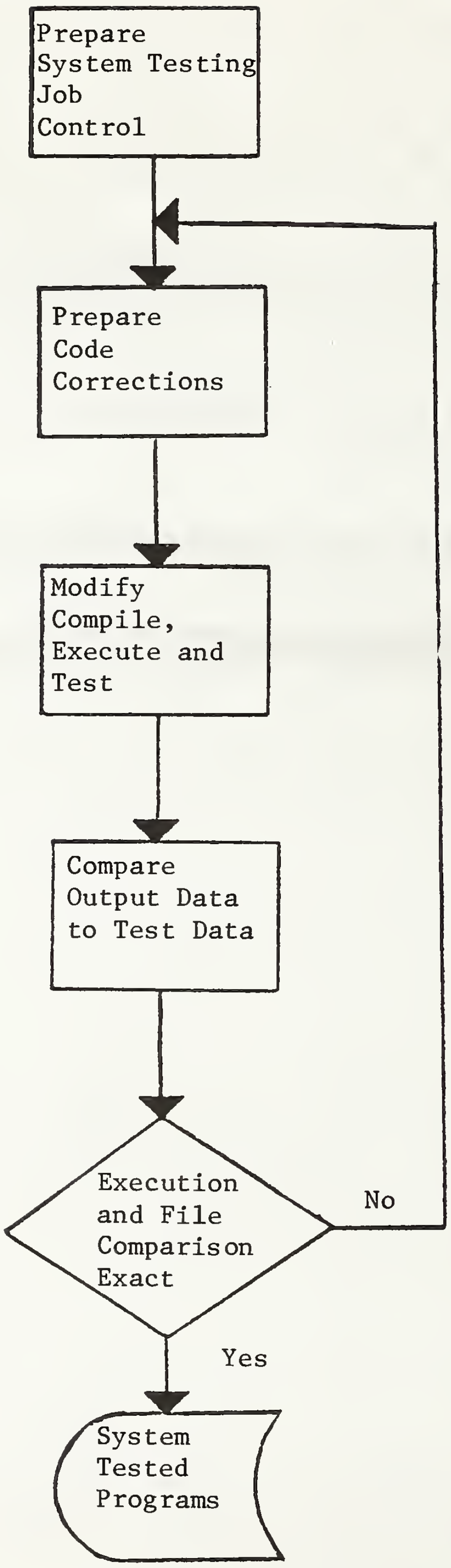
Prepare System Testing Job Control
Prepare Code Corrections
Modify Compile, Execute and Test
Compare Output Data to Test Data
Execution and File Comparison Exact
No
Yes
System Tested Programs

Figure 4: System Testing Phase

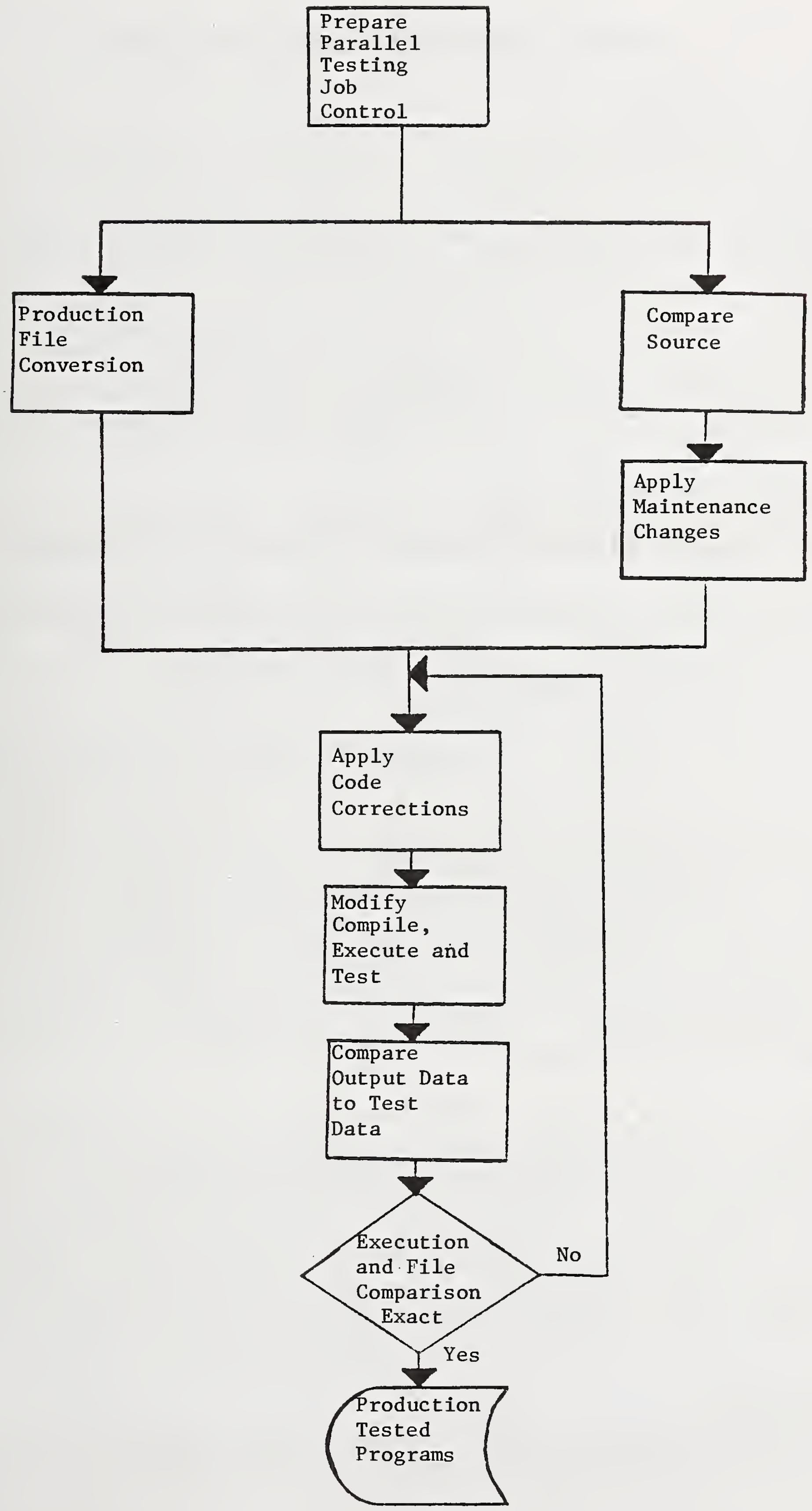
Prepare Parallel Testing Job Control
Production File Conversion
Compare Source
Apply Maintenance Changes
Apply Code Corrections
Modify Compile, Execute and Test
Compare Output Data to Test Data
Execution and File Comparison Exact
No
Yes
Production Tested Programs

Figure 5: Parallel Testing Phase

3. DATABASE MANAGEMENT CONSIDERATIONS

The batch file environment is by far the most prevalent computer system environment undergoing conversion today. Many of these computer systems that are undergoing conversion were first installed fifteen years ago and longer. Consequently, the viewpoint of the people interviewed, and the suggested techniques for handling conversions, are mostly oriented toward batch file environment systems. Future conversions will in all probability be more concerned with increasingly complex systems, such as those involving database management systems (DBMS). A conversion study would not be complete unless it discussed some of these technical advances which are being used in developing today's systems.

Database Management Systems

Some general characteristics of DBMS's are that they are proprietary, written in assembly language, optimized for a particular hardware/software environment, and incompatible with other DBMS syntax and semantics. These characteristics form massive technical and contractual barriers to converting the DBMS itself to the target system.

The vast majority of commercially available DBMS's operate on a single vendor's hardware system. There are presently only a few exceptions. One database management system runs on ten different hardware systems, one system on three hardware systems, and a few on two hardware systems. However, in general, when moving a system, which utilizes a DBMS, from one environment to another, the degree of difficulty of the conversion is increased compared to moving a system which does not utilize a DBMS. The major reason for this increased difficulty is that incompatibilities exist among different DBMS's. The syntax and semantics used for one DBMS are not acceptable to another DBMS.

DBMS's, in general, have different data definition languages, data manipulation languages (COBOL, FORTRAN, and other higher level languages), and query languages. Except in those cases where the same DBMS is implemented on several different hardware systems, many programming changes are required in order to accommodate the target DBMS. Furthermore, redesign techniques are usually necessary with the target DBMS application in order to provide adequate performance.

Data Definition

DBMS's support three major data views or models: network, hierarchic, and tabular (relational). A network structure is a more general data structure than a hierarchic structure. If the target DBMS data definition language and data structure is at least as general as the source DBMS data definition language and data structure, then the data definition conversion will be easier. Translation tools have been used to translate portions of one data definition language to another. However, in most cases the process is done manually.

Data Manipulation Language

Various DBMS's provide different levels of Data Manipulation Language (DML) capabilities. For example, a DML statement in one DBMS is only capable of selecting at most a single record, while a DML in another DBMS can select 1000 records. Converting programs between these DBMS systems is much more involved than the simple replacement of one DML statement for another. Additional code is needed to form equivalences between the two DMLs. Specialized automated tools have been used in translating portions of one DML to another DML. However, no general tools are available, and tools that are available generally are not capable of doing major portions of the work. Developing general solutions to the DBMS program conversion problems must be considered an area of present and future research.

