1989

Software engineering with database management systems.

Karatasios, Labros G.

Monterey, California. Naval Postgraduate School
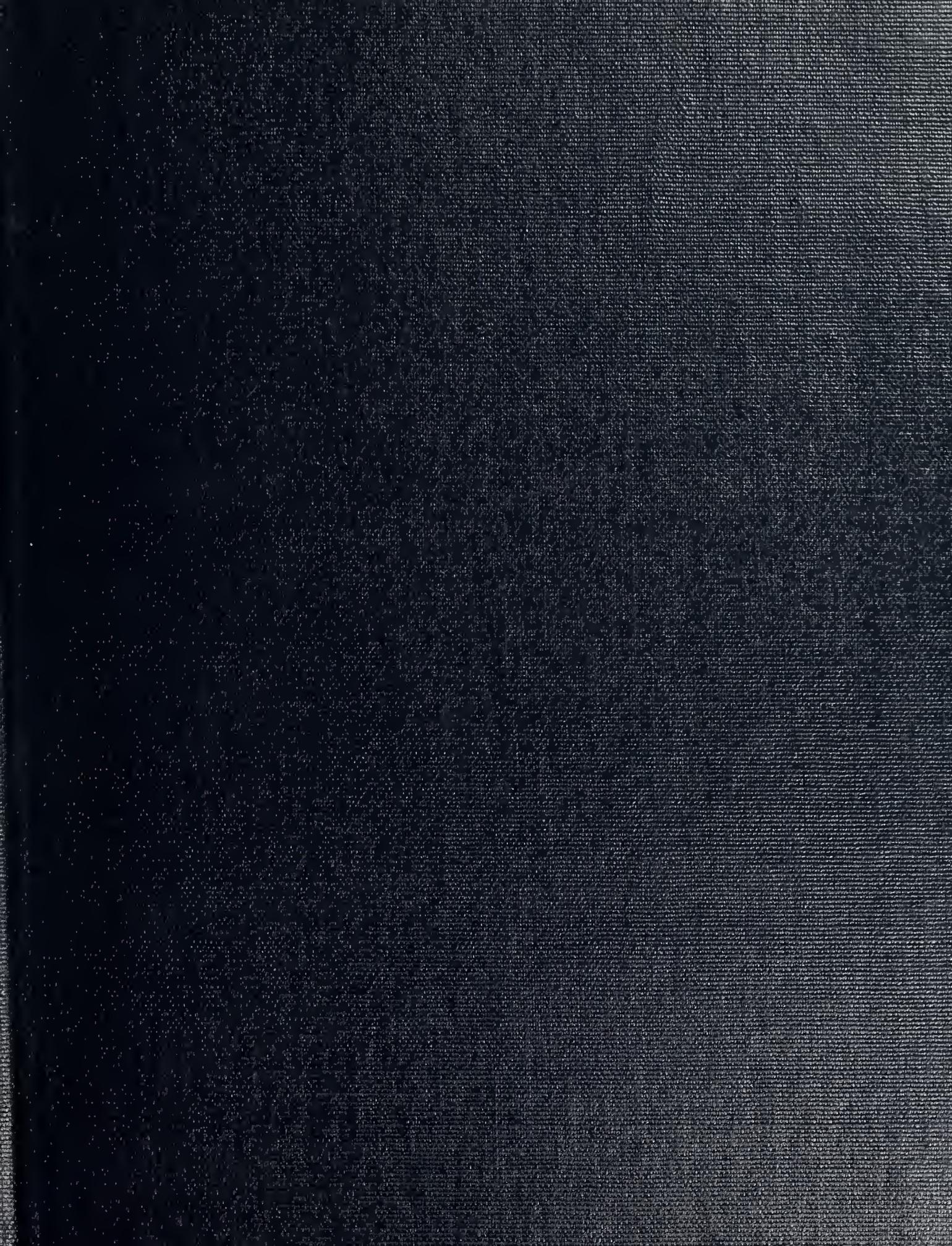
NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SOFTWARE ENGINEERING WITH DATABASE
MANAGEMENT SYSTEMS

by

Labros G. Karatasios

March 1989

Thesis Advisor: S. H. Parry

Approved for public release, distribution unlimited

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	public release, distribution unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL (If applicable) 37	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000

8a. NAME OF FUNDING / SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS

	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification) SOFTWARE ENGINEERING WITH DATABASE MANAGEMENT SYSTEMS

12. PERSONAL AUTHOR(S)
Karatasios, Labros G.

13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 1989 March	15. PAGE COUNT 189

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17.	COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Database Design; Personnel Systems; Greek Navy
			Personnel Database

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
The purpose of this thesis is to communicate a general knowledge of software engineering principles that can be applied to the development of a software system. Fundamental Software Engineering concepts are first discussed and then applied to a personnel database management system which is featured throughout the thesis. The individual tools and techniques that are used in each phase of the system development are widely known in the computer science community and each has been employed successfully in certain situations.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Prof S. H. Parry	22b. TELEPHONE (Include Area Code) (408)646-2779 22c. OFFICE SYMBOL 55Py

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted
All other editions are obsolete

☆ U.S. Government Printing Office: 1986—606-24-

Software Engineering with Database Management Systems

by

Labros G. Karatasios
Major, Hellenic Army
B.A., Hellenic Army Academy, 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 1989

ABSTRACT

The purpose of this thesis is to communicate a general knowledge of software engineering principles that can be applied to the development of a software system. Fundamental Software Engineering concepts are first discussed and then applied to a personnel database management system which is featured throughout the thesis. The individual tools and techniques that are used in each phase of the system development are widely known in the computer science community and each has been employed successfully in certain situations.

TABLE OF CONTENTS

I. INTRODUCTION

During the past thirty years the advances in computer hardware have been so phenomenal that even the pioneers in the computing industry are amazed at how much progress has been made. Progress in computer hardware with respect to cost, speed of computations and decrease in size is measured by several orders of magnitude and the most amazing is the forty percent compound annual rate of reduction in the unit cost of memory and storage.

Unfortunately, our ability to build software, which is necessary to interface with the computer hardware, has not progressed as rapidly. As the range of computer applications has grown and the complexity of tasks computers can handle has increased, the cost of developing software for a computer system has increased so much that it has become by far the most costly component in the system. The main cause of the increase in the cost of software was that the existing methodologies for software development were inadequate. As a result, many software systems failed, and many others were late, unreliable, difficult to maintain, their performance was poor and they cost much more than originally predicted.

Much research has been conducted during the last twenty years on software development techniques and methodologies that would end the "software crisis". A new technological discipline, Software Engineering, has been developed to improve the quality of software products and to increase the productivity of software engineers.

The term "Software Engineering" was chosen as the title of a NATO conference in 1968 in order to express "the need for

1

software manufacture to be based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering". [Ref. 1: p. 13] Since then a lot of activity has been focused on software engineering. The Institute of Electrical and Electronics Engineers (IEEE) has been publishing the "Transactions on Software Engineering" since 1975. The Association for Computing Machinery (ACM) has founded a Special Interest Group on Software Engineering (SIGSOFT). Several conferences and symposia on this new discipline have been sponsored by many organizations. The chairman of the IEEE Richard Fairley stated in 1979 that "Software engineering has evolved into a major subdiscipline of computer science and engineering. Although much remains to be done, a body of knowledge and a set of guidelines have emerged which incorporate traditional engineering values into the production and maintainance of software systems". [Ref. 2] During recent years a great number of well known and unknown computer scientists and theorists have defined numerous techniques and methodologies on the development of software products. Much discussion and controversy follow the announcement of every new method or technique. Some of these techniques are more controversial than others. Some of them do not work at all in certain situations while they are very effective in others. Some techniques are very promissing; Ed Yourdon, in one of his many books on the structured techniques, trying to convince the reader how good these techniques are, he writes: "... what the techniques can do is impressive. Chances are that if you use the techniques, you'll be sipping your mint julep in your Mediterranean villa before long. If you don't use the techniques, well, chances are you'll be replaced by someone who does." [Ref. 3:p. 3]

After so many years of analyzing and studying the mechanisms of software development, software engineering has not yet been able to develop a universally accepted methodology for building software, and software development is still considered an art by most people. The truth is that many of the developed techniques really work. Studies conducted by large software organizations have proven that the programmer productivity and the software quality has improved significantly with the use of certain techniques. For example, IBM studies report an average of forty percent productivity savings in real-time, business and systems software projects utilizing structured programming. [Ref. 4] In other projects, however, it has been found that the principles of software engineering do not always guarantee success.

Although a definite trend exists in academic computer science circles toward the recognition of software development as an engineering discipline, such recognition is much less pronounced among software houses and other software producers. One of the reasons is that the established software engineering principles and techniques are very often too technical and require a higher-level of knowledge in Computer Science to understand them. Unfortunately, the main body of programmers consists of people with litle or no higher education background. In fact only a very small fraction of the people who are programming computers have ever studied computer science in any depth. In the U.S.A., some 29,600 bachelor's degrees were awarded in computer and information sciences in the six-year period 1972-1977. [Ref. 5: Pg. 169] These courses of study were first offered in the 1960's and are still growing; therefore it can be assumed that a comparable number of degrees were awarded in the years before 1972. Although we do not have

3

the statistics for the years after 1979 it is estimated that the total number does not exceed one hundred thousand. This number is quite small in comparison with the 300,000 programmers working full time and another 300,000 who work part time. [Ref. 6: Pg. 6] On the other hand, the existing literature on software engineering is anything but reliable. The books one can find on software development tools and techniques are usually very abstract and difficult to read. The majority of such books concentrate on only one phase in the software life-cycle and provide no interface with the rest of the phases. Most of them do not provide a case study to illustrate the theoretical part and those who do, usually give very simplified examples that have nothing to do with the complexity of the real world and they also shift to a different case study as they go from one phase to another. As a result of the above situation the books on software engineering concepts have a very small number of readers, limited to academians and computer science students in the universities. On the other hand, books of the type "How to become a programmer in ten easy steps" or books with applications software that can be extended or modified to fit someone's needs are becoming best sellers.

The purpose of this thesis is to communicate a general knowledge of software engineering principles that can be applied to the development of a software system. A personnel database management system was chosen to serve as an example throughout the thesis. The individual tools and techniques that are used in each phase of the system development are widely known in the computer science community and each has been employed successfully in certain situations. This does not mean that the presented techniques are the only techniques a software engineer can use. On the contrary there are as good

4

or even better techniques in certain cases. What the reader must understand is two things:

a. The development of any software system must follow certain steps. Different people have given different names to these steps and others have combined or analyzed them. Whether the six-step decomposition of a system's life-cycle followed in this thesis is better than others is not a key issue. What is important is to recognize that phases exist and organize the approach to account for them.

b. The development of software systems can be fascilitated by software engineering tools and techniques that allow the developer to continuously assess the extend of his progress and the validity of his decisions. The tools and techniques used in the development of the example software system are not to be considered as the most recommended ones or as suitable for all cases. Each software engineer may choose the set of tools and techniques that he thinks is most appropriate for the system he is developing. The idea is that such sets do exist and the thesis provides such an example.

The reason why a database management system (DBMS) was selected to serve as a case study in the thesis is that a DBMS is like any other software system, only it is somewhat more complicated because it involves the design of a database. An additional reason is that DBMS's are very popular today and it is likely that most software engineers will face the problem of developing such a software system.

5

II.   THE SOFTWARE DEVELOPMENT PROCESS

To better control the development of a software system, software engineering has identified a sequence of stages through which the system passes; these stages are collectively called the software development life-cycle. The stages of the software life-cycle followed in this thesis are described below.

A.   PROBLEM DEFINITION

This first stage helps the software engineer understand the problem and define the objectives and the scope of the system he will develop.

B.   FEASIBILITY STUDY

During this stage it is determined if the problem can be solved and a number of solutions that might satisfy the user's needs within the defined scope are identified. At the end of this stage the software engineer obtains a decision from the user or management whether to proceed with the development of the system.

C.   ANALYSIS

This is the most important stage in the system life-cycle. The software requirements (i.e., the system's functions and operational constraints) are specified and documented during analysis.

D.   SYSTEM DESIGN

The software engineer determines how in general the system will be implemented by identifying different alternative

strategies and selecting the one that best satisfies the system's needs.

E. DETAILED DESIGN

The designer takes the software requirements prepared during analysis and based on the decision made during system design, organizes them in a way suitable for computer execution.

F. IMPLEMENTATION

In this stage results produced during the detailed design are implemented in source code. It is also the purpose of this stage to verify that this source code implements correctly the design specifications.

There is no single decomposition of the software development process. The one presented here works in practice but modifications may be required for other applications. The important thing is that all the different decompositions of the system life-cycle that exist require all of the above functions to be performed, no matter what name they assign to each stage, whether they combine two or more stages in one, or further decompose one stage.

The following six chapters of the thesis develop each stage of the software development process. Each chapter is divided into two parts. In the first part the theory of the particular stage is given and in the second part this theory is applied to a real database management system for implementation in the Greek Army.

7

III. PROBLEM DEFINITION

The first step in the system life-cycle is the Problem Definition during which the analyst understands the problem and defines the objectives and the scope of a system that solves it.

A. THEORY

It is very unlikely that an analyst would be asked to design a system that has no precedent. Most of the time a system that works already exists but which may have some problems in its performance. These problems may be recognized by the user himself or by the management. If the problem is serious and its solution is considered imperative, then an analyst will be asked to offer his services.

The first thing the analyst must do, in any case, is to understand the existing problem and prepare a written statement of his understanding. Many analysts consider this step as needless, but it has been proven that in a number of cases the delivered system solved a different problem from the one the user had in mind, just because the analyst ignored this step.

B. IMPLEMENTATION

1. The Greek Army

Before introducing the problem that we as analysts will be asked to analyze and try to solve, a very brief overview of the Greek Army structure and the soldier life-cycle will be presented.

The Greek Army consists of the Operational Arms (Infantry, Artillery, Armor, Corps of Engineers and Signal

Corps) and the Support Services. The Arms and Services are manned by officers, non-commissioned officers and soldiers. The officers and NCOs are for the most part professional career personnel who graduate from the Military Academy and the NCO School. The soldiers are citizens who have been conscripted to serve in the Army for a period of 22 months. Every two months a new group of soldiers is conscripted. Every Greek citizen has to join the Army, usually at the age of 21, unless he has been medically proven to be physically or mentally incapable.

Every soldier passes through the following stages during his time in the military:
- Enlistment
- Basic Training, lasting two months
- Specialty Training, the duration of which varies
  depending on the specialty the soldier has been assigned.
- Service in one or more of the units of the Arm or Service
  to which the soldier belongs.
- Separation from military service.

Each Arms or Service Directorate has its own Personnel Office, located in the General Staff, which manages its personnel and is responsible for the smooth transition of its soldiers from one stage to the next.

2. The Problem Environment

The Personnel Office of the Signal Corps Directorate, among its other responsibilities, performs the following functions related to the management of its soldiers:
- Maintains updated files of the soldiers in the
  Signal Corps.
- Maintains updated files of the Signal Corps Units.

- Estimates the number of soldiers that must be trained in each specialty.
- Performs the assignments of the soldiers that are necessary to fill the vacancies created by the retirement or transfer of other soldiers.

The personnel responsible for carrying out the above procedures consists of one captain, 2 sergeants and 3 soldiers acting under the supervision of a colonel who is the director of the Personnel Office. The director, after almost 2 years of supervising the Personnel Office, has come to the conclusion that there are problems in its performance.

One of the problems is that too often the personnel office is not able to prepare the list of the soldiers' transfers and assignments on time. This is due to the great volume of transactions involved and frequently because of the untimely arrival of the required information from other subordinate units. At other times, errors have been discovered in some of the lists and reports generated by the Personnel Office. FInally, the most important problem is that during the scheduling of the soldiers' assignments, the office personnel evaluate the needs of the military units, but are rarely able to evaluate the soldiers needs as well.

The Colonel reported his observations to the Signal Corps Director and suggested that the personnel system should be studied to find the means to eliminate the problems. He also suggested that simply to increase the personnel in his office should not be considered as a solution, since this would only create more problems in other areas.

The Signal Corps Director understood the importance of the reported problems and asked the Studies Directorate to assign a systems analyst to investigate a possible solution.

3. The Problem Definition

First the analyst must understand the problem. Next he must define the objectives of a new system that would solve the problem. He must also estimate an approximate cost of the new system. He finally prepares a written statement of the scope and objectives.

Usually the most difficult part in this procedure is to estimate the cost of the new system. At such an early stage we do not usually expect someone to define accurately the cost of a system, but to set an upper limit. How can such a limit be estimated? The operating cost of the existing system must be considered first. The system is manual, therefore its main cost is the wages of the employees. This cost is calculated to be approximately $50,000 annually. One of the constraints for the new system was that it should not increase the personnel. This means that the new system may be less expensive than the old one because it may require fewer people. Of course there is no way that the system could operate without any people at all, but the analyst feels that it could use two fewer people than the old one, i.e., a sergeant and a soldier could be transferred to another office resulting in annual savings of about $10,000. Therefore with a new system that costs $20,000 we could have a return on investement in about two years. This sounds like a reasonable upper limit. In a few cases of course the management might agree for political reasons to spend even more on a new system than it was spending on the old one. Our case belongs to this category. The new system will satisfy the soldiers needs for transfers and this will have a positive effect on the soldiers' morale. We cannot evaluate morale in terms of money. For the sake of generality the systems scope will be defined at $20,000.

Next the analyst prepares a written statement of the
scope and objectives which summarizes his understanding of the
problem (see Fig. 3.1).

STATEMENT OF SCOPE AND OBJECTIVES

DATE : 29 January 1987

THE PROJECT : PERSONNEL MANAGEMENT

PROJECT OBJECTIVES : To investigate the potential for a
new system to perform the functions of SCD/
Personnel Office/Soldier Section. This sys-
tem compared to the existing one should be:
. More accurate
. More timely
. Less error prone
. Will consider not only the military units
needs but the soldiers needs as well.

KEY SELECTION CRITERIA : The number of employees must not
increase.

PROJECT SCOPE : The preliminary cost estimate of the sys-
tem is $20,000 with a precision of 30% .

FEASIBILITY STUDY : In order to investigate the potenti-
al for this project more fully, a fea-
sibility study lasting approximately
2 weeks is recommended. The cost of
this study will not exceed $1,000 and
is included in the project scope.

Fig. 3.1. The Statement of Scope and Objectives.

IV. FEASIBILITY STUDY

A. THEORY

1. Why a Feasibility Study?

The initial phase of the Specification Stage ended with the preparation of a written statement of the system's scope and objectives. What must be the next step?

Many analysts tend to become attached to the first possible solution they think of and start proceeding in this direction. This is a very bad habit that quite often leads to catastrophic results. If during the process this solution is found to be infeasible then a large amount of time and effort has been wasted. Even worse is the case when the analyst does not want to accept that he failed and tries to integrate the system, changing part or all of its functions, so that they fit into the one designed. As a result of the above tendancy, many systems have been delivered with excessive delays, over budget and often without solving the problem. In order to avoid these undesirable effects, a feasibility study must always follow the problem definition.

The primary objectives of a feasibility study are to determine if the problem can be solved and to recommend a solution strategy. There is no standard format for a feasibility study. Many people have described a number of ways on how to conduct such a study. The reader can find some of them in [Ref. 7], [Ref. 8], [Ref. 9] and [Ref. 10]. There are many differences in the methods and steps they follow and even in the contents of the study. The method described in [Ref. 10] is one of the most complete and easiest to follow and therefore will be used as a reference point for the feasibility study.

13

The main steps of a feasibility study are:

- Confirm the problem definition.
- Study the existing system.
- Develop a high-level logical model.
- Confirm the logical model.
- Develop and evaluate alternative solutions.
- Select one alternative.
- Prepare a development plan.
- Write and present the feasibility study.

A brief discussion on each of these steps follows.

2. Confirm the problem definition

It is very probable that the analyst's understanding of the problem as described in the problem definition is not quite accurate. For this reason the analyst must contact the user and the management and confirm that what he has in mind is exactly what they want.

3. Study the existing system

A new system almost always replaces an existing one. We usually want the new system to perform basically the same functions as the old system but in a faster, more reliable, cheaper or generally more enhanced way. Sometimes it is desirable that the new system includes a few new functions that the old system did not perform or to eliminate some functions that are no longer needed. In any case most of the functions in both systems overlap. Therefore, the study and documentation of the existing system can provide information that will be very useful in the following steps.   It is not always easy to understand an existing system. During this step of the feasibility study the analyst must identify and inter- view key people in the existing system and collect documents, distribution lists, or reports produced by the system.   It is

14

usually easier to study an automated system than a manual one because the documentation is better organized. Sometimes it is helpful to summarize the gained knowledge in a systems flow-chart. In any case the analyst must keep in mind that he is not asked to document the existing system in detail, but only to understand it.

    4. Develop a high level model

        The next step is to develop a logical model that stresses the functions that must be performed by the new system without considering the specific physical way these functions are implemented in the existing system. In other words this model will represent the existing system as it ought to be rather than as it actually is.

        One of the techniques that is often used to develop a high-level logical model of a system is the Data Flow Diagram (DFD).

    5. Confirm the logical model

        In some cases the new and the old systems are perfor-ming exactly the same functions. In most cases however, the new system includes some additional functions. The analyst, working with the user, reviews the logical model developed in the previous step and adds new features to the model or eli-minates some features. The final product will be a logical model of the system that the user had in mind when he asked for an analyst.

    6. Develop and Evaluate Alternative Solutions

        With the agreed logical model of the system in mind the analyst will develop a variety of possible solutions to the problem. There are a number of different techniques that can help the analyst to generate these high-level solutions. It is beyond the scope of this thesis to describe each of

these techniques. A simple and effective method is described below.

First, the analyst generates a set of alternatives without considering any restrictions or constraints. At this point he may think of any system that could possibly solve the problem: a manual system, a fully automated system, a batch system on a mainframe, an interactive system on a main-frame or on a microcomputer, a DBMS or any possible combina-tion of such systems.

Next, the analyst sould consider the technical feasi-bility for each of these alternatives. For example if one possible solution is to create a data-base system on a micro-computer using a DBMS package that needs more memory space than is available on the micro, then this system must be dis-carded.

The remaining possible solutions are examined for eco-nomical feasibility. In other words the alternatives that can not be accomplished within the specified scope are ignored.

Finally, political feasibility is considered. If, for example, one solution is to replace or reduce the number of personnel working in a department - especially in a goverment organization - this may present a serious political problem in the organization and it is very probable that the management would not accept this as a solution.

7. <u>Select one alternative</u>

Based on the set of alternatives that have passed the above feasibility tests, the analyst should select the alter-native that he believes is the best. Keep in mind that the management usually bases its decision on the cost savings or the positive return on investment relative to the existing

system. Therefore a cost / benefit analysis should accompany the recommendation for an alternative.

8. Initiate a development plan

Assuming that management accepts the alternative recommended by the analyst, they need to know how long it will take to do the job, how many people from the organization may be involved and what is the approximate cost of each step in the process. The analyst must provide this information in the form of an initial implementation plan of the proposed solution.

9. Document and Present the Feasibility Study

The analyst collects and compiles the results of his feasibility study and prepares a written report. There is no standard format for the feasibility study report. The analyst will decide which is the best way to document his work during this study. However, for the reader who feels he needs a guideline, Figure C.3 in [Ref. 10] provides an outline of a typical feasibility study.

B. IMPLEMENTATION

1. Confirm the Problem Definition

The problem definition statement prepared during the first step of the specification stage included an understanding of the system objectives and scope. But is this exactly what the director of the SCD had in mind when he asked for a new system? The first task is to confirm that the problem definition is correct.

The problem is with the performance of the Soldiers Section of the Personnel Office in the SCD which is headed by a colonel. He is the one who suggested that a better personnel management system is needed. The analyst visits

him and asks if he agrees with his view of the problem. He confirms that the objectives are clearly stated, but he thinks that the scope is too costly. The SCD is not willing to allocate more than $15,000 for this project. Therefore the scope of the system is changed to accomodate a cost of $15,000. Now both the scope and objectives have been confirmed.

2. Study the existing system

The main purpose of this step is to understand the existing system. If it is known what the existing system does, then it is easier to find one or more ways to design a new system that eliminates the problems.

The sources of information that will help in the understanding of the existing system are two: people who work in the system and documents (reports, forms etc) used or produced by the system. During the interview the Colonel indicated that the best source of information is his assistant. The interview with this officer led the analyst to interviews with some other key people inside and outside the SCD and finally he came up with the following description of the system:

New soldiers enter the Signal Corps every two months. On the first day of each odd month a new group of draftees reports to the Enlistment Center (EC) where they are examined for physical and mental ability. The EC then prepares a list with the names and other information of the soldiers who were judged acceptable and sends one copy to the Basic Training Center (BTC) and one to the SCD.

Those draftees found capable of becoming soldiers are sent to the BTC where they receive training for two months in the basic subjects that every soldier must know. The average

18

number of soldiers that enter the Signal Corps every two months is about 1,000.

The SCD matches the number of new soldiers with the general needs for specialized personnel and calculates the number of soldiers that must be trained in each of the 15 different specialties. One list with these numbers is sent to the BTC which, after interviews with the soldiers, decides in which specialty each soldier will be trained.

The BTC prepares a list with the names of the soldiers and the specialty selected for each one and sends one copy to the SCD for information and one copy to the Special Training Center (STC) to assist in scheduling the training of the new incoming soldiers.

After the end of their basic trainig the soldiers are sent to the STC where they will receive training in the specialty they were assigned. The duration of this training is one, two, or three months, depending on the specialty. One week before the end of the training of each specialty the STC sends a report to the SCD with the names of the trained soldiers.

The SCD, based on this report and the personnel needs of the Signal Corps units, selects the units to which the newly trained soldiers will be assigned and sends a copy of the assignments list to the STC and the interested units. This procedure takes place during the first five days of each month.

Each unit of the Signal Corps submits two reports to the SCD at the end of each month. The first includes the names of the soldiers who retired or were separated from active duty during the month and the second includes the changes (if any) in the status of the soldiers in the unit. The SCD

uses these reports to update the soldiers' individual files and the units' files.

The above description, although very general, is sufficient for understanding the current system. Of course during the study of the system notes have been taken on many details that will help later to design the new system. For example, copies of the various lists, reports or forms that are produced or used by the system were requested.

The main observations about the existing system are described below.

(1) The basic functions that the Soldiers Section in the Personnel Office performs are as follows:
   - Calculates the number of soldiers that must be trained in each specialty.
   - Updates the soldiers' files.
   - Updates the units' files.
   - Performs the assignments of the soldiers.

(2) The Personnel Office does not exist in a vacuum. A number of other organizations (EC, BTC, STC and units) must provide it with timely information in order to be able to accomplish its mission. The delays reported in the problem definition have their primary origin in the untimely arrival of these data.

(3) Almost all the procedures that take place in the system are purposely concentrated at the beginning or the end of each month. This makes the problem of manually synchronizing these procedures even more complicated and the presence of errors more likely.

(4) The volume of information that is manipulated is so large that it is almost impossible to evaluate the

soldiers' requests when performing the assignments with a manual system.

(5) The general observation is that the existing system is well designed and the problems that have been reported are mainly caused by the relative difficulty that characterizes manual systems in dealing with great volumes of information under pressure of time.

3. Develop a high-level model

Now that the analyst has an understanding what the existing system does, the next step is to construct a high-level logical model of the system. This model will assist in the design of the new system. It can also be used as a communications tool with the people in the Personnell Office and the Management.

There are many techniques for describing a physical system. Some analysts prefer the systems flowcharts. Others think that data flow diagrams are better. The Data Flow Diagram (DFD) will be used to describe the personnel system. The reason is that at this early stage in the process physical implementation details should be avoided. The analyst wants to summarize the functions that are performed by the system, but not to be influenced by the specific way in which these functions are performed by the existing system. The DFD is excellent for a high-level description of a system because it stresses functional rather than physical implementation. It is also a diagram that is very easy to understand. There are only four symbols used in this diagram (fig.4.1). A source or a destination of data is represented by a square; a process in the system that transforms data by a circle or a rectangle with rounded corners; a data store by an open-ended rectangle and data flow by an arrow.

21
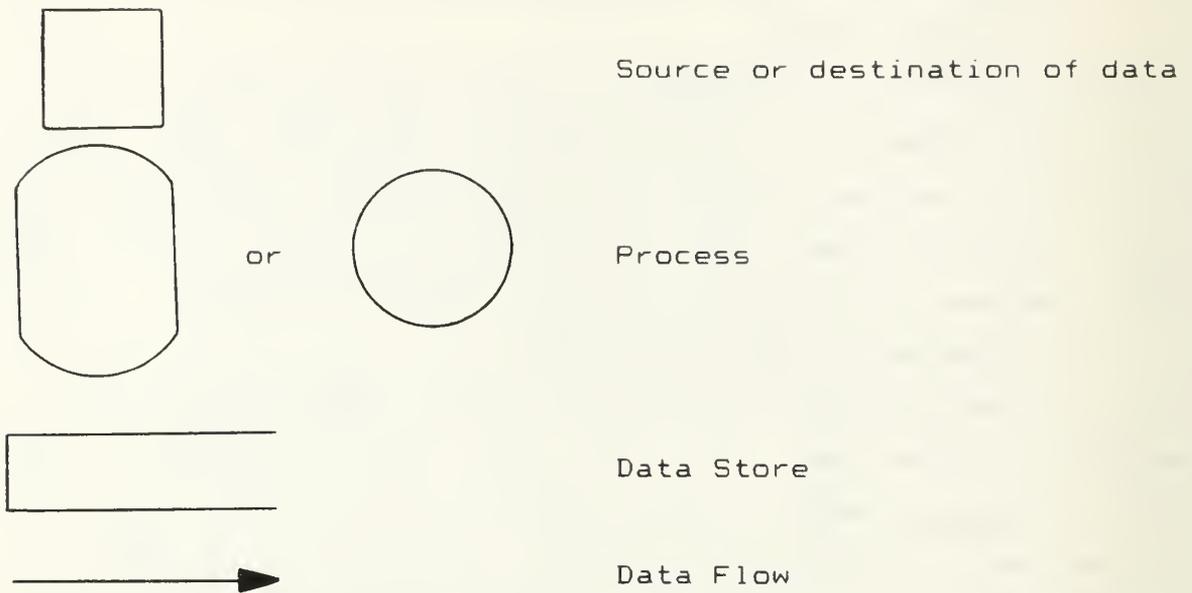
Figure 4.1 The Symbols of a Data Flow Diagram

In order to develop a DFD of the system, its four ele-
ments must be identified, begining with an identification of
the sources and destinations. From the description of the
system it is determined that there are four sources or desti-
nations of data: the EC, BTC, STC, and the units.

After reviewing the system description it is found
that the processes the system performs are:

  - Update Soldiers files

  - Update Units files

  - Estimate the number of soldiers for each specialty

  - Perform assignements.

The description of the system is reviewed again,
identifying the data flows and the data stores. At the end of
the enlistment the EC sends a list with the soldiers who
joined the Signal Corps to the SCD. This list is a data flow.
Is it also a data store? The purpose of this list is to update
the soldiers' files. Most of the time, this updating is not

22

performed immediately after the list is received. Therefore
a data store is needed to keep the data until the user is
ready to use them. Continuing in the same way the remainder of
the data flows and data stores are identified.

The next step is to develop the DFD (Figure 4.2). Note
that the Processes and the Data Stores are numbered so that
the reference to them is facilitated.
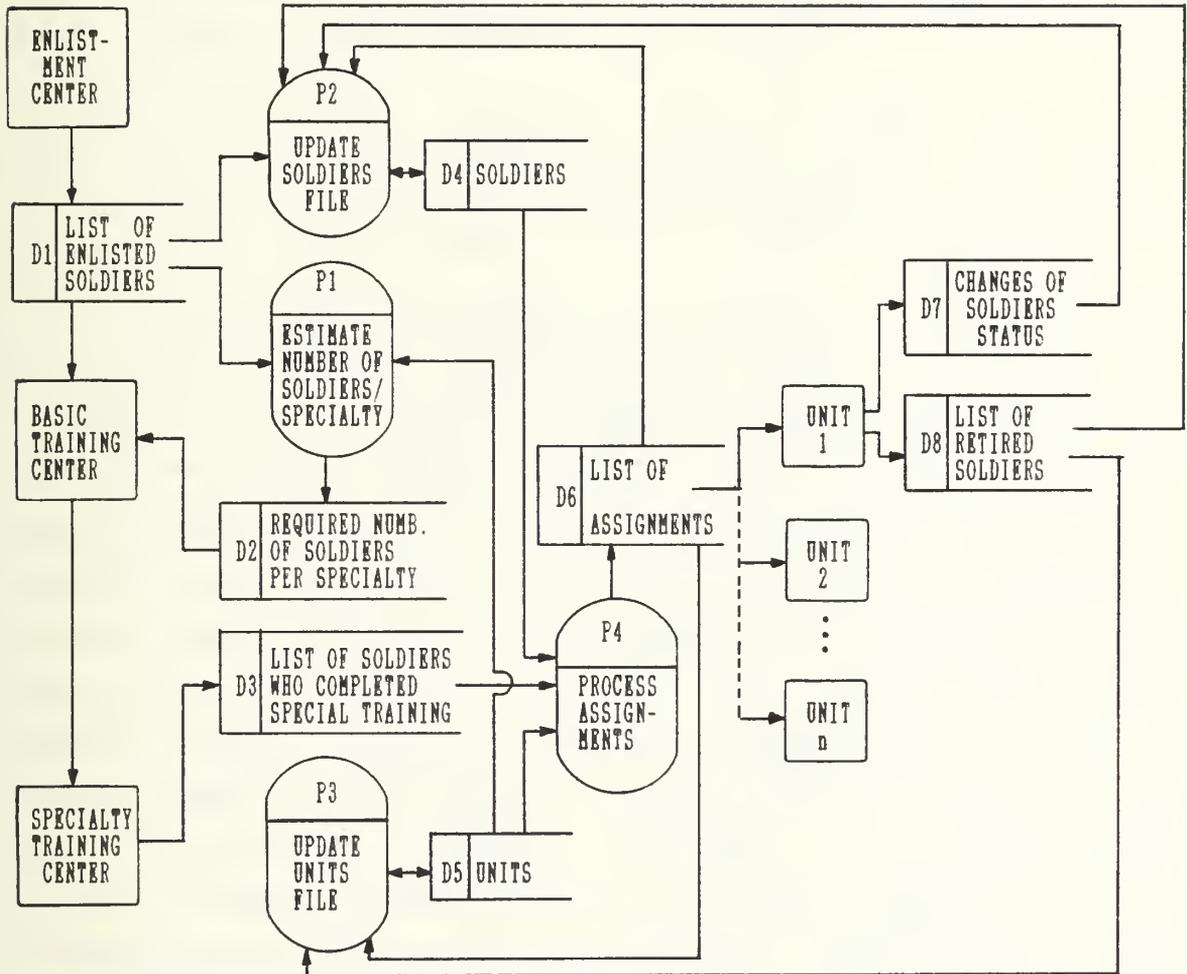
Figure 4.2   The Data Flow Diagram of the existing system

4.   Confirm the logical model

One of the major advantages of a DFD is that it is an
excellent communications tool. Thus it can be used to explain
our understanding of the system to the management.

23

The DFD is presented to the Colonel and he agrees that this is a good and accurate representation of the system. He also reminds the analyst that the details of the implementation of the Assignments process will be changed to meet the soldiers needs. There is no need to change any other process in the system.

5. Develop and Evaluate Alternative Solutions

The analyst now faces the question of how to solve the problem. He must consider and evaluate several possible solutions.

One simple solution could be to improve the manual procedures of the existing system. It is true, however, that people are not very effective when dealing with large amounts of complex data. Therefore, the use of a computerized system would be better. Should a traditional file processing or a database system be used? Database processing has a number of advantages over the file systems. These advantages become clearer as the volume and complexity of data increases. Clearly it is better to choose a Database system. On what computer should the database be implemented? The Generel Staff has its own Computer Center that runs a number of applications, such as payroll, mobilization procedures etc. There is enough capacity to run our database on the mainframe. There are some disadvantages to this solution, however. The director of the Personnel Office, would not have complete control of the soldiers' management. Also it is very likely that sooner or later the other Arms will ask to use the mainframe for their needs, but the Computer Center as it is now organized cannot undertake the management of all the Greek army soldiers.

Another way is to have a centralized machine in the SCD with a centralized database and implement some kind of a

network with the SCD as the central node and the EC, BTC, STC and the units as peripheral nodes. This may appear to be extremely expensive but it is not necessary to implement a hardware network. For example the data could be transfered via courier. The advantage of this fully centralized database is that it guarantees the integrity of data and it also helps the other units to benefit from the system by automating part of their work. This solution also would put full control of the whole system in the hands of the director of the Personnel Office, something very important for elimination of delays and other timing problems. The only disadvantage is that this is still a very expensive solution, far beyond the scope of the project.

Finally, a third way is to implement the database on a microcomputer system. This microcomputer would be positioned in the Personnel Office. The data from the units outside the SCD will continue to be delivered in the same way (manually written reports or lists) and entered into the computer. All the functions performed by the Personnel Office will be auto-mated, speeding up execution time and eliminating delays caused by the people in the office, but it does not address delays from outsiders. One possible disadvantage of using a microcomputer is that it may impose a limit in growth which means that we are not going to be able to go beyond micro-computer storage capability. One of the advantages of this solution is that it solves the delay and error problems inside the SCD. Even more important is that this system is able to solve the soldiers' assignments problem, which is one of our major objectives. Another benefit of this system is that it can be easily expanded to the other Arms.

What would be the cost of such a system? First the
type of microcomputer that would be appropriate for this
database must be determined. The main datastore is the SOLDIER
file which contains about 10,000 records, with each record
having about 400 characters. There is always an overhead of
about 20% for pointers, links etc, which results in about 500
characters per record. Therefore the total memory space needed
for this file is: 10,000 x 500 = 5 Mbytes. If an IBM/AT micro-
computer with a 20 Mbyte hard disk is considered, the system
can be implemented. The cost of such a microcomputer together
with the appropriate peripherals is about $6,000.

6. Select one Alternative

The analyst feels that the idea of using the General
Staff mainframe is the least desirable·by almost everyone. The
network solution is the most attractive but far beyond the
given scope limits. The solution of implementing a database on
a microcomputer is well within the limits of both the scope
and the objectives of the project and this is what the analyst
will recommend.

7. Initiate a Development Plan

The purpose of this step of the feasibility study is
to provide the management with a rough implementation plan of
the proposed solution, together with an estimate of the
approximate costs for each step in the plan.

The implementation schedule prepared by the analyst is
shown in Figure 4.3. Note that this is a rough estimate of the
implementation. The cost figures were calculated on the basis
of the salary of a major (usual rank of an analyst). The total
cost of this system will be $14,000 ($6,000 for hardware plus
$8,000 for implementation).

STEP	PERSONNEL TIME (months)	ELAPSED TIME (months)	COST
Feasibility	0.5	Completed	$ 1,000
Analysis	1.0	1.0	$ 2,000
Design	1.5	1.5	$ 3,000
Implementation	1.0	1.0	$ 2,000
TOTAL	4.0	3.5	$ 8,000

Figure 4.3 The implementation plan

8. Document and present the Feasibility Study

The feasibility study is now completed. The analyst writes a report including the results that he thinks are necessary to support his recommendation. Then he presents his report in a meeting with the director of the Personnel Office and the director of the SCD. They agree with the proposed solution and schedule. Now the analyst is ready to move to the next step in the process: the analysis.

V. ANALYSIS

A. THEORY

1. The Purpose of the Analysis Phase

The purpose of analysis is to define the system in terms of what is to be produced. The question of how the system will provide the required features will be answered later, during the design phase. Therefore, analysis is a logical process which does not really solve the problem, but decides what should be done to solve it. It is obvious that the importance of this stage is paramount. Decisions made here will influence the rest of the development process.

Many studies have been conducted which document that system analysis errors are extremely expensive to correct, especially if they are discovered late in the system development process Figure 5.1 [Ref. 11] illustrates the relative cost to make a change as a function of the phase in which the change is made. Note it is about 100 times as costly to correct a specification error during system testing as it is to correct it during analysis.

Another stydy by James Martin also documented that about 50% of the number of errors and 75% of the cost of error correction in operational systems is caused by errors during analysis. Finally there is strong empirical connection between failure to define a system adequately during analysis and failure to produce it. Nevertheless a great number of managers and users think of analysis as a time consuming process that must be minimized. This attitude has its origin in older times when coding was really the dominating process in system development.

28
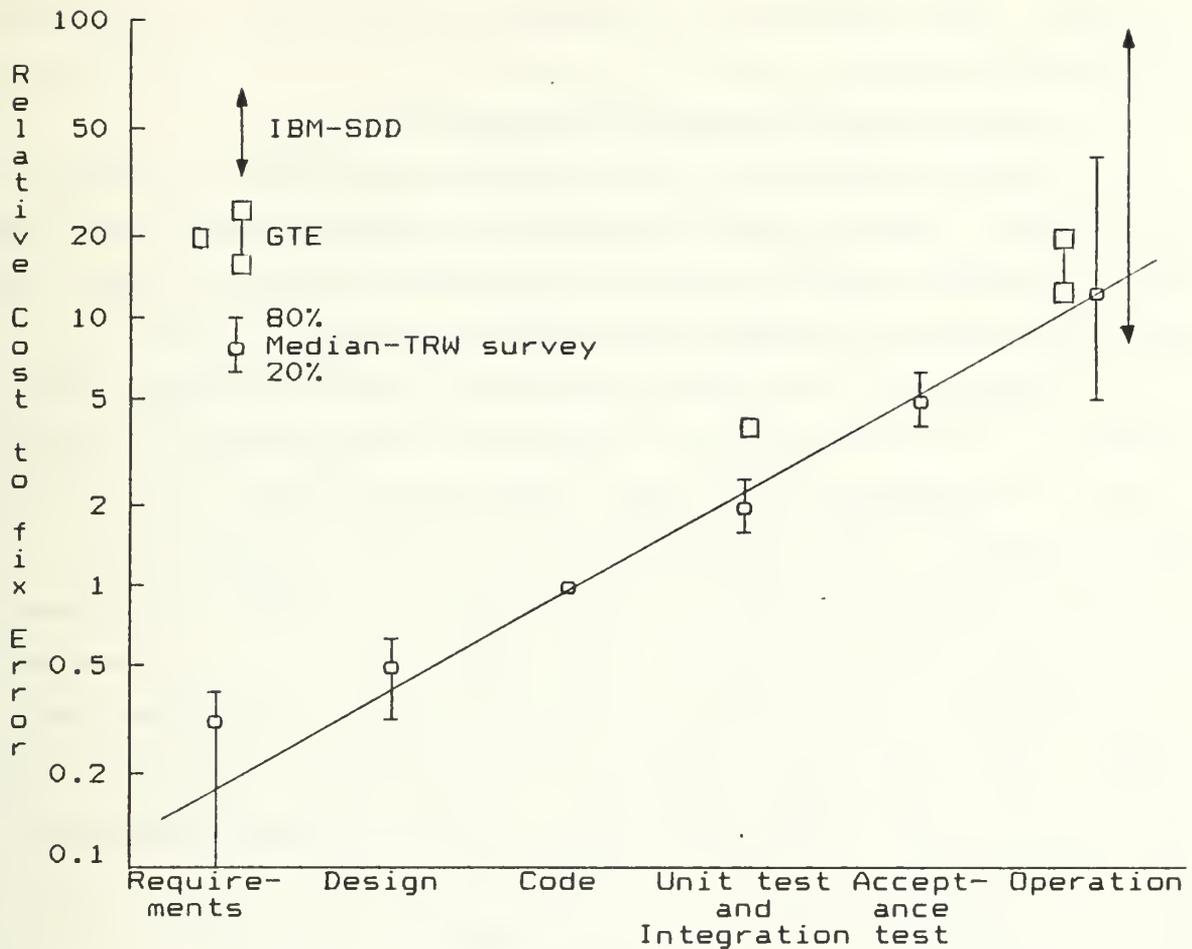
PHASE in which error was detected and corrected

Figure 5.1   Relative cost to correct an error

The main deliverable of the Analysis phase is a
document called Functional Specification or System Definition.
Very often this document is considered as being a training
manual, an operational handbook, or a management summary, but
it is not. The only purpose of this document is to provide a
specification of the functions to be performed by the system
and the constraints within which it must work. The Functional
Specifications will become the starting point for the Design
phase that follows analysis. Depending on the size and
complexity of the system, the functional specifications

document may consist of a few pages, or it may be packaged in several volumes.