Query Language

Many DBMS's offer the users the capability of writing predefined programs in an English-like language and invoking these programs with parameter substitution. This capability is usually an extension of the query capabilities offered. These predefined programs must be rewritten since little commonality exists among DBMS query languages or predefined programs.

Data Conversion

In a database environment, all aspects of the conversion create some special difficulties. However, data conversion has its own set of problems. The data in a DBMS is structured. The logical structure of the data, which can be network, hierarchic, or tabular (relational), must be represented in some linear representation on tape for most conversions. There is no present standard or procedure for representing structured data in a linear fashion. Presently, vendors represent their data in a unique manner which would be of little or no use to another DBMS. Moving from a network to a hierarchic system, for example, leaves open questions as to how the network would be represented in the less complex hierarchy. This representation is generally done via redundancies in the hierarchic structures. Problems occur whenever more complex structures are mapped onto less complex structures because either structural information that cannot be directly modeled in the new DBMS must be dropped, or the information must be represented in a way permissible but not necessarily desirable in the target DBMS.

Effectiveness

For a DBMS to be effective, it must closely support the application, and make efficient use of the hardware/software environment. Therefore, most DBMS conversions are generally redesigns. Different data definitions, physical access methods, or additional indices may be required to keep performance at an acceptable level and to maximize hardware/software system utilization.

Conversion from a manual environment, file environment, or DBMS environment to a DBMS environment relies heavily on all the techniques, knowledge, and good management necessary to establish a DBMS successfully in an organization. Good database management techniques rather than good conversion techniques seem to dominate. Some good management techniques described in the Data Base Directions Report [2] are:

* Establish review points
* Involve end users
* Keep the scope reasonable
* Encourage a prototype
* Shift responsibility to the DBMS experts
* Phase the implementation
* Don't be the first to use a new technology
* Get adequate management commitment
* Select an important portion of the ultimate system and have visible results in four to six months
* Provide adequate training for your technical staff

* Orient and educate your users

* Implement a data dictionary/directory

* Obtain help from the DBMS and hardware vendor

Despite the problems a DBMS creates in the conversion process, there are many advantages in using a DBMS. With a DBMS the user can carry analysis to new levels of sophistication which were considered too difficult or unreasonable in a file environment.

In a conversion that involves a DBMS, personnel requirements are much more stringent. A complete understanding of the target DBMS and its peculiarities is required. Adequate training in the target DBMS is a necessity, and those who were not trained in DBMS concepts previously may feel threatened by the change.

After hearing the special problems involved in DBMS conversion, one might wonder why users would want to utilize a different DBMS while retaining the same hardware/software environment. In the Data Base Directions report [2] some reasons for this change were given:

* Improved functions (more complete)

* Better performance

* Improved query capability

* Development of richer data structures

* More efficient usage of the computer resources through decreased cycles and/or space

* Improved or added communication functions

* Availibility of transaction processing

* Distributed processing capability

* Abandonment of a complex data structure

Summary

A DBMS environment can significantly increase the difficulty of conversion. For example, a conversion that involves COBOL programs utilizing a DBMS can be as much as ten times more costly than conversion involving solely COBOL programs in a file environment. General DBMS conversion tools do not exist. The general solutions to DBMS program conversion problems are still in the research stages. The technology to assist with DBMS data conversion problems exists, yet no common data interchange forms are available. DBMS conversion in general has not reached the level of refinement that a COBOL to COBOL file environment conversion has achieved.

There are many DBMS's available in the marketplace with little commonality among them. Conversions involving a DBMS require significant knowledge of the target DBMS. However, additional capabilities provided by the DBMS in many circumstances outweigh the increased problems it causes in conversion.

4. CONVERSION EXPERTS AND TOOLS

In order to get an implementor's perspective on conversion, NBS interviewed four companies specializing in different aspects of conversion. These companies will be referred to as A, B, C, and D.

4.1 Company A

Company A employs under 500 people. About 100 people work on conversion projects. Company A has performed conversion services for several Federal agencies.

Company A prefers to do as much work as possible on their own premises in order to control the environment. They have their own computers, and will rent time on target computers that they do not own. They offer fixed price/fixed schedule contracts and provide a warranty for the completed package. For a typical job, Company A will perform all of the translations and unit testing and will assist the client with the planning and data preparation. The client performs the remainder of the effort.

Company A feels that management problems are more difficult than technical ones. Management considerations in conversion involve:

* controlling large volumes of materials
* coordinating 20 to 40 runs/program
* ensuring quality and standards
* limiting operational overhead
* minimizing parallel costs
* maintaining good morale
* training

<u>Company A's Working Environment</u>

The organization of Company A's conversion team is very structured. This rigid structure is characterized by specialized assignments for each individual, and productivity statistics about each programmer. Each individual working on a conversion project has a well-defined and unique piece of the conversion assigned to him. He is concerned only with his piece and knows little of the overall conversion effort or strategy. He is not only logically, but physically removed from the rest of the team working on conversion, since each member of the team works in his own individual cubicle. Interaction and communication among members of the team are not encouraged. Batch, rather than interactive processing, is encouraged since machine down time in an interactive mode leads to lost productivity.

Company A's Conversion Philosophy

Company A considers that the source code for the programs to be converted and the data files constitute the input to the conversion process. Program documentation is neither required nor desired by Company A. Comments within the source code are disregarded. In most cases, Company A considers an understanding of the source program unnecessary for them to perform a conversion.

Some other thoughts by Company A which give some insight into their philosophy are listed below:

* The biggest problems in converting COBOL programs involve differences in file structure, differences in syntax, and different collating sequences.

* FORTRAN programs are much more difficult to convert than COBOL programs. The author(s) or individual(s) responsible for the FORTRAN program must provide assistance to the conversion team for FORTRAN conversions. Precision, which varies due to different word sizes on different machines, is a major problem. Difference in word sizes can also cause problems with respect to the different number of characters stored in a single word.

* Company A has a complex data base for conversion estimates, to which they attribute the high degree of success they have attained in projecting conversion costs. The data base is not complete for FORTRAN programs.

* Tools can be helpful in conversion, but they are not an end in themselves. Programmers must understand what they can do, and the decision as to whether or not to use a tool must be an economic decision. There is a need for more general tools that can be modified for each job.

* Company A defines completion in conversion in terms of files and program testing. 100% of file conversion must be complete, and 70% of the statements in the programs must be executed.

* Production runs have a negative impact on conversion. Company A controls the situation by bringing in their own computers.

* Performance requirements, such as execution speed, throughput, etc. are not usual considerations for Company A. However, if these considerations are in the RFP, Company A will adhere to them.

* Agencies undergoing conversions should have an up-to-date test file which can be used during the testing phase of conversion.

4.2 Company B

Company B employs approximately 500 people. Twenty percent of Company B's resources are used for management consulting, 40% for programming at the customer site and 40% for performing conversions. Company B said that they would not bid on Government RFP's because the Government is unrealistic about what they expect to get within a fixed length of time and the Government is lacking in expertise to evaluate the quality of the code produced. Additionally, Company B feels that the people problem in conversion is very difficult in the government. Government employees are not willing to face up to conversion because they are unwilling to face change.

Company B's Conversion Philosophy

Company B's conversion philosophy is quite different from Company A's. Company B seems to emphasize a level above line-for-line conversion. Their emphasis is on quality control and they claim that their converted programs are more maintainable and readable. Company B believes that variable and procedure identifiers, as well as statement labels, are easier to maintain if the names are understandable rather than coded. They maintain their own internal programming standards to which their programmers adhere. They emphasize documentation and aim to produce user documentation along with the converted software. Company B defines a successful conversion as follows:

"... a technical operating environment moved to another machine with little break in functioning and leaving it in an enhanced state."

Company B employs programmer/analysts, rather than just programmers. People working on conversion interface with the project as a whole, all the way from design through implementation. This is in marked contrast to Company A, where each individual working on conversion is concerned only with the segment which has been assigned to him.

Company B offers the following sequence for a conversion:

* Do a conversion study.

  . Develop contingencies.

  . Delineate scope of project.

  . Perform a technical survey - note hardware, proprietary software, line counts, languages, programs. Note: Company B either does the survey itself or trains customers to do the survey.

  . Identify conversion alternatives.

  . Define factors - What it takes and how much complexity.

  . Run through estimation modelling program. Note: Company B owns a proprietary program to do this job.

* Estimate time/cost.

* Produce alternative summary.

Company B will perform benchmarks at a cost of less than $50 per program before contracting a conversion job. The benchmarks are limited to 10 representative programs of the complete job. Two average programs, two easy programs, and two of the most difficult programs in the system must be included within the 10 representative programs.

Company B believes that the following must be considered before buying a new system:

* Longevity of hardware.

* Cost of conversion.

* Applications.

* Maintainability.