2. Techniques of use during Analysis

Unfortunately, as was also the case with the Feasibility Study, there is no universally adopted method for the Analysis phase and the system's functional specifications are sometimes produced without following any method at all.

Because of the great importance of this phase, many attempts have been made by computer science people to bring a level of formalism to the production of the functional specifications. As a result of this effort a great number of different techniques have been produced. These techniques range from manually driven techniques to fully computerized ones. Some of them provide a way to generate the system definition, whatever form this may take. Others simply provide a way of presenting the definition. The best technique is one that combines both results for the system definition process. Again some techniques are very effective in certain aspects and for particular types of systems. For the reader who is interested in the details of any particular analysis technique, the names of the most popular ones together with the reference of a description of the technique follow:

- Structured Analysis. [Ref. 12]
- PSL/PSA. [Ref. 13]
- Sructured Analysis and Design Technique (SADT). [Ref. 14]
- Controlled Requirements Expression (CORE). [Ref. 15]
- Software Requirements Engineering Methodology. [Ref. 16]
- Finite State Machines (FSM). [Ref. 17]
- Petri Nets. [Ref. 18]
- Jackson System Development (JSD). [Ref. 19]
- Software Development System (SDS/RSRE). [Ref. 20]

30

All of these techniques in some way model the system being defined. The difference is that each technique focuses on a different aspect of the system such as data flow, data structure, control of flow, etc. Therefore, before choosing an analysis technique one should first identify which aspect of the system is the most important. In the following section a very brief and simplified description of the Structured Analysis method [Ref. 12] is given and will be followed during the implementation of the Analysis phase.

3. A brief description of Structured Analysis

De Marco [Ref. 12] and Gane and Sarson [Ref. 21] have defined a methodology which is primarily involved with the application of a particular set of tools to the production of a Structured Functional Specification. These tools are: The Data Flow Diagrams, the Data Dictionary and Process Description Tools. The Data Flow Diagrams (DFD) and their use were described in Chapter IV.

The Data Dictionary (DD) is used to define the data flows and data stores that appear in the DFDs. In other words the DD is a collection of data about data. The basic idea is to provide information on the definition, structure and use of each data element an organization uses. A data element is a unit of data that cannot be decomposed. A DD usually consists of a listing of all elements found in each data store. For each data element, information about its name, aliases or synonyms, description, format, location and other characteristics is recorded in the DD.

The Process Description Tools are used to define the processes in the DFD that cannot be further decomposed and are called primitive processes (or primitives). Primitives are not described in terms of further DFDs and so require some other

31

means of definition. Structured Analysis uses Structured English, Decision Tables and Decision Trees for this purpose. For each of the primitive processes a description called mini-spec is written using these tools.

One way to produce the Functional Requirements using the Structured Analysis tools is described below.

First explode the DFD which was produced during the Feasibility study by taking each process in the DFD and breaking it down into its subfunctions. These lower level functions, together with their own data stores and data flows, become processes on a new more detailed version of the DFD. This decomposition continues to the point of code generation, at which the analysis phase ends.

The next step is to define the data flows and stores down to the element level. For each process in the exploded DFD the elements that must appear in the output and input are identified. Then a list is made of each data store and data flow together with the data elements it contains. Finally, the DD is constructed in which information is recorded about the name, description, format, use, etc of each data element.

The third step during Structured Analysis is to describe each process at a high level, using Structured English, Decision Tables or Decision Trees.

Note that this process is not linear. For example, while the analyst is defining the data elements he may find that he must go back and change the DFD, or a process description might imply the addition of new data elements in a data store.

The exploded DFD, the Data Dictionary and the process descriptions form the Functional Specifications of the system, which after being reviewed and approved by the user, are presented to the management.

B. THE IMPLEMENTATION OF STRUCTURED ANALYSIS

1. Explode the Data Flow Diagram

The DFD developed during the Feasibility study phase
(Fig. 4.2) is the starting point for the Analysis phase. This
DFD contains four processes, one for each major function in
the Personnel Office. During this phase the analyst will con-
sider each process separately and try to decompose it into
lower-level processes.

a. Decompose process P1

Consider the first process, P1. The purpose of
this process is to calculate how many of the new enlisted
soldiers are needed to train in each specialty. Figure 5.2
shows the DFD of this process.
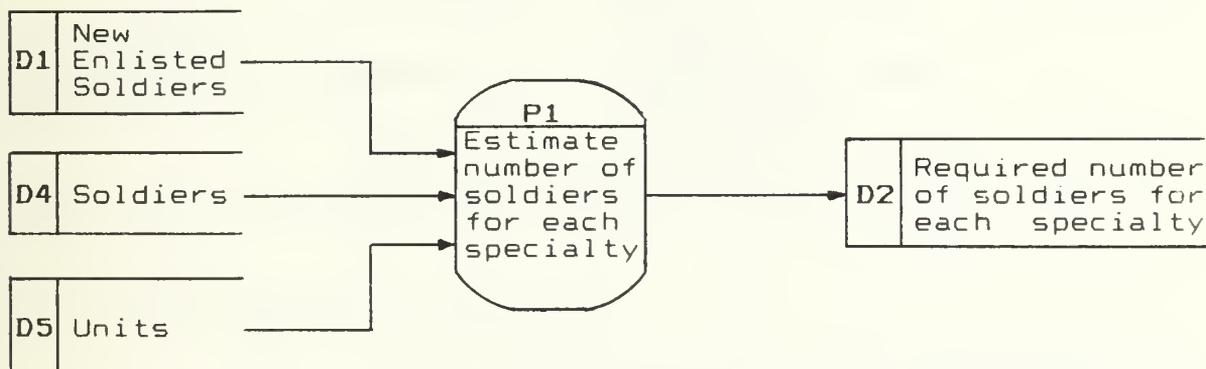
Figure 5.2   The DFD for process P1

Trying to decompose P1 we find that it cannot be
divided into functional groups to form separate processes.
Therefore process P1 will not be further decomposed.

b. Decompose process P2

The DFD for process P2 is shown in figure 5.3.
This process updates the Soldiers file with information from
the following incoming data stores: D1 (new enlisted soldiers),
D3 (soldiers who completed specialty training), D6 (assign-
ments), D7 (changes of status) and D8 (retired soldiers).

33
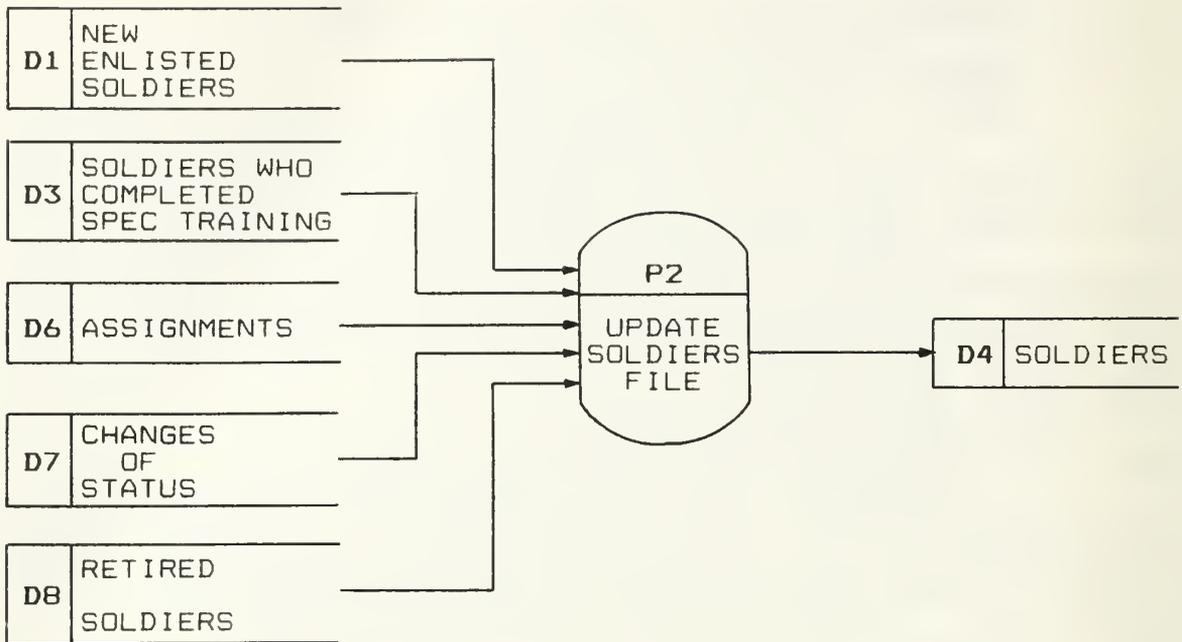
Figure 5.3   The DFD of original process P2

                These subfunctions are performed by this process:

  - P2.1 : Add new records to Soldiers file.

  - P2.2 : Update records in Soldiers file.

  - P2.3 : Delete records from Soldiers file.

                The new Data Flow Diagram for process P2 after

this decomposition is shown in figure 5.4.
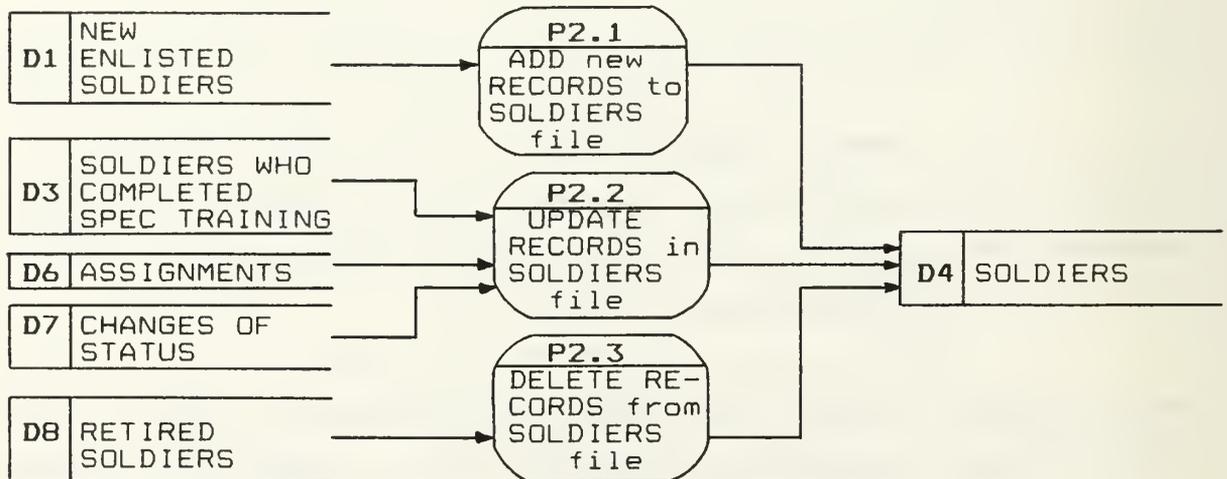
Figure 5.4   Decomposition of process P2

34

Some of the processes in this new DFD require further decomposition. For example process P2.2 uses three different data stores that enter the system at different times to update the Soldiers file. Therefore it could be decomposed into three subfunctions shown in figure 5.5.
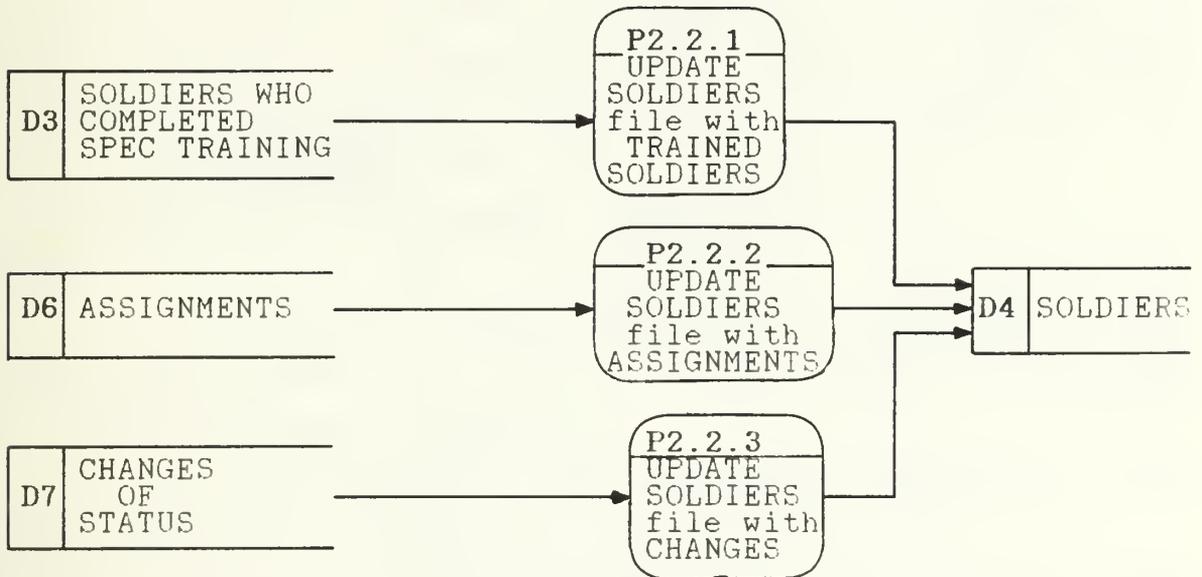
Figure 5.5 Decomposition of process P2.2

c. Decompose process P3

The function of P3 is to update the Units file with the information contained in the data stores D6 (assignments) and D8 (retired soldiers). Figure 5.6 shows the original version of the DFD for this process.
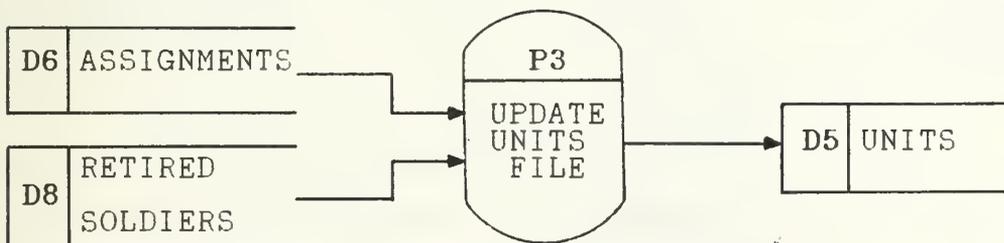
Figure 5.6 The DFD for process P3

35

The explosion of process P3 is shown in figure 5.7.
Two new processes are created:

- P3.1 : Update the Units file with the retired soldiers.
- P3.2 : Update the Units file with the new assignments.
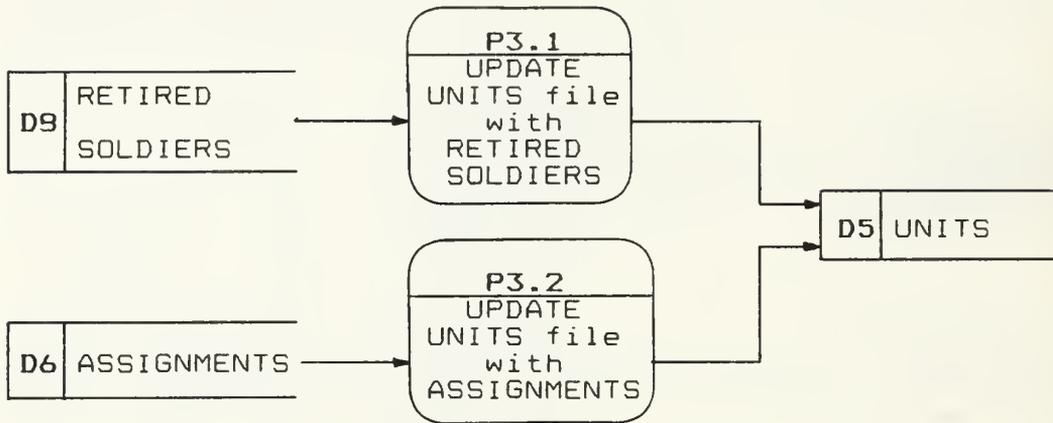
Figure 5.7   Decomposition of process P3

d.   Decompose process P4

Finally process P4 is considered.   This process is
used to assign the newly trained soldiers to units.   The Data
Flow Diagram for P4 is shown in figure 5.8.
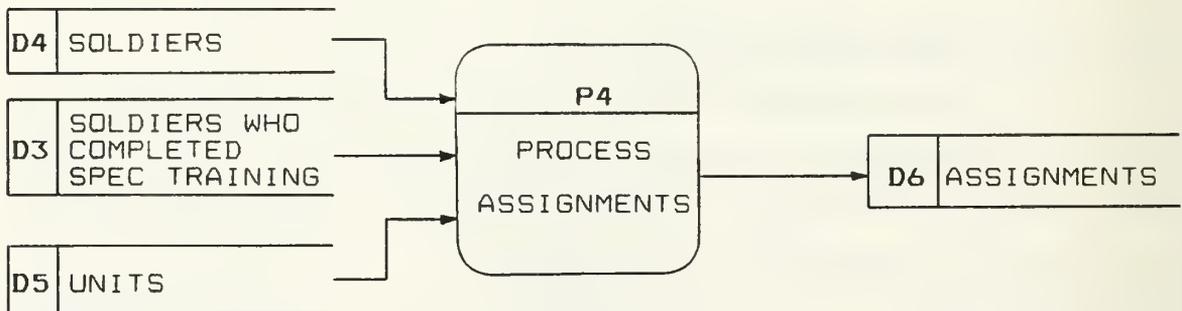
Figure 5.8   DFD of process P4

Three subfunctions of process P4 can be identified
as follows.   The first subfunction (process P4.1) uses data
store D3 (soldiers who completed specialty training) to deter-
mine who is going to be transfered and data store D4 (soldiers)
to get information about each soldier.   This information is

36

used by P4.1 to calculate the transfer points for each soldier
that will decide the priority for transfer among the soldiers.
Thus the need for a new data store D11 (Soldier Qualification
Points) to store this information is identified.

The second subfunction (process P4.2) uses data
store D3 to get the number of soldiers to be assigned and data
store D5 (Units) to read the number of existing and required
soldiers in each unit. It then calculates the number of soldi-
ers of each specialty that will be assigned to each unit. This
information will be recorded in a new data store D12 (Unit
Needs).

Finally, a third subfunction (process P4.3) will
assign each soldier to a unit using the information contained
in the data stores D11 and D12.

The decomposition of process P4 is shown in figure
5.9.
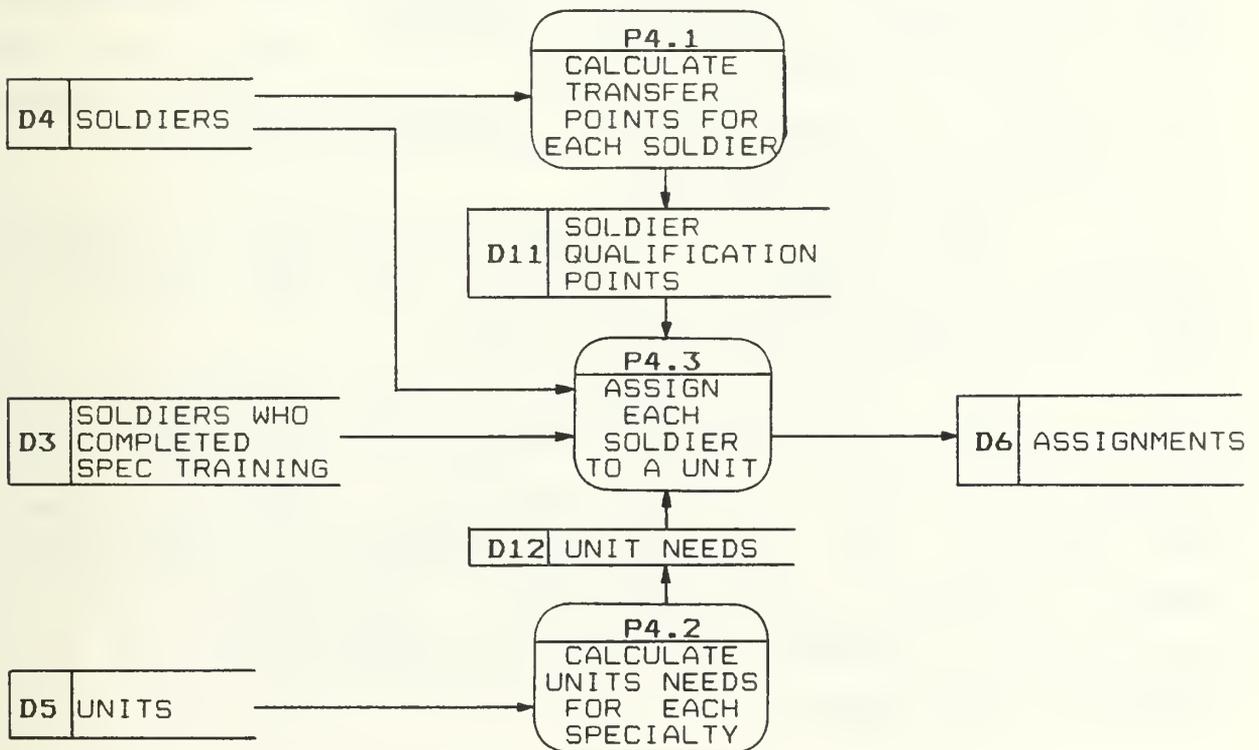
Figure 5.9   Decomposition of process P4

37

2.   Define the Data Elements

One objective of analysis is to define the data flows and data stores down to the element level. To accomplish this each process in the data Flow Diagram is considered separately and the data that must appear as its outputs and inputs are defined.

a.   Using process P1

Process P1 calculates the number of soldiers to be trained in each specialty. Therefore its output, which is stored in data store D2, will contain at least two elements: The SPECIALTY and the REQUIRED SOLDIERS per specialty.

To calculate the Required Soldiers, process P1 uses the following elements:

(1) Total number of new enlisted soldiers.

(2) Total number of soldiers per specialty
    required to satisfy the current unit's needs.

(3) Total number of soldiers per specialty
    to retire in the next 4 months.

(4) Number of soldiers currently undergoing
    training in each specialty.

Data store D1 (New Enlisted Soldiers) provides information for element (1). Therefore, D1 must contain a field: TOTAL NUMBER OF ENLISTED SOLDIERS.

Next consider element (2). Data store D5 (Units) in the form that it is used now consists of a UNIT NAME field followed by one field for each SPECIALTY, which is divided into subfields for the EXISTING, REQUIRED and COMPLEMENT number of soldiers for this Specialty. Thus, element (2) can be calculated by adding all the Complement fields of one specialty in all units.

38

As for element (3) the remaining time of service
for each soldier is needed. One way to obtain this information
is to include a field REMAINING TIME in D4. Process P1 will
read this number for each soldier and decide if it is greater
or less than 4 months. But the Remaining Time of Service is
a value that must be continuously updated (it changes every
day). Is there a more convenient way to derive the desired
information? One way is to calculate the date when the remai-
ning service time becomes 4 months. Then P1 will compare the
current date with this date and if the current date is already
past this date then the remaining time will be less than four
months. This date is calculated by adding the duration of
service to the date of enlistment to get the retirement date
and then subtract 4 months from it. Therefore data store D4
must provide the elements DURATION of SERVICE and DATE of
ENLISTMENT. For purposes of speeding up the execution of P1,
it may be better to store this date in D4 after it is calcula-
ted for the first time.

Finally, element (4) (i.e., the number of soldiers
currently enrolled in training in a given specialty X) must be
calculated. We should be able to find this number by counting
the number of records in D4 with COMPLETED TRAINING = False
and SPECIALTY = X. Unfortunately the SPECIALTY field cannot
be updated before the completion of specialty training. There-
fore a new data store will be needed to provide the soldiers
enrolled in training in each specialty. This will become data
store D14 (CURRENTLY TRAINING SOLDIERS) and it will contain
the fields ID_NUMBER and SPECIALTY. This data store will be a
list submitted by the Training Center every month together
with D3.

39

The needed number of soldiers per specialty can now be determined if the specialty name is known. None of the data stores in the DFD seems to contain this information, though. Obviously something is missing. The user is asked where the names of the specialties are stored. The answer is that they do not use a data store for the names of the ten specialties, because they can easily remember them or they can look them up on a piece of paper, obviously not feasible for a computerized application. A new data store is required to keep the names of the specialties. This data store will be D9 (SPE-CIALTIES) with one data element SPECIALTY NAME.

The DFD of process P1 is updated to show the new data stores D9 and D14 (Figure 5.10).
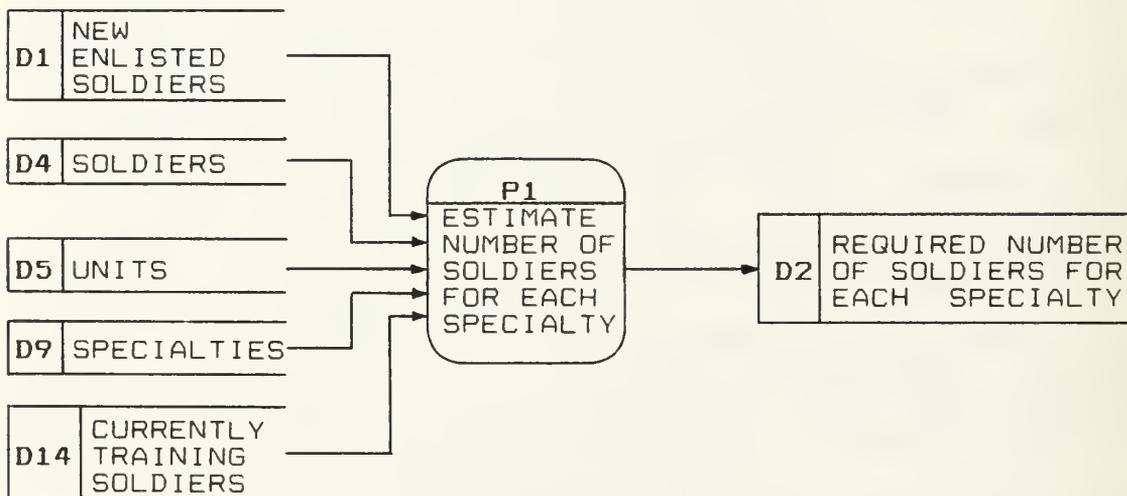
Figure 5.10   Process P1 after the addition of D9 and D14

Following is a list of the data stores used by process P1, together with the data elements they contain:

- D1 : NEW ENLISTED SOLDIERS
    1. Total number of enlisted soldiers

- D2 : REQUIRED SOLDIERS FOR EACH SPECIALTY
    1. Specialty
    2. Required soldiers for specialty training

40

- D4 : SOLDIERS
     1. Date Enlisted
     2. Service duration
     3. Date when remaining service equals four months

- D5 : UNITS
     1. Unit name
     2. Specialty
     3. Required number of soldiers
     4. Existing number of soldiers
     5. Complement number of soldiers

- D9 : SPECIALTIES
     1. Specialty

- D14: CURRENTLY TRAINING SOLDIERS
     1. ID number
     2. Specialty

b. Using the remaining processes in the DFD

Proceeding in the same manner the data elements that are required by the rest of the processes in the Data Flow Diagram are identified. New data stores are added and the DFD is expanded where necessary. At the end of this process the DFDs for the processes P1 through P4 have taken the form shown in figures A.2 through A.5 of Appendix A. Also the data stores are shown in section B.1 of Appendix B. Note that a new data store, HISTORY, was added to keep the information about a soldier after he has retired and before his record is deleted from the Soldiers file.

3. Construct the Data Dictionary

For each data element already registered in a data store, information is recorded about its name, aliases, description, format, location, etc.   The Data Dictionary is shown in sections B.1 and B.2 of Appendix B.

4. Write Process Descriptions

As previously discussed the purpose of this step is to describe each process in the DFD at a high level. There are many available tools to be used for this purpose such as, Decision Tables, Decision Trees and Structured English. Structured English was chosen to document the processes. The

41

complete documentation is shown in figures C.1 through C.11 of Appendix C.

    5.   Review the Functional Requirements

        Working with the user the Functional Requirements are reviewed to decide if they are complete.

        During this review it was found that there are two more processes that need to be added to the system.

        The data stores D1, D3, D7, D8 and D14 enter the system in manual form. The computer cannot process manual files.   Therefore a new process P5 (ENTER DATA) is required which will transform the manual data stores into electronic files that can be manipulated by the computer. Figure 5.11 shows the DFD for this new process. The exploded process P5 is shown in figure A.6 of Appendix A and its algorithm description in sections C.12 through C.16 of Appendix C. Finally a last process P6 (GENERATE REPORTS) is needed. This process prints out the reports that the Personnel Office of the SCD
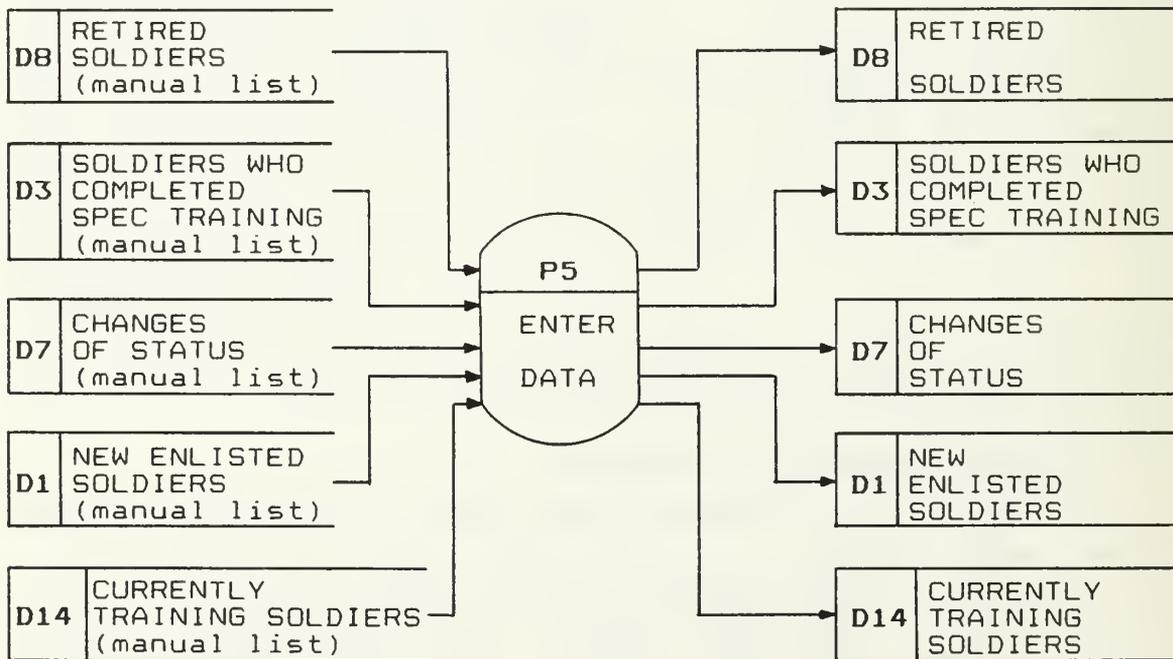
Figure 5.11   The DFD of process P5

42

must send to the training centers and the units (i.e., the Assignments list and the list with the Training needs).

The Data Flow Diagram for process P6 is shown in figure 5.12 and its exploded form in figure A.7 of Appendix A. Also the algorithm description for this process is contained in sections C.17 and C.18 of Appendix C.
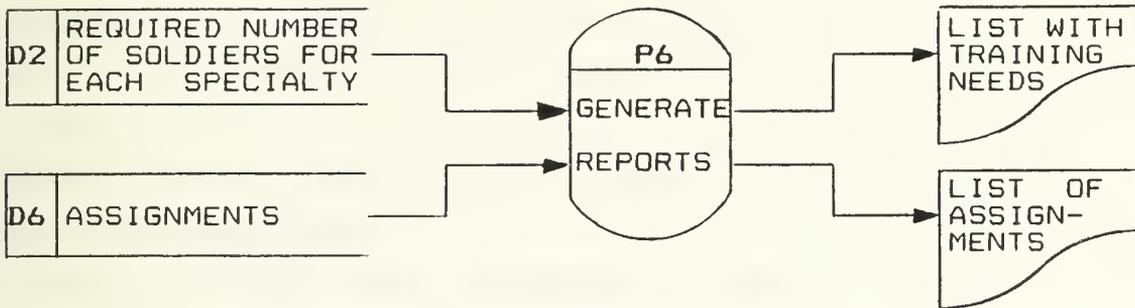
Figure 5.12   The DFD of process P6

The final version of the system Data Flow Diagram is shown in figure A.1 of Appendix A.

6.   Present the Functional Requirements to the Management

At the beginning of this chapter it was mentioned that the process of producing the Functional Requirements for the system is not linear. Very often the analyst backtracks and repeats certain steps in the process. Finally it is felt that the Functional Requirements are complete and they are presented to the management. The contents of Appendices A, B and C form the Functional Requirements of the system.

43

VI. SYSTEM DESIGN

A. THEORY

1. The Purpose of the System Design

During the analysis stage the analyst defined **what** is going to be produced. The next step is to answer the question of **how** to produce the system and this is the purpose of the Design phase.

The majority of the literature that exists on the different steps in the development of a system views Design as one big step which takes the Functional Specifications produced during Analysis as its input and designs a system that satisfies them. A number of people, however, believe that it is better to break the Design stage in two phases: System Design and Detailed Design.

During System Design it is determined <u>in general</u> how the system will be implemented by identifying different alternative solutions and selecting one alternative that best satisfies the system needs.

During Detailed Design it is determined how <u>specifically</u> the selected alternative should be implemented.

However, no matter which design approach is followed the system designer must always keep in mind that the final product of this phase should be a system design which first provides all the required functions and second can be easily implemented.

2. Identify Alternative Solutions

System Design begins with a search for different systems that meet the user's requirements. To make this search easier each one of the system's components is considered

44

separately. A system usually consists of four components: Data, Hardware, Software and People.

a. Data alternatives

The main function of any computer system is data processing. The way in which data is organized greatly affects the system structure and most of the time its effectiveness. There are two primary data alternatives.

One way is to organize data in files, which exist independently of each other, and their structure is distributed across the application programs. Another alternative is to utilize a database.

There are advantages and disadvantages related to each of the two alternatives and the system designer should evaluate them and choose the one that best satisfies the needs of the particular application. Sometimes, a combination of these two alternatives should be used.

b. Software

The evaluation and selection of the appropriate software for the system is usually a difficult and time consuming task. During this step different programming languages must be evaluated that can be used to write the applications programs. The operating system to be used should also be considered. The most difficult decision, however, is to choose a Data Base Management System (DBMS) when a database is involved.

The functions provided by a DBMS must match closely the user requirements. Unfortunately, most of the existing DBMSs do not provide the same functions and the same interfaces and therefore, we must proceed very carefully in choosing the correct DBMS.

Gordon Everest in his book "Database Management" has devoted a whole chapter to the DBMS selection and aquisi-

tion process [Ref. 22]. For the reader who needs more details on this process, the reference provides a very good guide.

There is a substantial difference between a DBMS for a personal computer and one for a large system. First of all the size of the databases that must be managed by a perso- nal computer DBMS is many orders of magnitude less than the size of a commercial database. Also the personal systems are usually intended for use by only one person, while the large mainframe systems are serving concurrently many different applications and users. Because of these differences a commer- cial DBMS should be much more sophisticated and able to coordinate the complex database activities. Therefore, it is always much easier to select one of the many low-cost DBMS packages available in the market for microcomputers.

c. Hardware

In most cases the management wants the new system to run on existing hardware. If this is possible without any major additions or modifications, it is wise to keep the current system in place. Sometimes, however, new applications require a new computer system. Also the involvement of a database usually implies the use of special hardware with more main and secondary storage space, faster CPU etc. The DBMS vendor will provide information on the hardware needed to be used.

d. People

Finally the people who are involved in the system are considered. We must decide who the end-users are going to be and the extent of training required.

If a database is involved then it must be decided whether a Data Base Administrator (DBA) will be needed. The function of the DBA is "to protect the database and to ensure

that it is structured and used so as to provide maximum bene-
fit to the community of users" [Ref. 9: p. 30]. When a great
number of different applications and users are using the data-
base the presence of a DBA becomes a necessity. The DBA could
be a single person or a whole team. The cost of the DBA staff
should be considered as part of the cost of the alternative
being evaluated.

3. Select one alternative

The next step is to select one of the different alter-
native sets of the system components identified during the
previous step.

A number of different techniques exists that assist in
the selection of an alternative solution. David Kroenke
[Ref. 9] has described three such techniques which are briefly
reviewed below.

The first technique is called Subjective Evaluation.
This approach is the cheapest, quickest and most frequently
used. The purpose is to subjectively evaluate each alternative
and to make an intuitive decision. Thus the criteria for
comparing alternatives are first identified. Next the criteria
are weighted according to their relative importance. Then,
each alternative is subjectively scored by the members of the
evaluation team. The total score for each alternative is then
calculated and the alternative with the highest total score is
selected.

Cost/benefit analysis is another technique, which
gives a reasonable picture of the costs and benefits associa-
ted with each alternative solution, so that management can
compare the alternatives and decide which one is a good
investment. First the costs of developing each alternative
system are identified. The cost of each phase in the system

47

development is estimated separately and the total cost of development is the sum of these costs. Next the cost of maintaining and operating the system after it is implemented is estimated and is added to the development cost. The expected benefits from the use of the system is estimated next.

Note that the development costs occur only once. The operating costs and the benefits occur continuously after the system becomes operational and are not equally distributed across time. Also, as is the case with most computerized systems, they usually become obsolete in a few years. Therefore it is wise to estimate the return on investment for any system for a period no longer than five years. If the benefits during this period do not exceed the sum of the development and operation costs then this system might not be feasible.

A third technique suggested by Kroenke is to use a combination of both of the above techniques. Subjective evaluation can be used to reduce the number of alternatives to two or three and then perform a cost/benefit analysis on these alternatives to choose the best one.

B. THE IMPLEMENTATION OF SYSTEM DESIGN

1. Identify Alternative Solutions

During the Feasibility Study phase a number of alternative solutions to the problem were developed and evaluated. The result of that evaluation was that the implementation of a database system on a microcomputer is the most desirable solution. These results will be used during System Design to help make the effort of identifying alternative solutions easier.

a. Data Alternatives

The system can be implemented using different data processing technologies. One way is to continue using the

existing manual files for all processes in the system, except for the assignments processing for which traditional file processing can be used. Another option is to implement the whole system using file processing. Finally, a third option is to store all data in a single database.

    b. Software

        In the market there are a large number of DBMSs (over one hundred) that run on microcomputers. Most of these DBMSs provide a programming language that can be used to write application programs when all processing requirements cannot be handled by the database functions provided by the DBMS. Some of these full function DBMSs are listed below:

  - DATAFLEX from Data Access Corp., Miami, FL
  - dBASE II, III from Ashton-Tate, Culver City, CA
  - INFORMIX from Relational Database Systems, Sunnyvale, CA
  - MAG/base III from Micro Applications Group, Canoya Park CA
  - MDBS+QRS from Micro Data Base System Inc., Lafayette, IN
  - OPTIMUM from Uveon, Denver, CO
  - ORACLE from Oracle Corp, Menlo Park, CA
  - Q PRO-4 from Quick'n Easy Products, Langhorne, PA
  - R:BASE from Microrim, Bellevue, WA
  - REVELATION from Cosmos, Seattle, WA
  - UNIFY from Unify Corp., Portland, OR

    c. Hardware

        As described in the feasibility study the micro-computer solution is the only acceptable one for the current application.

    d. People

        Currently six men are working in the Soldier Section of the SCD Personnel Office (a captain, two sergeants and three soldiers). None of these needs to be replaced if the

49

current system is maintained. If it is decided to implement a database system using a microcomputer then at least four people will be needed: One captain, one sergeant and two soldiers. These people will be required to have some knowledge of microcomputers.

2. Select one Alternative

After the evaluation of the different alternative solutions and in agreement with the results and constraints of the feasibility study, the system components are selected as follows:

a. Data

A database will be used to store all the data in the system.

b. Software

The dBASE III Plus package was chosen as the DBMS. During the evaluation process a number of other DBMS packages were found that provide almost the same functions as dBASE III Plus. The prices of these packages were also similar. Some of the reasons for selecting dBASE III Plus are:

- Product stability.

  dBASE III Plus (was dBASE II before) has been in the market for a long period and the number of people using it is continuously increasing.

- Maintenance support.

  There are many enhancements of the product and the users interviewed were generally satisfied by the way the vendor responds to occurring problems.

- Documentation and Training.

  The documentation is very well written and readable. There are also several references one can find that make the learning and use of dBASE III Plus easier.

The dBASE III Plus features, limitations and software/hardware requirements are described below.

dBASE III Plus is a relational DBMS which runs on IBM microcomputers or compatible machines and stores information in relational data tables.

The database can be processed in two ways. One way is interactive command processing in which the data in the database is manipulated by means of commands entered interactively from the keyboard, and the results are displayed on an output device such as a monitor or a printer. A second way is through application programs. An application program is a collection of dBASE III Plus commands stored in a file. These programs can be loaded and executed by the DBMS. This capability makes dBASE III Plus useful for application development.

The database files used by dBASE III Plus can hold a maximum of:
- One billion records
- Two billion bytes
- 4000 bytes per record
- 128 fields per record
- 254 bytes per field

dBASE III Plus allows a maximum of 15 files to be open at once including database, index, memo, procedure and program files. This sounds like a serious limitation but if the application programs are carefully designed these problems can be avoided. Note that this limitation is imposed by PC_DOS and not by dBASE III.

dBASE III Plus is designed to run on the IBM PC, IBM PC XT, IBM PC AT and the IBM-compatible microcomputers. It requires MS_DOS or PC_DOS version 2.0 or later. The minimum memory requirement is 256 K under DOS version 2.0 or 2.1.

51

dBASE III requires more memory to run under DOS 3.0 or above and if you want to replace the dBASE III program editor with an external editor then at least 384 K of memory will be required. For a serious application development the microcomputer should have at least 512 K main memory, a 360 K floppy disk and a 10 megabyte hard disk.

    c.   Hardware

    An IBM personal computer AT with a 20 Mbyte hard disk and two 360 K floppy disks is recommended for implementation. Also a monitor and a printer will be used as output devices.

    d.   People

    One captain, one sergeant and two soldiers with some knowledge of computer science and especially on microcomputers will be required.

VII. DETAILED DESIGN

A. THEORY

1. Purpose of the Detailed Design

As stated before the purpose of the Detailed Design phase is to determine **how specifically** the system will be implemented.

During System Design the solution strategy which best satisfies the user's requirements was selected. If this solution involves a database then the Detailed Design should be divided in two tasks: The Database Design and the Design of the Applications Programs.

2. Database Design

The Database Design is usually divided into two stages: Logical Database Design which is entirely independent of limitations imposed by the hardware or any particular DBMS and Physical Database Design which is dependent on the DBMS selected during System Design.

a. Logical Database Design

During this stage the database logical structure is developed by determining the actual contents of the database in a way that satisfies the user requirements without using any particular DBMS.