Company B's thoughts about conversion:

* Freeze program for language translation.

* Character set changes are difficult.

* Autocoder should be rewritten, not translated.

* Some compilers have extensions which make programs non-standard.

* Do not let hardware vendors do conversions.

* Fixed price conversions are dreadful.

* Compare costs for contracting with a vendor and with another government agency.

* About 70% of conversions today involve COBOL (to and from).

* A conversion might reveal serious bugs in a program which can be very embarrassing to the customer.

<u>Company B's proprietary tools are</u>:

* <u>Translators</u> -

  RPG to COBOL (full ANS COBOL on any computer)
  DOS COBOL to OS COBOL
  DOS ALC to OS ALC
  360 BAL to 360/370 OS ALC
  SYSTEM 3 COBOL to OS COBOL

* <u>Utility software</u> -
  Blocking/deblocking records
  File matching

4.3 Company C

Company C visited NBS to present its view of conversion. Company C provides a wide variety of services including consulting, systems design, programming support and conversion services. They define software conversion to be "the total migration of an automated data processing system from one computer's hardware and operations environment to another, including: software, data base files, job control language, and operating procedures." Company C contrasts software conversion with software development by pointing out that software conversion entails: a different technical/management relationship, a different operating environment, different staffing considerations, and different risks.

Company C's Conversion Philosophy

Company C breaks down the conversion process into three distinct stages: conversion planning, workload conversion, and production testing. Each of these stages is, in turn, broken down into more specific stages. The percentages of time shown are those experienced in the company's latest conversion project.

* Conversion Planning (26%)
  - .Development of Conversion Plans (8%)
  - .Preparation of Conversion Work Packages (2%)
  - .Ensure 100% COBOL/FORTRAN (1%)
  - .Test Data Preparation and Validation (15%)

* Workload Conversion (54%)
  - .Software Translation (3%)
  - .Clean Compile (10%)
  - .Unit Test (20%)
  - .Production Preparation (7%)
  - .Systems Test (10%)
  - .Production Data Conversion (4%)

* Production Testing (20%)
  - .Insert Maintenance Changes (2%)
  - .Full Systems Test (6%)
  - .Parallel Operation (10%)
  - .Production Cutover (2%)

Company C concentrated on the procurement aspect of conversion more than did other software firms with whom we spoke. They presented a plan detailing the suggested criteria one would use to evaluate proposals for conversions. Their evaluation criteria were broken down into the following categories:

* Personnel Qualifications - 30 points
    - .Relative Experience of Project Team - 20 points
    - .Mix of Skills of Proposed Personnel - 5 points
    - .Relative Experience of Project Manager - 5 points

* Experience of Firm and Corporate Management - 20 points
    - .Degree of General Corporate Experience in Software Conversion - 10 points
    - .Level of Management Participation in the Conversion Project - 5 points
    - .Quality, Extent, Depth and Variety of Prior Experience - 5 points

* Technical Approach to the Statement of Work - 35 points
    - .Degree of Use of Conversion Aids Proposed - 10 points
    - .Conceptual Soundness of Approach - 10 points
    - .Schedule and Plan to Implement the Approach - 10 points
    - .Use of Project Management Techniques - 5 points

* Responsiveness and Thoroughness of Proposal - 5 points

* Innovative Approaches in Proposal - 10 points

4.4 Company D

Company D's views were solicited because of their extensive experience in developing tools to aid in language translation.

The following describes automated tools developed by Company D for use in language translation:

* A meta-language translator - a machine-independent tool for developing language processors. This is a proprietary program which has been used by a government agency with non-disclosure provisions. A meta-language (a modified BNF) is used for inputting the syntax and semantic primitives of the host language. The meta-language translator can be considered a building tool for language translators or a tool to develop a tool.

* A meta-language assembler - developed under contract with a government agency for translating one assembly language to another assembly language.

* A program evaluator and tester - used to validate a

converted system. It assists in debugging, testing, and documenting FORTRAN programs by inserting code for automatically gathering run time statistics. The program evaluator and tester flags conversion problems, such as differences in word size.

* A tool for developing a language to analyze operating performance - like the meta-language translator, this tool is a language translator builder.

* A high order language evaluation tool - checks behavior of language constructs during program development.

* A high order language definition language - a meta-language to provide a medium for expressing the following: definitions of high order language words for inclusion in a dictionary, information on syntactic order and precedence, information to establish proper binding of words, and semantic information for code generation.

The most difficult problems encountered by Company D in translation are listed below:

* Multiple assignments - e.g., A,B =C

* Differences in the value of loop variables on exit from the loop after N iterations.

* Different scoping rules.

* Mixed mode expressions.

* Different operator sets.

* Different rules of precedence.

* Differences in data structure - e.g., 1's complement vs 2's complement.

* I/O differences.

* Reserved words in target language used as variable names in the host language.

* Different character codes.

* Different collating sequences.

* Algorithms which are required to handle semantic differences.

* Lexical differences - can be especially difficult when a

one-to-one correspondence does not exist.

* Structural differences (e.g., PASCAL's rigid block structure) can cause problems.

Company D feels that syntax differences are the easiest to solve.

5. FEDERAL GOVERNMENT AGENCIES: CONVERSION EXPERIENCE

A vitally important perspective on Federal Government conversion problems can be obtained from Federal Government agencies who have undergone a large scale conversion. In each case described below, the agency was very candid and helpful in describing its conversion experiences. Many of the results of the agencies' conversions were not totally satisfactory, yet each agency was willing to share its failures, as well as its successes, in order for other agencies to benefit from its experiences.

It should be noted that although the agencies' files contain confidential information and are vulnerable to theft, the extent of their software security techniques is not enough to cause any concern for the conversion process. Passwords and authorization lists (read only, write only, etc.,) are used. Other than that, the implemented security measures are physical.

This chapter identifies computer hardware systems as necessary by their trade names to provide an understanding of the conversion effort. This identification in no case implies a recommendation or endorsement by the National Bureau of Standards.

Summaries of the experiences of three different agencies follow. These agencies will be referred to as A, B, and C.

5.1 Agency A

The history of changes that occurred in Agency A's computer resources is probably typical of changes that took place in most Government agencies during the 1960's and 1970's. Agency A entered the computer field near the end of the 1950's after acquiring a small scale computer. In the early 1960's, an IBM 1401 was acquired to replace the original computer. By the middle of the 1960's, two hardware/software systems similar to the IBM 360/370 architecture were acquired. These computers were the source computers in their recent conversion described below. Old applications were converted from machine to machine, and in many cases improved. New applications were added.

By the middle of the 1970's, four large application systems were running on the two source computers. These systems were high volume batch file systems, utilizing magnetic tape and sequential file processing. Very few data were stored on disk. These systems were very similar, in many cases, to the original designs that were made for the IBM 1401.

As time passed, Agency A found that their computer operations were exceeding the capacity of their hardware/software system. In addition, the hardware vendor was no longer maintaining these machines. The computer system had become obsolete.

Plans were formulated to upgrade the hardware/software environment to take advantage of the latest technology. Interactive processing and a database management system were elements of the new plan. A Dual Honeywell Series 60 computer was purchased at a cost of approximately two million dollars. It was estimated that it would take three years to make the existing application systems operate on the new hardware.

Agency A found themselves in a very difficult situation. The application system they possessed needed to be redesigned, but they did not have the manpower to convert the system or even to prepare specifications for the desired redesigned system. Functional specifications did not exist for the source system, and documentation was not complete. These circumstances forced Agency A to contract out for the conversion effort, with recoding as the chosen conversion technique. Agency A planned to redesign the system with in-house personnel after the conversion contract was completed.

Planning For The Conversion

The following steps were taken early in 1975 to plan the conversion from the source computer to the Honeywell Series 60:

* An inventory was taken on all subsystems - program names, functions, file sizes, record layouts, utility programs, interface programs, etc. were documented.

* Primary users were consulted for comments on output and where the output could be improved.

* RFP's were prepared, which included:

    . Number of lines of code by programs.

- The number and types of files.
- The kinds of utilities required.
- Source code.
- File formats.

It should be noted that neither the RFP's nor the contracts specified the operating efficiency of the final product. This turned out to be a mistake since a translator is not geared to fine tune the logic of the program in order to increase efficiency. The use of translators to perform the majority of the code conversion dramatically increased the size of the programs. One reason for this increase in size is that the source computer is character-oriented and the Honeywell is word-oriented. Logical data (yes/no) which requires eight bits in the source computer (one byte) was converted to a full word on the Honeywell. In general, character processing on a word-oriented machine is very inefficient. Program efficiency, however, could have been guaranteed if the contracts had included execution time constraints, and other performance characteristics.

Two companies were given contracts to convert portions of the application system. The first contractor was to convert 37 smaller systems, including the payroll system, at a cost of $150,000. The majority of these programs were in COBOL, with about one-third being written in assembly language. Approximately 100,000 lines of code constituted the 37 systems, making the cost for converting a line of code less than two dollars.

Two separate contracts were awarded because there was a logical break in the conversion work, with the second job being extremely large and significantly different. The company awarded the second contract, having had much experience in large conversion efforts, was better able to perform the second job. Information pertaining to the conversion under the second contract follows.

Second Contract

This job alone contained 4 million records which were stored on approximately 1500 reels of magnetic tape. The source system encompassed some 650 files, containing about 90 different formats. File and display formats had to be reformatted, necessitating changes to programs. The source code consisted of some 270,000 lines of ALC and 150,000 lines of COBOL, some of which was code which could never be executed. About seven agency staffers prepared the test data, a task which took one and a half years to complete.

The terms of the contract included the following:

* The contract was to run for three years.

* The company had translators which converted directly from the source machine to the Honeywell Series 60. They also had translators which converted from the source machine to the IBM 360/370 as well as translators which converted from the IBM 360/370 to the Honeywell Series 60. The company chose to utilize the two-step approach of converting from the source machine to the IBM 360/370 and then from the IBM 360/370 to the Honeywell Series 60. First, all code (assembly and COBOL) was converted to COBOL from the source computer to the IBM 360/370. The second step was a conversion from the IBM 360/370 (COBOL) to Honeywell Series 60 (COBOL). The conversion was to be performed at the contractor's home office.

* The contractor was to modify and optimize (e.g. eliminate excess passes through the files) the programs after the conversion.

* For testing, a criteria of 70% code execution was established.

Amendments had to be added to the contract as time progressed. Because of the ramifications of the line-to-line conversion from the assembly language to COBOL, the converted system took 40% more time to execute than the old system. Since the contract made no reference to execution or turnaround time, the contractor could not be held accountable for the poor efficiency of the converted system.

The costs for this contract are summarized below:

$ 366,000	optimization
28,000	systems analysis
1,192,000	actual conversion
82,000	Implementation support on-site
$1,668,000	
100,000	Installation and maintenance
$1,768,000	Total

In addition to the above costs, labor costs were incurred by ten agency employees working on the project for two years.

<u>New Procurement</u>

The magnitude and ramifications of the conversion effort were underestimated by Agency A. The converted system took 40% more time to execute than the old system. Something needed to be done since the capacity of the Honeywell had been exceeded. Rather than augmenting their Honeywell computer by adding greater capacity in the form of more memory or faster peripherals, Agency A has recently opted to procure a new computer to alleviate the problem of a saturated hardware/software system. Agency A feels that this acquisition will subsequently make the Honeywell system available for other useful purposes.

Agency A Recommendations and Comments

The following are the recommendations and comments from Agency A personnel, based upon their experiences in this conversion effort, and do not necessarily reflect the authors' opinions:

* Technical expertise should be available in the Federal government to assist agencies.

* Top management always underestimates the scope of effort required for a conversion.

* Contracting out 100% of the conversion is not possible. Only about 30% to 40% of the work can usually be done via contract. The remainder of work is done in-house, and consists of the most difficult tasks.

* Some factors causing their problems were:

  . Application systems were up to 10 years old.

  . Systems were poorly documented.

  . Many changes in system design, which were not reflected in the system documentation, were made over the years, making the programs difficult to modify and convert.

  . Agency personnel did not have detailed knowledge of the application systems.

  . Contractors were not familiar with the system functions.

    . Moving from a character oriented computer to a word oriented computer created horrendous problems during file converrsion.

* New development is better done in-house.

* Administering, monitoring and testing are important tasks.

* Although Agency A has language standards, they still use vendor's extensions.

* Database management systems, similar to an end-user oriented and an inverted file type, should become standard.

* Standards on the data element level would be useful. Definition of elements should be consistent.

* Hardware costs are easier to estimate than software costs.

* The magnitude of the conversion effort was underestimated.

* Contracting out a conversion may cause a moratorium on modifications to the system.

* Conversion is a specialty, a unique talent.

5.2 Agency B

Agency B underwent a conversion of a large batch financial system. Agency B personnel who were responsible for the conversion did not understand the reason for the conversion. Their application programs, written in a mixture of COBOL and assembly language, were successfully running on an IBM 370 Series. Even more puzzling to Agency B personnel was the new hardware for the target system - a Honeywell Series 60, which was estimated to be 3 times slower than the IBM computer it replaced.

Agency B had about 200 programs consisting of approximately 600,000 lines of code. About seven percent of the code was assembly language. Like Agency A, the original programs were written many years ago (approximately twenty), have many patches, and have evolved from several conversions and additions from previous years. Because most of the staff members were new to the agency (a typical characteristic of computer groups today), the people interviewed were only knowledgeable of the most recent conversion, which is discussed below.

A lack of personnel and a short time frame allotted for the conversion caused Agency B to contract out for conversion services. The RFP went out with the intent of awarding a fixed price/fixed schedule contract. The contractor was to supply the following:

* The guidelines for the data preparation phase, which was to be performed by Agency B.

* Translations, using their proprietary translators to convert from the mixed COBOL 68 and assembly enviroment to COBOL 74 on the target machine.

* Comprehensive testing of the system.

The contract was awarded, and in early 1978 the contractor began the conversion effort. The contract called for the conversion of all the programs first to COBOL 68 and then from COBOL 68 to COBOL 74. However, the translator to convert directly to COBOL 74 became available prior to the translation phase, and was, therefore, used. The programs were delivered in stages between the middle of 1978 and the early part of 1979. The total cost of the contract was $1.8 million (approximately $3.00 per line of code).

Problems Encountered During The Conversion

Agency B performed the data preparation, which turned out to be a horrendous job. Documentation for the source system was not available, and most of the programmers were new to the agency and unfamiliar with the programs and files. In addition, the guidelines on data preparation provided by the contractor were inadequate.

Because of the difficulties encountered in the data preparation, Agency B was three months late in turning over the materials to the contractor (program source listings, input files, output files, and source programs on tape). This slippage in time caused the vendor to delay the completion date. The vendor has asked for additional payments because of this delay.

Results Of The Conversion

The conversion was completed by the contractor. However, the converted system ran ten to fifteen times slower than the source system. Consequently there were not enough hours in a day to run the converted system. Part of the reason for the increased running time was the difference in hardware speed - the target computer was 3 times slower than the source computer. Another major reason was the inefficiencies caused in converting assembly language programs and their files to COBOL programs and COBOL files. A major cause of the inefficiency was the input-output processing. In the assembly language version variable length records were moved in blocks of 3000 characters. Equivalent COBOL statements moved data one character at a time.

Testing was also a problem. The small representative test files proved to be inadequate in testing. When the production data was run through the programs, new errors were found. For this reason Agency B used the production files for test files, as well as for maintenance purposes.

Because the contractor's staff worked at their own headquarters and not at the same site as Agency B's staff, communication problems developed. The contractor used a slightly different version of the hardware vendor's software; the difference caused problems during installation. Many of these problems were aggravated by the extremely large size of Agency B's programs.

Another problem occurred in job control language. There was no direct correspondence between job control language on the IBM and Honeywell systems. For example, one cannot concatenate files on the target operating system. In some cases, Agency B had to write software to perform functions that the operating system performed on the source machine.

Agency B Recommendations And Comments

The following are the recommendations and comments from Agency B personnel, based upon their experiences in this conversion effort, and do not necessarily reflect the authors' opinions:

* Avoid conversion when possible. A conversion should only be attempted when time and resources are available.

* Data preparation, translation, and testing should all be done at one site.

* Consideration should be given to the tradeoffs between transportability and efficiency of the resultant code.

* Assembly language programs should be converted to assembly language programs.

* There is an uncertainty that generalized tools are the complete answers for conversion problems. To be useful, tools have to be very specific.

* The target machine's compiler and operating system should be thoroughly tested. Newly released versions of software are dangerous, and cause problems in conversion.

* If possible, let the vendor do the data preparation.

* Before a job is declared complete, the testing should be thoroughly done.

* A fixed price/fixed schedule contract is not recommended.

* The RFP should require volume testing and timing tests as part of the acceptance criteria.

Agency B also recommended the following standards or guidelines:

* Data preparation guidelines

* Programming standards for structured program design

* Programming standards for efficiency

* Guidelines for performance tuning and optimization

* Guidelines for choosing between recoding and redesign conversion techniques

* Guidelines for preparing an RFP (Request for Proposals). It is important to write the RFP properly. Efficient programs should be included as deliverables. The running time should be specified in the RFP.

* Guidelines for managing a conversion

5.3 Agency C

Like Agency B, Agency C also has a large financial system which underwent conversion. Unlike Agency B, Agency C's system is on-line and uses a DBMS. In addition, Agency C's conversion was performed by in-house personnel. The system was converted from an IBM 370 Series to a Honeywell Series 60. The database/data communications environment was changed from a network DBMS primarily used with a programmer-oriented data manipulation language and its communications monitor to a CODASYL-based DBMS which comes with its own data communications monitor.

Background

Around 1969 or 1970, Agency C contracted with a private vendor to develop a pilot program for automating its records. The pilot program, a totally automatic process for claims processing, first tied in two city offices with updating capabilities. Three cities were later added with read only authorization.