What are the steps that should be followed during logical database design? Although many techniques have been defined, unfortunately once again there is no algorithm to follow. David Kroenke in his book "Database Processing" [Ref. 9: p. 177] writes:

"... database design is an intuitive and artistic process. There is no algorithm for it. Typically, database design is an iterative process; during each iteration the goal is to get closer to an acceptable design..."

The different techniques that exist for logical database design vary from very general and abstract to very detailed techniques that focus only on specific aspects of the database . The process described next provides a simple way to create a logical database design and includes the major steps found in most techniques.

First the data to be stored in the database is identified. Using the Data Dictionary (DD) prepared during Analysis, synonyms are identified. Synonyms are two or more different names for the same data element. It is desired to remove synonyms from the database in order to eliminate ambiguity and redundancy. For this reason all different names of the same data element are replaced with one standard name, and a new revised version of the Data Dictionary results.

During the analysis phase every data element that is needed by the system was recorded in the DD. Next a more detailed examination of the DD is required in order to identify data elements that cannot be part of the database or must be further analyzed. In this way a final version of the DD is created.

The next step in the logical database design process is to specify the logical database records. By examining the DFD of the system the data stores and data flows which should become records in the database are identified. The data elements, alias fields, that each record should contain are already listed in the DD. Obviously this step is very straightforward and should not be difficult to execute. However, some additional items must be acomplished. One is to determine which records should be combined or separated. A closer examination of the process descriptions of the functional requirements may reveal that some processes utilize only a small part

54

of the information stored in some of the records. Performance could then be improved by breaking such records into more records according to how they are used by each process. In other cases two or more records should be combined into one if, for example, they are used simultaneously by most processes. In his effort to determine which records should be joined or separated the system designer should also be guided by the anticipated future use of the records.

Now that the final structure of the database has been defined one final item is required: determine the primary key for each record. It is required that each record be uniquely identified in any of the files in the database. To do this one or more of the fields in the records are used as identifiers. These fields are called keys. It must be ensured that a selected key uniquely identifies a record and this is not always an easy job.

b. Physical Database Design

This stage takes the logical database design as its input and transforms it into a form that is acceptable to the hardware and to the Data Base Management System (DBMS) that will be used.

Since this transformation varies greatly from one DBMS to another it is not possible to provide a general description of the process. In order to be able to perform a physical database design using a specific DBMS, the designer should study its documentation first.

3. Design of the Application Programs

After the database has been designed the next step is to design the application programs. This design will later result in code using the Data Manipulation Language (DML) provided by the DBMS. At this stage we work independently of

55

any particular programming language, designing the programs that will call on the DBMS in order to provide the desired database service.

The designer's goal is to identify the applications programs and then develop a set of specifications for each program that will contain the required information to support writing the actual code during the next stage: the Implementation.

It is amazing that so many people have described so many different program design techniques. Terms like Modular design, Top-down design, Bottom-up design, Structured design, Stepwise Refinement, Systematic Programming, Transform Analysis, Transaction Analysis etc, are well known to almost every computer science student. There is also extensive debate as to which is the best technique. In this presentation of a strategy for designing application programs no particular design technique will be followed in detail, although there is some similarity with Structured Programming. A good designer should know as many different techniques as he can but should not commit himself to only one of them.

The programs in an application do not run independently of each other. One program is usually called by another and either during or after execution, passes control to another program. The hierarchy and the flow of control among the application programs of a system can be represented using a Structure Chart.

A structure chart is a pictorial representation that uses simple boxes and statements to describe the functions in a system. To illustrate this concept Figure 7.1 shows the structure chart for a very simple problem: boil an egg. Notice that the processing flow in the chart is from left to right.

Also note that all subfunctions are grouped under a main control function.


```
                   ┌─────────┐
                   │  BOIL   │
                   │ AN EGG  │
                   └────┬────┘
         ┌──────────────┼──────────────┬──────────────┐
   ┌─────┴─────┐  ┌─────┴─────┐  ┌─────┴─────┐  ┌──────┴──────┐
   │ PLACE EGG │  │ COVER EGG │  │   TURN    │  │  BOIL EGG   │
   │ INTO POT  │  │WITH WATER │  │ HEAT ON   │  │FOR 5 MIN.   │
   └───────────┘  └───────────┘  └───────────┘  └─────────────┘
```
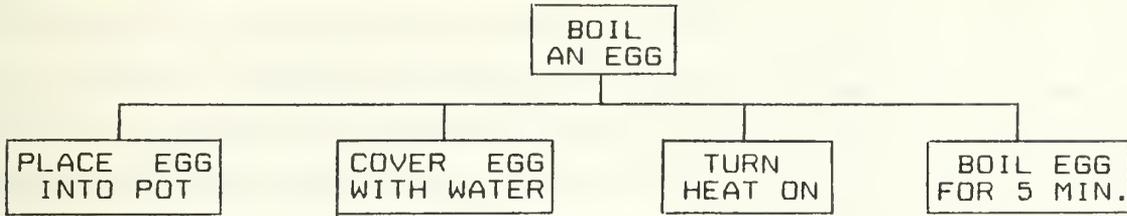
Figure 7.1 A simple Structure Chart

The first step in the process of designing the applications programs is to construct a Structure Chart. The Data Flow Diagram of the analysis stage will provide all the information needed. The processes in the DFD will become programs in the Structure Chart. During analysis each process was decomposed into subprosesses up to the point where each procces performed only one single task. Therefore, there is no need for further decomposition of the programs. However, these programs must be grouped under control programs which will determine the order of execution. A main control program will be on top of the other control programs.

In order to group the processes under common control modules automation boundaries are drawn in the DFD. As an example, suppose that certain processes in the DFD are performed daily, while others are performed weekly or monthly. A line is drawn to surround all daily processes and another line for the weekly or monthly processes, thus establishing automation boundaries in the DFD. Care must be taken not to include in the same automation boundary processes with timing conflicts. Next a structure chart is prepared for the processes in each automation boundary. Finally, by connecting all these structure charts under a main control program, a structure chart for the whole system is realized.

57

In order to make the reference to any of the modules
in the chart simpler, names and numbers are assigned. It would
be wise to use names that conform to the constraints of the
programming language to be used. Numbers assigned to each box
in an orderly fashion facilitate reference even more. A good
method for assigning numbers to modules that also shows the
hierarchy and functional dependence among the modules in a
structure chart is described below. (A full description of
this method is contained in [Ref. 23]).

a. Number the main control module by suffixing a digit with
   as many zeros as the number of subordinate levels in the
   structure chart. In the example in Figure 7.2 there are
   three such levels. Therefore, the main control module
   should be numbered "1000".

b. Assign numbers to the modules in a subordinate level by
   incrementing the left-most zero digit of the controlling
   module by one proceeding from left to right.

c. Repeat step (b) until all modules have been assigned
   numbers.
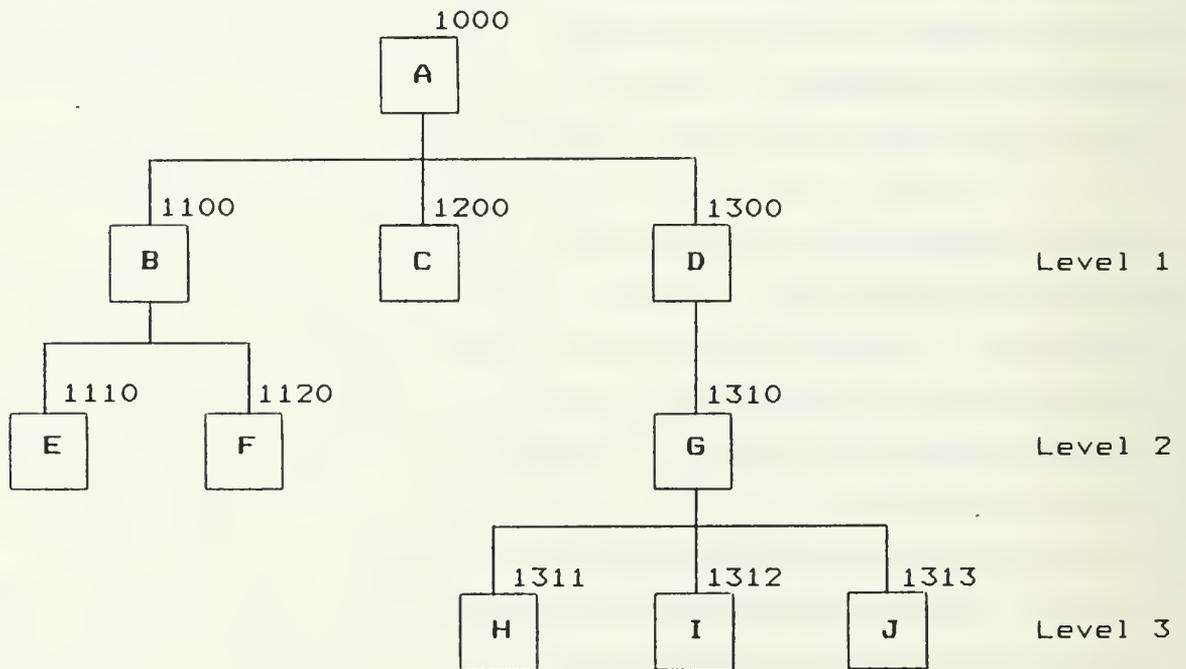
Figure 7.2   Numbering the modules in a Structure Chart

58

The advantage of this numbering scheme is that by knowing the number of a module one can tell which module invokes it, by replacing the last non-zero digit with a zero. For example module 1312 is called by module 1310, which in turn is called by module 1300 which is finally called by module 1000. Therefore it is easy to construct the structure chart by simply knowing the numbers of all the modules.

The weakness of this method is that it cannot be applied if a certain module in the system invokes more than nine other modules, an occurence that is very unusual for small and medium systems.

The second step in the design of the application programs is to create a table that contains a brief description of each module in the structure chart. This table is usually called Visual Table of Contents or VTOC and serves as a quick reference guide when someone wants to know the purpose of a program.

Finally the third step is to document the programs in the structure chart by creating an IPO (Input/Output/Process) chart for each program. As its name implies, the IPO chart of a module shows the inputs to, outputs from and the process performed by the module. The data stores and flows shown in the DFD are the sources of the inputs and outputs. The algorithm descriptions of the Functional Specification will be used to document the process. To describe the process steps on each IPO chart Structured English, pseudocode, or a flowchart may be used.

The system Structure Chart, the VTOC and the IPO charts produced during this stage provide an increasing level of detail and together they constitute a complete design of the applications programs.

B.   IMPLEMENTATION OF THE DETAILED DESIGN

   1.   Database Design

      a.   Logical Database Design

         (1) Identify the data to be stored

            The process is initiated with the Data Dictio-
nary (DD) constructed during analysis. A summary of all the
data items described in the DD is shown in Figure 7.3 (for the
moment do not consider the last three columns). Each data
element is now examined in detail.

            First synonyms are identified. One example is
the data elements UNIT NAME, UNIT, UNIT ASSIGNED TO, and
ASSIGNED UNIT, found in the data stores D5, D8, D6 and D4,
respectively, which are in fact the same data element under
different names. The term UNIT is chosen to be the common name
and a reference number is entered in the column SYNONYMS.
Similarly DATE ASSIGNED of D4 and DATE OF REPORT of D6 can be
represented as one element. The remainder of the data elements
in the DD are similarly analyzed. The result of this work is
shown in the column SYNONYMS in Figure 7.3.

            Next data that cannot be included in the data-
base are considered. For example, consider the data element
TOTAL NUMBER OF ENLISTED of data store D1. It is undesirable
to include this element in every record in D1. Unfortunately,
the relational model does not allow records of variable length.
The solution is to create a separate file or to ask the user
to enter this information when needed. The second solution is
prefered and the DELETE column for this element is marked.

            Data that must be analyzed are considered next.
For example, SOLDIER NAME is a data element consisting of
three subfields: FIRST NAME, MIDDLE INITIAL and LAST NAME.
This structure is not allowed by the relational model.

Therefore, this element is divided into three data elements and the ANALYZE column in the DD is marked.

S/N	DATA ELEMENT	TYPE	WIDTH	1	2	3	4	5	6	7	8	9	10	11	12	13	14	ANALYZE	SYNONYMS
1	ID_NUMBER	NUMERIC	7	+		+	+		+	+	+		+	+			+		
2	SOLDIER NAME	CHAR	36	+		+	+		+		+		+					+	
3	DATE ENLISTED	DATE	8	+			+							+					
4	SERVICE DURATION	NUMERIC	2	+			+												
5	CLASS	CHAR	3	+			+							+					
6	MARITAL STATUS	CHAR	7	+			+												
7	NUMBER OF CHILDREN	NUMERIC	1	+			+												
8	FINANCIAL STATUS	CHAR	1	+			+												
9	FAMILY SUPPORTER	CHAR	3	+			+												
10	NUMBER OF BROTHERS IN SERVICE	NUMERIC	1	+			+												
11	SPECIAL REASONS FOR TRANSFER	LOGICAL	1	+			+												
12	PREFERED UNITS	CHAR	21	+			+											+	
13	ADDRESS	CHAR	69	+			+							+				+	
14	TOTAL NUMBER OF ENLISTED	NUMERIC	4	+														+	
15	SPECIALTY	CHAR	8		+	+	+	+	+		+	+	+	+	+	+			
16	REQUIRED SOLDIERS FOR SPEC TRAINING	NUMERIC	4		+														
17	ASSIGNED UNIT	CHAR	7				+												37
18	DATE ASSIGNED	DATE	8				+												
19	COMPLETED TRAINING	LOGICAL	1				+												
20	DATE WHEN REMAINING SERVICE < 4 MON.	DATE	8				+												
21	UNIT NAME	CHAR	7					+							+	+			37
22	REQUIRED NUMBER OF SOLDIERS	NUMERIC	3						+										
23	EXISTING NUMBER OF SOLDIERS	NUMERIC	3						+										
24	COMPLEMENT NUMBER OF SOLDIERS	NUMERIC	3						+										
25	UNIT ASSIGNED TO	CHAR	7							+									37
26	DATE OF REPORT	DATE	8							+									18
27	CHANGED NAME	CHAR	36								+								2
28	CHANGED SERVICE DURATION	NUMERIC	2								+								4
29	CHANGED MARITAL STATUS	CHAR	7								+								6
30	CHANGED NUMBER OF CHILDREN	NUMERIC	1								+								7
31	CHANGED FINANCIAL STATUS	CHAR	1								+								8
32	CHANGED FAMILY SUPPORTER	CHAR	3								+								9
33	CHANGED NUMBER BROTHERS IN SERVICE	NUMERIC	1								+								10
34	CHANGED SPECIAL REASONS FOR TRANSFER	LOGICAL	1								+								11
35	CHANGED PREFERED UNITS	CHAR	21								+								12
36	CHANGED ADDRESS	CHAR	69								+								13
37	UNIT	CHAR	7									+							
38	DATE RETIRED	DATE	8									+	+						
39	QUALIFICATION POINTS	NUMERIC	3											+					
40	NEEDS	NUMERIC	3												+				

Figure 7.3   The data items of the Data Dictionary

61

Finally, the type and width of each data element is considered. Why should FAMILY SUPPORTER be a three-byte character field instead of an one-byte logical field? Or the MARITAL STATUS field can be changed to an one-byte character field which will store M for Married, D for Divorced, S for Single and W for Widowed provided that a short explanation is given to the user on the screen when he runs the program. Also, there is no need for ID_NUMBER to be a numeric field since there are no arithmetic computations performed on it. A character field occupies less memory space and can be manipulated much faster than a numeric field. A summary of the Data Dictionary is shown in Figure 7.4.

(2) Specify the logical database records

Each data store in the DFD will normally be transformed into a database file. The data elements of each file are shown in the Data Dictionary in Figure 7.4. What files should be combined or separated? To determine this the chart in Figure 7.5 will be utilized. This chart shows which data items of each data store are used by each process. For example process P2.1 uses the ID_NUMBER of both data store D1 and D4. With the help of this chart consider data store D4 which is the largest data store in the system. D4 also contains more records than any other data store. This means that significant memory space will be occupied by D4 and therefore, the process of loading it will take considerable time. On the other hand observe that some of the processes make use of only a small part of this data store. For example, process P4.3 only uses the names of the three units of preference. After carefully looking at all processes it is decided that data store D4 should be divided into four data stores as shown in Figure 7.6.

S/N	DATA ELEMENT	TYPE	WIDTH	|	DATA STORE 1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	ID_NUMBER	CHAR	7	|	+		+	+		+	+	+		+	+			+
2	FIRST NAME	CHAR	15	|	+		+	+		+	+	+		+				
3	LAST NAME	CHAR	20	|	+		+	+		+	+	+		+				
4	MIDDLE INITIAL	CHAR	1	|	+		+	+		+	+	+		+				
5	DATE ENLISTED	DATE	8	|	+			+							+			
6	SERVICE DURATION	NUMERIC	2	|	+			+			+							
7	CLASS	CHAR	3	|	+			+							+			
8	MARITAL STATUS	CHAR	1	|	+			+			+							
9	NUMBER OF CHILDREN	NUMERIC	1	|	+			+			+							
10	FINANCIAL STATUS	CHAR	1	|	+			+			+							
11	FAMILY SUPPORTER	LOGICAL	1	|	+			+			+							
12	NUMBER OF BROTHERS IN SERVICE	NUMERIC	1	|	+			+			+							
13	SPECIAL REASONS FOR TRANSFER	LOGICAL	1	|	+			+			+							
14	PREFERED UNIT 1	CHAR	7	|	+			+			+							
15	PREFEBED UNIT 2	CHAR	7	|	+			+			+							
16	PREFEBED UNIT 3	CHAR	7	|	+			+			+							
17	STREET	CHAR	15	|	+			+			+		+					
18	CITY	CHAR	20	|	+			+			+		+					
19	STATE	CHAR	15	|	+			+			+		+					
20	ZIP	CHAR	5	|	+			+			+		+					
21	PHONE	CHAR	14	|	+			+			+		+					
22	SPECIALTY	CHAR	8	|		+	+	+	+	+		+	+	+	+	+		+
23	REQUIRED SOLDIERS FOR SPECIALTY TRAINING	NUMERIC	4	|		+												
24	UNIT	CHAR	7	|				+	+	+		+				+	+	
25	DATE ASSIGNED	DATE	8	|				+		+								
26	COMPLETED TRAINING	LOGICAL	1	|				+										
27	DATE WHEN REMAINING SERVICE < 4 MON.	DATE	8	|				+										
28	REQUIRED NUMBER OF SOLDIERS	NUMERIC	3	|					+									
29	EXISTING NUMBER OF SOLDIERS	NUMERIC	3	|					+									
30	COMPLEMENT NUMBER OF SOLDIERS	NUMERIC	3	|					+									
31	DATE RETIRED	DATE	8	|										+	+			
32	QUALIFICATION POINTS	NUMERIC	3	|												+		
33	NEEDS	NUMERIC	3	|												+		

Figure 7.4   A summary   of the Data Dictionary

Another use   of the chart in Figure 7.5   is to help determine   which fields are redundant.   For example,   the FIRST NAME, LAST NAME and MIDDLE INITIAL fields can be removed from data stores   D3, D6   and D8   since no process uses   these fields.

S/N	DATA ELEMENT	DATA STORES IN WHICH THE ELEMENT EXISTS	P	P		R	R	O		C	C	R	R	S	S	S	R	R	S		S
			P1	P2.1	P2.2	P2.2.1	P2.2.2	P2.2.3	P2.3	P3.1	P3.2	P4.1	P4.2	P4.3	P5.1	P5.2	P5.3	P5.4	P5.5	P6.1	P6.2
1	ID_NUMBER	1,3,4,6,7,8,10 11,14		1,4	3,4	6,4		7,4	4,8 10	.		1,3,4		4,6 11	8	3	7	1	14		6
2	FIRST NAME	1,3,4,6,7,8,10		1,4				7,4	4,10								7	1			
3	LAST NAME	1,3,4,6,7,8,10		1,4				7,4	4,10								7	1			
4	MIDDLE INITIAL	1,3,4,6,7,8,10		1,4				7,4	4,10									1			
5	DATE ENLISTED	1,4,10		1,4					4,10									1			
6	SERVICE DURATION	1,4,7		1,4				7,4	4,10								7	1			
7	CLASS	1,4,10		1,4														1			
8	MARITAL STATUS	1,4,7		1,4				7,4	4,10								7	1			
9	NUMBER OF CHILDREN	1,4,7		1,4				7,4	4,10								7	1			
10	FINANCIAL STATUS	1,4,7		1,4				7,4	4,10								7	1			
11	FAMILY SUPPORTER	1,4,7		1,4				7,4	4,10								7	1			
12	NUMBER BROTHERS IN SERVICE	1,4,7		1,4				7,4	4,10								7	1			
13	SPECIAL REASON FOR TRANSFER	1,4,7		1,4				7,4	4,10								7	1			
14	PREFERRED UNIT 1	1,4,7		1,4				7,4	4,10			4		4			7	1			
15	PREFERRED UNIT 2	1,4,7		1,4				7,4	4,10			4		4			7	1			
16	PREFERRED UNIT 3	1,4,7		1,4				7,4	4,10			4		4			7	1			
17	STREET	1,4,7,10		1,4				7,4	4,10								7	1			
18	CITY	1,4,7,10		1,4				7,4	4,10								7	1			
19	STATE	1,4,7,10		1,4				7,4	4,10								7	1			
20	ZIP	1,4,7,10		1,4				7,4	4,10								7	1			
21	PHONE	1,4,7,10	2,4 5,9 2	1,4	3,4			7,4	4,10	8,5	5,6	3,11	3,5,9 12	6,11 12	8	3	7	1	14	2	6
22	SPECIALTY	2,3,4,5,6,8,9 10,11,12,14								8.5	5,6		5,12,13	6,12 6						2	6 6
23	REQUIRED SOLDIERS SPEC TRN.	2				6,4															
24	UNIT	4,5,6,8,12,13				6,4				8.5	5,6										
25	DATE ASSIGNED	4,6	4			4															
26	COMPLETED TRAINING	4	4																		
27	DATE REMAIN SERV. <4 MONTHS	4																			
28	REQUIRED NUMBER OF SOLDIERS	5							8,10	5	5										
29	EXISTING NUMBER OF SOLDIERS	5								5	5		5								
30	COMPLEMENT NUMB OF SOLDIERS	5	5												8						
31	DATE RETIRED	8,10													8						
32	QUALIFICATION POINTS	11										11									
33	NEEDS	12											12	12							

Figure 7.5 The relationship between data elements and processes.

64

FIELD NAME	Original Data Store D4	NEW DATA STORES			
		D4.1	D4.2	D4.3	D4.4
ID_NUMBER	+	+	+	+	+
FIRST NAME	+	+			
LAST NAME	+	+			
MIDDLE INITIAL	+	+			
DATE ENLISTED	+		+		
SERVICE DURATION	+		+		
CLASS	+		+		
MARITAL STATUS	+			+	
NUMBER OF CHILDREN	+			+	
FINANCIAL STATUS	+			+	
FAMILY SUPPORTER	+			+	
NUMBER OF BROTHERS IN SERVICE	+			+	
SPECIAL REASONS FOR TRANSFER	+			+	
PREFERED UNIT 1	+				+
PREFERED UNIT 2	+				+
PREFERED UNIT 3	+				+
STREET	+	+			
CITY	+	+			
STATE	+	+			
ZIP	+	+			
PHONE	+	+			
SPECIALTY	+		+		
UNIT	+		+		
DATE ASSIGNED	+		+		
COMPLETED TRAINING	+		+		
DATE WHEN REMAINING SERVICE < 4 MON	+		+		

Figure 7.6   The analysis of Data Store D4

Next, standard names are given to the files and the data elements. Although this step in the design is quite independent of any particular DBMS, it will save time if while naming the files and data elements, consideration is given to the selected DBMS, dBASE III Plus, which allows eight characters for record names and ten characters for field names. The data stores and the corresponding database files are shown in Figure 7.7.

Next the key field for each record must be determined. Most of the records in the data base contain an ID_NUMBER field, which is an excellent identifier of the record. In the few records that do not have an ID_NUMBER, the

S/N	DATA STORE	FILE NAME
1	D1	ENLISTED
2	D2	TRAIN_RQ
3	D3	COMPL_TR
4	D4.1	SLD_ADDR
5	D4.2	SLD_SERV
6	D4.3	SLD_TRAN
7	D4.4	SLD_PREF
8	D5	UNIT_ORG
9	D6	ASSIGNMS
10	D7	CHANGES
11	D8	RETIRED
12	D9	SPECS
13	D10	HISTORY
14	D11	TRSF_PTS
15	D12	UNIT_REQ
16	D13	UNITS
17	D14	TRAINEES

Figure 7.7   Naming the database files

UNIT, SPECIALTY or a combination of these two fields was selected as a key. Figure 7.8 shows the final structure of the database. A circled cross denotes that this field is a key.

Now that the logical structure of the database has been defined the memory space that each field and each record will occupy can be determined. The maximum number of records that each database file may contain is also known. Therefore, the memory space needed for each file and consequently for the whole database can be calculated. This calculation is shown in Figure 7.9.

b. Physical Database Design

The logical database sructure defined during the previous step will now be transfered into a physical structure. This means that the actual database will be created and its structure stored on a computer storage device, such as a disk, using the DBMS software package that has been selected, namely dBASE III Plus.

#	FIELD NAME	TYPE	WIDTH	ENLISTED	TRAIN-RQ	COMPL-TR	SLD-ADDR	SLD-SERV	SLD-TRAN	SLD-PREF	UNIT-ORG	ASSIGNMS	CHANGES	RETIRED	SPECS	HISTORY	TRSF-PTS	UNIT-REQ	UNITS	TRAINEES
1	ID_NUMBER	CHAR	7	⊕		⊕	⊕	⊕	⊕	⊕		⊕	⊕	⊕		⊕	⊕			⊕
2	F_NAME	CHAR	15	+			+						+			+				
3	L_NAME	CHAR	20	+			+						+			+				
4	M_INITIAL	CHAR	1	+			+						+			+				
5	DATE_ENL	DATE	8	+				+								+				
6	SERV_DUR	NUMERIC	2	+				+						+						
7	CLASS	CHAR	3	+				+								+				
8	MARIT_STAT	CHAR	1	+					+				+							
9	NUM_CHILD	NUMERIC	1	+					+				+							
10	FINAN_STAT	CHAR	1	+					+				+							
11	FAM_SUPP	LOGICAL	1	+					+				+							
12	BROTH_SERV	NUMERIC	1	+					+				+							
13	SPEC_REAS	LOGICAL	1	+					+				+							
14	PRF_UNIT1	CHAR	7	+						+			+							
15	PRF_UNIT2	CHAR	7	+						+			+							
16	PRF_UNIT3	CHAR	7	+						+			+							
17	STREET	CHAR	15	+			+						+			+				
18	CITY	CHAR	20	+			+						+			+				
19	STATE	CHAR	15	+			+						+			+				
20	ZIP	CHAR	5	+			+						+			+				
21	PHONE	CHAR	14	+			+						+			+				
22	SPECIALTY	CHAR	8		⊕	+		+			⊕	+				⊕	+	+	⊕	+
23	SPEC_REQ	NUMERIC	4		+															
24	UNIT	CHAR	7						+		⊕	+						⊕	⊕	
25	DATE_ASIGN	DATE	8						+			+								
26	END_TRAIN	LOGICAL	1						+											
27	DATE_4	DATE	8						+											
28	REQUIRED	NUMERIC	3								+									
29	EXISTING	NUMERIC	3								+									
30	COMPLEMENT	NUMERIC	3								+									
31	DATE_RET	DATE	8											+		+				
32	QUAL_PTS	NUMERIC	3														+			
33	NEEDS	NUMERIC	3															+		

Figure 7.8. The final structure of the database.

Creating a database using dBASE III Plus is very easy and is done by using the CREATE command. The database file ASSIGNMS is created as an example:

- First bring up dBASE III by typing:

  C> DBASE

- When the period prompt is received enter the command CREATE followed by the file name:

  . CREATE ASSIGNMS

67

- dBASE III Plus will then ask for the field name, type and
  width and if it is a numeric field, the width of the
  decimal part. Since this information has been defined in
  the logical design, it can easily be entered.

FILE NAME	RECORD SIZE (Bytes)	MAXIMUM NUMBER OF RECORDS	MAXIMUM FILE SIZE (Bytes)	N O T E S
ENLISTED	153 (a)	1,500	229,500	(a) One byte is used
TRAIN_RQ	13	10	130	as a flag by dBASE
COMPL_TR	16	1,100 (b)	17,600	III Plus
SLD_ADDR	113	16,500 (c)	1,864,500	(b) 400 soldiers comp-
SLD_SERV	53	16,500	874,500	lete training in
SLD_TRAN	14	16,500	231,000	1 month
SLD_PREF	29	16,500	478,000	(c) 1500 * 11 classes
UNIT_ORG	25	300 (d)	7,500	(d) 10 spec * 30 units
ASSIGNMS	31	1,100	34,100	(e) no more than 2% of
CHANGES	142	330 (e)	46,860	all soldiers
RETIRED	16	1,500	24,000	(f) Removed from data-
SPECS	9	10	90	base once every
HISTORY	140	9,000 (f)	1,260,000	year.
TRSF_PTS	19	1,100	20,900	
UNIT_REQ	19	300	5,700	
UNITS	8	30	240	
TRAINEES	16	10	160	
MAXIMUM DATABASE SIZE			5,095,180	

Figure 7.9 Memory requirements of the database

The file ASSIGNMS as entered in the computer is
shown in Figure 7.10

	Field Name	Type	Width	Dec
1.	ID_NUMBER	character	7	
2.	SPECIALTY	character	3	
3.	UNIT	character	7	
4.	DATE_ASIGN	date	8	

Figure 7.10   File ASSIGNMS

Working in the same way the remaining files in the
database are created.

68

2. Design the Application Programs

The design of the programs that will be used by the DBMS to process the data in the database is the next step. According to the method previously described, the DFD of analysis must be transformed into a structure chart.

The first step is to identify automation boundaries for the processes in the DFD. For example, consider process P2.1 which adds new records in data store D4 for the new enlisted soldiers. What other processes could be inside the same automation boundary with P2.1? Process P2.1 uses as input data store D1 (New Enlisted Soldiers) which is being updated by process P5.4 (Get Enlisted Soldiers data). Therefore, P5.4 and P2.1 can be included in the same boundary but it must be assumed that P5.4 will be executed first. Next the other processes are considered, and after having examined each one of them, the grouping of all processes in the DFD into automation boundaries is produced as shown in Figure 7.11.

T I M E	P R O C E S S
1st of each month	P5.1, P3.1, P2.3, P5.3, P2.2.3
2nd of each month	P4.1, P4.2, P4.3, P3.2, P2.2.2, P6.2
3rd of each odd month	P5.4, P2.1
10th of each odd month	P1, P6.1
24th of each month	P5.2, P5.5, P2.2.1

Figure 7.11 Automation boundaries

For each one of these boundaries a structure chart is constructed. On the top of each chart a control module is inserted, which controls the flow of execution among the processes. Inside the boxes of the structure charts a very brief imperative statement is provided which explains the

69

purpose of the module. The result of this work is shown in Figures 7.12 through 7.16.
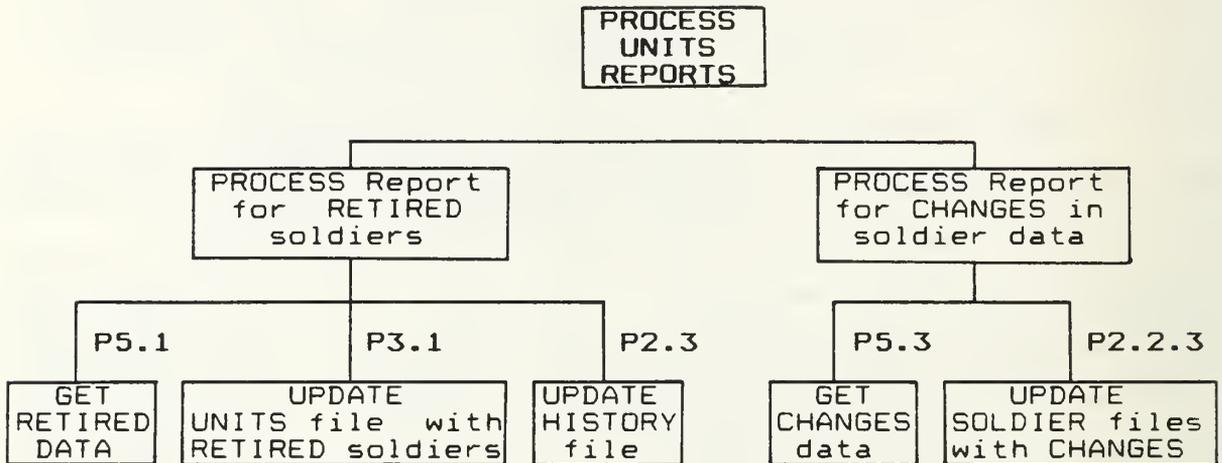
Figure 7.12   Structure chart for the processes:
              P5.1, P3.1, P2.3, P5.3 AND P2.2.3
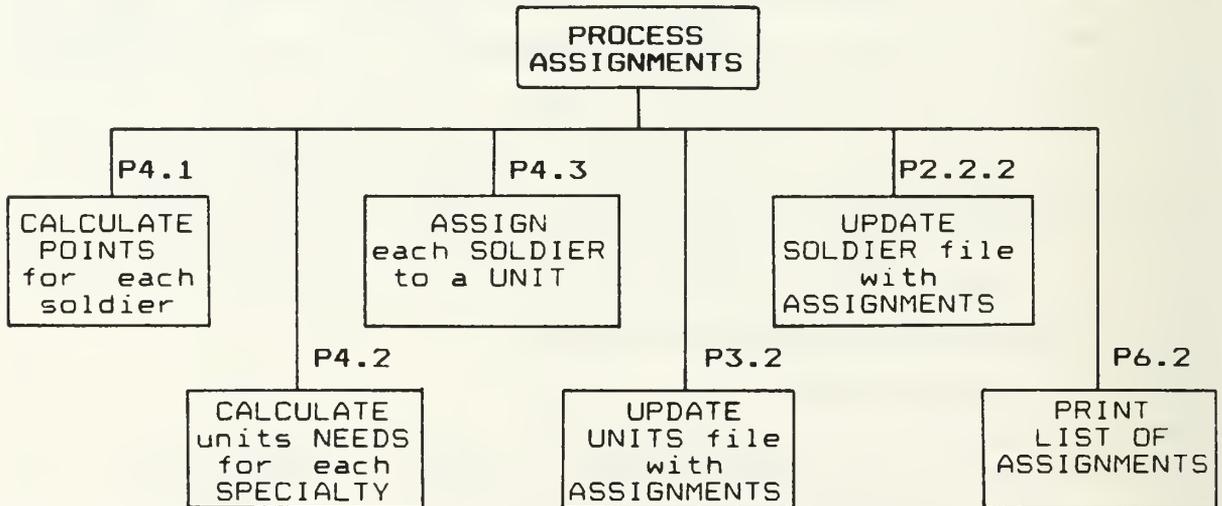
Figure 7.13   Structure chart for processes:
              P4.1, P4.2, P4.3, P3.2, P2.2.2 and P6.2

The system's structure chart is almost complete. The only thing remaining is to connect all structure charts under a main control module and number and name the modules in the chart. Figure D.1 in Appendix D shows the completed structure chart.

70
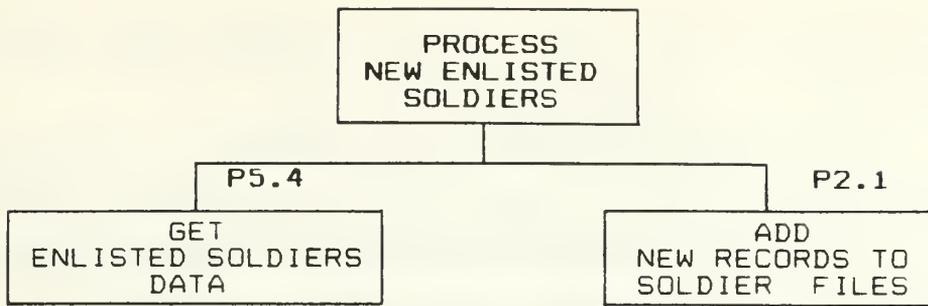
Figure 7.14 Structure chart for processes P5.4 and P2.1
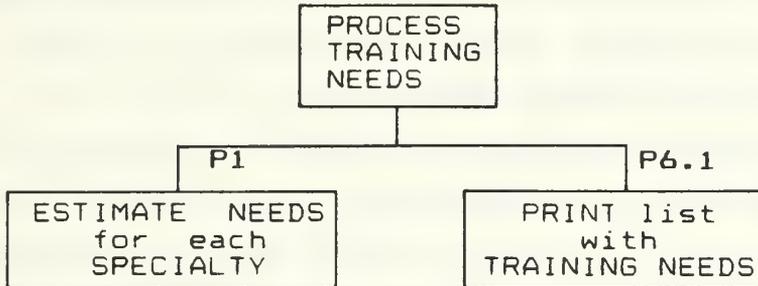
Figure 7.15 Structure Chart for processes P1 and P6.1
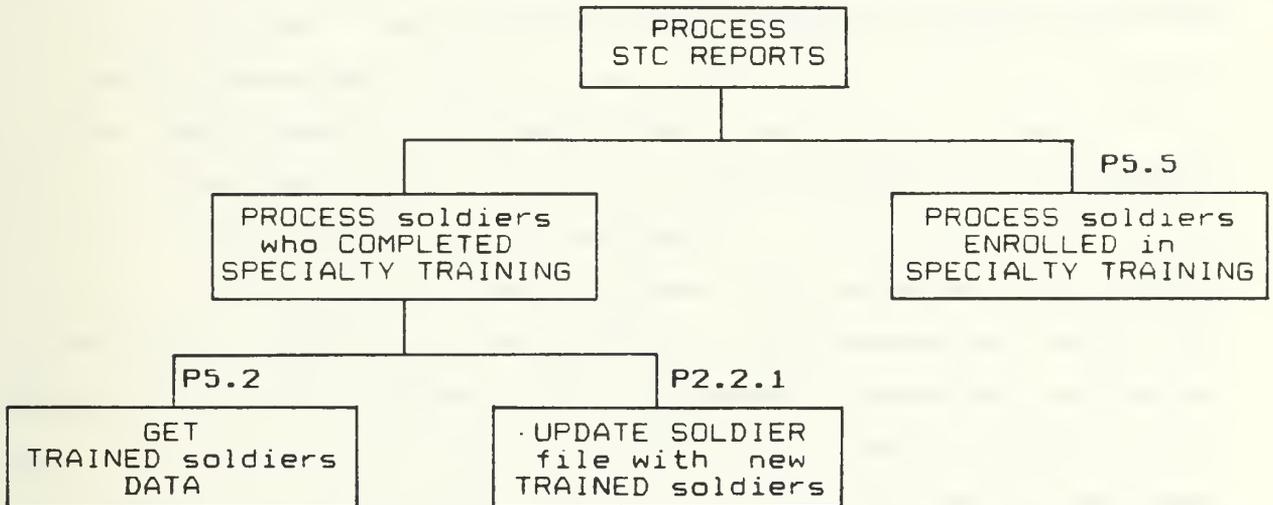
Figure 7.16 Stucture chart for processes P5.2, P2.2.1 and P5.5

The next step is to prepare a table that contains a brief description of the function of each module in the structure chart. This table gives some additional information and helps in clarifying some of the imperative statements in the boxes of the structure chart. The second ingredient of the

71

program design, the VTOC, is now complete (see Figure D.2 in Appendix D).

Finally the IPO charts are prepared, one for each box in the structure chart. These charts provide detailed information on the inputs that each module requires, the process performed by the module and the outputs produced.

When constructing an IPO chart, most designers prefer to use Structured English to describe the process performed by a module. Others prefer to use pseudocode, or logic flowcharts or a combination of these techniques.

The logic flowchart provides an excellent way for describing the steps of a process. It uses very few, simple symbols and is easy to follow. Many people, however, consider flowcharts as a bad choice. The reason is that flowcharts have been misused for many years. Programmers in the past were usually required to document their programs using flowcharts. What they really were doing was writing the program first and then drawing a flowchart that echoed the program code. Another reason why flowcharts are not very popular is that they are difficult to construct if the program is very complex. For the DBMS considered in this thesis, the processes of the system have been decomposed to the level where each module performs only one function. Therefore, logic flowcharts are prefered for the IPO charts to show the process performed by each program. The IPO charts for all twenty seven modules of the system structure chart are shown in Figures D.3.1 through D.3.27 of Appendix D.

The Structure chart together with the VTOC and the IPO charts form the design specifications of the applications programs. This design will be translated into source code during the Implementation phase.

VIII. <u>IMPLEMENTATION</u>

A. THEORY

1. <u>The Purpose of the Implementation Phase</u>

This phase is probably the largest and most difficult one. It may fail if the phases preceding it, especially the Analysis and Design phases, have not been adequately document- ed. If, however, these two phases have been successfully completed, the implementation phase will be straightforward.

The purpose of the Implementation phase is primarily to deliver a system ready for execution. The two major tasks accomplished during this phase are:

- To take the design specifications that resulted from the Detailed Design phase and translate them into source code.
- To verify that this source code implements correctly the design specifications.

2. <u>The steps of Implementation</u>

The steps that must be followed during this stage are:

a. Construct a test database

During the design phase the database is created by defining, compiling and storing its structure using the DBMS previously selected. This database, however, contains no real information. The purpose of constructing a test database is twofold: to test the accuracy of the database definitions and to facilitate the testing of the application programs.

b. Code the application programs

During this step the detailed design specificati- ons are transformed into source code. The first consideration is the order of program development. In other words it must be

73

decided which programs to develop first. There are two appro-
aches used by the majority of the programmers: the top-down
and the bottom-up approach.

During top-down program development the programmer
starts with the main application program and works down
through the system structure chart, leaving for last the
programs at the bottom of the chart. To test a finished
program at a higher level, the programmer creates dummy subpro-
grams that simulate the lower level programs called by the
higher level program. These programs are called stubs.

During bottom-up development the programmer starts
with the programs at the lowest level which do not depend on
any other program in the system. When all the independent
programs have been developed and tested, the programmer moves
to the programs that call them. Using this approach no dummy
subprograms are necessary.

    c. Test the system

To test the coded programs the test database is
used. First each program is tested independently, refered to
as a unit test. Finally the system programs as a whole are
tested which is called integrated test. Both unit and integra-
ted testing are difficult tasks. Each program should be tested
for its handling of abnormal situations and data entry errors,
attempting to use the system as the users will. Typical use
cycles should be exercised to make sure that the programs work
together as they were designed. Next, check the data files to
see if the application adds, updates and deletes data properly.
No matter how much time is spent on testing, it cannot be
overdone. Therefore it must be decided when sufficient testing
has been done to ensure that the system is not going to crash
after it is up and running.

d. Document the system

The user's opinion of a system is greatly influen-
ced by the quality of the documentation he is given and the
ease with which he can use this documentation. Documentation
helps the user to maintain the system. Users must be able to
read and understand the documentation in order to correct or
modify an application program. Well documented programs are
easier to maintain but incorrect and/or out of date documen-
tation is worse than none at all.

Well designed and carefully formatted code is the
start of a properly documented system. Document the programs
considering the people who will have to maintain the applica-
tion. Maintenance personnel do not trust documentation that is
not embedded in the code or otherwise "on line". Program
documentation starts with the program-header and includes
comment lines and line-by-line comments adjacent to the
program commands and statements.

The program header provides the name of the
program, a description of what it does, the date of the last
update and optionally the significance of each program para-
meter.