By 1976, they were ready to go into full production. An RFP was issued in two parts, one part for procuring the mainframe and one part for procuring the terminals. All products were tested (benchmarked) and off-the-shelf. The criteria for passing the benchmark test - response time less than five seconds - was decided by Agency C. Unfortunately, because of regulations governing fair competition practices, they were unable to benchmark the mainframes combined with the terminals. The contract, awarded in the latter part of 1977, went to Honeywell.

The present system calls for three regional data processing centers. Each center is equipped with a dual Honeywell Series 60 and three front ends for servicing approximately twenty regional offices. In the summer of 1979, over 50 cities and 100 minicomputers were added to the system. About 50 people oversee the program in Washington, D.C. To date, they have not yet achieved a stable environment that would accommodate the full load.

Although Honeywell has a group which performs conversions from various hardware systems to Honeywell, Agency C opted to do the conversion themselves because of the needed modifications to the system. About 15 in-house people trained for the job. For three months at the beginning of 1978, Agency C personnel were sent to Phoenix, Arizona, where Honeywell is located. They were put in an environment with no problems and were thus able to work 72 hours per week on the conversion.

The conversion involved about 90 programs written in ANSI COBOL '68 and utilizing the network DBMS. Since most of the programs were transaction processors, the bulk of the conversion was from the network DBMS to the CODASYL DBMS. Honeywell's CODASYL DBMS comes with a teleprocessing system, and a query system. The small amount of COBOL requiring conversion was converted manually to ANSI COBOL '74.

Before the machine vendor was selected, an RFP went out for a conversion tool (isolator or translator) to assist in converting from the network DBMS to the DBMS that was to be chosen. The contract was awarded and the translator was produced. Unfortunately, there was a seven month deadline for completion of the translation, and the translator was not ready for use until well into the fourth month after the start of this task. Hence, much of the the network DBMS to CODASYL DBMS translation was performed manually. Because of the tight time constraints for completing the installation phase of the inquiry subsystem, installation was started on schedule but before the system was ready.

Agency C converted all the files themselves without extensive use of specialized tools. Testing and development were done on the same computers that were used for the production runs.

Agency C Recommendations and Comments

The following are the recommendations and comments from Agency C personnel, based upon their experiences in this conversion effort, and do not necessarily reflect the authors' opinions:

* Conversion costs should be included in the procurement evaluation. The on-line conversion took 35,000 man hours, of which 10,000+ were overtime hours.

* More time should be allowed for the planning phase. Because the date of installation was shortly after the procurement date, there was no time to do things in a structured, planned, and effective way.

* Much time was spent adjusting the programs manually.

* The programs were more difficult to convert than the data and data definition portions of the DBMS. Much of the DBMS work was done by two people.

* Redesign should be concurrent with the conversion. This takes more time but would result in a more efficient system.

* Design the programs to take advantage of vendor's equipment.

* Use off-the-shelf software, not software that is being developed.

* Agencies should do their own conversions in order to have full control. Vendors have no vested interest in seeing the system operating efficiently.

* Tools were worthless for this job. The people were more important than the tools. Agency C selected their best people, who were very dedicated and well trained in the target computer and in the application. The terminal, front end, and TP systems were not known by everyone.

6. CONVERSION LITERATURE--A SURVEY

The following summaries of various documents pertaining to conversion do not constitute a comprehensive survey of the available conversion literature. However, some of the more significant documents are summarized in this chapter in order to provide the readers with a broad perspective of various aspects in the conversion process. The following summaries reflect the opinions of the authors of these documents and do not necessarily reflect the opinions of NBS.

SURVEY OF SOFTWARE CONVERSION AIDS by the US Army Computer Systems Support and Evaluation Agency [25]

The purpose of this report is to survey the translators, emulators and simulators that were available in December, 1975. Translators are defined to include any software available to translate programs from one language to another with little or no manual reprogramming. An emulator is defined as a hardware and software technique for execution of programs assembled or compiled for a different computer. A simulator is defined as a software-only technique to accomplish the same purpose.

The paper begins by summarizing some general software conversion methods:

1. The programs may be rewritten manually by installation programmers. This is the slowest and costliest method of conversion

2. The programs may be rewritten under contract by a software house.

3. The programs may be emulated in those cases where the computer on which the programs are to be executed is equipped with the necessary emulator hardware and software feature.

4. The programs, through the use of simulators, may be run on other computers.

5. The programs, through the use of a translator, may be converted into the language of other computers.

The remainder of the report deals with the results of the survey and consists of tables and product descriptions. The Emulator Cross-Reference Table shows which computers' programs can be emulated on other computers. The Simulator

Cross-Reference Table shows which computers' programs can be simulated on other computers. The Assembly Language to Assembly Language Conversion Table shows which computers' programs can be translated to other computers. The Assembly to COBOL Conversion Table shows which computers' assembly language programs can be translated to COBOL for other computers. The COBOL to COBOL Language Conversion Table shows which computers' COBOL programs can be translated to another computer's COBOL. The High Level to High Level Language Table (other than COBOL to COBOL) shows which computers' high level language can be translated to another computer's high level language. The Cross-Assembler Table shows where some computers' assembly language programs can be assembled on a different host computer system, and then executed on the target computer system for which the program was originally written.

The Other Software Aids Table shows which computers' data base, file, and library can be translated to another computer's data base, file, and library. The Product Description Table describes translators, emulators and simulators developed by the Army, Burroughs, CDC, GE, IBM, NCR, UNIVAC, and others.

HANDBOOK FOR ESTIMATING CONVERSION COSTS OF LARGE BUSINESS PROGRAMS by Paul Oliver [17]

This handbook is intended to assist a manager in making estimates of conversion costs for large business/administrative data processing systems. The handbook assumes that source programs are in a higher-level language (probably COBOL), but most of the procedures are applicable to any conversion. Useful estimates, according to the handbook, must include the following factors: cost, time schedules, precise definitions of end products, a list of all pertinent assumptions, and an analysis of risk (i.e., the probability that actual costs will exceed estimated costs.) Three types of estimates must be made by managers at different times during the conversion. A feasibility estimate is a gross estimate used to evaluate tradeoffs on alternative approaches to conversion; a commitment estimate is used to commit resources and make cost/quality tradeoffs; and an operational estimate specifies how project management will use its resources.

The handbook consists of tables which detail the tasks, inputs, and outputs needed for different stages of conversion. The stages of conversion addressed are feasibility analysis, planning, preparation, conversion of programs and test data, production (conversion), implementation/installation, translation and unit testing.

GUIDELINES TO SOFTWARE CONVERSION by Paul Oliver [16]

Five definitions set the scope of the paper. Conversion is defined to mean any change made to a program or systems of programs solely for the purpose of enabling such a program or system to execute correctly on a computer different from the one for which they were originally devised. Translation refers to a largely automated process of conversion in which the original programs themselves serve as adequate specifications for the new programs to be produced. Recoding is similar to translation except that the process is largely manual. Reprogramming refers to a conversion which may entail a system redesign (e.g., batch to on-line) but no significant functional redesign. Redesign refers to a conversion effort which involves functional redesign and is therefore akin to new development.

A conversion project overview is provided, addressing the areas of preparation, production, implementation, software tools, and managing the conversion project. The following points are emphasized in the first section:

1. Program translation (i.e., one-for-one, or close to it, conversion) should precede any modification. This is done to avoid intermingling, and thereby compounding any translation errors with modification errors.

2. The entire conversion process must be thoroughly and carefully documented. Documentation should include the following: converted source programs, flowcharts of the converted systems, listing of all job control language programs used, standard file labels, file conversion parameters, operating instructions and technical notes, and unit and system test reports.

3. Unit and integration testing should be repeated on converted programs and test data once these are installed on the target system.

4. Manpower and time are not generally interchangeable in a software production project, but within the bounds of common sense, they are in a conversion project.

5. Productivity rates also differ widely between conversion and development. Software development proceeds at approximately 12-20 lines per man-day for general application software, while conversion may proceed at rates as high as 400 lines of code per man-day.

6. The conversion staff should never be a part-time group which participates in conversion activities, but whose members continue to report to their parent organizations.

The next section of the paper addresses conversion problems. Some of the problems cited are: machine time requirements will conflict with production; computer words vary in the number of characters they contain, causing problems in numerical accuracy as well as data movement; the format and the amount of information required to define a file vary among languages.

The last section of the paper focuses on conversion cost estimates, with detailed data about conversion costs per line supplied.

Two appendices are included with the paper. Appendix A is entitled System Conversion Checklist, and Appendix B is entitled COBOL Programming Problem Sources.

PLANNING GUIDE FOR SOFTWARE CONVERSION by HEADQUARTERS, DEPARTMENT OF THE ARMY, OCTOBER, 1977 [11]

This guide has been developed as a step-by-step manual for the guidance of ADP managers who must plan in-house software conversion. The conversion planning guide is divided into four phases.

Phase I is project initiation. This entails the assignment of the conversion manager and the team.