Comment lines are used to describe the function
of a group of statements. Usually a well documented program
includes a comment line for each box in the program flowchart.
Line-by-line comments document exactly what each line of
program code is doing.

To complete the documentation, in addition to
"on line" documentation:

(1) Document procedures for the user on how to:

. Use the new system

. React in case that the system fails

75

(2) Document procedures for the operations personnel on how to:

. Act on a system failure

. Back-up the data base

e. Train users and operations personnel

Although the documentation given to the user and the operations personnel provides information on how to install, operate and check out the system, some training of personnel is required. Training will help them to understand the documentation, answer the questions, and clarify misunderstandings.

f. Test the new system in parallel with the old one

If possible, the new system should be run in parallel with the old one. In this way the transition becomes smoother and a comparison of the results of the two systems can be made.

B. IMPLEMENTATION

1. <u>Constructing a test data base</u>

The data base consists of 17 files which were created during the Detailed Design phase. The next step is to enter data in these files to facilitate the testing of the application programs. Using dBASE III Plus is very easy to add records to a data base file and fill them with information using the following commands:

- USE <file name>
- APPEND

A single blank record will be displayed on the screen and the user can fill in the empty fields. After the last field has been entered the user enters a <Pg Dn> and a new blank record is displayed. When finished the user presses

<Esc> to terminate the addition process. All data base files do not require initial loading. Some of these files are loaded by the system. For example, test data is not required for the ASSIGNMS file or the RETIRED file. This is done by the ASSIGNMT and GET-RET programs, respectively. The files in which test data is entered will be:
SLD-ADDR, SLD-SERV, SLD-TRAN, SLD-PREF, UNIT-ORG, SPECS, HISTORY and UNITS. Manual lists must also be prepared for NEW ENLISTED soldiers, soldiers who COMPLETED SPECIALTY TRAINING, CHANGES in soldiers status and soldiers currently in TRAINING.

2. Translating the design into dBASE III Plus code

The IPO charts and the logic flowcharts of the Detailed Design contain enough information to fully capture the program logic. Therefore the effort to translate each step in the flowcharts into one or more dBASE III commands is a straightforward process.

As mentioned before there are two different approaches to program development: bottom-up and top-down. It is the author's opinion that the bottom-up approach is easier to follow since the program can be tested immediately after writing it, instead of having to create "stubs" as in the top-down approach.

The listings for the programs PERS-MGT, ENL-SLDS, GET-ENL and ADD-SLD are shown in Appendix E (Sections E.1 through E.4).

3. Testing, debugging and documenting the system

Using the test data base constructed, the application programs are tested and debugged individually. Finally the system as a whole is tested.

77

The listing of program PERS-MGT (Section E.1 in Appendix E) provides an example of "on-line" documentation. Each programmer can use his own skills to create good and readable documentation.

IX. <u>CONCLUSION</u>

The steady decline in computer hardware costs not only makes it possible to add more applications on computers but it also distributes computing power to more and more new users. As a result the demand for software, that tells the computer exactly what steps to perform to convert its raw power into useful operations, is increasing steadily.

Therefore, improved techniques in software development become the key issue if further expansion of the use of computers is to be achieved. Software engineering is rapidly emerging as a discipline for managing the development of software systems, but like every new engineering discipline has not yet achieved widespread acceptance.

Due to the existing shortage in software engineers, a large number of people are building software systems who have limited or no knowledge of the software engineering principles. In fact most of them have obtained only a technical knowledge of one or two programming languages and one or two computer systems.

This thesis is intended especially for these people. Fundamental software engineering concepts were first discussed and then applied to a real software product which was featured throughout this thesis.

Although panacean tools and techniques for the software engineer do not exist, the value of software engineering principles remains great. Until an adequate number of software engineers using these principles has been developed, the "software crisis" will continue to be the major restraint on the progress of computer technology.

THE SYSTEM DATA FLOW DIAGRAMS
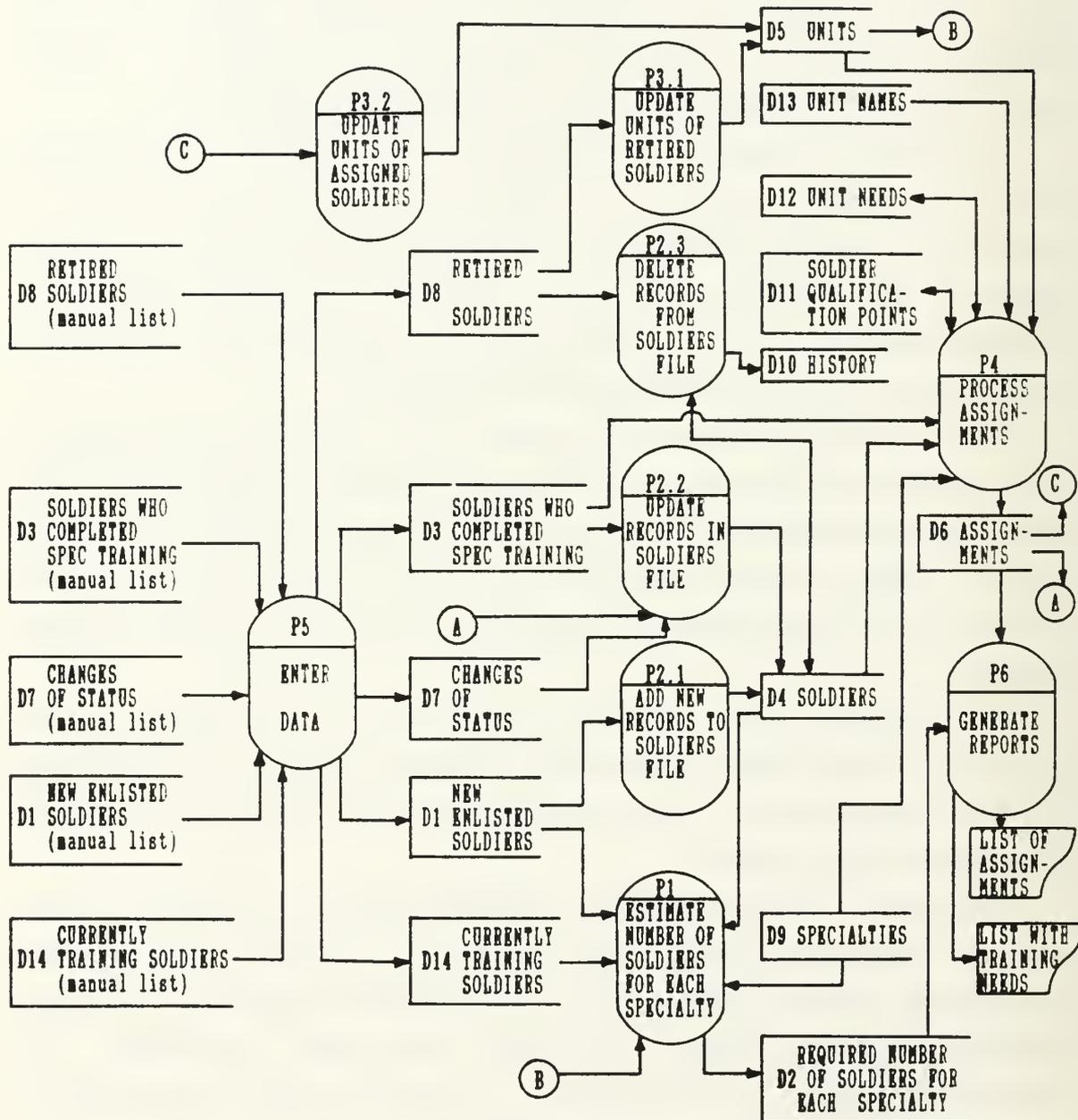
Figure A.1   The System Data Flow Diagram
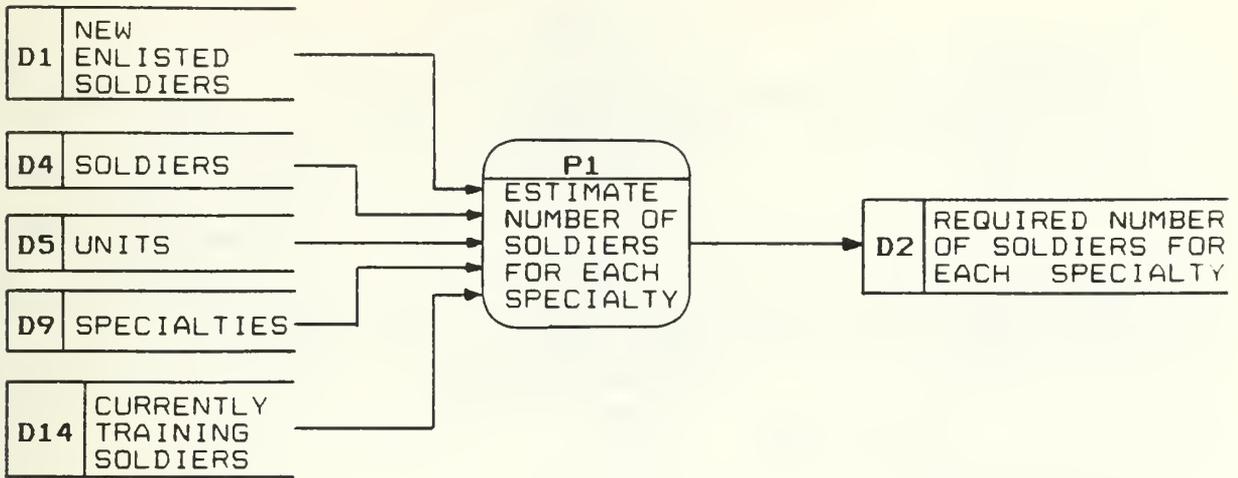
Figure A.2   Process P1
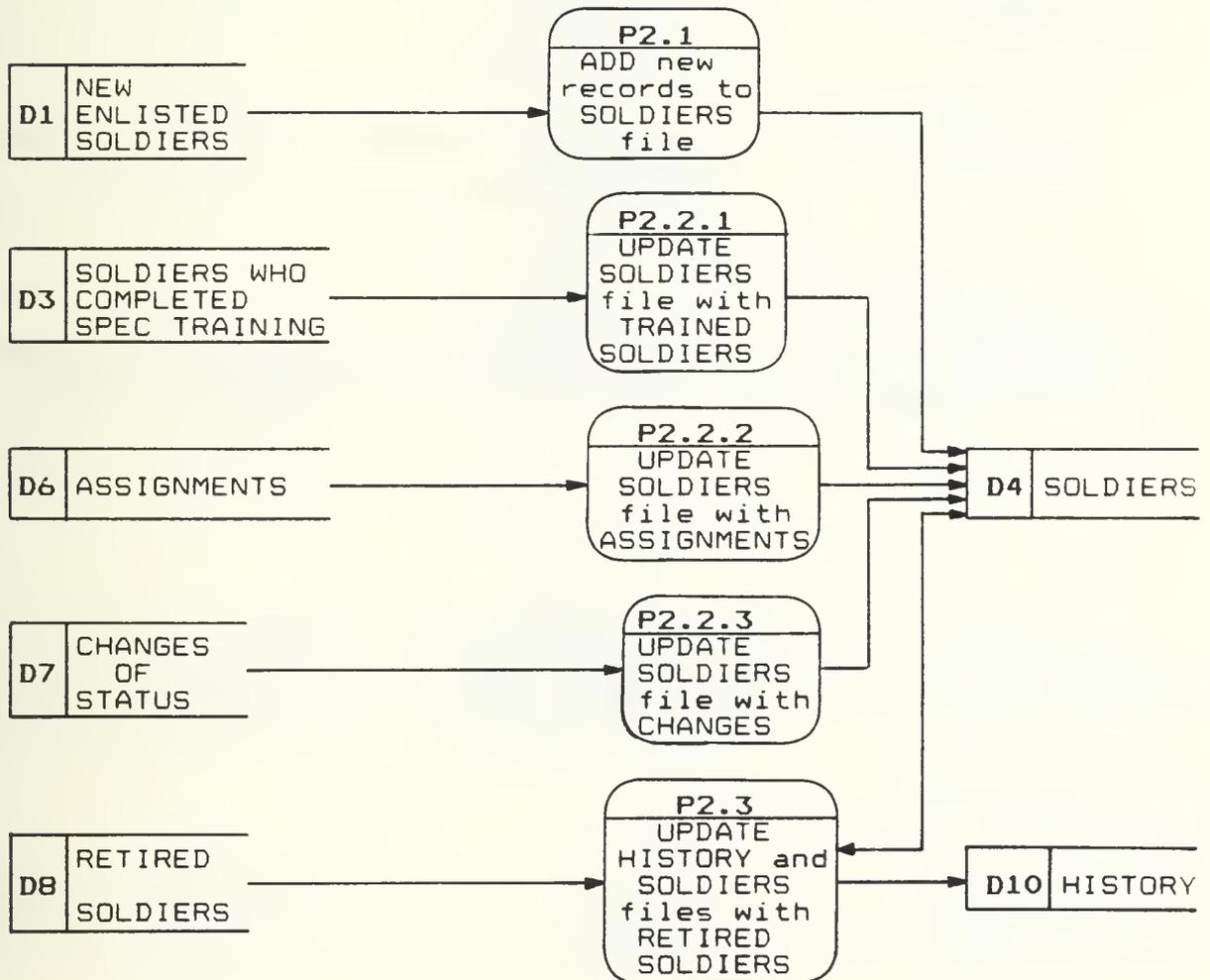
Figure A.3   Process P2
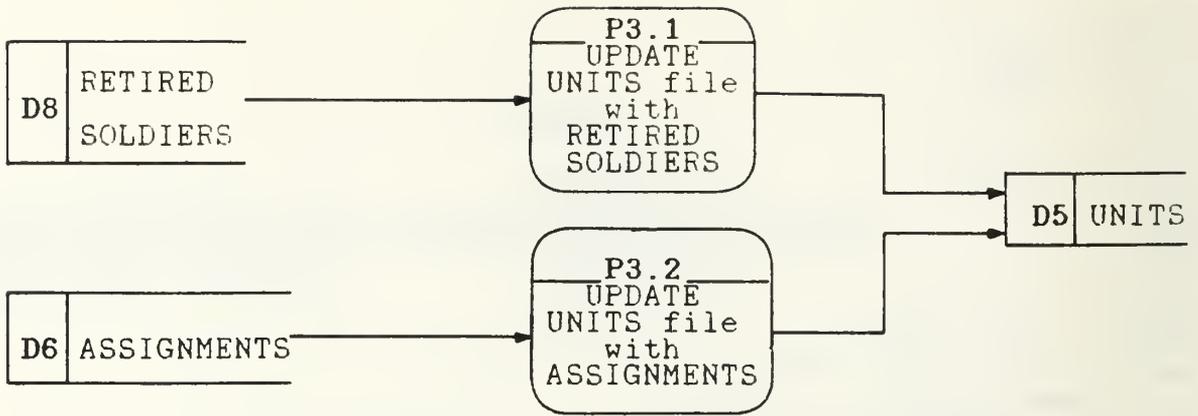
Figure A.4   Process P3
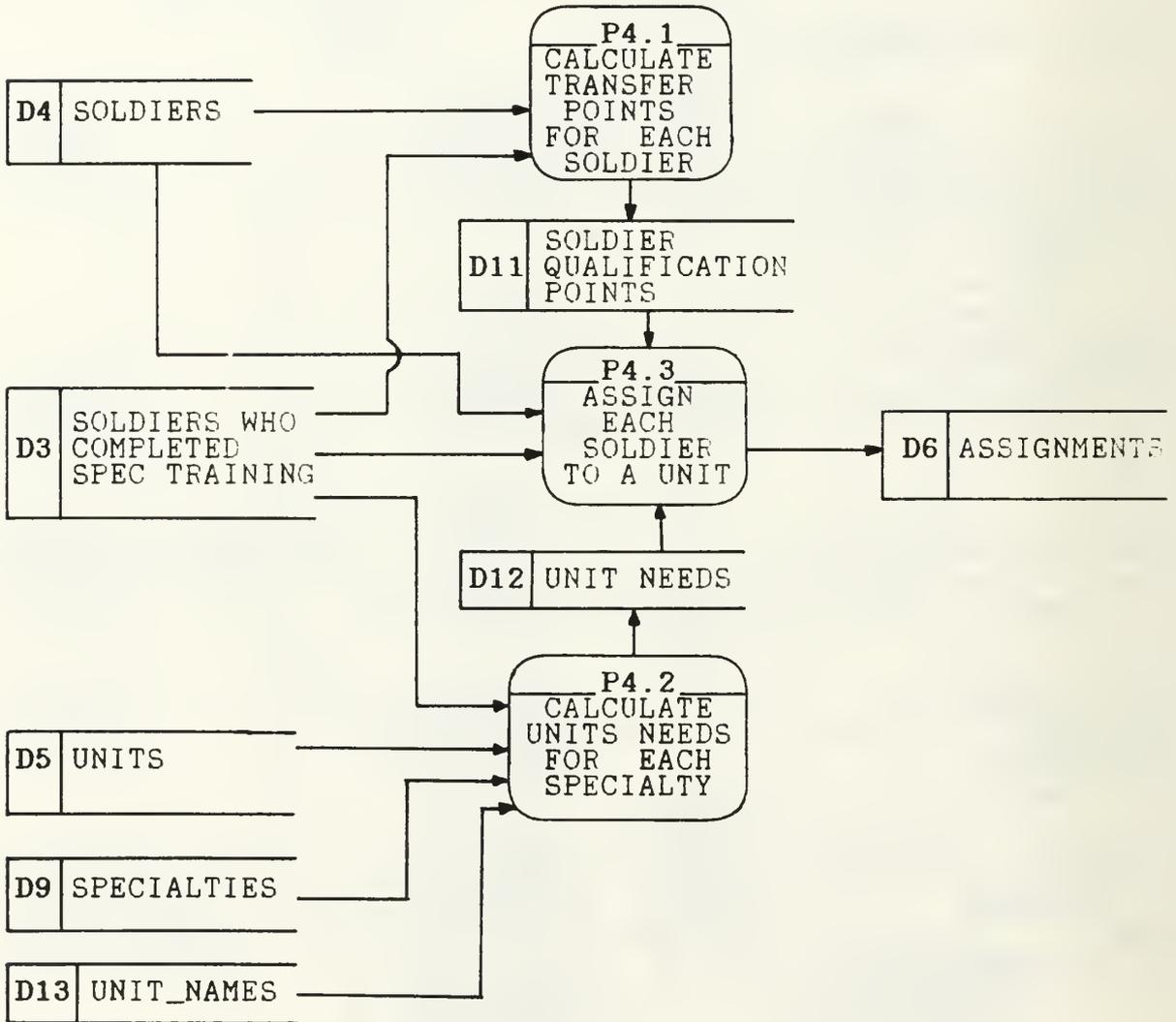
Figure A.5   Process P4

82
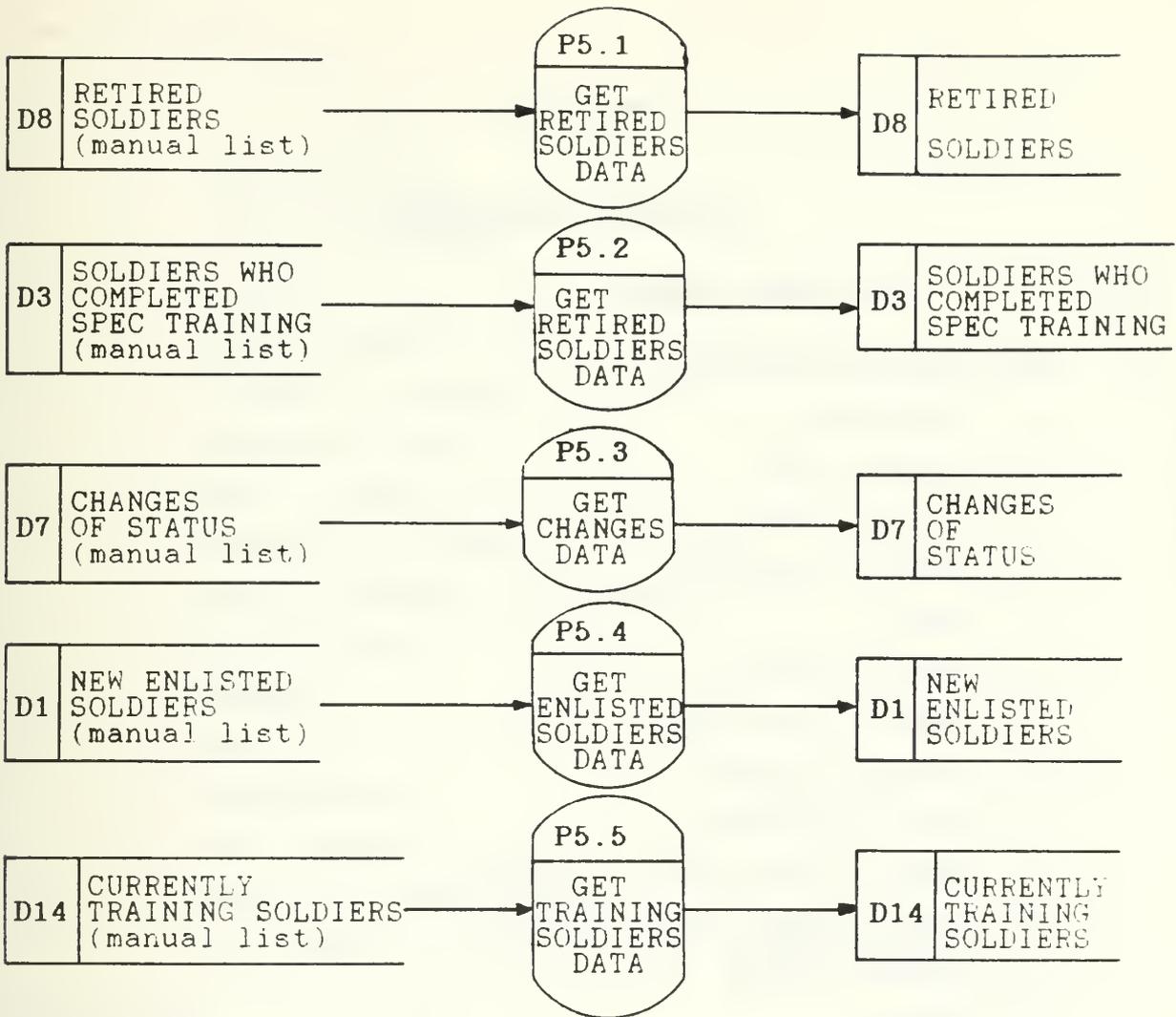
Figure A.6   Process P5
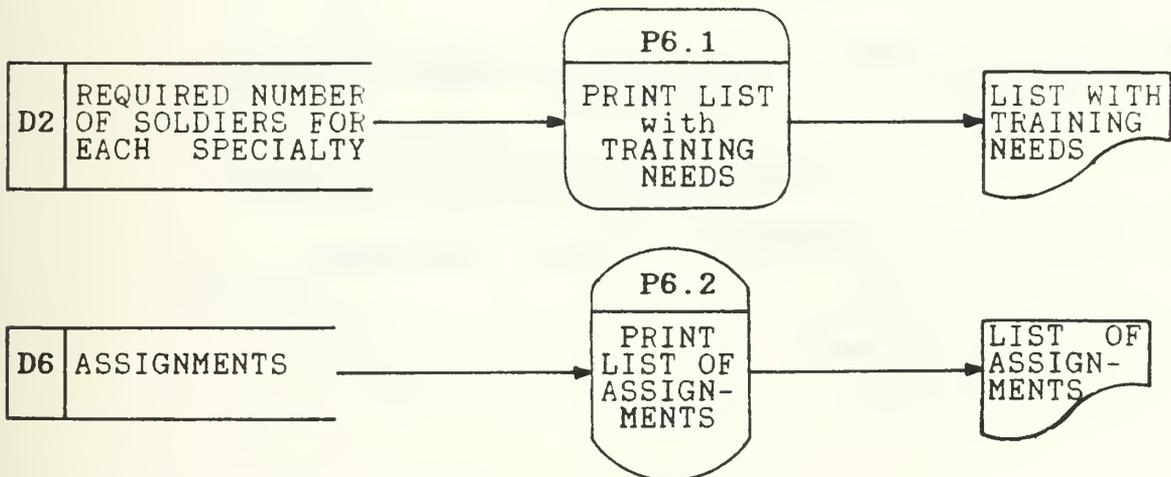
Figure A.7   Process P6

THE DATA DICTIONARY

Section B.1.   The Data Stores

D1 : NEW ENLISTED SOLDIERS

1.   ID_NUMBER

2.   SOLDIER NAME

3.   DATE ENLISTED

4.   SERVICE DURATION

5.   CLASS

6.   MARITAL STATUS

7.   NUMBER OF CHILDREN

8.   FINANCIAL STATUS

9.   FAMILY SUPPORTER

10. NUMBER OF BROTHERS IN SERVICE

11. SPECIALREASONS FOR TRANSFER

12. PREFERED UNITS

13. ADDRESS

14. TOTAL NUMBER OF ENLISTED

D2 : REQUIRED NUMBER OF SOLDIERS FOR EACH SPECIALTY

1.   SPECIALTY

2.   REQUIRED SOLDIERS FOR SPECIALTY TRAINING

D3 : SOLDIERS WHO COMPLETED SPECIALTY TRAINING

1.   ID_NUMBER

2.   SOLDIER NAME

3.   SPECIALTY

D4 : <u>SOLDIERS</u>

    1. ID_NUMBER

    2. SOLDIER NAME

    3. DATE ENLISTED

    4. SERVICE DURATION

    5. CLASS

    6. MARITAL STATUS

    7. NUMBER OF CHILDREN

    8. FINANCIAL STATUS

    9. FAMILY SUPPORTER

    10. NUMBER OF BROTHERS IN SERVICE

    11. SPECIAL REASONS FOR TRANSFER

    12. PREFERED UNITS

    13. ADDRESS

    14. SPECIALTY

    15. ASSIGNED UNIT

    16. DATE ASSIGNED

    17. COMPLETED TRAINING

    18. DATE WHEN REMAINING SERVICE EQUALS 4 MONTHS

D5 : <u>UNITS</u>

    1. SPECIALTY

    2. UNIT NAME

    3. REQUIRED NUMBER OF SOLDIERS

    4. EXISTING NUMBER OF SOLDIERS

    5. COMPLEMENT NUMBER OF SOLDIERS

D6 : <u>ASSIGNMENTS</u>

    1. ID_NUMBER

    2. SOLDIER NAME

    3. SPECIALTY

4. UNIT ASSIGNED TO

          5. DATE OF REPORT

   D7 : CHANGES OF STATUS

          1. ID_NUMBER

          2. CHANGED NAME

          3. CHANGED SERVICE DURATION

          4. CHANGED MARITAL STATUS

          5. CHANGED NUMBER OF CHILDREN

          6. CHANGED FINANCIAL STATUS

          7. CHANGED FAMILY SUPPORTER

          8. CHANGED NUMBER OF BROTHERS IN SERVICE

          9. CHANGED SPECIAL REASONS FOR TRANSFER ,

          10. CHANGED PREFERED UNITS

          11. CHANGED ADDRESS

   D8 : RETIRED SOLDIERS

          1. ID_NUMBER

          2. SOLDIER NAME

          3. SPECIALTY

          4. UNIT

          5. DATE RETIRED

   D9 : SPECIALTIES

          1. SPECIALTY

   D10: HISTORY

          1. ID_NUMBER

          2. SOLDIER NAME

          3. DATE ENLISTED

          4. CLASS

          5. ADDRESS

6.   SPECIALTY

        7.   DATE RETIRED

D11 : <u>SOLDIER QUALIFICATION POINTS</u>

        1.   ID_NUMBER

        2.   SPECIALTY

        3.   QUALIFICATION POINTS

D12 : <u>UNIT NEEDS</u>

        1.   SPECIALTY

        2.   UNIT NAME

        3.   NEEDS

D13 : <u>UNIT NAMES</u>

        1.   UNIT NAME

D14 : <u>CURRENTLY TRAINING SOLDIERS</u>

        1.   ID_NUMBER

        2.   SPECIALTY

Section B.2.   **The Data Elements**

Name: ID_NUMBER

Aliases:

Description: A number given to a soldier during enlistment,
               that uniquely identifies him.

Format: Numeric, PIC 9(7)

Location: D1, D3, D4, D6, D7, D8, D10, D11, D14

Name: SOLDIER NAME

Aliases: CHANGED NAME

Description: The name of a soldier in the form:
               First, Last, Middle Initial.

87

```
Format:        Character, PIC X(36)

Location:      D1, D3, D4, D6, D8, D10


Name:          DATE ENLISTED

Aliases:

Description:   The date of the soldier enlistment

Format:        Date, PIC X(8)

Location:      D1, D4, D10


Name:          SERVICE DURATION

Aliases:       CHANGED SERVICE DURATION

Description:   The duration of the soldier service time in months

Format:        Numeric, PIC 9(2)

Location:      D1, D4


Name:          CLASS

Aliases:

Description:   All soldiers enlisted in the same period belong in
               the same Class

Format:        Alphanumeric, PIC X(3)  (two digits followed by a
               letter.  eg. 87B, 87E, 88A)

Location:      D1, D4, D10


Name:          MARITAL STATUS

Aliases:       CHANGED MARITAL STATUS

Description:   The marital status of the soldier. (eg. married,
               single, divorced, etc)

Format:        Character PIC X(7)

Location:      D1, D4


Name:          NUMBER OF CHILDREN

Aliases:       CHANGED NUMBER OF CHILDREN
```

Description: The number of the soldier's children

Format: Numeric, PIC 9(1)

Location: D1, D4

Name: FINANCIAL STATUS

Aliases: CHANGED FINANCIAL STATUS

Description: The soldier financial status

Format: Character, PIC X(1)
             (G = Good, M = Medium, B = Bad)

Location: D1, D4

Name: FAMILY SUPPORTER

Aliases: CHANGED FAMILY SUPPORTER

Description: A soldier is considered family supporter if his
             father has died and he is the oldest son.

Format: Character, PIC X(3) (YES, NO)

Location: D1, D4

Name: NUMBER OF BROTHERS IN SERVICE

Aliases: CHANGED NUMBER OF BROTHERS IN SERVICE

Description: The number of a soldier's brothers serving the
             Armed Forces.

Format: Numeric, PIC 9(1)

Location: D1, D4

Name: SPECIAL REASONS FOR TRANSFER

Aliases: CHANGED SPECIAL REASONS FOR TRANSFER

Description: A logical field that becomes true if the soldier
             has special reasons to be transfered to a specific
             unit.

Format: Logical, T or F

Location: D1, D4

Name: PREFERED UNITS

Aliases: CHANGED PREFERED UNITS

Description: The names of three units the soldier prefers to be transfered to, in order of preference.

Format: Character, PIC X(21)

Location: D1, D4

Name: ADDRESS

Aliases: CHANGED ADDRESS

Description: The soldier civilian address

Format: Character, PIC X(69)
                    Street, PIC X(15)
                    City,   PIC X(20)
                    State, PIC X(15)
                    ZIP,   PIC X(5)
                    Phone, PIC X(14)

Location: D1, D4, D10

Name: TOTAL NUMBER OF ENLISTED

Aliases:

Description: The total number of new enlisted soldiers

Format: Numeric, PIC 9(4)

Location: D1

Name: SPECIALTY

Aliases:

Description: The name of the soldier's specialty

Format: Character, PIC X(8)

Location: D2, D3, D4, D5, D6, D8, D9, D10, D11, D12, D1

Name: REQUIRED SOLDIERS FOR SPECIALTY TRAINING

Aliases:

Description: The number of new enlisted soldiers who must be trained in each specialty to cover the units needs.

Format: Numeric, PIC 9(4)

Location: D2

Name: ASSIGNED UNIT

Aliases: UNIT, UNIT NAME, UNIT ASSIGNED TO

Description: The name of the unit a soldier is assigned to

Format: Character, PIC X(7)

Location: D4

Name: DATE ASSIGNED

Aliases: DATE OF REPORT

Description: The date when a soldier is assigned to a unit.

Format: Date, PIC X(8)

Location: D8

Name: COMPLETED TRAINING

Aliases:

Description: A logical field that becomes true when a soldier
             completes his specialty training.

Format: Logical, T or F

Location: D4

Name: DATE WHEN REM. SERVICE = 4 MONTHS

Aliases:

Description: The date when the remaining service time of the
             soldier becomes equal to 4 months. This date is
             used in calculating the training needs.

Format: DATE, PIC X(8)

Location: D4

Name: UNIT NAME

Aliases: ASSIGNED UNIT, UNIT ASSIGNED TO, UNIT

91

Description: The name of a unit.

Format: Character, PIC X(7)

Location: D5, D12, D13

Name: REQUIRED NUMBER OF SOLDIERS

Aliases:

Description: The number of soldiers of a specific specialty
             required to meet a unit's needs.

Format: Numeric, PIC 9(3)

Location: D5

Name: EXISTING NUMBER OF SOLDIERS

Aliases:

Description: The number of soldiers of a specific specialty
             in a unit.

Format: Numeric, PIC 9(3)

Location: D5

Name: COMPLEMENT NUMBER OF SOLDIERS

Aliases:

Description: The difference between the required and existing
             number of soldiers in a unit.

Format: Numeric, PIC 9(3)

Location: D5

Name: UNIT ASSIGNED TO

Aliases: UNIT, ASSIGNED UNIT, UNIT NAME

Description: The unit to which a soldier is assigned.

Format: Character, PIC X(7)

Location: D6

Name: DATE OF REPORT

Aliases: DATE ASSIGNED

Description: The date when a soldier is assigned to a unit.

Format: Date, PIC X(8)

Location: D6

Name: CHANGED NAME

Aliases: SOLDIER NAME

Description: The changed name of a soldier in the form:
Last, First, Middle Initial.

Format: Character, PIC X(36)

Location: D7

Name: CHANGED SERVICE DURATION

Aliases: SERVICE DURATION

Description: New service time for a soldier in months.

Format: Numeric, PIC 9(2)

Location: D7

Name: CHANGED MARITAL STATUS

Aliases: MARITAL STATUS

Description: The new marital status of a soldier.

Format: Character, PIC X(7)

Location: D7

Name: CHANGED NUMBER OF CHILDREN

Aliases: NUMBER OF CHILDREN

Description: The new number of children of a soldier.

Format: Nummeric, PIC 9(1)

Location: D7

Name: CHANGED FINANCIAL STATUS

Aliases: FINANCIAL STATUS

Description: The new financial status of a soldier.

Format: Character, PIC X(1)

Location: D7

Name: CHANGED FAMILY SUPPORTER

Aliases: FAMILY SUPPORTER

Description: The new status of the soldier relatively to
                 being a family supporter or not.

Format: Character, PIC X(3)

Location: D7

· **Name:**   CHANGED NUMBER OF BROTHERS IN SERVICE

Aliases: NUMBER OF BROTHERS IN SERVICE

Description: The new number of the soldier's brothers
                 serving the Armed Forces.

Format: Numeric, PIC 9(1)

Location: D7

Name: CHANGED SPECIAL REASONS FOR TRANSFER

Aliases: SPECIAL REASONS FOR TRANSFER

Description: This field is updated whenever the special
                 reasons for transfer change.

Format: Logical, T or F

Location: D7

Name: CHANGED PREFERED UNITS

Aliases: PREFERED UNITS

Description: The names of three units the soldier wants to be
                 transfered to, in order of preference.

Format: Character, PIC X(21)

Location: D7

Name: CHANGED ADDRESS

Aliases: ADDRESS

Description: The new address of a soldier.

Format: see ADDRESS

Location: D7

Name: QUALIFICATION POINTS

Aliases:

Description: A number calculated during the assignments process.
             The higher this number, the more probable for a
             soldier to be transfered to the unit he prefers.

Format: Numeric, PIC 9(3)

Location: D11

Name: UNIT

Aliases: ASSIGNED UNIT, UNIT NAME, UNIT ASSIGNED TO

Description: The unit name of a retired soldier.

Format: Character, PIC X(7)

Location: D8

Name: DATE RETIRED

Aliases:

Description: The date when the soldier retired from service.

Format: Date, PIC X(8)

Location: D8, D10

Name: NEEDS

Aliases:

Description: The number of soldiers of a specialty that a unit
             requires to acomplish its mission.

Format: Numeric, PIC 9(3)

Location: D12

APPENDIX C

PROCESS DESCRIPTIONS

Section C.1 : <u>Algorithm Description of Process P1</u>

<u>INPUT</u>   : D1, D4, D5, D9, D14

<u>OUTPUT</u>   : D2

<u>PROCESS</u> :

  Get TOTAL_ENL (total number of new enlisted soldiers)
      from D1.

  Get CURRENT DATE.

  Calculate TOTAL_REQ (total number of required soldiers for
      all spesialties) as follows:

  TOTAL_REQ = TOTAL_COMPL + TOTAL_RET + TOTAL_TR where:

      TOTAL_COMPL = The sum of all COMPLEMENT fields in D5

      TOTAL_RET = The number of records in D4 with DATE
                   WHEN REMAINING SERVICE EQUALS 4 MONTHS
                   earlier than CURRENT DATE.

      TOTAL_TR = The number of records in D14.

  For each Specialty i in D9 do the following:

      Calculate COMi (the number of soldiers needed to sa-
          tisfy the needs of all units for this specialty) by
          adding all COMPLEMENT fields for Specialty i in D5.

      Calculate TRi (number of soldiers currently training
          in Specialty i) by counting the records in D14 with
          SPECIALTY = i)

Calculate RETi (number of soldiers to retire within
the next 4 months) by counting the records in D4
with DATE WHEN REMAINING SERVICE EQUALS 4 MONTHS
earlier than CURRENT DATE and SPECIALTY =i.

Calculate REQi (total number of soldiers required to
fully satisfy the needs for Specialty i) as follows:
REQi = COMi + RETi - TRi

Calculate Xi (number of new enlisted soldiers to be
trained in Specialty i) as follows:
Xi = REQi * TOTAL_ENL / TOTAL_REQ

Append a record to data store D2.

Store i and Xi in D2.

Section C.2 : <u>Algorithm Description of Process P2.1</u>

<u>INPUT</u>   : D1

<u>OUTPUT</u> : D4

<u>PROCESS</u> :

For each record in D1 do the following:

Create a new record in D4.

Read all the fields from the record in D1 into the
respective fields of the record in D4.

Initialize the rest of the fields of the record in D4:

COMPLETED TRAINING = False

SPECIALTY = "ZZ...Z"

ASSIGNED UNIT = "ZZ...Z"

DATE ASSIGNED = 01/01/01

DATE WHEN REMAINING SERVICE EQUALS 4 MONTHS =

(DATE ENLISTED) + (DURATION OF SERVICE) - (4 Months)

Section C.3 : <u>Algorithm Description of Process P2.2.1</u>

<u>INPUT</u>   : D3

<u>OUTPUT</u>   : D4

<u>PROCESS</u> :

    For each record in D3 do the following:

        Read the ID_NUMBER and SPECIALTY.

        Find the record of the soldier in D4 using ID_NUMBER

          as a key.

        Update the SPECIALTY field.

        Write "True" into the COMPLETED TRAINING field.

Section C.4 : <u>Algorithm Description of Process P2.2.2</u>

<u>INPUT</u>   : D6

<u>OUTPUT</u>   : D4

<u>PROCESS</u> :

    For each record in D6 do the following:

        Read the ID_NUMBER, UNIT ASSIGNED TO and DATE OF

          REPORT.

        Find the record in D4 with the same ID_NUMBER.

        Update the ASSIGNED UNIT and ASSIGNED fields in this

          record.

Section C.5 : <u>Algorithm Description of Process P2.2.3</u>

<u>INPUT</u>   : D7

<u>OUTPUT</u>   : D4

<u>PROCESS</u> :

    For each record in D7 do the following:

        Read the ID_NUMBER and the rest fields.

        Find the record in D4 with the same ID_NUMBER.

If CHANGED SERVICE DURATION <> 99 then

(DATE WHEN REMAINING SERVICE EQUALS 4 MONTHS) =
(DATE ENLISTED) + (CHANGED DURATION OF SERVICE) -
(4 MONTHS).

If CHANGED NAME <> "ZZ...Z" then

update SOLDIER NAME in D4.

If CHANGED MARITAL STATUS <> "ZZ...Z" then

update MARITAL STATUS in D4.

If CHANGED FINANCIAL STATUS <> "ZZ...Z" then

update FINANCIAL STATUS in D4.

If CHANGED FAMILY SUPPORTER <> False then

update FAMILY SUPPORTER in D4.

If CHANGED NUMBER OF CHILDREN <> 9 then

update NUMBER OF CHILDREN in D4.

If CHANGED SPECIAL REASONS FOR TANSFER <> False then

update SPECIAL REASONS FOR TRANSFER in D4.

If CHANGED PREFERED UNIT 1 <> "ZZ...Z" then

update PREFERED UNIT 1 in D4.

If CHANGED PREFERED UNIT 2 <> "ZZ...Z" then

update PREFERED UNIT 2 in D4.

If CHANGED PREFERED UNIT 3 <> "ZZ...Z" then

update PREFERED UNIT 3 in D4.

If CHANGED STREET <> "ZZ...Z" then

update STREET in D4.

If CHANGED CITY <> "ZZ...Z" then

update CITY in D4.

If CHANGED STATE <> "ZZ...Z" then

update STATE in D4.

If CHANGED ZIP <> "ZZ...Z" then

update ZIP in D4.

```
If CHANGED PHONE <> "ZZ...Z"  then
    update PHONE in D4.
```

Section C.6 : <u>Algorithm Description of Process P2.3</u>

<u>INPUT</u>   : D4, D8
<u>OUTPUT</u>   : D4, D10
<u>PROCESS</u> :

    For each record in D8 do the following:
        Read the ID_NUMBER and the DATE RETIRED.
        Find the record in D4 with the same ID_NUMBER.
        Create a new record in D10.
        Transfer the following fields into the respective
            fields in D10:
            - ID_NUMBER
            - SOLDIER NAME
            - DATE ENLISTED
            - CLASS
            - ADDRESS
            - SPECIALTY
        Write DATE RETIRED into the respective field in D10.
        Delete the record from D4.

Section C.7 : <u>Algorithm Description of Process P3.1</u>

<u>INPUT</u>   : D8
<u>OUTPUT</u>   : D5
<u>PROCESS</u> :

    For each record in D8 do the following:
        Read the UNIT and SPECIALTY
        Find the unit record in D5, using UNIT and SPECIALTY
            as a key.

100

Decrement by 1 the EXISTING field.

Increment by 1 the COMPLEMENT field.

Section C.8 : <u>Algorithm Description of Process P3.2</u>

<u>INPUT</u>   : D6

<u>OUTPUT</u>   : D5

<u>PROCESS</u> :

    For each record in D6 do the following:

        Read the UNIT ASSIGNED TO and SPECIALTY.

        Find the record in D5, using UNIT ASSIGNED TO and
          SPECIALTY as a key.

        Increment by 1 the EXISTING field.

        Decrement by 1 the COMPLEMENT field.

Section C.9 : <u>Algorithm Description of Process P4.1</u>

<u>INPUT</u>   : D3, D4

<u>OUTPUT</u>   : D11

<u>PROCESS</u> :

    For each record in D3 do the following:

        Initialize variable POINTS = 0

        Read the ID_NUMER and SPECIALTY.

        Find the record in D4 with the same ID_NUMBER.

        If MARITAL STATUS = "MARRIED" then

          add 40 to POINTS.

        If NUMBER OF CHILDREN > 1 then

          add 70 to POINTS.

        If NUMBER OF CHILDREN = 1 then

          add 35 to POINTS.

        If FAMILY SUPPORTER = True then

          add 40 to POINTS.

If FINANCIAL ABILITY = "BAD" then

   add 20 to POINTS.

If FINANCIAL ABILITY = "MEDIUM" then

   add 10 to PPINTS.