Phase II is the conversion workload analysis. The first step in this phase is listing all application systems that have to be converted and their locations in the organization. This involves interviewing project leaders in order to understand their systems, reviewing each program to ensure that all subroutines and utilities are identified, and obtaining an application system flowchart to include with the data which shows program reliance and interaction. The next step in this phase is identifying program conversion problems. Conversion problems are divided into one of the following groups:

1. Logic changes involving a modification of the programs to facilitate the running of this program on another computer system.

2. Code replacement involving a modification to the programs but only affecting the program in a very limited area.

3. Recompilation of a program on the new system, allowing for minor syntax errors to be corrected.

Present data base management system functions/capabilities now in use should be identified as part of Phase II. Next, current hardware/software configurations should be identified. Documentation should be reviewed for adequacy and completeness. Finally, conversion resource requirements should be estimated.

Phase III is planning for conversion (pre-award). This phase involves the following: developing a pre-conversion plan, selecting the conversion order of programs and files, preparing a preliminary conversion schedule, establishing reporting requirements for the conversion effort, and developing an overall conversion training plan.

The last phase is planning for conversion (post-award). This phase involves the following: developing and completing conversion aids, training for project leaders, initiating specialized training for the conversion team, initiating generalized training for the conversion workforce, reevaluating application systems identified for conversion, identifying new ADP hardware and software, determining standards and procedures to be used in the pilot project, performing the pilot project, reviewing the pilot project, preparing the conversion plan, establishing production resources, approving the conversion plan, completing the production and scheduling documentation.

ADP CONVERSION COST-A STUDY FOR THE FY76 WORLDWIDE ADP SINGLE MANAGERS CONFERENCE [1]

This study addresses the cost of conversion from one ADP computer system to another, and the consideration of this cost in the acquisition of replacement ADP equipment.

The following is a list of the elements of a conversion which require the expenditure of resources:

1. Conversion of Application Programs - Those written in ANSI source languages are the least costly to convert. Complex programs, programs with a large number of patches, programs highly dependent on utilities unique to a site or machine, and programs without adequate documentation will all be more costly to convert. The availability of usable conversion utilities will reduce the cost of converting the application programs, but the cost of obtaining the utilities must be considered. Non-availability of test data to test all application programs will increase conversion costs.

2. Conversion of Data - If a DBMS is desired, the availability of utilities to convert data from the current file or

DBMS is required.

3. Conversion of Operating Procedures - Differences and similarities in the following areas will affect this cost: how the operating systems handle division of tasks, how much manual intervention is desired or required, how much control the user has (via JCL) over what the operating systems will do, and how each operating system handles error recovery.

4. Support Software.

5. Facilities Requirements - floor space, air conditioning, and electrical power.

6. Parallel Operations Cost.

7. Training - systems programmers, analysts, applications programmers, operators and sometimes users.

8. Acquisition Activity - Resources will be spent producing studies on all aspects of the conversion. Examples of these studies are: hardware differences, site requirements, systems software comparability, RFP preparation, and other documents.

9. Management and Administrative Cost.

Four preliminary activities in conversion have been identified and should be completed before the conversion is undertaken. They have been outlined below, according to their relative importance.

1. Collect data regarding programs, files, and operating environment, and update the data where necessary.

2. Standardize - translate programs to a standard programming language, such as ANSI COBOL.

3. Evaluate and select conversion alternatives - ranging from total redesign to emulation/simulation.

4. Decide whether the conversion is to be done by in-house or contractor personnel.

The cost of converting applications programs can be divided into four categories: analysis costs; programming manpower costs; machine costs; data conversion costs. Formulas to determine programming manpower costs are supplied in this study.

The principal cost element in computer conversion is the conversion of data from the old system to the new system. Considerations to be included in the data conversion are: size of the data base; structure of the files; access method; production of data to test the transformed system.

Many factors affect the actual cost of converting the operating procedures. Two of the major factors which affect this cost are the similarities of the JCL's for the two machines and the existence of documentation.

One of the factors which can be easily estimated is resource costs. This element of conversion includes the cost of support software. The support software on the target machine should be divided into the following three categories: commercially procured packages available on both the old and new machines; utilities which are supplied by the new vendor; locally developed utilities.

Facilities requirements are estimated by determining the facility required to support each alternative and estimating the cost to develop, acquire or modify each of these facilities. Costs are estimated in terms of such elements as floor space, power requirements, new air conditioning, etc.

The cost of parallel operations is the cost of running the new system while the old system continues to provide support to the users. A time-phased plan for conversion activities is required for this estimate.

Training costs are among the easiest to estimate. The total training cost is the cost of the man-hours lost during this training, plus the travel costs, plus the fees, if any, for the courses.

Acquisition activity cost must be considered a cost of conversion, particularly where one alternative is not to convert. This cost entails manpower allocated to the acquisition, costs for studies conducted, and a projected cost for travel associated with the acquisition.

The management and administrative costs detailed above cannot be ignored. It is suggested that the organization designated to perform the conversion be evaluated with the objectives of establishing a management:technician ratio. Commercial firms establish this ratio as high as 1:1. A more reasonable ratio in the Federal Government would be 1:2.

7. CONCLUSIONS

The interviews with Federal Government agencies who have undertaken large conversion efforts and conversion specialists who perform conversions as a profession, as well as the literature review, indicate that many problems exist regarding conversion. The conclusions are described under the following topics: management problems, conversion costs, conversion tools, personnel problems, effects of standards, and critical areas.

7.1 Management Problems

Management problems are significant because of the quantity and complexity of the resources that must be brought together and managed. The following paragraphs describe some of these management problems.

Resource Requirements

Top management must understand the reasonable time frame and costs needed to produce useable programs on the target machine. Resources commensurate with the job must be provided. In addition, management should keep in mind that a conversion schedule compressed at an inappropriate time may very well ensure unsatisfactory results. If conversion is viewed as the software-producing activity which it really is, and not as a mode of catching up to where an agency was, the results of conversion would be much more satisfactory.

Documentation

Programs should be designed with conversion in mind. This implies well-structured programs that are maintainable, readable, and understandable. Good documentation is an absolute necessity. Many agencies develop adequate user documentation, but very few develop adequate programmer documentation, especially comments. Very often, when programmer documentation is developed, it is not updated. Standards for documentation should be developed and used. Proper program documentation should begin during design. Pre-coding and post-coding documents should be produced. Documentation can not be left for the end of the project, but must be an integral part of development and, thus, must be produced during the analysis, design, coding and testing stages. Programs must be well commented. Agencies should have rules governing when and where comments must appear in a program.

Test Files

Adequate test files must be developed during program development and maintained for the life of the program. The test files should be modified to test new sections of code, when this code is added to the program. Test files and documentation, if both are kept up-to-date, can not only be of tremendous benefit during an eventual conversion, but will prove to be extremely valuable during maintenance.

Conversion Techniques

One of the most important decisions during a conversion effort is the choice of a conversion technique (recoding, reprogramming, redesign). Federal agency management should be integrally involved in choosing a conversion technique, since this choice will ultimately affect the size and quality of the converted system. However, our interviews with Federal agencies lead us to believe that a conscious decision as to which technique to use is usually not made by management within each agency. Very often, this decision is left up to the contractor who performs the conversion. It is in the best interests of the contractor to select recoding as the conversion technique that he will use, since it is the least costly technique and takes less time than the other techniques. Also, conversion contractors usually have data bases that they can use to estimate recoding costs, but not reprogramming or redesign costs. Reprogramming and redesign costs are much more difficult to estimate since a greater amount of human intervention is required in these techniques. Thus, translators and other automated tools used in reprogramming and redesign, make a much less significant impact on the conversion process, and thus, on the cost of conversion.

Management can, and should, help determine which conversion technique(s) are to be chosen for a particular conversion. This can be done either explicitly or implicitly.

An explicit choice involves examining the various conversion techniques, the programming languages involved, and the performance requirements of the system. The proper technique can then be chosen. One useful set of criteria that can be used in deciding among conversion techniques is listed below:

* Recoding - Source and target languages should be similar. The source and target computers should have comparable hardware/software capabilities. If recoding is used when hardware/software capabilites are not comparable, then the capabilities of the new computer may not be utilized, often resulting in

inefficient programs.

* Reprogramming - Source and target languages are dissimilar (e.g., converting from assembler to COBOL). Line-for-line translation (or recoding), when the source and target languages are dissimilar, may well result in converted code which is much larger than the source code. For example, in a conversion of assembly language to COBOL, assume that the first assembly language statement loads register 1 with variable A; the second statement loads register 2 with variable B; the third statement adds the contents of register 1 to register 2; and the fourth statement puts the new contents of register 2 in variable C. Some translators, performing line-for-line conversion, may pick this up and generate four COBOL statements corresponding to these assembly language statements. In turn, when these COBOL statements are compiled and assembled, they will produce 16 assembly language statements, producing a four-fold code expansion. A good programmer, using reprogramming rather than recoding, would produce COBOL code saying "ADD A TO B GIVING C", which would compile much more efficiently.

* Redesign - The source program is many generations old. The design is out-of-date, and the program is poorly structured and documented. The agency desires a more modular, maintainable and readable program.