If NUMBER OF BROTHERS IN SERVICE > 1 then

   add 20 to POINTS.

If NUMBER OF BROTHERS IN SERVICE = 1 then

   add 10 to POINTS.

If SPECIAL REASONS FOR TRANSFER = True then

   add 10 to POINTS.

Add a new record to D11.

Write the ID_NUMER, POINTS and SPECIALTY into the

   respective fields in D11.

Section C.10 : <u>Algorithm Description of Process P4.2</u>

<u>INPUT</u>   : D3, D5, D9, D13
<u>OUTPUT</u> : D12
<u>PROCESS</u> :

   For each record in D9 do the following:

   Read the SPECIALTY name.

   Calculate TOTAL_ASSIGN (number of soldiers to be as-
      signed for this specialty), by counting the records
      in D3 with the same SPECIALTY name.

   Calculate TOTAL_REQ (number of soldiers of this spe-
      cialty required for all units), by summing up the
      COMPLEMENT fields for this specialty in D5.

   For each record in D13 do the following:

   Read the UNIT NAME.

   Find the record in D5, using SPECIALTY and UNIT NAME
      as a key.

102

Read the COMPLEMENT field in this record.

Calculate:

NEEDS = (TOTAL_ASSIGN / TOTAL_REQ) * COMPLEMENT

Create a record in D12.

Write the UNIT NAME, SPECIALTY and NEEDS into the
respective fields.

Section C.11 : <u>Algorithm Description of Process P4.3</u>

<u>INPUT</u>   : D3, D4, D11, D12

<u>OUTPUT</u>   : D6

<u>PROCESS</u> :

Get CURRENT DATE.

DATE OF REPORT = CURRENT DATE + 7 DAYS.

Sort D11 by SPECIALTY and QUALIFICATION POINTS.

For each record in D11 do the following:

Read the ID_NUMBER and SPECIALTY.

Find the record in D4 with the same ID_NUMBER.

Read the PREFERED UNIT 1, PREFERED UNIT 2 and PREFERED
UNIT 3.

Find the record in D12 using PREFERED UNIT 1 and
SPECIALTY as a key.

If NEEDS > 0 then

assign the soldier to prefered unit 1 as follows:

- Subtract 1 from the NEEDS field in D12.

- Write the ID_NUMBER, SPECIALTY and PREFERED UNIT 1
in a new record in D6.

Else find the record in D12 using PREFERED UNIT 2 and
SPECIALTY as a key.

103

If NEEDS > 0   then

assign the soldier to prefered unit 2   as above.

Else find the record in D12   using PREFERED UNIT 3

and SPECIALTY as a key.

If NEEDS > 0   then

assign the soldier to prefered unit 3 as above.

Else assign the soldier to the first unit in D12

in which the NEEDS for this SPECIALTY is > 0.

Section C.12 : <u>Algorithm Description of Process P5.1</u>

<u>INPUT</u>   : Manual list of Retired soldiers

<u>OUTPUT</u>   : D8

<u>PROCESS</u> :

Delete all previous records from D8.

For each record in the manual list do the following:

Append a record to D8.

Read the ID_NUMBER, SOLDIER NAME, SPECIALTY, UNIT and

DATE RETIRED fields into the respective fields in D8.

Section C.13 : <u>Algorithm Description of Process P5.2</u>

<u>INPUT</u>   : Manual list of Soldiers who completed spec. training

<u>OUTPUT</u>   : D3

<u>PROCESS</u> :

Delete all previous records from D3.

For each record in the manual list do the following:

Append a record to D3.

Read the ID_NUMBER, SOLDIER NAME and SPECIALTY fields

into the respective fields in D3.

Section C.14 : <u>Algorithm Description of Process P5.3</u>

<u>INPUT</u>   : Manual list of changes in soldier status

<u>OUTPUT</u>   : D7

<u>PROCESS</u> :

    Delete all previous records from D7.

    For each record in the manual list do the following:

        Append a record to D7.

        Read all fields of the manual list record into the
        respective fields of the record in D7.

Section C.15 : <u>Algorithm Description of Process P5.4</u>

<u>INPUT</u>   : Manual list of new enlisted soldiers

<u>OUTPUT</u>   : D1

<u>PROCESS</u> :

    Delete all previous records from D1.

    For each record in the manual list do the following:

        Append a record to D1.

        Read all fields in the manual list record into the
        respective fields of the record in D1.

Section C.16 : <u>Algorithm Description of Process P5.5</u>

<u>INPUT</u>   : Manual list of soldiers enrolled in special training

<u>OUTPUT</u>   : D14

<u>PROCESS</u> :

    Delete all previous records from D14.

    For each record in the manual list do the following:

        Append a record to D14.

        Read the ID_NUMBER and SPECIALTY fields from the list
        into the respective fields of the record in D14.

105

Section C.17 : <u>Algorithm Description of Process P6.1</u>

<u>INPUT</u>   : D2

<u>OUTPUT</u>   : Printed list of TRAINING NEEDS

<u>PROCESS</u> :

    Prepare the system printer to print.

    Print the list according to a desired format.

Section C.18 : <u>Algorithm Description of Process P6.2</u>

<u>INPUT</u>   : D6

<u>OUTPUT</u>   : Printed list of ASSIGNMENTS

<u>PROCESS</u> :

    Prepare the system printer to print.

    Print the list according to a desired format.

A P P E N D I X   D

APPLICATION   PROGRAMS   DESIGN

1000

PERS_MGT
PERSONNEL
MANAGEMENT

1100
UNIT_REP
PROCESS
UNITS
REPORTS

1200
ASSIGNT
PROCESS
ASSIGNMENTS

1300
ENL_SLDS
PROCESS new
ENLISTED
SOLDIERS

1400
TRAINING
PROCESS
TRAINING
NEEDS

1500
STC_REPS
PROCESS
STC
REPORTS

1110
RET_REP
PROCESS
RETIRED
SOLDIERS
REPORT

1210   P4.1
CALC_PTS
CALCULATE
TRANSFER
POINTS for
each soldier

1310 P5.4
GET_ENL
GET
ENLISTED
SOLDIERS
DATA

1410   P1
TRN_NEED
ESTIMATE
NEEDS
for each
SPECIALTY

1510
COMP_TRN
PROCESS sldrs
who COMPLETED
SPECIALTY
TRAINING

1111 P5.1
GET_RET
GET
RETIRED
DATA

1220   P4.2
CLC_NEED
CALCULATE
UNIT NEEDS/
SPECIALTY

1320   P2.1
ADD_SLD
ADD new
RECORDS to
SOLDIER files

1420   P6.1
PR_TRAIN
PRINT LIST
with TRAINING
NEEDS

1511   P5.2
GET_TRND
GET TRAINED
SOLDIERS
DATA

1112   P3.1
UPDNRET
UPDATE UNITS
file with
RETIRED
SOLDIERS

1230   P4.3
ASGN_SLD
ASSIGN
each SOLDIER
to a UNIT

1512 P2.2.1
UPSLDTRN
UPDATE
SOLDIER file
with new TRAINED
SOLDIERS

1113 P2.3
UPD_HIST
UPDATE
HISTORY
File

1240   P3.2
UPDUNASN
UPDATE
UNIT file
with
ASSIGNMENTS

1520   P5.5
SP_TRNE
PROCESS
SOLDIERS
ENROLLED in
SPECIALTY
TRAINING

1120
CHNG_REP
PROCESS
REPORT for
CHANGES in
soldier data

1250 P2.2.2
UPSLDASN
UPDATE
SOLDIER
file with
ASSIGNMENTS

1121 P5.3
GET_CHNG
GET
CHANGES
DATA

1260   P6.2
PR_ASIGN
PRINT
LIST of
ASSIGNMENTS

1122 P2.2.3
UPSLDCHN
UPDATE
SOLDIER file
with CHANGES

Figure D.1   The System Structure Chart

108

PROGRAM LEVEL				DESCRIPTION
0	1	2	3	
1000				Main control program of the PERSONNEL MANA-GEMENT SYSTEM. It displays a menu screen and depending on the user's choice it gives control to one of five functions.
	1100			- Enters the units reports into the system and performs the necessary updates to the system files.
		1110		- Enters the units report for RETIRED SOL-DIERS to the system and updates the UNIT_ORG, HISTORY, SLD_ADDR and SLD_SERV file.
			1111	- Creates the RETIRED file and updates it with the data from the unit report.
			1112	- Reads each record in the RETIRED file and updates the UNIT_ORG file.
			1113	- Reads each record in the RETIRED file, updates the HISTORY file and deletes the soldier record from the SLD_ADDR and SLD_SERV files.
		1120		- Enters the units reports for CHANGES in SOLDIER DATA to the system and updates the SLD_ADDR, SLD_SERV, SLD_TRANSF and SLD_PREF files.
			1121	- Creates the CHANGES file and updates it with the data from the unit report.
			1122	- Reads each record in the CHANGES file and updates the SLD_ADDR, SLD_SERV, SLD_TRANSF and SLD_PREF files.
	1200			- Calls other modules which assign soldiers to units, update the units and soldier files and print the list of assignments.
		1210		- Calculates transfer qualification points for each soldier and updates the TRSF_PTS file.
		1220		- Calculates the units needs for soldiers of each specialty and updates the UNIT_REQ file.
		1230		- Considers the soldier transfer points, his preferences and the unit needs and assigns each soldier to a unit. This decision is entered into the ASSIGNMS file.

Figure D.2   The Visual Table of Contents (VTOC)
            of the Personnel Management System (Contin.)

PROGRAM LEVEL				DESCRIPTION
0	1	2	3	
		1240		- Reads each record in the ASSIGNMS file and updates the UNIT_ORG file.
		1250		- Reads each record in the ASSIGNMS file and updates the SLD_SERV file.
		1260		- Prints the list of ASSIGNMENTS.
	1300			- Enters the EC report for new ENLISTED SOLDIERS into the system and updates the ENLISTED, SLD_ADDR, SLD_SERV, SLD_TRANSF and SLD_PREF files.
		1310		- Creates the ENLISTED file and updates it with the data from the EC report.
		1320		- Reads each record in the ENLISTED file and updates the SLD_ADDR, SLD_SERV, SLD_TRANSF and SLD_PREF files.
	1400			- Calculates the training needs for each specialty and prints a list of the needs.
		1410		- Calculates the training needs for each specialty andstores them in the TRAIN_REQ file.
		1420		- Prints the list with the training needs for each specialty.
	1500			- Enters the STC reports into the system and updates the necessary files.
		1510		- Enters the STC report for SOLDIERS WHO COMPLETED SPECIALTY TRAINING into the system and updates the COMPL_TR and SLD_SERV files.
			1511	- Creates the COMPL_TR file and updates it with the data from the STC report.
			1512	- Reads each record in the COMPL_TR file and updates the SLD_SERV file.
		1520		- Creates the TRAINEES file and updates it with the data from the STC report for the soldiers currently enrolled in special training.

(contin.) **Figure D.2** The Visual Table of Contents (VTOC) of the Personnel Management System

```
SYSTEM        : PERSONNEL MANAGEMENT
MODULE NAME : PERS_MGT
MODULE No    : 1000
DESIGNER     : Labros Karatasios        DATE : 4/30/1987
```

```
INVOKED BY MODULE :            INVOKES MODULES :

                               1100, 1200, 1300, 1400,
                               1500
```

```
INPUTS :                       OUTPUTS :

ENLISTED,  TRAIN_RQ,           ENLISTED,  TRAIN_RQ,
COMPL_TR,  SLD_ADDR,           COMPL_TR,  SLD_ADDR,
SLD_SERV,  SLD_TRAN,           SLD_SERV,  SLD_TRAN
SLD_PREF,  UNIT_ORG,           SLD_PREF,  UNIT_ORG,
ASSIGNMS,  CHANGES,            ASSIGNMS,  RETIRED,
RETIRED,   SPECS,              HISTORY,   TRSF_PTS,
TRAINEES,  TRSF_PTS,           UNIT_REQ,  TRAINEES
UNIT_REQ,  UNITS
```

```
PROCESS :  See  Flowchart  in  Figure  D.3.1.a
```

Figure D.3.1   The IPO chart of program PERS_MGT

111

1000

PERS-MGT

CLEAR
SCREEN

CLOSE
FILES

DISPLAY
MAIN MENU

READ
CHOICE

CHOICE=1 — Y → DO PROCEDURE UNIT-REP

N

CHOICE=2 — Y → DO PROCEDURE ASSIGNMT

N

CHOICE=3 — Y → DO PROCEDURE ENL-SLDS

N

CHOICE=4 — Y → DO PROCEDURE TRAINING

N

CHOICE=5 — Y → DO PROCEDURE STC-REPS

N

CHOICE=6 — N → DISPLAY ERROR MESSAGE

Y

END
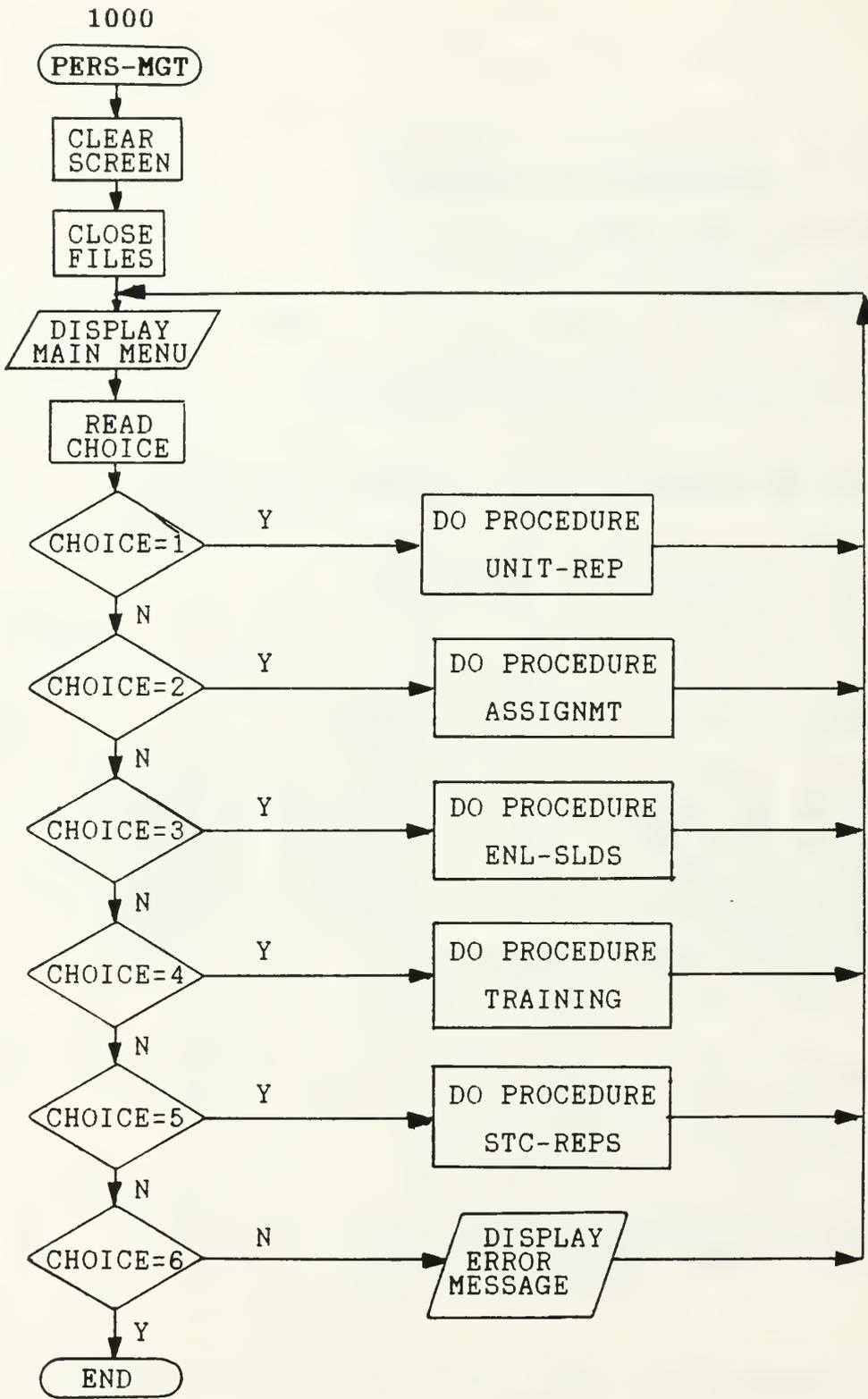
Figure D.3.1.a   The flowchart of program   PERS-MGT

112

```
┌────────────────────────────────────────────────────────────────┐
│  SYSTEM        : PERSONNEL MANAGEMENT                           │
│  MODULE NAME : UNIT_REP                                         │
│  MODULE No    : 1100                                            │
│  DESIGNER     : Labros Karatasios        DATE : 4/30/1987       │
├────────────────────────────────────────────────────────────────┤
│                                                                │
│   ┌──────────────────────────┐   ┌──────────────────────────┐  │
│   │ INVOKED BY MODULE :      │   │ INVOKES MODULES :        │  │
│   │      1000                │   │      1110, 1120          │  │
│   │                          │   │                          │  │
│   └──────────────────────────┘   └──────────────────────────┘  │
│                                                                │
│   ┌──────────────────────────┐   ┌──────────────────────────┐  │
│   │ INPUTS :                 │   │ OUTPUTS :                │  │
│   │ Manual list of retired   │   │ RETIRED, UNIT_ORG,       │  │
│   │ soldiers, RETIRED,       │   │ HISTORY, SLD_ADDR,       │  │
│   │ UNIT_ORG, SLD_SERV,      │   │ SLD_SERV, SLD_TRAN,      │  │
│   │ SLD_ADDR, Manual list    │   │ SLD_PREF, CHANGES        │  │
│   │ of changes, CHANGES      │   │                          │  │
│   └──────────────────────────┘   └──────────────────────────┘  │
│                                                                │
│   ┌──────────────────────────────────────────────────────────┐ │
│   │ PROCESS :  See  Flowchart  in  Figure  D.3.2.a          │ │
│   └──────────────────────────────────────────────────────────┘ │
│                                                                │
└────────────────────────────────────────────────────────────────┘
```

Figure D.3.2   The IPO chart of program UNIT_REP
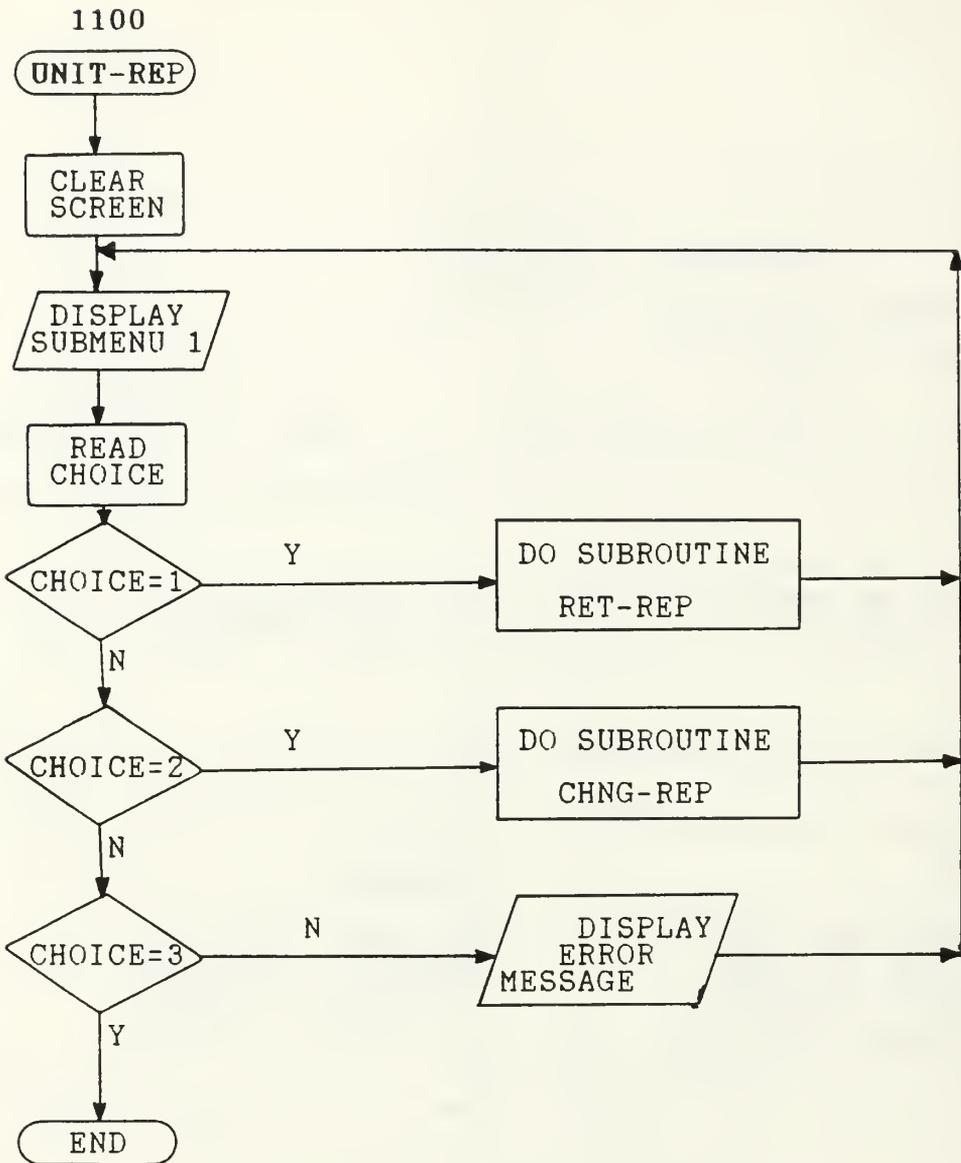
Figure D.3.2.a   The flowchart of program UNIT-REP

114

```
┌─────────────────────────────────────────────────────────────────┐
│                                                                 │
│  SYSTEM      : PERSONNEL MANAGEMENT                              │
│                                                                 │
│  MODULE NAME : RET_REP                                          │
│                                                                 │
│  MODULE No   : 1110                                            │
│                                                                 │
│  DESIGNER    : Labros Karatasios       DATE : 4/30/1987        │
│                                                                 │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│   ┌───────────────────────────┐   ┌───────────────────────────┐ │
│   │ INVOKED BY MODULE :       │   │ INVOKES MODULES :         │ │
│   │      1100                 │   │ 1111, 1112, 1113          │ │
│   │                           │   │                           │ │
│   └───────────────────────────┘   └───────────────────────────┘ │
│                                                                 │
│   ┌───────────────────────────┐   ┌───────────────────────────┐ │
│   │ INPUTS :                  │   │ OUTPUTS :                 │ │
│   │ Manual list of retired    │   │ RETIRED, UNIT_ORG,        │ │
│   │ soldiers, RETIRED,        │   │ SLD_ADDR, SLD_SERV,       │ │
│   │ UNIT_ORG, SLD_SERV,       │   │ SLD_TRAN, SLD_PREF,       │ │
│   │ SLD_ADDR                  │   │ HISTORY                   │ │
│   └───────────────────────────┘   └───────────────────────────┘ │
│                                                                 │
│   ┌───────────────────────────────────────────────────────────┐ │
│   │ PROCESS :  See  Flowchart  in  Figure  D.3.3.a            │ │
│   └───────────────────────────────────────────────────────────┘ │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

Figure D.3.3   The IPO chart of program RET_REP

115
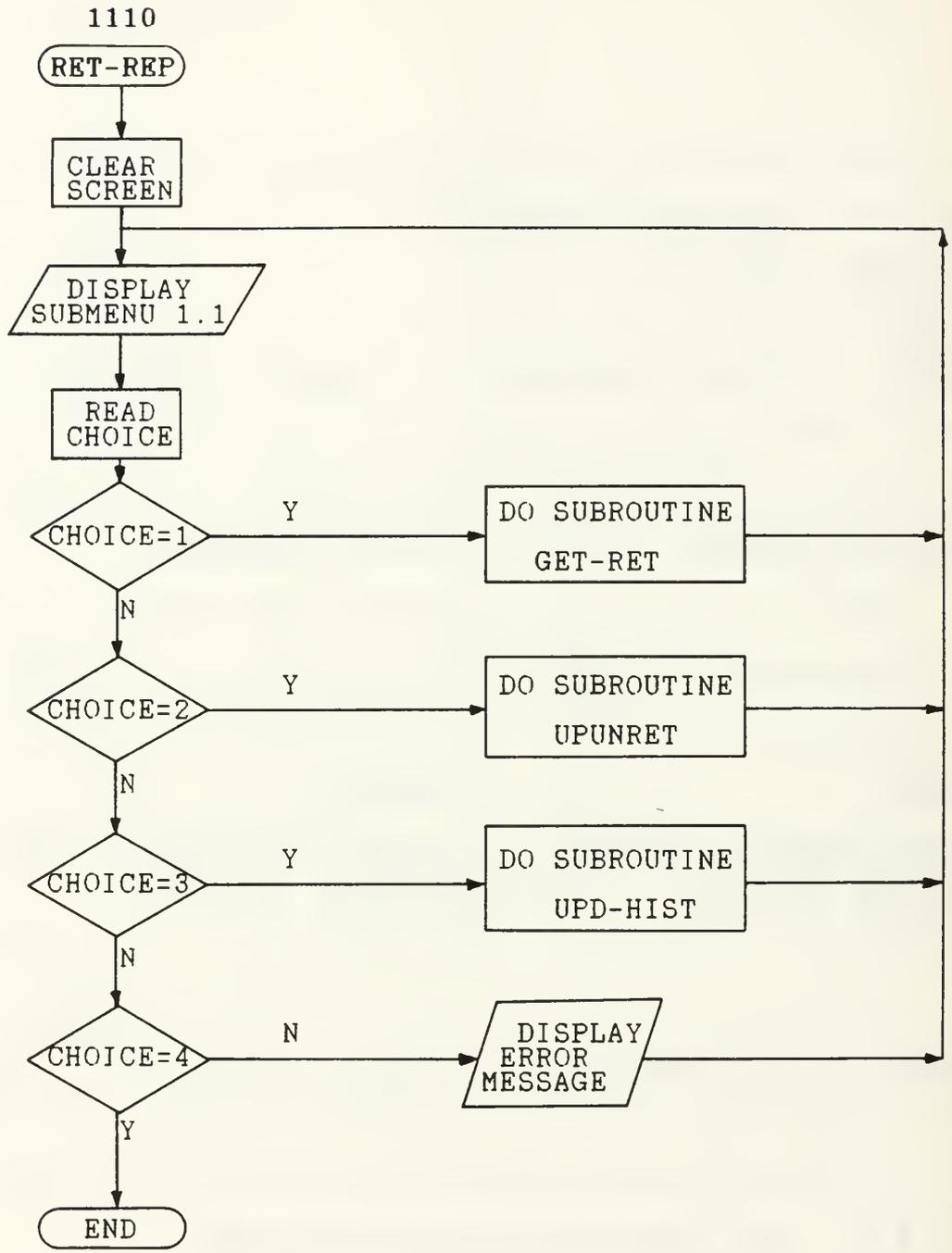
Figure D.3.3.a   The flowchart of program RET-REP

116

```
┌─────────────────────────────────────────────────────────────────┐
│                                                                   │
│  SYSTEM        : PERSONNEL MANAGEMENT                              │
│  MODULE NAME : GET_RET                                             │
│  MODULE No    : 1111                                              │
│  DESIGNER     : Labros Karatasios      DATE : 4/30/1987           │
│                                                                   │
├─────────────────────────────────────────────────────────────────┤
│                                                                   │
│   ┌─────────────────────────┐      ┌─────────────────────────┐   │
│   │ INVOKED BY MODULE :      │      │ INVOKES MODULES :        │   │
│   │      1110                │      │                          │   │
│   │                          │      │                          │   │
│   └─────────────────────────┘      └─────────────────────────┘   │
│                                                                   │
│   ┌─────────────────────────┐      ┌─────────────────────────┐   │
│   │ INPUTS :                 │      │ OUTPUTS :                │   │
│   │ Manual list of retired   │      │ RETIRED                  │   │
│   │ soldiers                 │      │                          │   │
│   └─────────────────────────┘      └─────────────────────────┘   │
│                                                                   │
│   ┌──────────────────────────────────────────────────────────┐   │
│   │ PROCESS :  See  Flowchart  in  Figure  D.3.4.a            │   │
│   └──────────────────────────────────────────────────────────┘   │
│                                                                   │
└─────────────────────────────────────────────────────────────────┘
```

Figure D.3.4   The IPO chart of program GET_RET

117
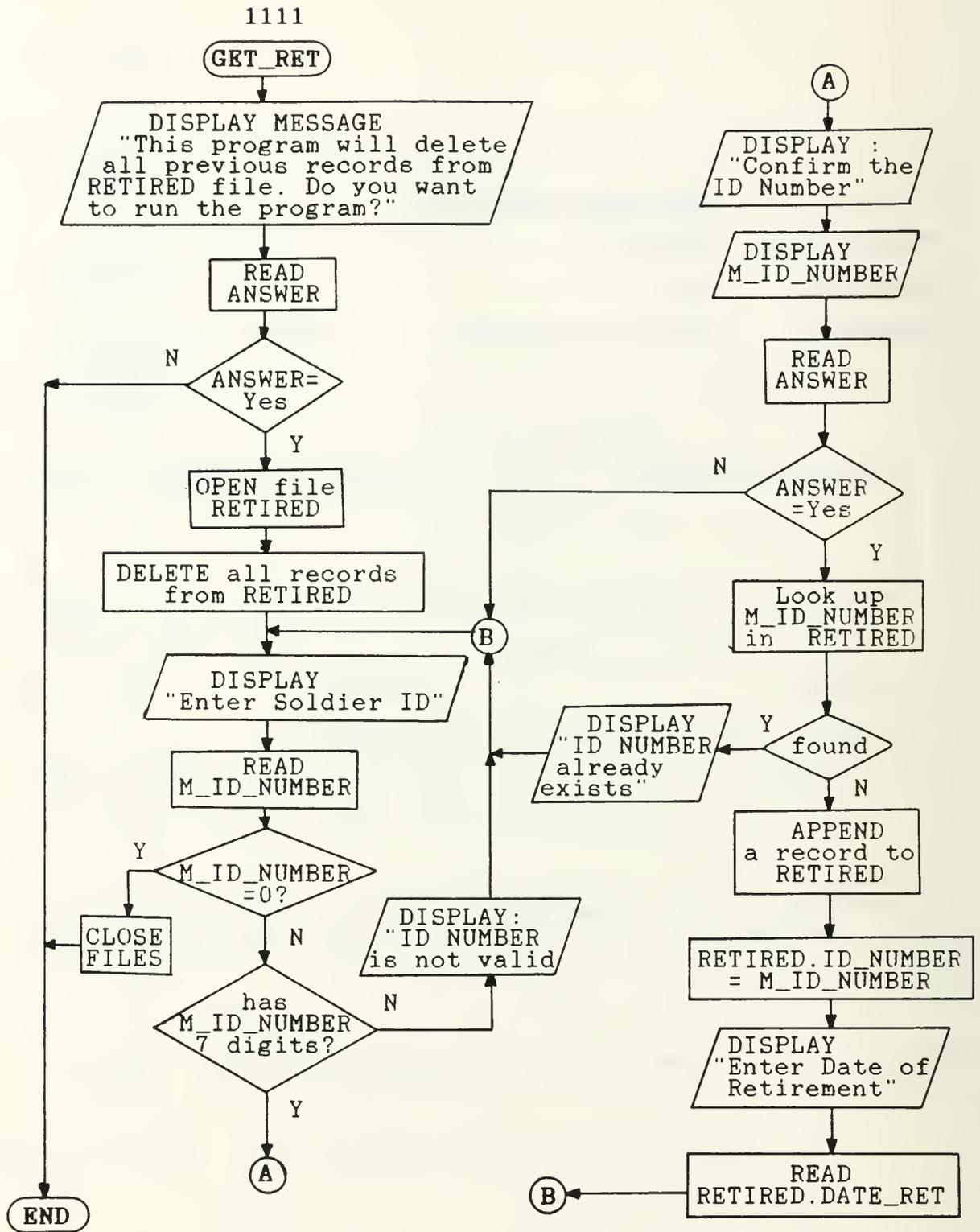
Figure D.3.4.a   The flowchart of program GET-RET

```
SYSTEM        : PERSONNEL MANAGEMENT
MODULE NAME : UPUNRET
MODULE No    : 1112
DESIGNER     : Labros Karatasios        DATE : 4/30/1987
```

INVOKED BY MODULE :
    1110

INVOKES MODULES :

INPUTS :
RETIRED, UNIT_ORG,
SLD_SERV

OUTPUTS :
UNIT_ORG

PROCESS : See Flowchart in Figure D.3.5.a

Figure D.3.5   The IPO chart of program UPUNRET

1112
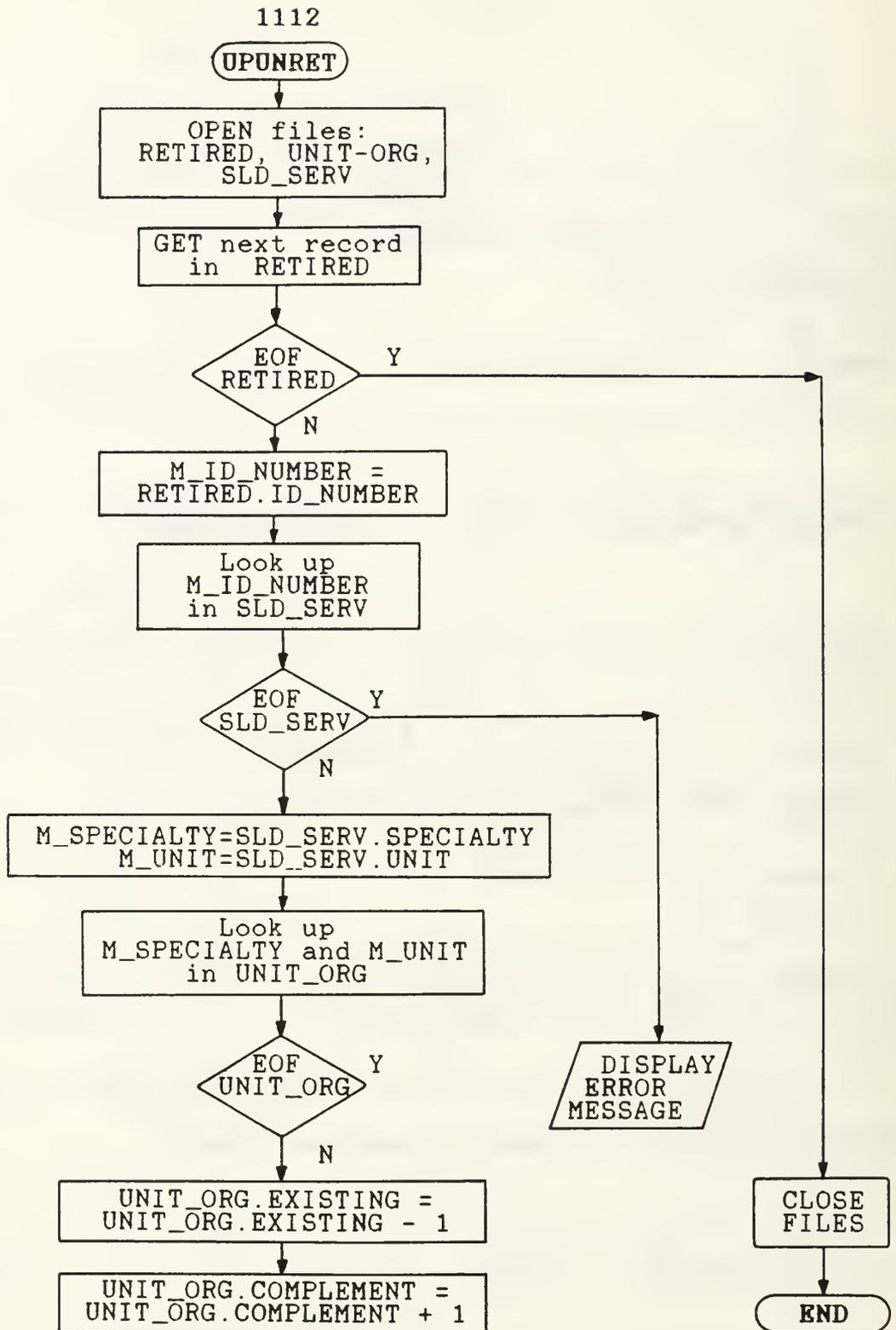
Figure D3.5.a   The flowchart of program UPUNRET

120

```
┌─────────────────────────────────────────────────────────────────┐
│  SYSTEM        : PERSONNEL MANAGEMENT                            │
│  MODULE NAME : UPD_HIST                                          │
│  MODULE No    : 1113                                            │
│  DESIGNER     : Labros Karatasios        DATE : 4/30/1987       │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│    ┌──────────────────────────┐    ┌──────────────────────────┐ │
│    │ INVOKED BY MODULE :      │    │ INVOKES MODULES :        │ │
│    │      1110                │    │                          │ │
│    │                          │    │                          │ │
│    └──────────────────────────┘    └──────────────────────────┘ │
│                                                                 │
│    ┌──────────────────────────┐    ┌──────────────────────────┐ │
│    │ INPUTS :                 │    │ OUTPUTS :                │ │
│    │ RETIRED, SLD_ADDR,       │    │ HISTORY, SLD_ADDR,       │ │
│    │ SLD_SERV                 │    │ SLD_SERV, SLD_TRAN,      │ │
│    │                          │    │ SLD_PREF                 │ │
│    └──────────────────────────┘    └──────────────────────────┘ │
│                                                                 │
│    ┌───────────────────────────────────────────────────────┐   │
│    │ PROCESS :  See  Flowchart  in  Figure  D.3.6.a        │   │
│    └───────────────────────────────────────────────────────┘   │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

Figure D.3.6   The IPO chart of program UPD_HIST

121
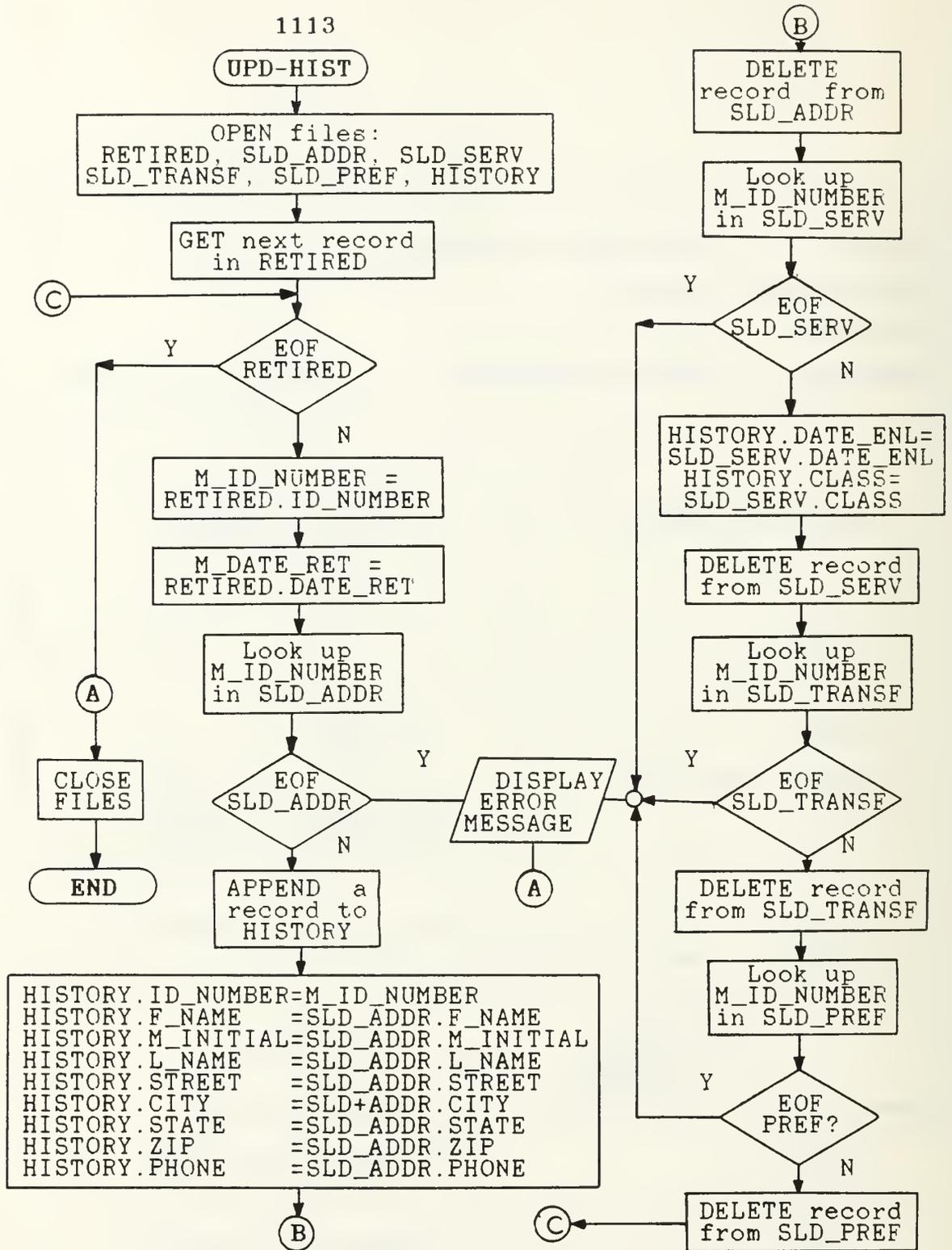
Figure D.3.6a   The flowchart of program UPD-HIST

122

```
SYSTEM         : PERSONNEL MANAGEMENT
MODULE NAME : CHNG_REP
MODULE No    : 1120
DESIGNER     : Labros Karatasios          DATE : 4/30/1987
```

```
INVOKED BY MODULE :              INVOKES MODULES :
      1100                       1121, 1122
```

```
INPUTS :                         OUTPUTS :
Manual list of changes,          CHANGES,   SLD_ADDR,
CHANGES                          SLD_TRAN,  SLD_PREF
```

```
PROCESS :  See  Flowchart  in  Figure  D.3.7.a
```

Figure D.3.7   The IPO chart of program CHNG_REP
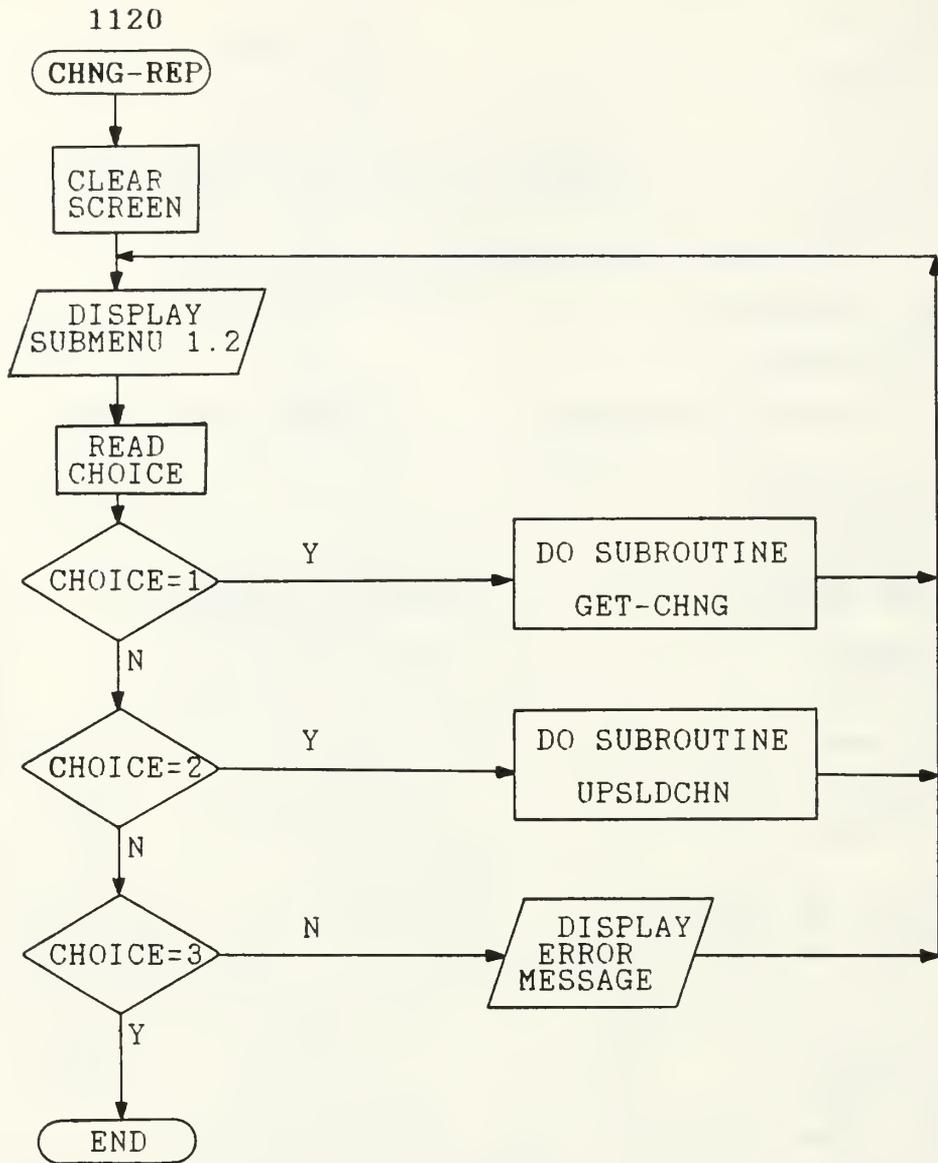
Figure D.3.7.a   The flowchart of program CHNG-REP

124

```
┌─────────────────────────────────────────────────────────────────────┐
│                                                                       │
│  SYSTEM      : PERSONNEL MANAGEMENT                                    │
│  MODULE NAME : GET_CHNG                                                │
│  MODULE No   : 1121                                                    │
│  DESIGNER    : Labros Karatasios        DATE : 4/30/1987              │
│                                                                       │
├───────────────────────────────────────────────────────────────────────┤
│                                                                       │
│   ┌──────────────────────────────┐   ┌──────────────────────────────┐ │
│   │  INVOKED BY MODULE :         │   │  INVOKES MODULES :           │ │
│   │       1120                   │   │                              │ │
│   │                              │   │                              │ │
│   └──────────────────────────────┘   └──────────────────────────────┘ │
│                                                                       │
│   ┌──────────────────────────────┐   ┌──────────────────────────────┐ │
│   │  INPUTS :                    │   │  OUTPUTS :                   │ │
│   │  Manual list of changes      │   │  CHANGES                     │ │
│   │                              │   │                              │ │
│   └──────────────────────────────┘   └──────────────────────────────┘ │
│                                                                       │
│   ┌───────────────────────────────────────────────────────────────┐  │
│   │  PROCESS :   See  Flowchart  in  Figure  D.3.8.a              │  │
│   └───────────────────────────────────────────────────────────────┘  │
│                                                                       │
└─────────────────────────────────────────────────────────────────────┘
```

Figure D.3.8   The IPO chart of program GET_CHNG

125
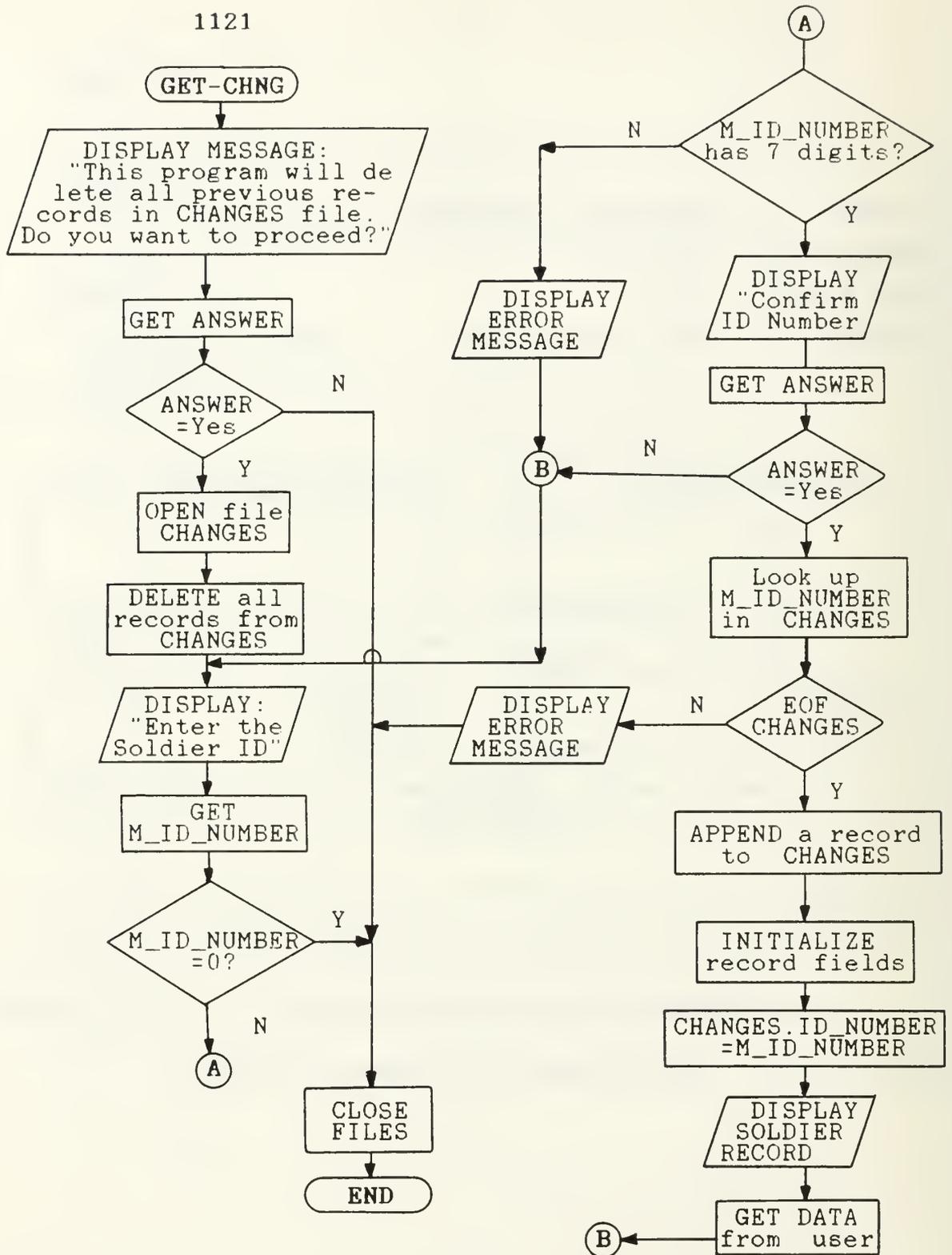
Figure D.3.8a The flowchart of program GET-CHNG

```
SYSTEM         : PERSONNEL MANAGEMENT
MODULE NAME : UPSLDCHN
MODULE No      : 1122
DESIGNER       : Labros Karatasios        DATE : 4/30/1987
```

```
INVOKED BY MODULE :                    INVOKES MODULES :

      1120



INPUTS :                               OUTPUTS :

CHANGES                                SLD_ADDR, SLD_SERV,
                                       SLD_TRAN, SLD_PREF



PROCESS :  See  Flowchart  in  Figure  D.3.9.a
```

Figure D.3.9   The IPO chart of program UPSLDCHN
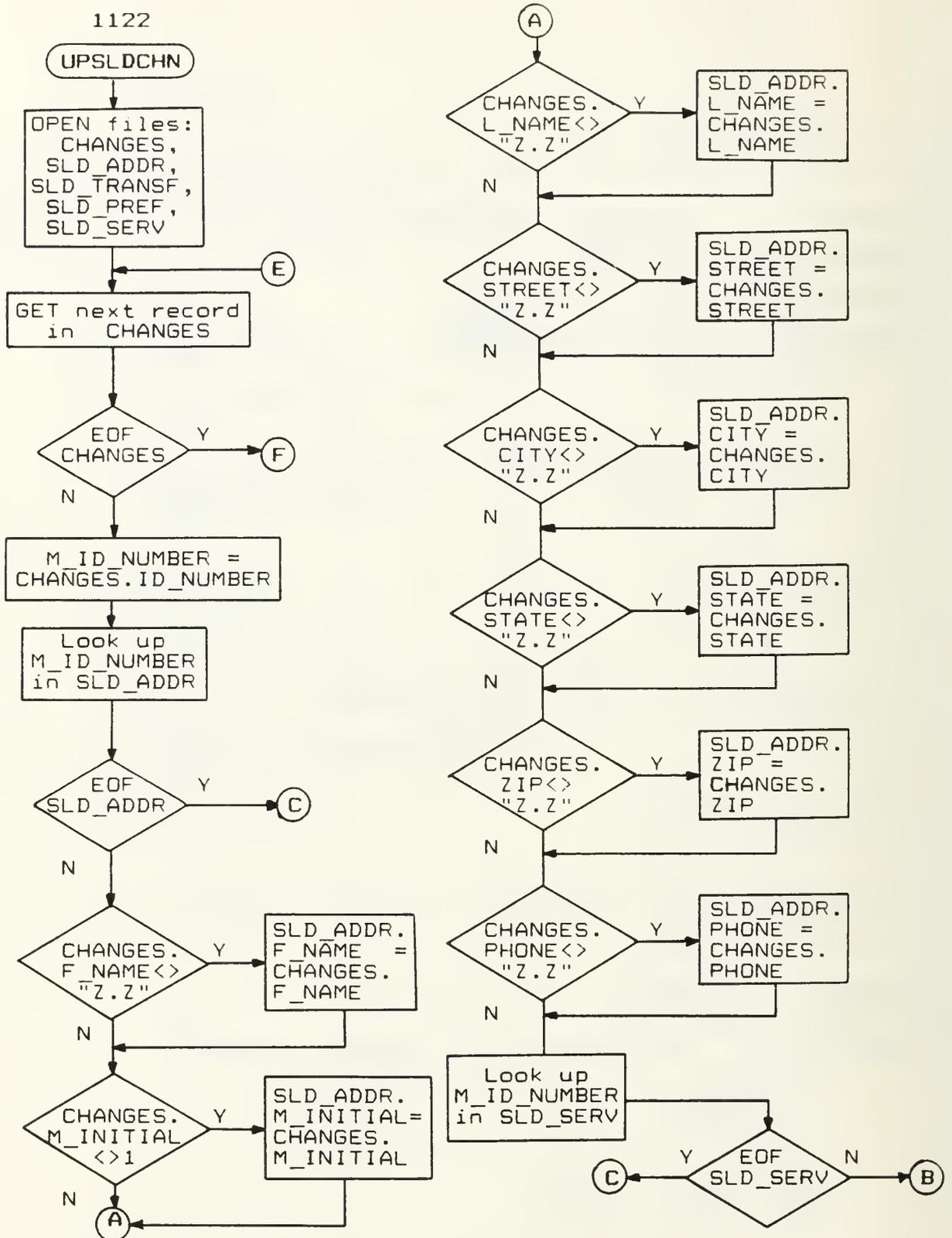
Figure D.3.9a   The flowchart of program UPSLDCHN (part 1 of 2)

128
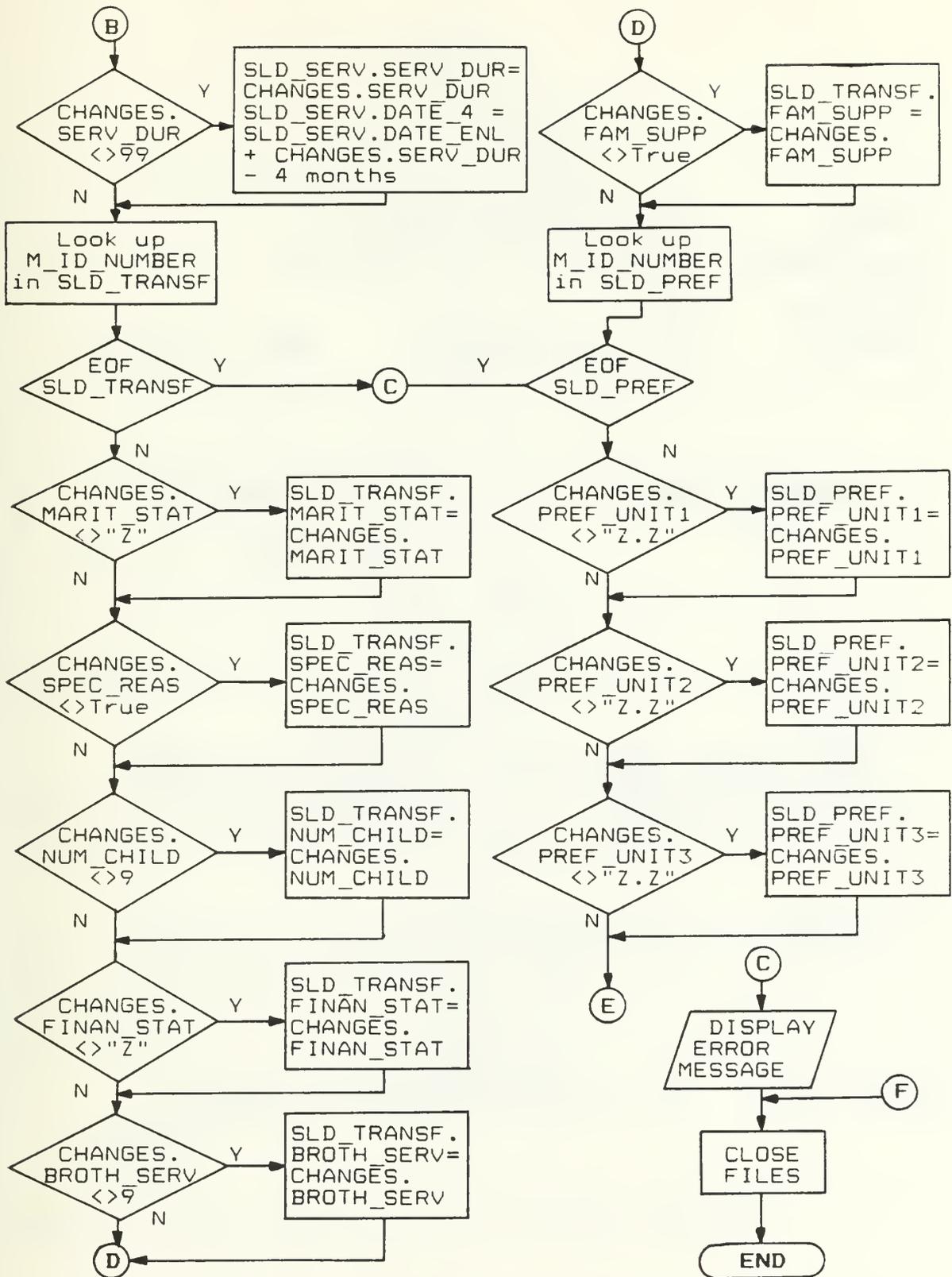
Figure D.3.9a   The flowchart of program UPSLDCHN (part 2 of 2)

129

```
SYSTEM        : PERSONNEL MANAGEMENT

MODULE NAME : ASSIGNMT

MODULE No    : 1200

DESIGNER     : Labros Karatasios        DATE : 4/30/1987
```

```
INVOKED BY MODULE :              INVOKES MODULES :

      1000                       1210, 1220, 1230, 1240,
                                 1250, 1260
```

```
INPUTS                           OUTPUTS :

COMPL_TR, SLD_TRAN,              TRSF_PTS, UNIT_REQ,
SPECS, UNITS, UNIT_ORG,          ASSIGNMS, UNIT_ORG,
TRSF_PTS, SLD_PREF,              SLD_SERV, Printed list of
UNIT_REQ, ASSIGNMS               assignments
```

```
PROCESS :  See  Flowchart  in  Figure  D.3.10.a
```

Figure D.3.10   The IPO chart of program ASSIGNMT

130

1200

```
        ┌─────────────┐
        │  ASSIGNMT   │
        └──────┬──────┘
               │
        ┌──────┴──────┐
        │ CLEAR SCREEN│
        └──────┬──────┘
               │
        ╱───────────────╲
       ╱  DISPLAY program ╲
      ╱  PURPOSE. Ask us-  ╲
      ╲  er if he wants    ╱
       ╲ to proceed.      ╱
        ╲───────┬───────╱
                │
         ┌──────┴──────┐
         │    GET      │
         │   ANSWER    │
         └──────┬──────┘
                │
   N         ╱──┴──╲
 ◄──────────╱ ANSWER ╲
            ╲ =Yes   ╱
             ╲──┬──╱
                │ Y
         ┌──────┴──────┐
         │     DO      │
         │  CALC_PTS   │
         └──────┬──────┘
                │
         ┌──────┴──────┐
         │     DO      │
         │  CLC_NEED   │
         └──────┬──────┘
                │
         ┌──────┴──────┐
         │     DO      │
         │  ASGN_SLD   │
         └──────┬──────┘
                │
         ┌──────┴──────┐
         │     DO      │
         │  UPDUNASN   │
         └──────┬──────┘
                │
         ┌──────┴──────┐
         │     DO      │
         │  UPSLDASN   │
         └──────┬──────┘
                │
         ┌──────┴──────┐
         │     DO      │
         │  PR_ASIGN   │
         └──────┬──────┘
                │
         ┌──────┴──────┐
         │  CLOSE all  │
         │   files     │
         └──────┬──────┘
                │
   ┌────────────┘
 ┌─┴──────┐
 │  END   │
 └────────┘
```
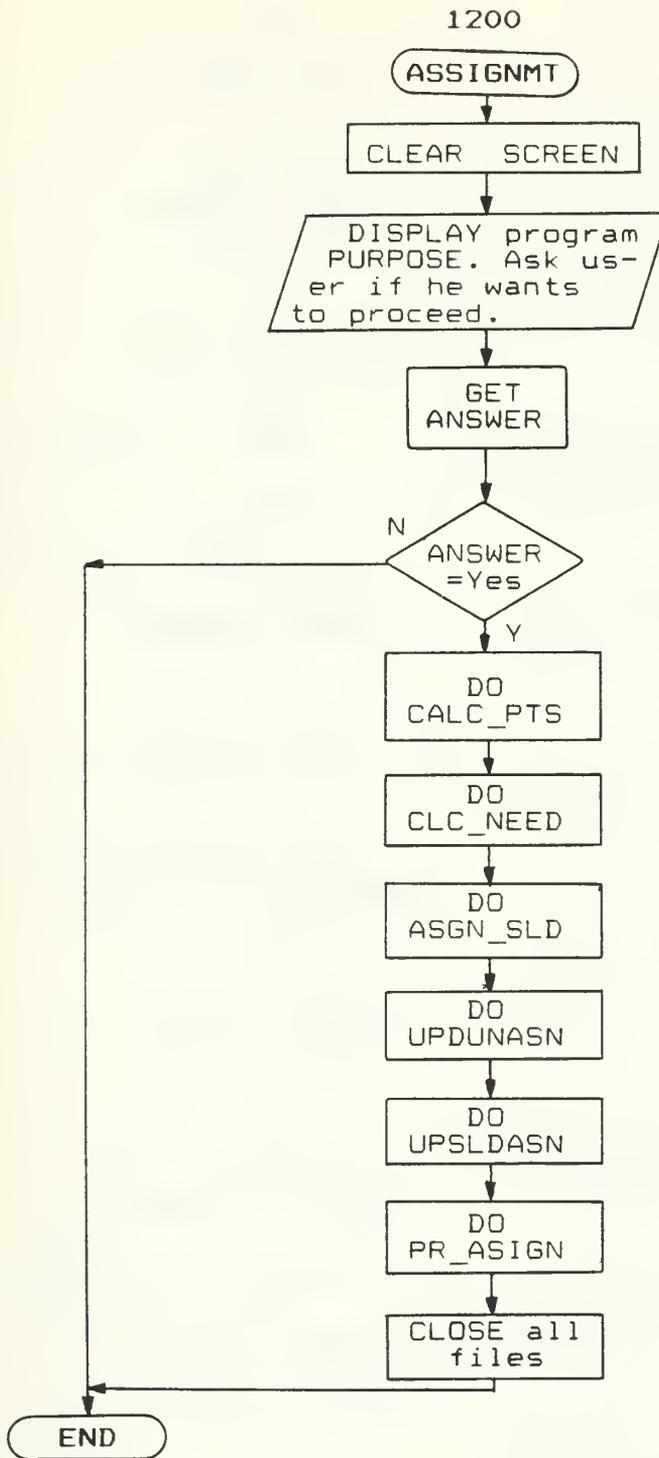
Figure D.3.10a   The flowchart of program ASSIGNMT

131

```
SYSTEM        : PERSONNEL MANAGEMENT
MODULE NAME : CALC_PTS
MODULE No    : 1210
DESIGNER      : Labros Karatasios         DATE : 4/30/1987
```

```
INVOKED BY MODULE :              INVOKES MODULES :
     1200
```

```
INPUTS :                         OUTPUTS :
COMPL_TR, SLD_TRAN               TRSF_PTS
```

```
PROCESS :  See  Flowchart  in  Figure  D.3.11.a
```

Figure D.3.11   The IPO chart of program CALC_PTS

1210

CALC-PTS

OPEN files:
COMPL_TR,
SLD_TRANSF,
TRSF_PTS

B

GET next record
in COMPL_TR

EOF
COMPL_TR — Y

N

POINTS = 0

M_ID_NUMBER=
COMPL_TR.
ID_NUMBER

M_SPECIALTY=
COMPL_TR.SPECIALTY

Look up
M_ID_NUMBER
in SLD_TRANSF

EOF
SLD_TRANSF — Y

DISPLAY
ERROR
MESSAGE

N

SLD_TRANSF.
MARIT_STAT=
"M" — N

Y

POINTS =
POINTS + 40 → A

CLOSE
FILES

END

A

SLD_TRANSF.
NUM_CHILD
>1 — N

Y

POINTS=
POINTS + 70

SLD_TRANSF.
NUM_CHILD
=1 — N

Y

POINTS=
POINTS + 35

SLD_TRANSF.
FINAN_STAT
="B" — N

Y

POINTS=
POINTS + 20

SLD_TRANSF.
FAM_SUPP
=True — N

Y

POINTS=
POINTS + 40

SLD_TRANSF.
FINAN_STAT
="M" — N

Y

POINTS=
POINTS + 10

SLD_TRANSF.
BROTH_SERV
>1 — N

Y

POINTS=
POINTS + 20

SLD_TRANSF.
SPEC_REAS
=True — N

Y

POINTS=
POINTS + 10

SLD_TRANSF.
BROTH_SERV
=1 — N

Y

POINTS=
POINTS + 10

APPEND a
record to
TRSF_PTS

TRSF_PTS.ID_NUMBER=
M_ID_NUMBER
TRSF_PTS.SPECIALTY=
M_SPECIALTY
TRSF_PTS.QUAL_PTS=
POINTS

B

Figure D.3.11a   The flowchart of program CALC-PTS

133

```
┌─────────────────────────────────────────────────────────────────┐
│                                                                 │
│  SYSTEM      : PERSONNEL MANAGEMENT                              │
│  MODULE NAME : CLC_NEED                                          │
│  MODULE No   : 1220                                             │
│  DESIGNER    : Labros Karatasios        DATE : 4/30/1987        │
│                                                                 │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│    ┌───────────────────────────┐   ┌───────────────────────┐   │
│    │  INVOKED BY MODULE :      │   │  INVOKES MODULES :    │   │
│    │       1200                │   │                       │   │
│    │                           │   │                       │   │
│    └───────────────────────────┘   └───────────────────────┘   │
│                                                                 │
│    ┌───────────────────────────┐   ┌───────────────────────┐   │
│    │  INPUTS :                 │   │  OUTPUTS :            │   │
│    │  SPECS, COMPL_TR, UNITS,  │   │  UNIT_REQ             │   │
│    │  UNIT_ORG                 │   │                       │   │
│    └───────────────────────────┘   └───────────────────────┘   │
│                                                                 │
│    ┌─────────────────────────────────────────────────────────┐ │
│    │  PROCESS :  See  Flowchart  in  Figure  D.3.12.a        │ │
│    └─────────────────────────────────────────────────────────┘ │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

Figure D.3.12   The IPO chart of program CLC_NEED
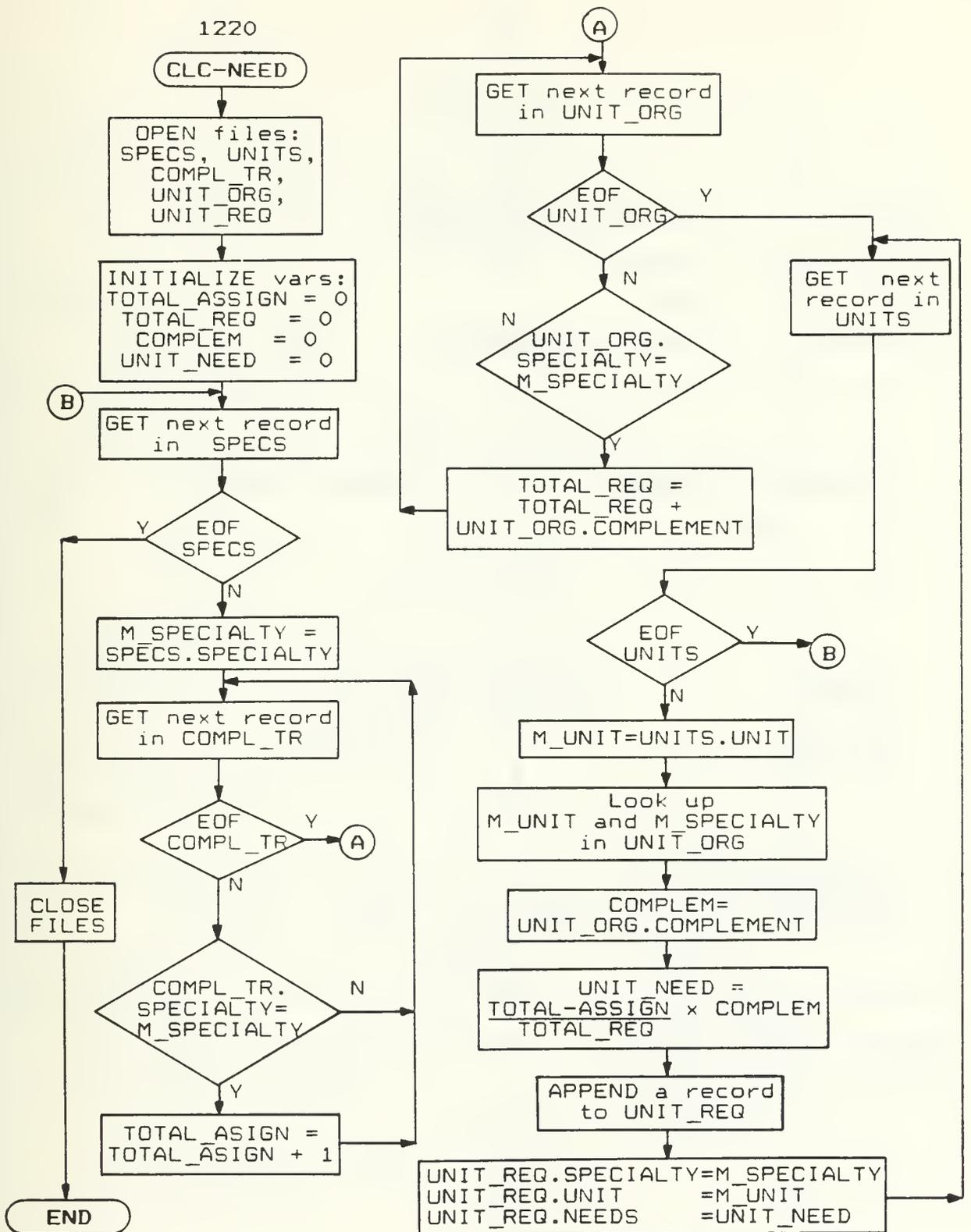
Figure D.3.12a   The flowchart of program CLC-NEED

135

```
SYSTEM        : PERSONNEL MANAGEMENT
MODULE NAME   : ASGN_SLD
MODULE No     : 1230
DESIGNER      : Labros Karatasios        DATE : 4/30/1987
```

```
INVOKED BY MODULE :              INVOKES MODULES :
      1200


INPUTS :                         OUTPUTS :
TRSF_PTS, SLD_PREF,              ASSIGNMS
UNIT_REQ


PROCESS :   See  Flowchart  in  Figure  D.3.13.a
```

Figure D.3.13   The IPO chart of program ASGN_SLD
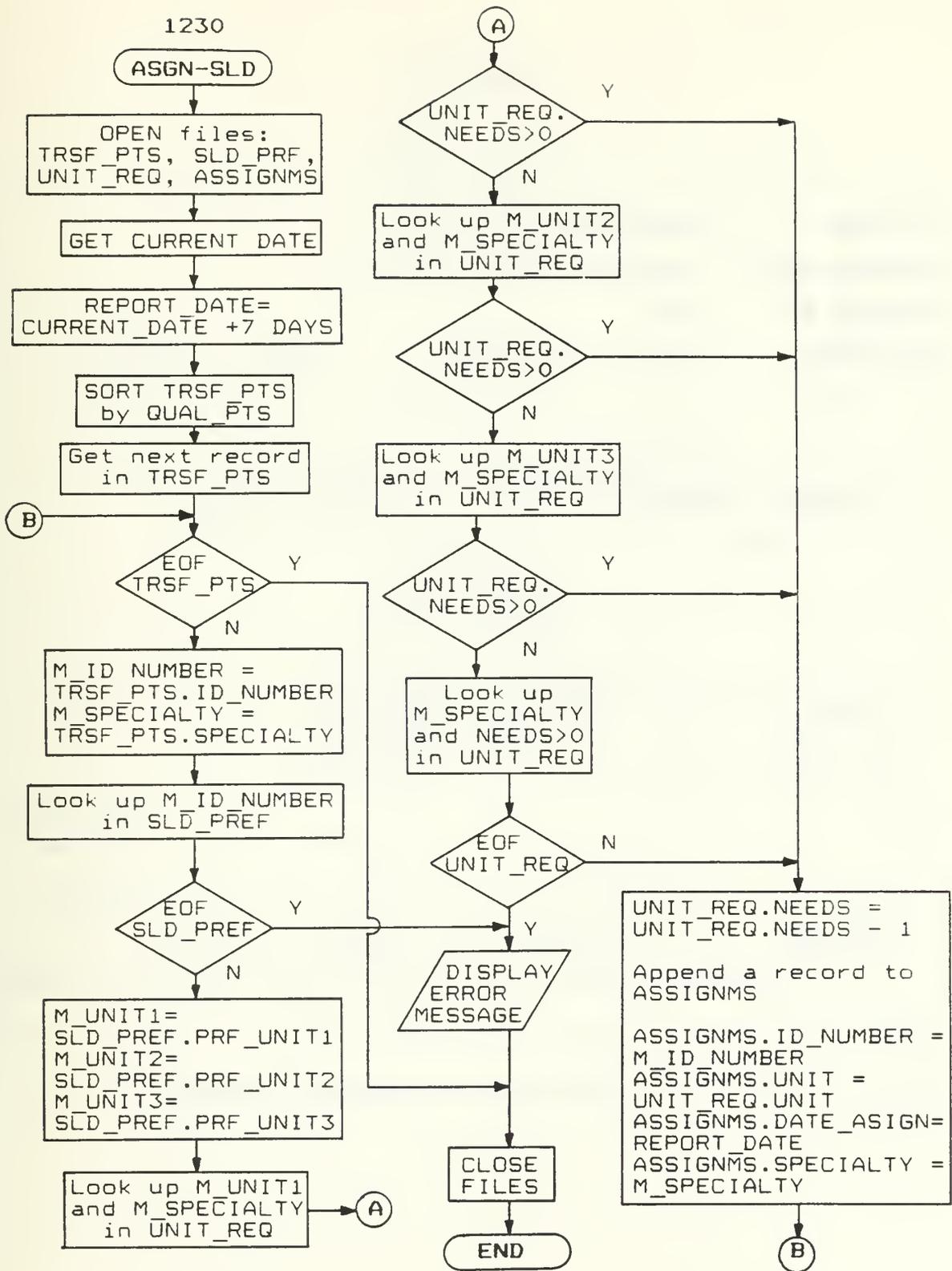
Figure D.3.13a   The flowchart of program ASGN-SLD

137

```
┌─────────────────────────────────────────────────────────────────────┐
│                                                                       │
│  SYSTEM      : PERSONNEL MANAGEMENT                                    │
│  MODULE NAME : UPDUNASN                                                │
│  MODULE No   : 1240                                                    │
│  DESIGNER    : Labros Karatasios          DATE : 4/30/1987            │
│                                                                       │
├─────────────────────────────────────────────────────────────────────┤
│                                                                       │
│   ┌──────────────────────────┐      ┌──────────────────────────┐     │
│   │  INVOKED BY MODULE :      │      │  INVOKES MODULES :        │     │
│   │       1200                │      │                           │     │
│   │                           │      │                           │     │
│   └──────────────────────────┘      └──────────────────────────┘     │
│                                                                       │
│   ┌──────────────────────────┐      ┌──────────────────────────┐     │
│   │  INPUTS :                 │      │  OUTPUTS :                │     │
│   │  ASSIGNMS                 │      │  UNIT_ORG                 │     │
│   │                           │      │                           │     │
│   └──────────────────────────┘      └──────────────────────────┘     │
│                                                                       │
│   ┌───────────────────────────────────────────────────────────┐     │
│   │  PROCESS :   See   Flowchart   in   Figure   D.3.14.a       │     │
│   └───────────────────────────────────────────────────────────┘     │
│                                                                       │
└─────────────────────────────────────────────────────────────────────┘
```

Figure D.3.14   The IPO chart of program UPDUNASN

138

1240
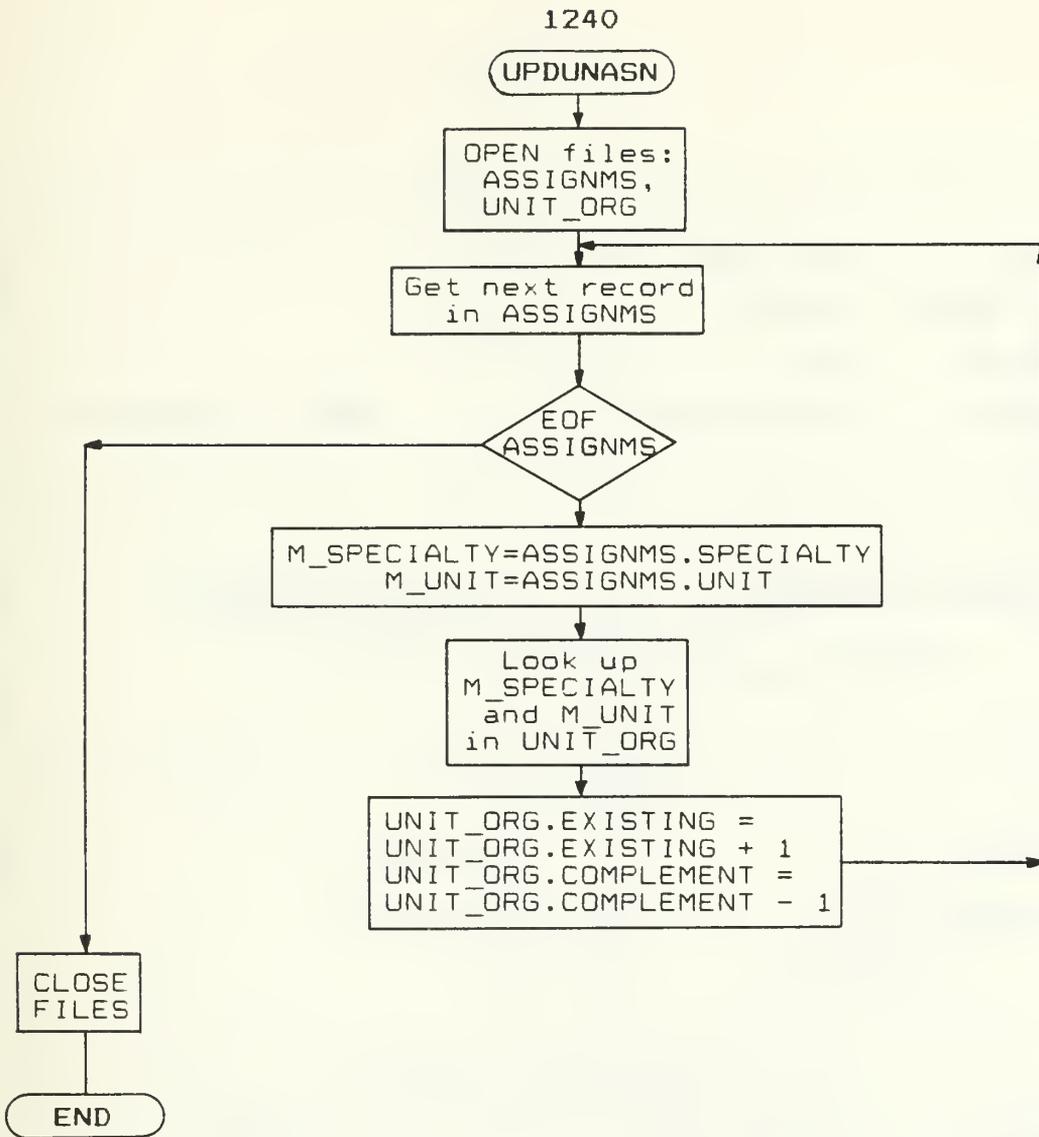
Figure D.3.14.a   The flowchart of program UPDUNASN

139

```
SYSTEM      : PERSONNEL MANAGEMENT
MODULE NAME : UPSLDASN
MODULE No   : 1250
DESIGNER    : Labros Karatasios       DATE : 4/30/1987
```

```
INVOKED BY MODULE :          INVOKES MODULES :
      1200


INPUTS :                     OUTPUTS :
ASSIGNMS                     SLD_SERV


PROCESS : See Flowchart in Figure D.3.15.a
```
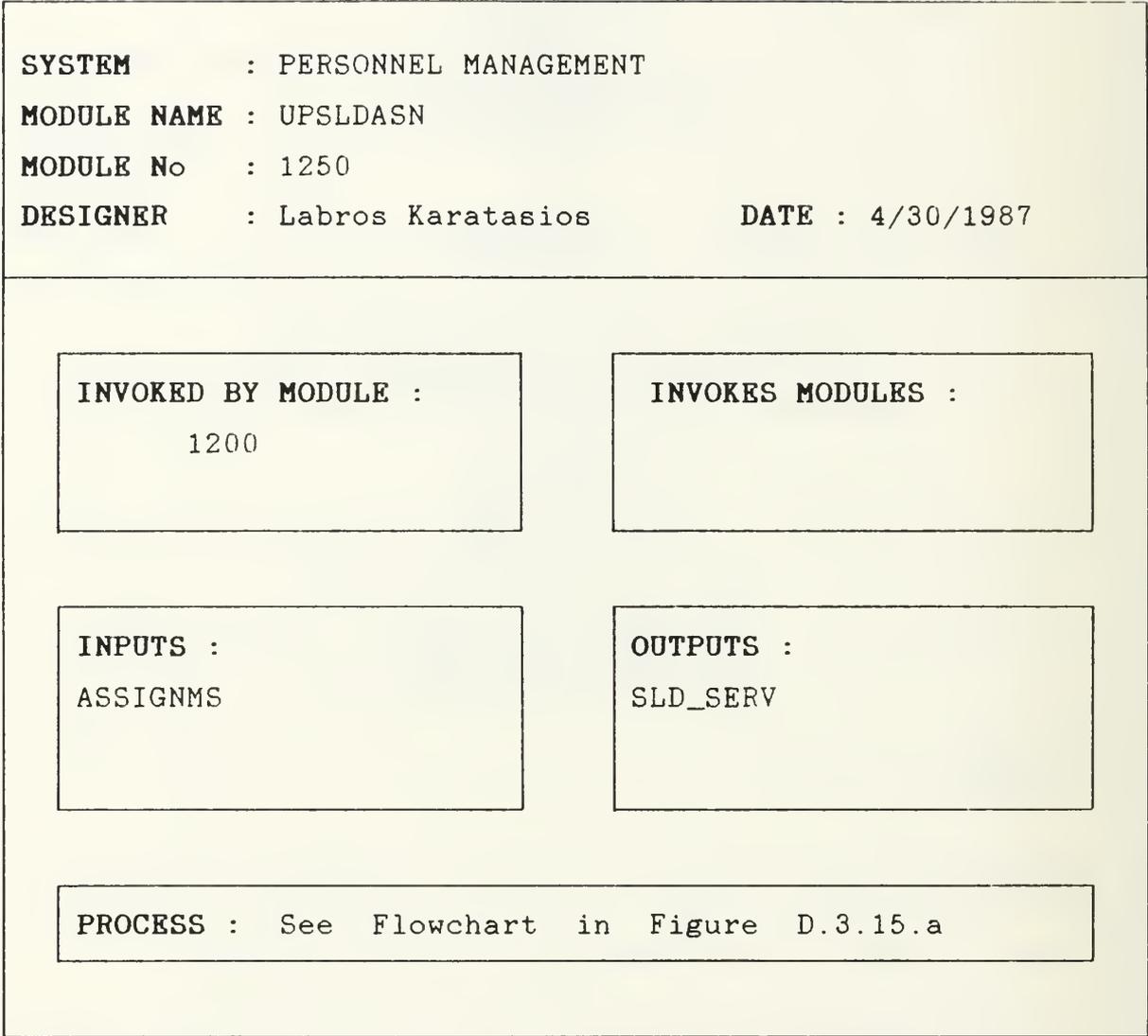
Figure D.3.15   The IPO chart of program UPSLDASN
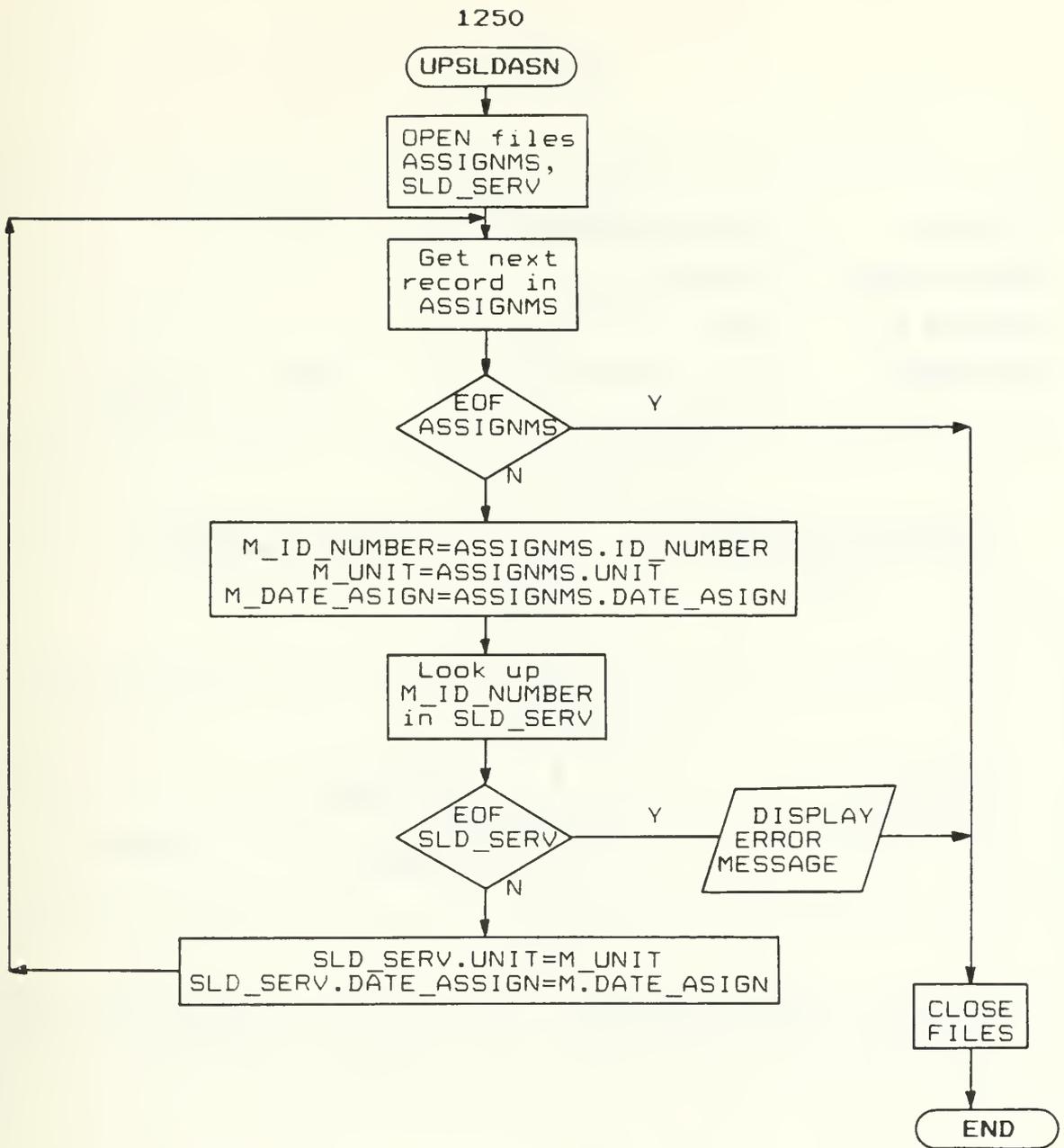
Figure D.3.15a   The flowchart of program UPSLDASN