A manager can implicitly specify the appropriate conversion technique by embedding performance requirements into the conversion RFP. In this way, the contractor is legally obliged to address constraints specified in the RFP (such as response time, memory requirements, size of the target program, etc.). The choice of recoding as a conversion technique will often fail to result in the target program satisfactorily meeting these performance constraints. In fact, translators, which use the recoding philosophy, have been known to produce an object expansion rate of 10 to 15 times. Thus, if the new computer is six times faster than the old one, the converted programs will execute at about one-half the speed as before conversion. When performance requirements are embedded in the RFP, the contractor is placed into the ideal situation, from the agency's standpoint. The contractor will still attempt to utilize the most economical and timeliest technique he can, but this technique must also meet the performance requirements. Thus, hopefully, the product that the agency will obtain is the one that is the least costly and most efficient, but also satisfactorily meets minimum performance requirements.

Estimating the cost of producing a system with specified performance constraints is very difficult. If performance constraints are included, the contract will probably not be a fixed price contract.

Many agencies fail to include any performance constraints in the RFP, asking instead for functionally equivalent source code. This can lead to degradation in performance, as illustrated in the assembler to COBOL translation documented above, and may result in a converted system which does not meet the user's needs.

7.2 Conversion Costs

Conversion companies maintain data bases containing information enabling them to make fairly accurate conversion estimates. Much of the information on these data bases is historical, resulting from past conversion projects. The high degree of confidence placed in the accuracy of this costing information is demonstrated by the reluctance of many conversion companies to accept anything but fixed-price contracts. However, the conversion companies which were interviewed had very little costing information pertaining to languages other than COBOL. Furthermore, the costing information contained in these data bases was derived from conversions utilizing recoding, as opposed to reprogramming or redesign.

One of the causes of widespread cost overruns for Federal agency conversions is the lack of a comprehensive cost assessment methodology. Since various methods are used to determine conversion costs, parameters included in some methods are omitted or regarded as of less significance in others. This often results in conversion costs excessive enough to offset a substantial percentage of the savings expected from expanded applications, increased memory, and faster throughput. Cost benefit analysis, however practical, has seldom been a prerequisite for undertaking software conversion within the Federal sector. Typically, many costs associated with conversion are hidden within the total project cost. The gross underestimation of conversion costs is due, in part, to the lack of a distinction, by management, between the costs of conversion elements and those of ongoing operations.

Agencies need information to help them make conversion estimates. Although much of the conversion being done in Federal agencies is either from assembler-to-COBOL or from COBOL-to-COBOL, information is also needed concerning conversion costs of other languages, particularly FORTRAN. As explained earlier in this chapter, Federal agencies should examine various conversion techniques before beginning coding for a conversion. Part of this examination should include a cost benefit analysis of each of the techniques. Thus, much more detailed cost information is needed concerning reprogramming and redesign, as well as recoding.

7.3 Conversion Tools

Conversion specialists tell us that many customers want to know what types of conversion tools are available. Information concerning the availability and the usefulness of various tools is certainly a prerequisite for planning, staffing and organizing a conversion project. Tools, such as translators, file comparators and source code reformatters, can be extremely helpful in conversion projects and are, in fact, being used by a wide variety of conversion houses and Federal agencies.

However, the limitations of tools must be realized. Automated tools usually solve the easy problems which are encountered in conversion, such as differences in syntax between two programming languages. The semantics of a block of code, on the other hand, can sometimes be determined only by interfacing directly with the person or persons who wrote the code. Automated tools usually won't translate the correct semantics of a block of code if the semantics cannot be identified by merely scanning the code. Look at the following block of FORTRAN code written for a CDC 6700 computer which stores ten characters per word.

```
      COMMON A(4), B(4)
      DO 100 I=1,4
      B(I)=A(I)
  100 CONTINUE
```

In attempting to convert this program to an IBM 360/370 series computer, a translator would probably not modify these lines of code, since they are syntactically correct statements for IBM FORTRAN as well as CDC FORTRAN. However, the arrays A and B may very well have been used to store a 40 character string on the CDC machine, 10 characters in each element of the array. In order for the semantics of this block of code to be preserved on an IBM 360/370, which stores 4 characters per word, the following block of code would have to be substituted for the above one.

```
      COMMON A(10), B(10)
      DO 100 I=1,10
      B(I)=A(I)
  100 CONTINUE
```

A translator would not "be smart enough" to make this substitution. It would have no way of knowing if array dimensions and/or loop counters are set up a certain way in order to store and manipulate character data, and are, thus, dependent on the word size of the machine. Conversely, if the array dimensions and/or loop counters are set up for numerical computations, they are independent of the word size.

Tools are not an end in themselves. They are just one part of a complex management strategy, and the people organizing and directing conversion projects must know how and when to use these tools. Also, most of the existing tools were written for, and execute correctly on, only one computer. A substantial effort is usually required to modify these tools to work in a different environment. Therefore, more general tools, which work in a broader environment, are desirable. Examples of such tools are translator generators and compiler compilers. A greater benefit can accrue from conversion tools if the tools themselves are written in higher level languages, with the objective of being portable, maintainable and modifiable.

7.4 Personnel Problems In Conversion

Conversion is viewed, by managers and programmers alike, as an undesirable activity. Managers are often threatened by conversion. They have not planned for conversion, nor have they budgeted for conversion. Eventually they are forced to convert, and must perform the conversion in a timely manner, with as little disruption to the ongoing system as possible.

Programmers seem to view conversion with equal disdain. They are "stuck" with a program that they neither designed nor coded, and now they must get it to execute correctly on a machine with which they are not familiar. Additionally, conversion is much more mechanical and not nearly as intellectually stimulating as new coding or design. All of the above people problems are compounded by the fact that the natural instinct of most individuals is to resist change, and cling to that with which they are most familiar.

There are no easy solutions to the above people problems. Managers need to be made aware that conversion is a part of their job. They must budget for conversion and develop a comprehensive conversion plan, like the one outlined in Chapter 2. Conversion must be scheduled, and computer time must be proportioned between conversion and ongoing production jobs.

Certainly, the image of conversion as an undesirable activity which is assigned to the low man on the totem pole must be changed. Perhaps, the utilization of tools to assist with the repetitive aspects of conversion can allow programmers more time to spend on the more challenging aspects of conversion. If possible, agencies should try to get programmers involved in the early stages of conversion, especially in the study of conversion techniques. In this way, programmers are made to feel more a part of the total project, rather than seeing themselves as coders being used for short term conversion jobs.

7.5 The Effects Of Standards On Conversion

ANSI Standards and Federal Information Processing Standards (FIPS) can help to reduce conversion problems. However, standards such as these must include features that are used and needed by the user community. If the standard for a specific higher level language does not include these features, vendors will almost definitely supply them as "extensions" to the language. When application programs incorporate these extensions, the conversion problem is compounded since the syntax, as well as the semantics, of these extensions will invariably differ from vendor to vendor.

7.6 Critical Areas

As a result of the information derived from this study, the following areas have been identified as most critical, and in need of immediate attention:

1. Guidance is needed concerning the selection of the most appropriate conversion technique to employ in a conversion effort. Some of the conversion techniques which should be examined are recoding, reprogramming, and redesign.

2. Guidance is needed concerning the best way to plan for future conversions during new development. The philosophy which should be encouraged is that new programs should be designed and developed with future conversion in mind.

3. Guidance is needed concerning the type of information which should be included in an RFP to perform a conversion.

4. Guidance is needed concerning conversion cost estimation. The following two areas need to be addressed: (a) costs for different techniques which can be used in conversion, such as recoding, reprogramming and redesign; (b) an examination of the ways in which costs differ when utilizing different programming languages in conversion.

5. A common data interchange format should be provided, capable of supporting file and database management environments, that would provide assistance in data conversion.

6. An in-depth study of the problems of database management system conversion is needed which would provide guidance to minimize present and future database management system conversion problems.

7. A data dictionary capability should be available to Federal Government agencies to assist in the planning and data preparation phases of a conversion. A data dictionary is a software tool which can contain information concerning all the programs and data in an organization.

8. Research directed toward general solutions to the program conversion problems that exist in a database management environment is necessary.

9. A cross reference list should be provided, which lists all translators (and other tools) and where these tools can be obtained. This list should also provide brief descriptions of the tools and their functions.

10. Guidance is needed to help overcome management problems involved in conversion, including a detailed approach to addressing each of the management problems.

In addition to the critical areas, identified above, other areas have also been identified. These other areas are not as critical as the ones identified above, but need to be addressed sometime in the future in order to encourage more efficient and more effective conversion. These areas are listed below:

1. More generalized conversion tools need to be designed and developed.

2. Guidance is needed concerning the complete life cycle of test files. The proper technique to be used in developing test files to be used in conversion should be addressed, as well as the update and maintenance which is needed in order to keep them current with program changes.

3. Training courses and workshops on conversion should be provided.

4. Guidance is needed concerning how to determine when to perform a conversion in-house and when to contract out for the conversion. The different types of people who are needed for various types of conversions need to be identified.