```
+-----------------------------------------------------------------------+
|                                                                       |
|  SYSTEM       : PERSONNEL MANAGEMENT                                   |
|  MODULE NAME : PR_ASIGN                                                |
|  MODULE No    : 1260                                                   |
|  DESIGNER     : Labros Karatasios         DATE : 4/30/1987            |
|                                                                       |
+-----------------------------------------------------------------------+
|                                                                       |
|    +----------------------------+   +----------------------------+    |
|    | INVOKED BY MODULE :        |   | INVOKES MODULES :          |    |
|    |      1200                  |   |                            |    |
|    |                            |   |                            |    |
|    +----------------------------+   +----------------------------+    |
|                                                                       |
|                                                                       |
|    +----------------------------+   +----------------------------+    |
|    | INPUTS :                   |   | OUTPUTS :                  |    |
|    | ASSIGNMS                   |   | Printed list of assign-    |    |
|    |                            |   | ments                      |    |
|    |                            |   |                            |    |
|    +----------------------------+   +----------------------------+    |
|                                                                       |
|    +-------------------------------------------------------------+    |
|    | PROCESS :  See  Flowchart  in  Figure  D.3.16.a            |    |
|    +-------------------------------------------------------------+    |
|                                                                       |
+-----------------------------------------------------------------------+
```

Figure D.3.16   The IPO chart of program PR_ASIGN

1260
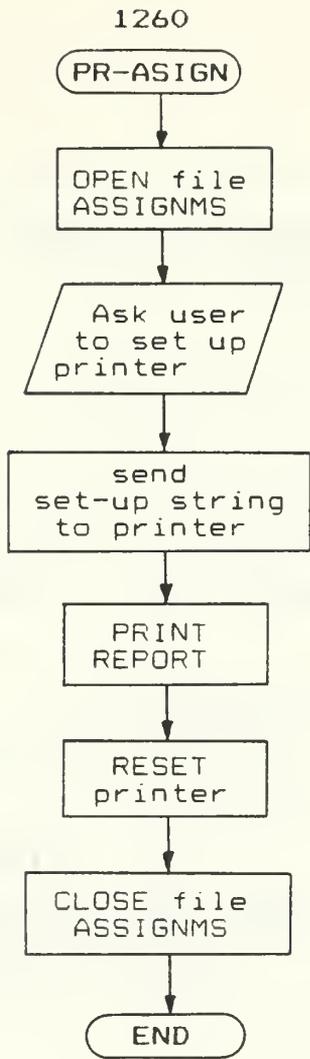
Figure D.3.16a The flowchart of program PR-ASIGN

143

```
SYSTEM       : PERSONNEL MANAGEMENT
MODULE NAME  : ENL_SLDS
MODULE No    : 1300
DESIGNER     : Labros Karatasios        DATE : 4/30/1987
```

```
INVOKED BY MODULE :              INVOKES MODULES :

      1000                           1310, 1320
```

```
INPUTS :                         OUTPUTS :
Manual ENLISTED file,            ENLISTED, SLD_ADDR,
ENLISTED                         SLD_SERV, SLD_TRAN,
                                 SLD_PREF
```

```
PROCESS :  See  Flowchart  in  Figure  D.3.17.a
```

Figure D.3.17   The IPO chart of program ENL_SLDS
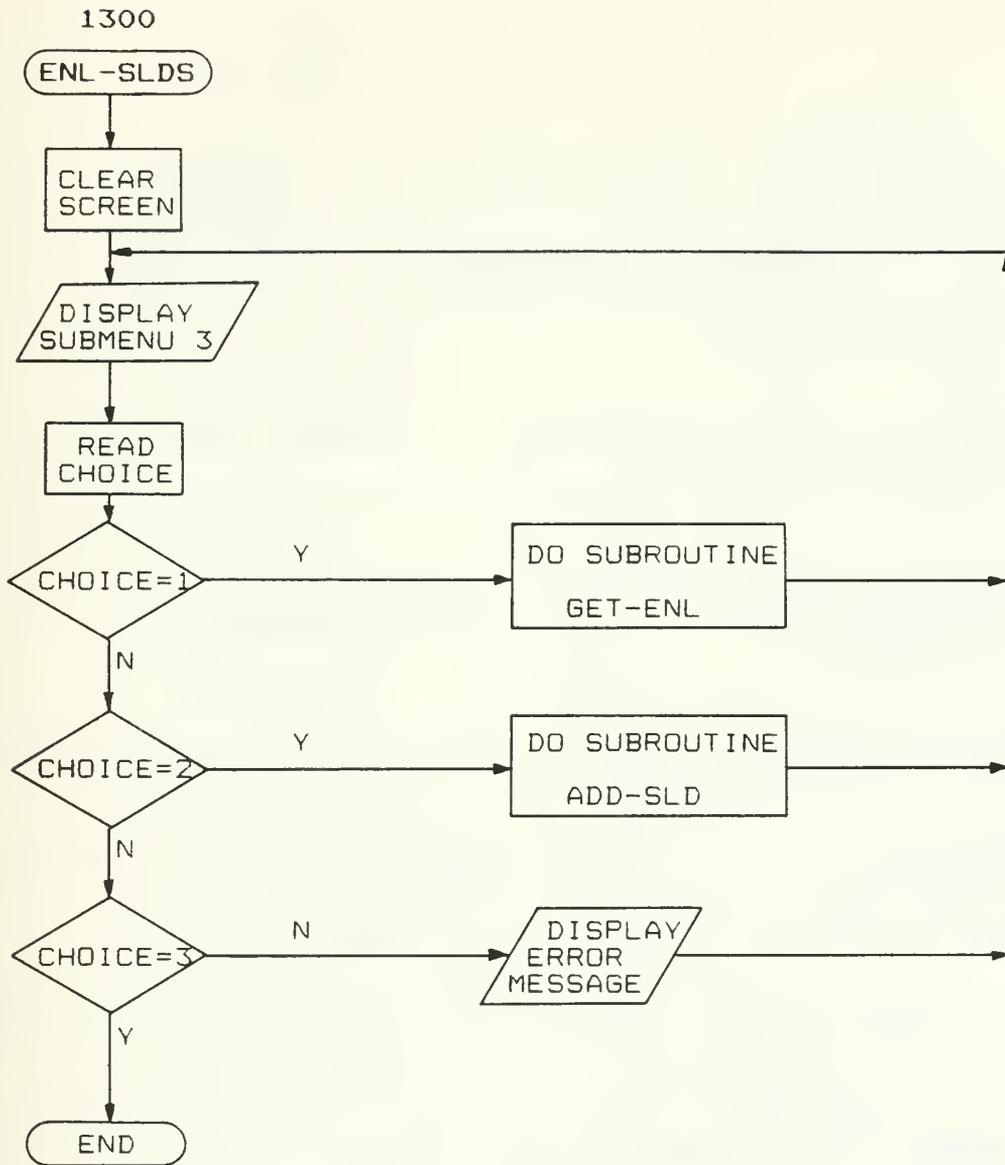
Figure D.3.17.a   The flowchart of program ENL-SLDS

145

```
┌─────────────────────────────────────────────────────────────────┐
│                                                                   │
│  SYSTEM      : PERSONNEL MANAGEMENT                                │
│  MODULE NAME : GET_ENL                                             │
│  MODULE No   : 1310                                                │
│  DESIGNER    : Labros Karatasios        DATE : 4/30/1987           │
│                                                                   │
├─────────────────────────────────────────────────────────────────┤
│                                                                   │
│    ┌─────────────────────────┐      ┌─────────────────────────┐   │
│    │ INVOKED BY MODULE :      │      │ INVOKES MODULES :       │   │
│    │      1300                │      │                         │   │
│    │                          │      │                         │   │
│    └─────────────────────────┘      └─────────────────────────┘   │
│                                                                   │
│    ┌─────────────────────────┐      ┌─────────────────────────┐   │
│    │ INPUTS :                 │      │ OUTPUTS :               │   │
│    │ Manual ENLISTED file     │      │ ENLISTED                │   │
│    │                          │      │                         │   │
│    └─────────────────────────┘      └─────────────────────────┘   │
│                                                                   │
│    ┌──────────────────────────────────────────────────────────┐  │
│    │ PROCESS :   See  Flowchart  in  Figure   D.3.18.a         │  │
│    └──────────────────────────────────────────────────────────┘  │
│                                                                   │
└─────────────────────────────────────────────────────────────────┘
```

Figure D.3.18   The IPO chart of program GET_ENL
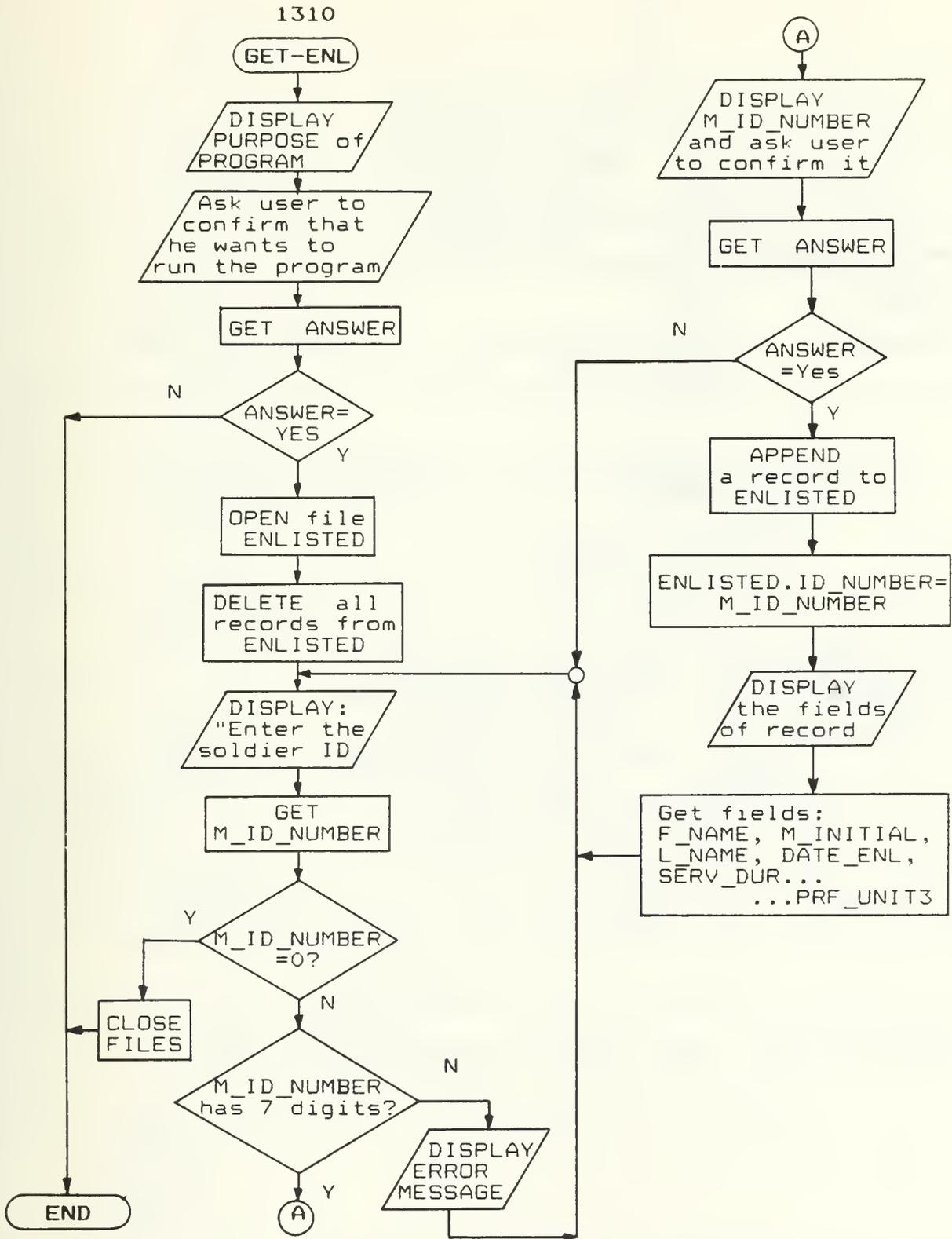
Figure D.3.18a   The flowchart of program GET-ENL

147

```
SYSTEM        : PERSONNEL MANAGEMENT
MODULE NAME : ADD_SLD
MODULE No    : 1320
DESIGNER      : Labros Karatasios        DATE : 4/30/1987
```

```
INVOKED BY MODULE :          INVOKES MODULES :

     1300


INPUTS :                     OUTPUTS :
ENLISTED                     SLD_ADDR, SLD_SERV,
                             SLD_TRAN, SLD_PREF


PROCESS :  See  Flowchart  in  Figure  D.3.19.a
```

Figure D.3.19   The IPO chart of program ADD_SLD
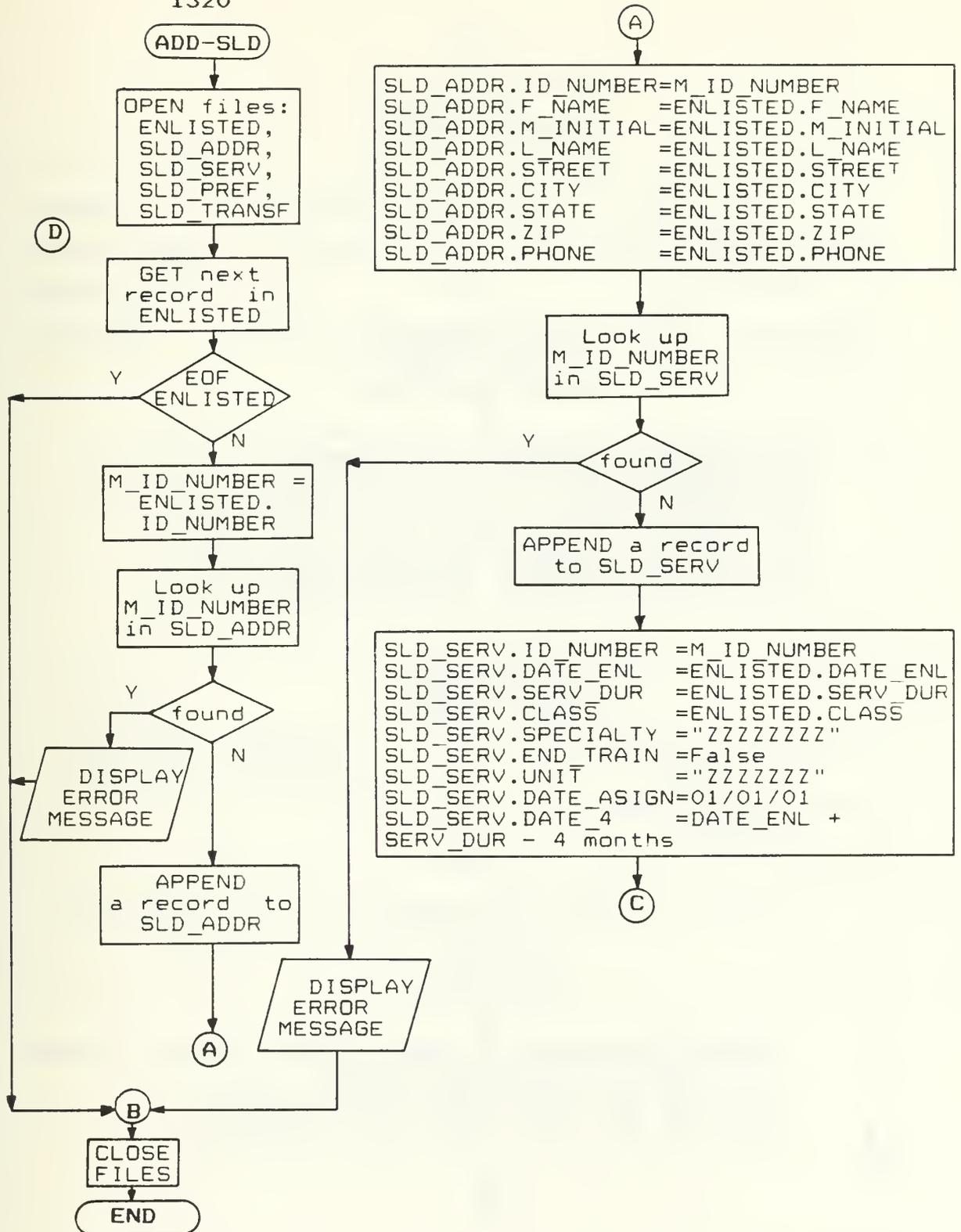
Figure D.3.19a   The flowchart of program ADD-SLD (part 1 of 2)

149
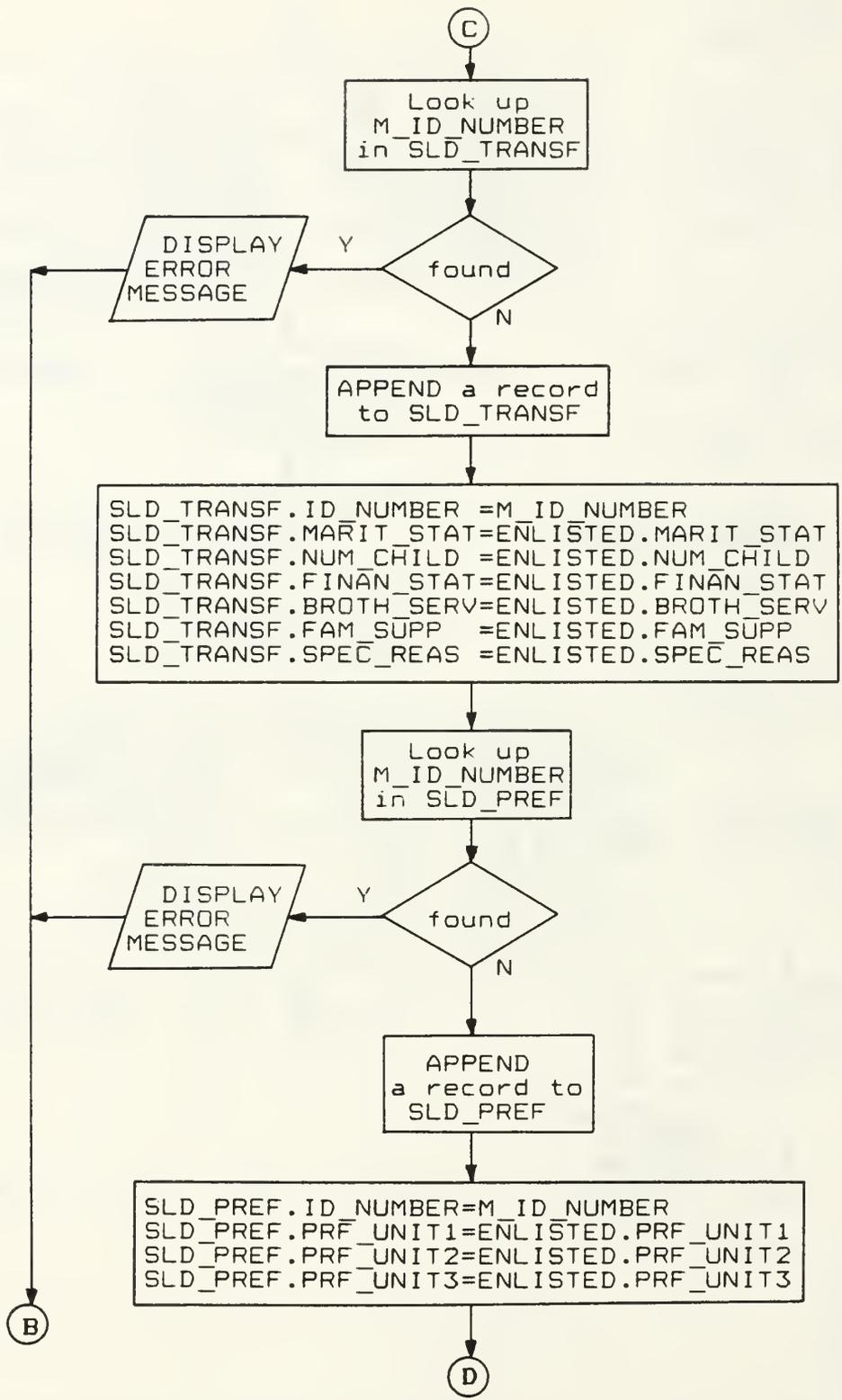
Figure D.3.19a   The flowchart of program ADD-SLD (part 2 of 2)

```
┌─────────────────────────────────────────────────────────────────────┐
│                                                                       │
│  SYSTEM        : PERSONNEL MANAGEMENT                                  │
│                                                                       │
│  MODULE NAME : TRAINING                                               │
│                                                                       │
│  MODULE No    : 1400                                                  │
│                                                                       │
│  DESIGNER     : Labros Karatasios        DATE : 4/30/1987            │
│                                                                       │
├─────────────────────────────────────────────────────────────────────┤
│                                                                       │
│   ┌─────────────────────────────┐   ┌─────────────────────────────┐  │
│   │  INVOKED BY MODULE :        │   │  INVOKES MODULES :          │  │
│   │       1000                  │   │       1410, 1420            │  │
│   │                             │   │                             │  │
│   └─────────────────────────────┘   └─────────────────────────────┘  │
│                                                                       │
│   ┌─────────────────────────────┐   ┌─────────────────────────────┐  │
│   │  INPUTS :                   │   │  OUTPUTS :                  │  │
│   │  ENLISTED, SLD_SERV,        │   │  TRAIN_REQ, Printed list    │  │
│   │  UNIT_ORG, SPECS,           │   │  with training needs        │  │
│   │  TRAINEES, TRAIN_REQ        │   │                             │  │
│   └─────────────────────────────┘   └─────────────────────────────┘  │
│                                                                       │
│   ┌───────────────────────────────────────────────────────────────┐  │
│   │  PROCESS :  See  Flowchart  in  Figure  D.3.20.a              │  │
│   └───────────────────────────────────────────────────────────────┘  │
│                                                                       │
└─────────────────────────────────────────────────────────────────────┘
```

Figure D.3.20   The IPO chart of program TRAINING
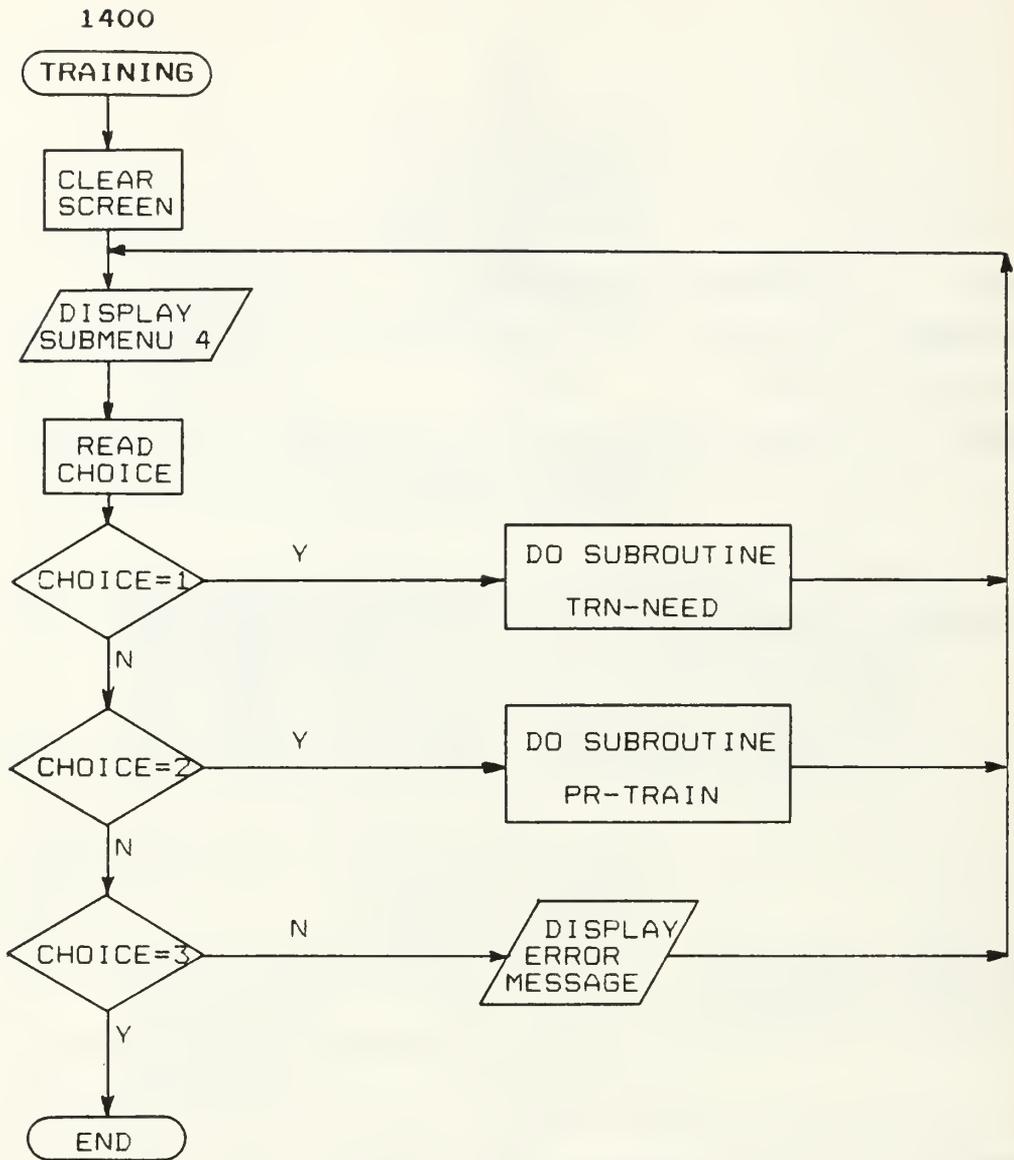
Figure D.3.20.a   The flowchart of program TRAINING

```
┌─────────────────────────────────────────────────────────────────────────┐
│                                                                           │
│   SYSTEM        : PERSONNEL MANAGEMENT                                     │
│                                                                           │
│   MODULE NAME : TRN_NEED                                                   │
│                                                                           │
│   MODULE No    : 1410                                                      │
│                                                                           │
│   DESIGNER     : Labros Karatasios         DATE : 4/30/1987               │
│                                                                           │
├─────────────────────────────────────────────────────────────────────────┤
│                                                                           │
│   ┌───────────────────────────────┐   ┌─────────────────────────────┐   │
│   │ INVOKED BY MODULE :           │   │ INVOKES MODULES :           │   │
│   │       1400                    │   │                             │   │
│   │                               │   │                             │   │
│   │                               │   │                             │   │
│   └───────────────────────────────┘   └─────────────────────────────┘   │
│                                                                           │
│   ┌───────────────────────────────┐   ┌─────────────────────────────┐   │
│   │ INPUTS :                      │   │ OUTPUTS :                   │   │
│   │ ENLISTED, SLD_SERV,           │   │ TRAIN_REQ                   │   │
│   │ UNIT_ORG, SPECS,              │   │                             │   │
│   │ TRAINEES                      │   │                             │   │
│   └───────────────────────────────┘   └─────────────────────────────┘   │
│                                                                           │
│   ┌───────────────────────────────────────────────────────────────┐     │
│   │ PROCESS :  See  Flowchart  in  Figure  D.3.21.a               │     │
│   └───────────────────────────────────────────────────────────────┘     │
│                                                                           │
└─────────────────────────────────────────────────────────────────────────┘
```

Figure D.3.21   The IPO chart of program TRN_NEED

153
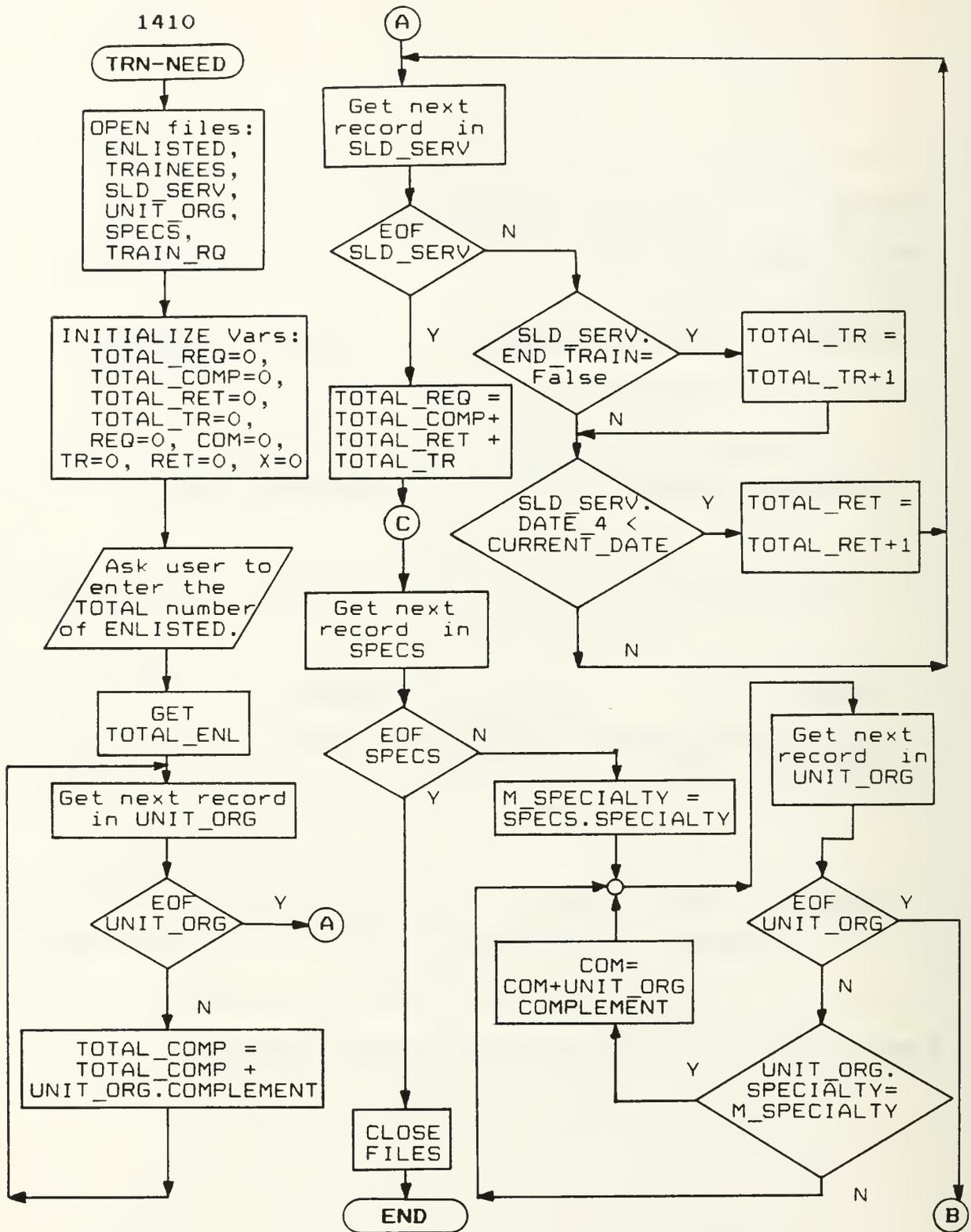
Figure D.3.21a   The flowchart of program TRN-NEED (part 1 of 2)

154

Figure D.3.21a   The flowchart of program TRN-NEED (part 2 of 2)

155

```
SYSTEM       : PERSONNEL MANAGEMENT

MODULE NAME : PR_TRAIN

MODULE No    : 1420

DESIGNER     : Labros Karatasios        DATE : 4/30/1987
```

```
INVOKED BY MODULE :              INVOKES MODULES :
        1400
```

```
INPUTS :                         OUTPUTS :
TRAIN_REQ                        Printed list with
                                 with training needs
```

```
PROCESS :  See  Flowchart  in  Figure  D.3.22.a
```
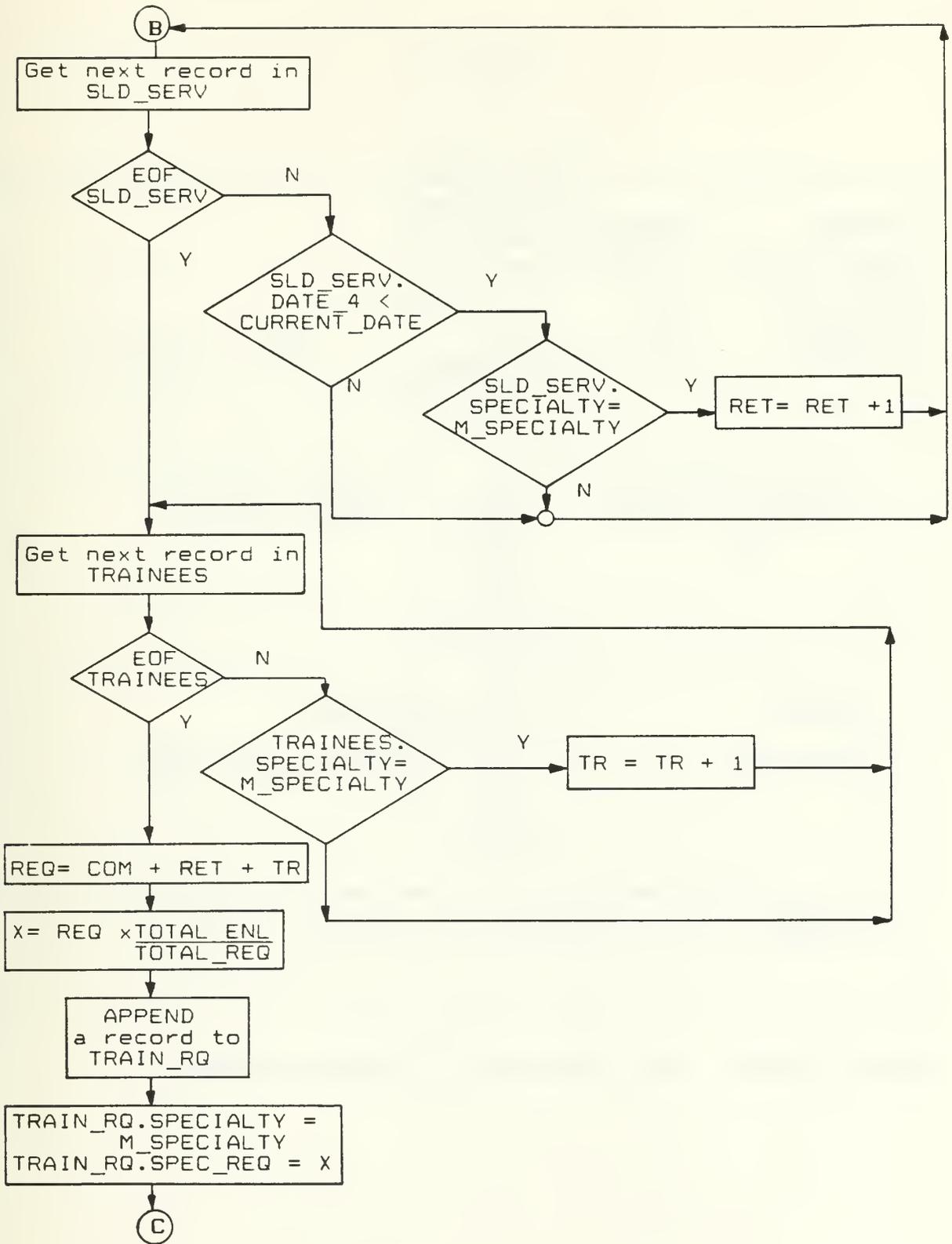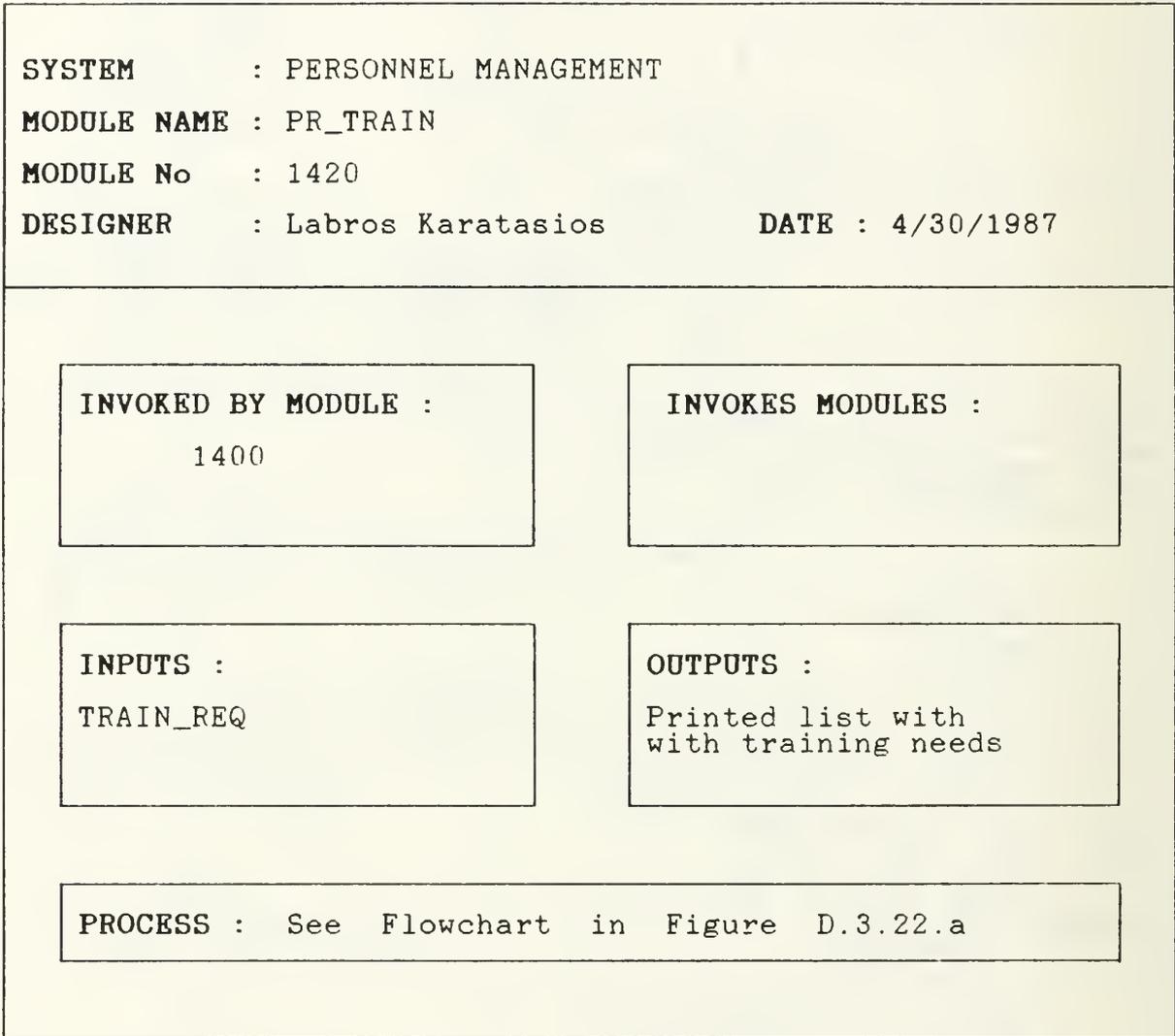
Figure D.3.22   The IPO chart of program PR_TRAIN

1420

```
   ( PR-TRAIN )
        │
        ▼
┌─────────────────┐
│  OPEN file      │
│  TRAIN_RQ       │
└─────────────────┘
        │
        ▼
   ╱─────────────╱
  ╱  Ask user   ╱
 ╱  to set-up  ╱
╱   printer   ╱
────────────
        │
        ▼
┌─────────────────┐
│  SEND set-up    │
│  string to      │
│  printer        │
└─────────────────┘
        │
        ▼
┌─────────────────┐
│  PRINT          │
│  REPORT         │
└─────────────────┘
        │
        ▼
┌─────────────────┐
│  RESET          │
│  printer        │
└─────────────────┘
        │
        ▼
┌─────────────────┐
│  CLOSE file     │
│  TRAIN_RQ       │
└─────────────────┘
        │
        ▼
    ( END )
```
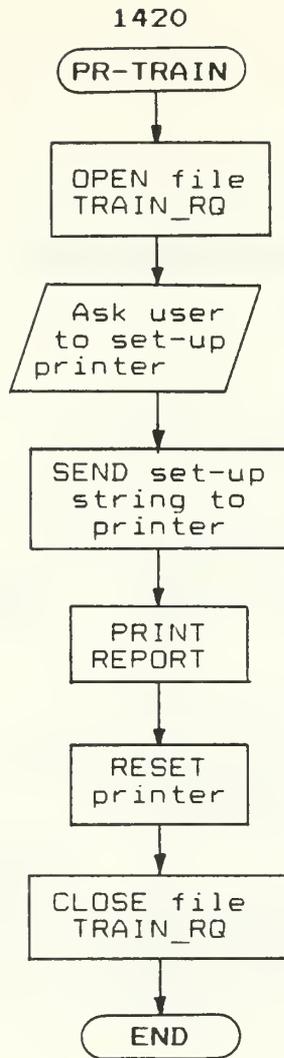
Figure D.3.22a   The flowchart of program PR-TRAIN

157

```
SYSTEM        : PERSONNEL MANAGEMENT
MODULE NAME : STC_REPS
MODULE No   : 1500
DESIGNER     : Labros Karatasios        DATE : 4/30/1987
```

```
INVOKED BY MODULE :                INVOKES MODULES :