5. Guidance is needed concerning the operational considerations of conversion.

6. Guidance is needed for scheduling a conversion project.

7. Guidance is needed concerning data preparation for conversion.

8. Guidance is needed concerning the ways in which to explore alternatives to conversion.

As a result of the conclusions stated in this study, NBS has initiated a comprehensive program to develop a set of standards and guidelines which will help reduce the costs and increase the efficiency of conversions done by Federal agencies. Subject to the availability of funds and appropriate human resources, the following products will be produced by NBS:

* A guideline for the evaluation and selection of conversion techniques

* A guideline on how to plan for future conversions by utilizing better system design, development and documentation techniques

* A guideline for conversion cost estimation

* A guideline for developing an RFP for conversion services

* An annotated conversion tool directory

* A guideline on how to program to achieve portability

* A guideline on database management conversion providing a tutorial on the special problems encountered and an analysis of alternative techniques

* A data translation standard providing a data interchange form on magnetic tape supporting file and database management environments

* A data dictionary standard to assist Federal Government agencies with their planning and data preparation

Appendix A: Questions For Vendors and Federal Agencies

Note: Most of the questions are relevant for both vendors and Federal agencies. However, some of the questions apply only to vendors or only to Federal agencies.

1. Reason for Conversion (e.g., new computer acquired, desire to share program with another installation, etc.)

2. Type of conversion - size of program, converting from what to what, costs, major problems encountered, languages.

3. How could development process have been improved to minimize the conversion problems that have resulted? (planning for conversion)

4. Were you satisfied with the conversion effort that was done?

5. Was the contractor late or on schedule?

6. Was the contractor's cost estimate accurate?

7. Did the converted system run as efficiently as the old one? If not, how much inefficiency was introduced? Could this have been avoided? Was a choice given to you - more efficiency for more cost?

8. How much direct contact was needed with the person who wrote the code for the program?

9. What types of automated tools were helpful in conversion? What new tools would be helpful if they could be developed?

10. What was the quality of the documentation of the software to be converted? Where was it deficient? Did you desire documentation as part of the conversion task deliverables? Was the documentation produced for you acceptable?

11. What was the quality of the code that was converted? Was it well structured and did it contain sufficient comments?

12. Did you utilize any in-house programming or documentation standards?

13. How can (Federal) standards aid in the conversion process?

14. How can software conversion costs be reduced?

15. Do you feel that your management recognized the fact that most production programs will eventually be converted to newer equipment? How can they be educated, so that they will "plan" for conversion during development?

16. Did the system being converted utilize a commercial DBMS? Did this present any special problems?

17. Have the differences in JCL caused significant problems in conversion?

18. Did you use vendor-unique features? If so, why? What percentage of conversion problems are caused by using vendor-unique features? Could the same thing have been accomplished using standard features? If so, at what price? (loss of efficiency, loss of capability, etc.)

19. Did you ever consider doing the conversion in-house? Why or why not? Why did you choose to contract out the conversion effort?

20. What are the human problems involved with conversion? Is it hard to train people in conversion techniques? Did you try? Are your people properly trained in conversion? What are your attitudes toward conversion? Are your best or worst people usually put on conversion projects?

21. At what point does conversion become impractical, and re-design more cost effective?

22. How much of the conversion effort involved was really new development?

23. Did your system have security constraints?

24. How did the conversion effort interface with operations? Did the contractor use your computer to do the conversion? Was there enough time available because of the current workload?

Appendix B: Bibliography and References

Note: Many of the following documents can be obtained from the National Technical Information Service by referencing the appropriate NTIS number.

1. A study for The FY76 Worldwide ADP Single Managers Conference
ADP Conversion Cost
Randolph AFB Tx, 3-5 December 1975

2. Berg, John, L., editor
Data Base Directions - The Conversion Problem
National Bureau of Standards and the ACM (draft to be published in 1980)

3. Computerworld
"Users Underestimate Conversion Costs : Survey"
October 8, 1979, p. 55

4. Conference Book of the National Symposium on Computer Systems Enhancement
Converting Today's Systems To Tomorrow's Technology
Data Processsing Management Association, November 13-15, 1979

5. Cooper, Roger
"Upgrading Federal Computers Through Existing Systems"
Government Executive, August 1979

6. Dooley, Ann
"Conversion Causes Welfare Payment Delays"
Computerworld, October 15, 1979

7. Federal Information Processing Standard 43
Aids for COBOL Program Conversion (FIPS PUB 21 to FIPS PUB 21-1)
National Bureau of Standards, December 1, 1975

8. Fleiss, J., et al.
Programming for Transferability
Prepared for Rome Air Development Center, NTIS AD-750 897, September 1972

9. Frank, Werner L.
The New Software Economics
Library of Congress Catalog No. 79-90434, 1979

10. Fry, J., et al.
"An Assessment of the Technology for Data and Program-

Related Conversion"
AFIPS Conference Proceedings,
National Computer Conference, Anaheim, California, June 5-8, 1978
pp. 887-907

11. Headquarters, Department of the Army
Army Automation Planning Guide for Software Conversion
Department of the Army Technical Bulletin, October 1977

12. Lynn, C., et al.
"Program Conversion - One Successful Paradigm"
AFIPS Conference Proceedings, Volume 48,
National Computer Conference, 1979

13. Morgan, L.W., et al.
Conversion of CCF UNIVAC Software
MITRE Corporation MTR-4710, January 1978

14. NTIS
Computer Software Transferability and Portability
A Bibliography with Abstracts,
NTIS/PS-79/0567, May 1979

15. Oliver, Paul
Bibliography of Conversion References
Federal COBOL Compiler Testing Service,
NTIS ADA052462, April 4, 1978

16. Oliver, Paul
"Guidelines to Software Conversion"
AFIPS Conference Proceedings,
National Computer Conference, Anaheim, California, June 5-8, 1978, pp. 877-886

17. Oliver, Paul
Handbook for Estimating Conversion Costs of Large Business Programs
Department of the Navy,
NTIS AD-A065145, February 14, 1979

18. Oliver, Paul
Survey of Conversion Support Software
Federal COBOL Compiler Testing Service,
NTIS ADA053741, March 16, 1978

19. Rand Information Systems
Conversion Special Report: Changing to New Technology
Commissioned by the Data Processing Management Association Education Foundation for the National Symposium on Computer Systems Enhancement, November 13-15, 1979

20. Rand Information Systems
Questions and Answers on Conversion

21. Razza, Sal
"Software Portability"
Computerworld, November 19, 1979, p. 57

22. Schneider, Daniel B.
Computer Systems Conversion
A Management Perspective
U.S. Department of Justice,
NTIS PB-297 604, October 1978

23. The Comptroller General of The United States
Report to the Congress
Millions in Savings Possible in Converting Programs from One Computer to Another
GAO FGMSD-77-34, September 15, 1977

24. The Comptroller General of the United States
The Federal Information Processing Standards Program: Many Potential Benefits, Little Progress, and Many Problems
GAO FGMSD-78-23, April 19, 1978

25. US Army Computer Systems Support and Evaluation Agency
Survey of Software Conversion Aids (Emulators, Simulators, Translators)
1 December 1975

NBS-114A (REV. 9-78)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP 500-62	2. Gov't. Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE COMPUTER SCIENCE & TECHNOLOGY: CONVERSION OF FEDERAL ADP SYSTEMS: A TUTORIAL	5. Publication Date August 1980
	6. Performing Organization Code
7. AUTHOR(S) Joseph C. Collica, Mark W. Skall, Gloria R. Bolotsky	8. Performing Organ. Report No.
9. PERFORMING ORGANIZATION NAME AND ADDRESS NATIONAL BUREAU OF STANDARDS DEPARTMENT OF COMMERCE WASHINGTON, DC 20234	10. Project/Task/Work Unit No.
	11. Contract/Grant No.
12. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (*Street, City, State, ZIP*) Same as Item 9	13. Type of Report & Period Covered Final
	14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

Library of Congress Number: 80-600106

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

16. ABSTRACT (*A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.*)

This tutorial report was undertaken to provide a better understanding of conversion of Federal Government ADP Systems. Three sources were used for gathering the required information to prepare this tutorial: (1) interviews with commercial conversion experts; (2) interviews with Federal Government agency personnel who have recently experienced conversions; (3) current literature. The first three chapters comprise the tutorial. The next three chapters discuss the information gathered from the above three sources. The last chapter summarizes the authors' conclusions, while highlighting the major problem areas requiring guidance.

17. KEY WORDS (*six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons*)

Conversion costs; conversion problems; conversion tools; database management; Federal agencies; language translators; maintenance; portability.

18. AVAILABILITY [✓ Unlimited]	19. SECURITY CLASS (THIS REPORT) UNCLASSIFIED	21. NO. OF PRINTED PAGES 73
☐ For Official Distribution. Do Not Release to NTIS		
[✓] Order From Sup. of Doc., U.S. Government Printing Office, Washington, DC 20402	20. SECURITY CLASS (THIS PAGE) UNCLASSIFIED	22. Price $4.00
☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161		

USCOMM-DC