     1000                              1510, 1520
```

```
INPUTS :                          OUTPUTS :

Manual list of trained            COMPL_TR, SLD_SERV,
soldiers, COMPL_TR,               TRAINEES
Manual list of trainees
```

```
PROCESS :   See   Flowchart   in   Figure   D.3.23.a
```

Figure D.3.23   The IPO chart of program STC_REPS
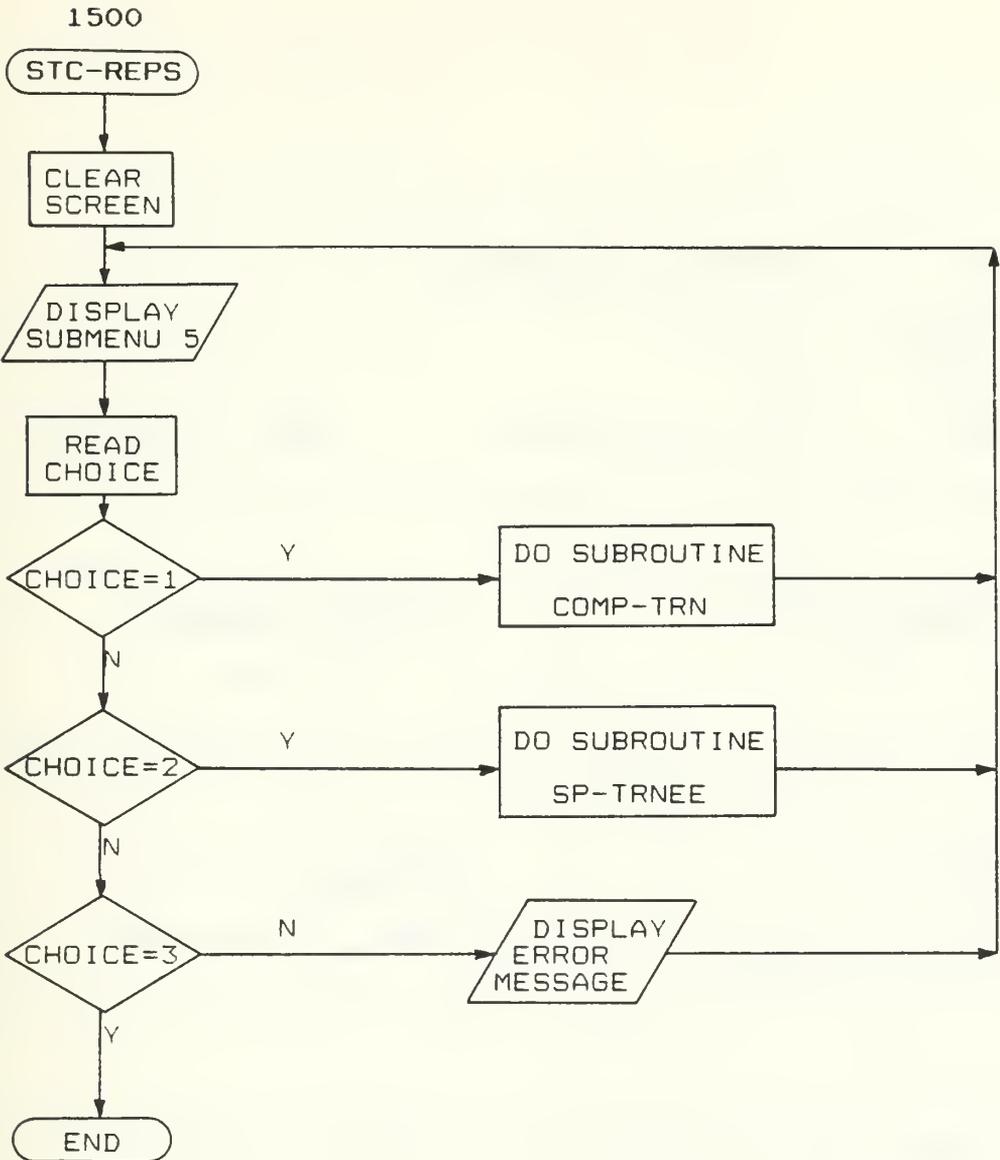
Figure D.3.23.a   The flowchart of program STC-REPS

```
SYSTEM        : PERSONNEL MANAGEMENT
MODULE NAME : COMP_TRN
MODULE No    : 1510
DESIGNER     : Labros Karatasios        DATE : 4/30/1987
```

```
INVOKED BY MODULE :               INVOKES MODULES :
     1500                              1511, 1512


INPUTS :                          OUTPUTS :
Manual list of trained            COMPL_TR, SLD_SERV
soldiers, COMPL_TR


PROCESS :  See  Flowchart  in  Figure  D.3.24.a
```
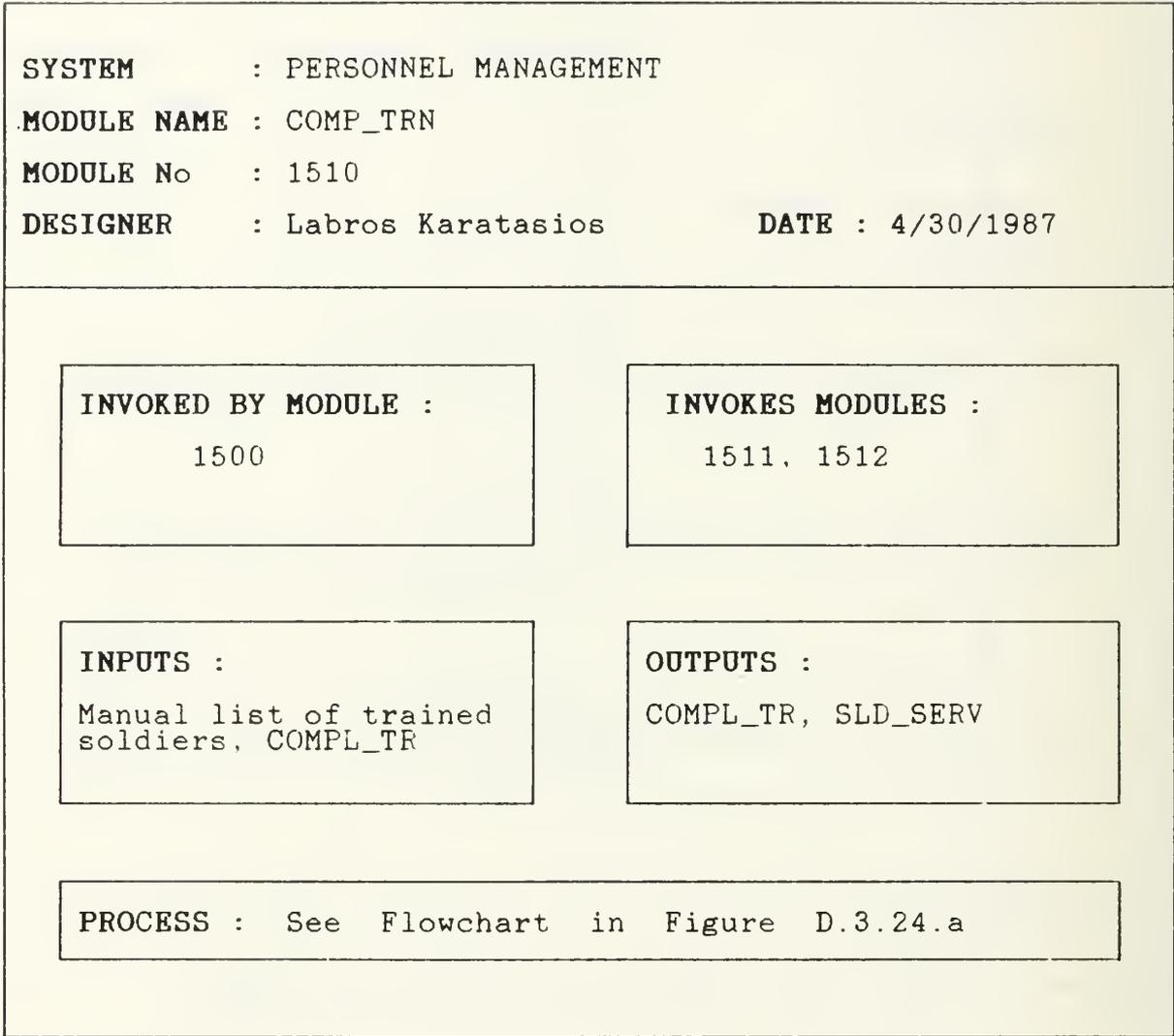
Figure D.3.24   The IPO chart of program COMP_TRN
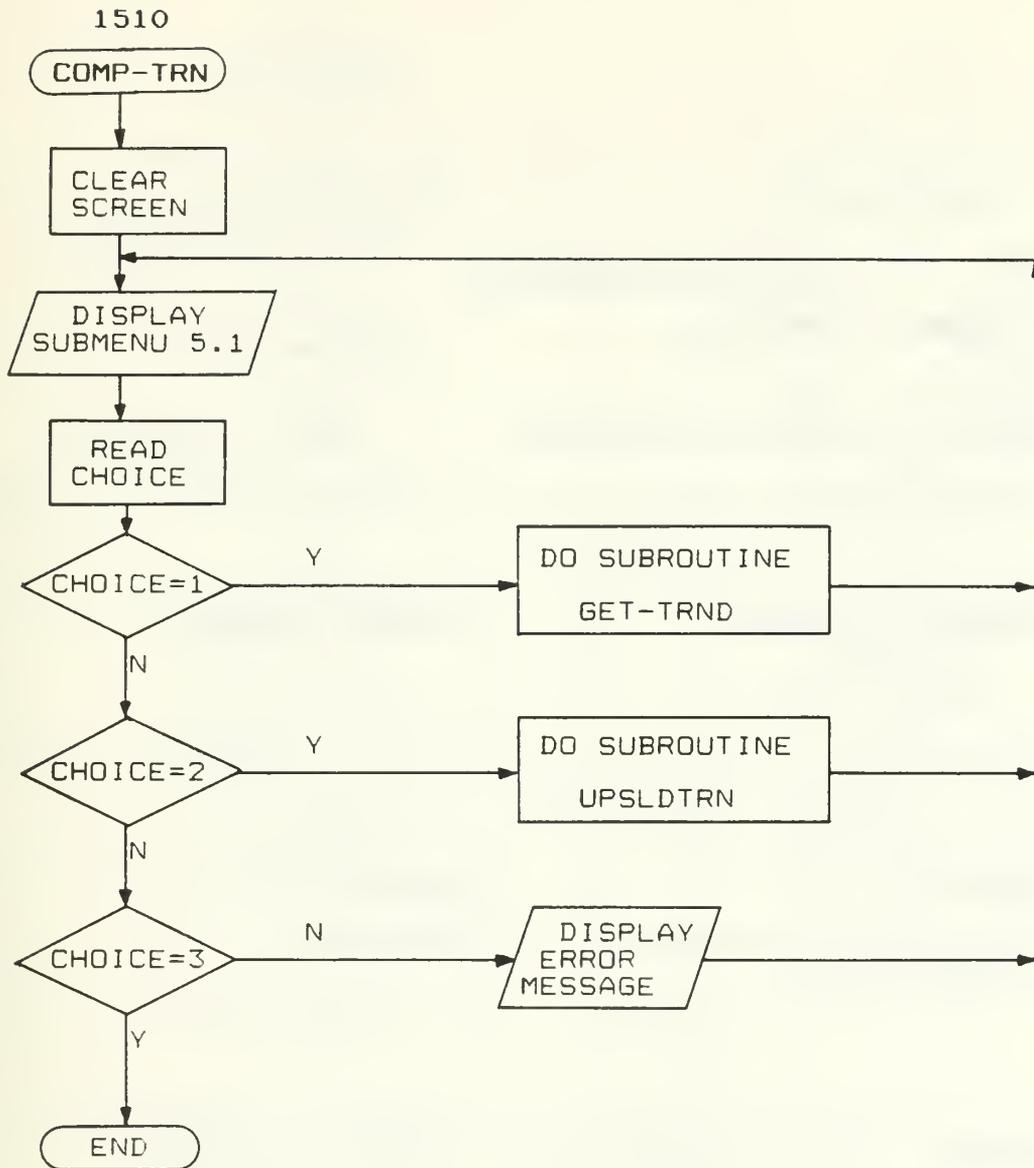
Figure D.3.24.a   The flowchart of program COMP-TRN

```
+------------------------------------------------------------------+
|                                                                  |
|  SYSTEM       : PERSONNEL MANAGEMENT                              |
|  MODULE NAME  : GET_TRND                                          |
|  MODULE No    : 1511                                             |
|  DESIGNER     : Labros Karatasios        DATE : 4/30/1987        |
|                                                                  |
+------------------------------------------------------------------+
|                                                                  |
|    +-----------------------------+   +------------------------+  |
|    | INVOKED BY MODULE :         |   | INVOKES MODULES :      |  |
|    |       1510                  |   |                        |  |
|    |                             |   |                        |  |
|    |                             |   |                        |  |
|    +-----------------------------+   +------------------------+  |
|                                                                  |
|    +-----------------------------+   +------------------------+  |
|    | INPUTS :                    |   | OUTPUTS :              |  |
|    | Manual list of trained      |   | COMPL_TR               |  |
|    | soldiers                    |   |                        |  |
|    |                             |   |                        |  |
|    +-----------------------------+   +------------------------+  |
|                                                                  |
|    +------------------------------------------------------------+|
|    | PROCESS :  See  Flowchart  in  Figure  D.3.25.a           ||
|    +------------------------------------------------------------+|
|                                                                  |
+------------------------------------------------------------------+
```
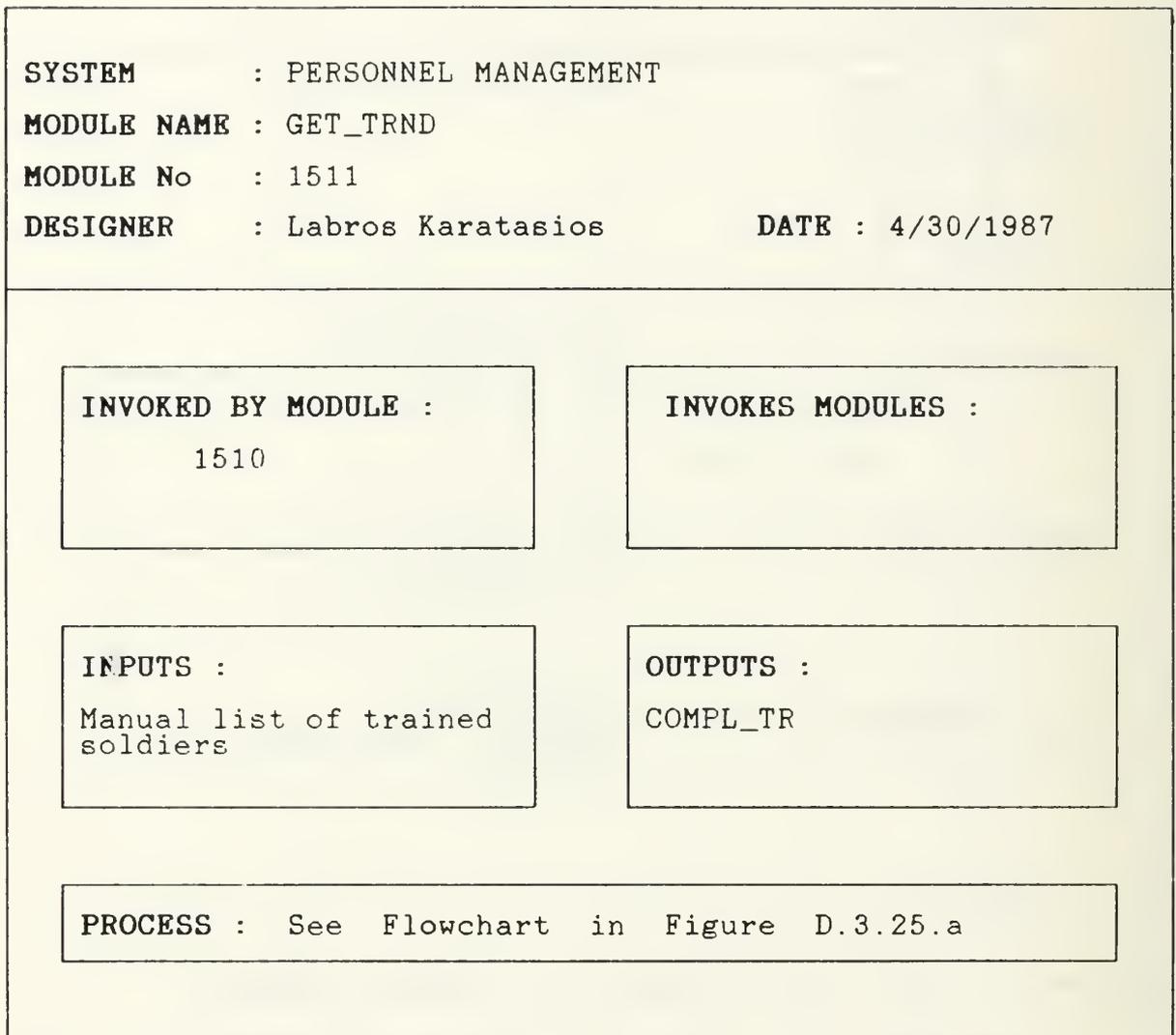
Figure D.3.25   The IPO chart of program GET_TRND

162
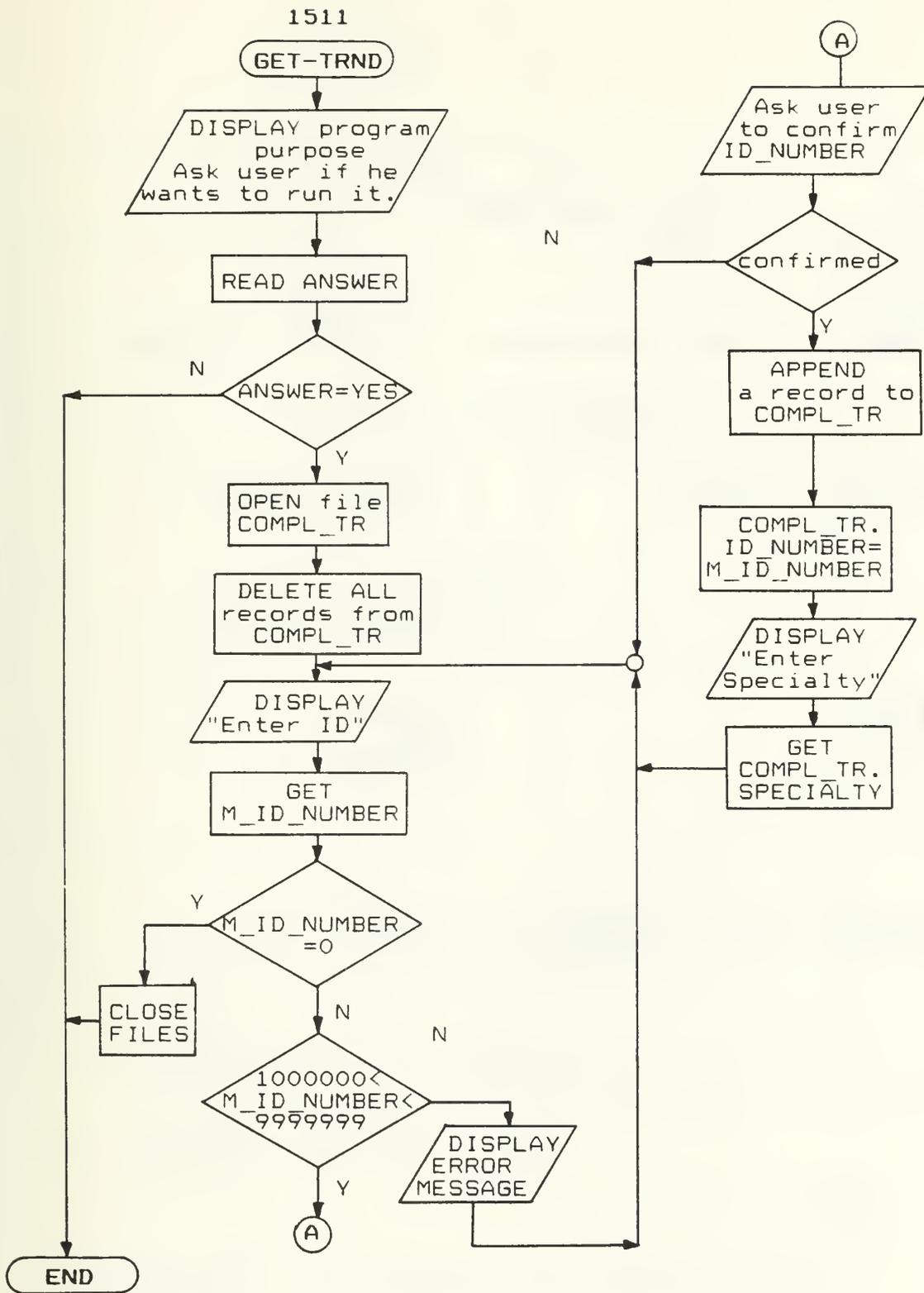
Figure D.3.25a   The flowchart of program GET-TRND

163

```
SYSTEM      : PERSONNEL MANAGEMENT
MODULE NAME : UPSLDTRN
MODULE No   : 1512
DESIGNER    : Labros Karatasios        DATE : 4/30/1987
```

```
INVOKED BY MODULE :              INVOKES MODULES :
      1510



INPUTS :                         OUTPUTS :
COMPL_TR                         SLD_SERV



PROCESS :  See  Flowchart  in  Figure  D.3.26.a
```

Figure D.3.26   The IPO chart of program UPSLDTRN
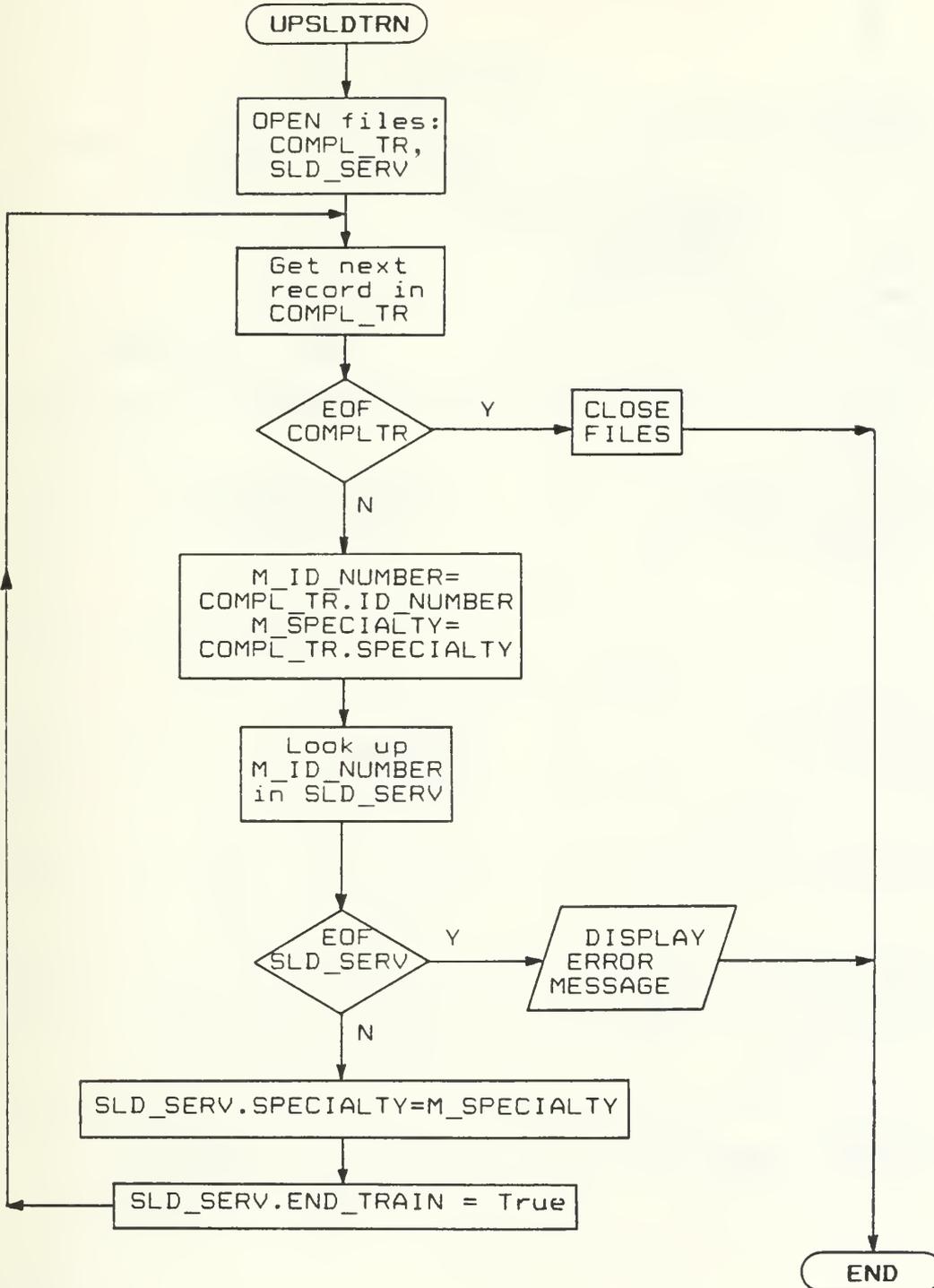
Figure D.3.26a   The flowchart of program   UPSLDTRN

```
┌─────────────────────────────────────────────────────────────────┐
│                                                                   │
│  SYSTEM      : PERSONNEL MANAGEMENT                                │
│  MODULE NAME : SP_TRNEE                                            │
│  MODULE No   : 1520                                               │
│  DESIGNER    : Labros Karatasios        DATE : 4/30/1987          │
│                                                                   │
├─────────────────────────────────────────────────────────────────┤
│                                                                   │
│    ┌──────────────────────────┐    ┌──────────────────────────┐   │
│    │ INVOKED BY MODULE :      │    │ INVOKES MODULES :        │   │
│    │      1500                │    │                          │   │
│    │                          │    │                          │   │
│    └──────────────────────────┘    └──────────────────────────┘   │
│                                                                   │
│    ┌──────────────────────────┐    ┌──────────────────────────┐   │
│    │ INPUTS :                 │    │ OUTPUTS :                │   │
│    │ Manual list of trainees  │    │ TRAINEES                 │   │
│    │                          │    │                          │   │
│    └──────────────────────────┘    └──────────────────────────┘   │
│                                                                   │
│    ┌─────────────────────────────────────────────────────────┐   │
│    │ PROCESS :  See  Flowchart  in  Figure  D.3.27.a          │   │
│    └─────────────────────────────────────────────────────────┘   │
│                                                                   │
└─────────────────────────────────────────────────────────────────┘
```
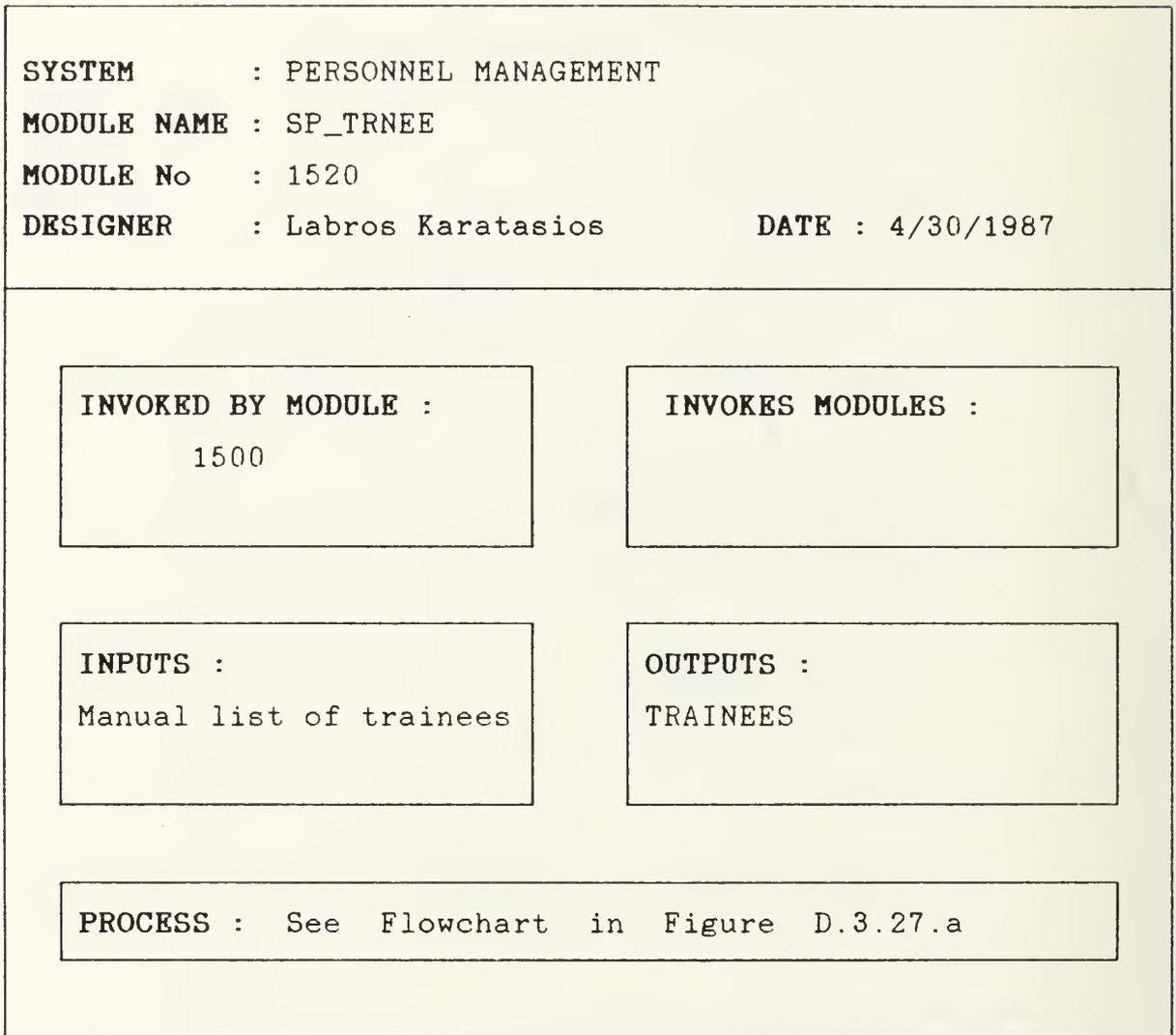
Figure D.3.27    The IPO chart of program SP_TRNEE
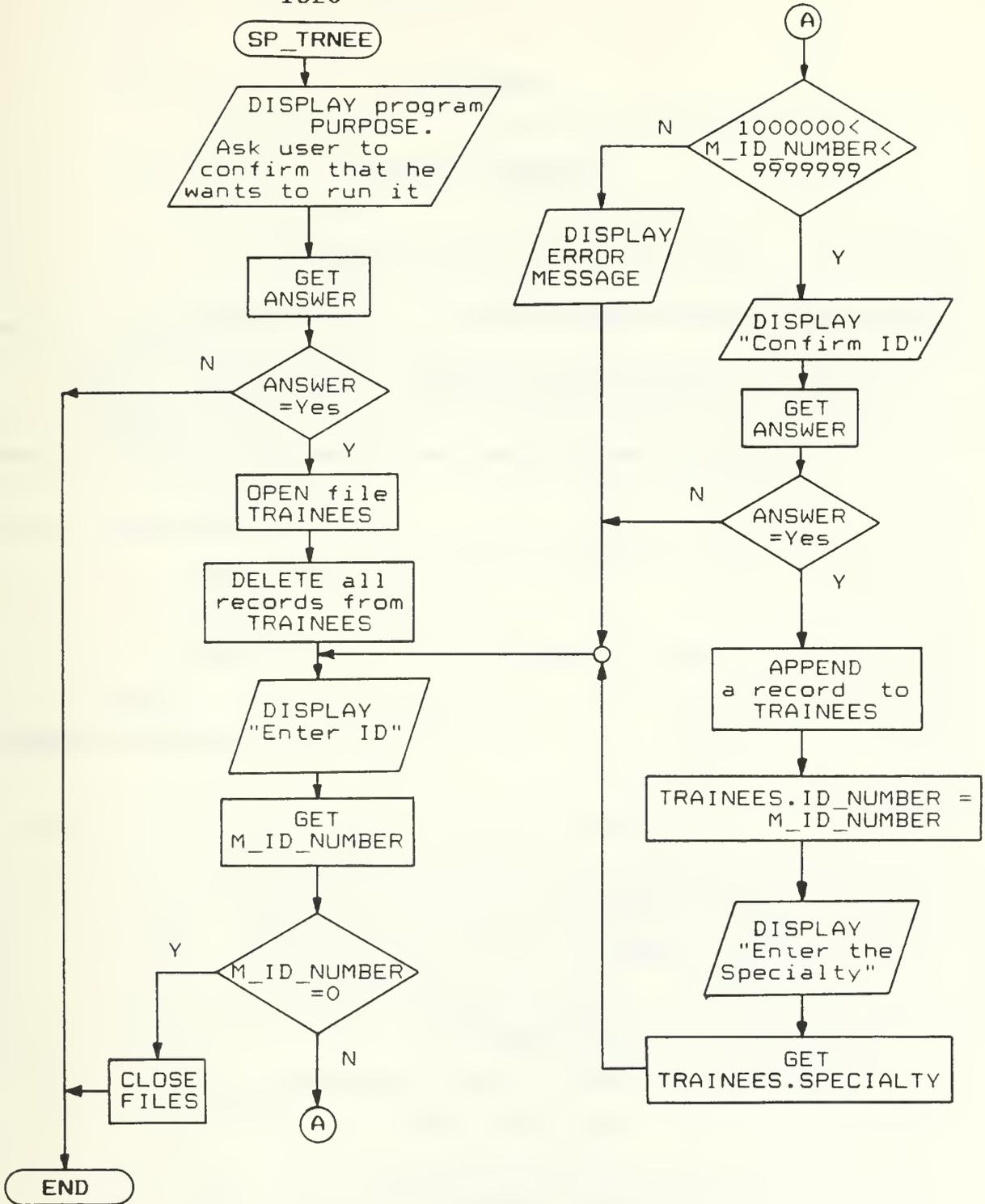
Figure D.3.27a   The flowchart of program SP-TRNEE

167

# APPENDIX E

## PROGRAM LISTINGS

### Section E.1  The listing of program PERS-MGT

```
******************************************************************
*                                                                *
*   PERS-MGT :   Main control program.   It displays   a menu    *
*                screen   and depending on the user's choice      *
*   5/12/87      it gives control to one of five processes.      *
*                                                                *
******************************************************************

CLEAR                                        &&  Clear the screen

* Initialize basic dBASE III Plus functions

CLEAR ALL

SET TALK OFF

SET BELL OFF

STORE " " TO CHOICE                && Initialize variable CHOICE

* Main loop

DO WHILE .T.

    * Display Main Menu

    @  2,27  SAY  "PERSONNEL MANAGEMENT SYSTEM"

    @  3,38  SAY  "MENU"

    @  6,20  SAY  "1.   Units Reports"

    @  8,20  SAY  "2.   Assignments"

    @ 10,20  SAY  "3.   New Enlisted Soldiers"

    @ 12,20  SAY  "4.   Training needs"

    @ 14,20  SAY  "5.   STC  Reports"

    @ 16,20  SAY  "6.   Exit Program"

    @ 20,20  SAY  "Your selection, please "  GET CHOICE

    READ
```

```
    * Execute selected process
    DO CASE
        CASE CHOICE = "1"
            DO UNIT-REP          && Process Units Reports
        CASE CHOICE = "2"
            DO ASSIGNMT          && Process Soldiers Assignments
        CASE CHOICE = "3"
            DO ENL-SLDS          && Process New Enlisted Soldiers
        CASE CHOICE = "4"
            DO TRAINING          && Estimate Training Needs
        CASE CHOICE = "5"
            DO STC-REPS          && Process STC Reports
        CASE CHOICE = "6"
            CLEAR ALL
            CLEAR
            RETURN               && Exit program
    ENDCASE
ENDDO
```

Section E.2 The listing of program ENL-SLDS

```
******************************************************************
*                                                                *
*  ENL-SLDS :  Program to enter the report for new enlisted      *
*              soldiers into the system  and to update  the      *
*  5/12/87    soldier files.                                     *
*                                                                *
******************************************************************
CLEAR                                    && Clear the screen
STORE " "  TO EC
DO WHILE .T.
    @  2,24  SAY  "ENLISTED SOLDIERS PROCESSING"
    @  3,38  SAY  "MENU"
    @  6,20  SAY  "1.  Input new enlisted data"
```

169

```
        @  8,20   SAY   "2.   Process new data"
        @ 10.20   SAY   "3.   Return to main menu"
        @ 14,20   SAY   "Your selection, please"
     GET EC
     READ
     DO CASE
        CASE EC = "1"
             DO GET-ENL
        CASE EC = "2"
             DO ADD-SLD
        CASE EC = "3"
             CLEAR ALL
             CLEAR
             RETURN
        OTHERWISE
             * Display error message
             CLEAR
             @ 15,15   SAY "Your selection must be  1, 2 or 3 "
             CLEAR ALL
             RETURN
     ENDCASE
ENDDO
```

Section E.3   The listing of program GET-ENL

```
********************************************************************
*                                                                  *
*   GET-ENL :   This program creates the ENLISTED file  and        *
*               updates it interactively with the data from        *
*   5/12/87      the EC report.                                     *
*                                                                  *
********************************************************************
CLEAR
CLEAR ALL
```

170

```
SET TALK OFF
SET BELL OFF
* Define what the program does
@ 4,5 SAY "This program  allows you to put  new soldiers into"
@ 6,5 SAY "the system.  Note: This program  also deletes  the"
@ 8,5 SAY "last group of soldiers that were put in the system"
STORE " " TO C
@ 10,5 SAY "Type Y to proceed, anything else to abort."
GET C PICTURE "!"
READ
CLEAR
IF C <> "Y"
   RETURN
ENDIF
RUN DEL ENLISTED.DBF
RUN COPY TE.DBF ENLISTED.DBF
USE ENLISTED
DO WHILE .T.
   CLEAR
   @ 2,5 SAY "Enter 0 to exit"
   STORE SPACE(7) TO MID_NUMBER
   @ 4,5 SAY "Enter ID number "
   GET MID_NUMBER
   READ
   DO CASE
      CASE VAL(MID_NUMBER) = 0
           RETURN
      CASE VAL(MID_NUMBER) < 1000000
           @ 7,5 SAY "Invalid ID number"
           DO WHILE VAL(MID_NUMBER) < 1000000
               STORE SPACE(7) TO MID_NUMBER
```

171

```
                    STORE SPACE(36) TO CL
                    @ 4,5 SAY "Enter ID number"
                    GET MID_NUMBER
                    READ
                    IF VAL(MID_NUMBER) = 0
                        RETURN
                    ENDIF
                ENDDO
            ENDCASE
            STORE " " TO CONF
            @ 6,5 SAY "Please confirm the above number (Y to confirm)"
            GET CONF  PICTURE "!"
            READ
            IF CONF <> "Y"
                LOOP
            ENDIF
            STORE SPACE(20) TO MCITY, ML_NAME
            STORE SPACE(15) TO MF_NAME, MSTREET, MSTATE
            STORE SPACE(14) TO MPHONE
            STORE SPACE(7)  TO MPRF_UNIT1, MPRF_UNIT2, MPRF_UNIT3
            STORE SPACE(5)  TO MZIP
            STORE SPACE(3)  TO MCLASS
            STORE SPACE(1)  TO MM_INITIAL, MMARIT_STAT, MFINAN_STAT
            STORE " " TO MFAM_SUPP, MSPEC_REAS
            STORE CTOD("  /  /  ") TO MDATE_ENL
            STORE 0 TO MSERV_DUR, MNUM_CHILD, MBROTH_SERV
            CLEAR
            @ 1,24 SAY "Entering new soldier into system"
            @ 2,34 SAY "ID number:"+MID_NUMBER
            @ 4,5  SAY "First Name "  GET MF_NAME     PICTURE "a"
            @ 4,32 SAY "M. Initial "  GET MM_INITIAL  PICTURE "!"
```

```
@ 4,47 SAY "Last Name "    GET ML_NAME      PICTURE "a"
@ 6,5   SAY "Street "      GET MSTREET      PICTURE "a"
@ 6,30 SAY "City "         GET MCITY        PICTURE "a"
@ 8,5   SAY "State "       GET MSTATE       PICTURE "a"
@ 8,28 SAY "ZIP "          GET MZIP         PICTURE "a"
@ 8,39 SAY "Phone "        GET MPHONE
@ 10,5 SAY "Date entered Service "    GET MDATE_ENL
@ 10,37 SAY "Number of months of service "
GET MSERV_DUR  PICTURE "99"
@ 10,69 SAY "Class " GET MCLASS  PICTURE "99!"
@ 12,5  SAY "Marital status: (D)ivorced, (M)arried, ";
        "(S)ingle, (W)idowed "  GET MMARIT_STAT PICTURE "!"
@ 14,5 SAY "Number of children " GET MNUM_CHILD PICTURE "9"
@ 14,30 SAY "Financial status: (G)ood, (M)edium, (B)ad "
        GET MFINAN_STAT  PICTURE "!"
@ 16,5 SAY "Family Supporter (T/F) "
        GET MFAM_SUPP   PICTURE "!"
@ 16,30 SAY "Number of brothers in service "
        GET MBROTH_SERV  PICTURE "9"
@ 18,5  SAY "Priority for transfer (T/F) "
        GET MSPEC_REAS   PICTURE "!"
@ 20,5  SAY "List unit preferences  #1: "
        GET MPRF_UNIT1   PICTURE "a"
@ 20,39 SAY "#2: "  GET MPRF_UNIT2   PICTURE "a"
@ 20,52 SAY "#3: "  GET MPRF_UNIT3   PICTURE "a"
READ
IF MFAM_SUPP = 'T'
   STORE .T. TO MFAM_SUPP
ELSE
   STORE .F. TO MFAM_SUPP
ENDIF
```

```
    IF MSPEC_REAS = 'T'
        STORE .T. TO MSPEC_REAS
    ELSE
        STORE .F. TO MSPEC_REAS
    ENDIF
    USE ENLISTED
    APPEND BLANK
    REPLACE ID_NUMBER WITH MID_NUMBER, F_NAME WITH MF_NAME
    REPLACE L_NAME WITH ML_NAME, M_INITIAL WITH MM_INITIAL
    REPLACE DATE_ENL WITH MDATE_ENL, CLASS WITH MCLASS
    REPLACE SERV_DUR WITH MSERV_DUR, MARIT_STAT WITH MMARIT_STAT
    REPLACE NUM_CHILD WITH MNUM_CHILD
    REPLACE FINAN_STAT WITH MFINAN_STAT
    REPLACE FAM_SUPP WITH MFAM_SUPP
    REPLACE BROTH_SERV WITH MBROTH_SERV
    REPLACE SPEC_REAS WITH MSPEC_REAS
    REPLACE PRF_UNIT1 WITH MPRF_UNIT1
    REPLACE PRF_UNIT2 WITH MPRF_UNIT2
    REPLACE PRF_UNIT3 WITH MPRF_UNIT3
    REPLACE STREET WITH MSTREET, CITY WITH MCITY
    REPLACE STATE WITH MSTATE, ZIP WITH MZIP, PHONE WITH MPHONE
    STORE " " TO DA
    @ 22,5 SAY "Do you want to enter another soldier? "
    GET DA PICTURE "!"
    READ
    IF DA = "Y"
        LOOP
    ELSE
        RETURN
    ENDIF
ENDDO
```

Section E.4 The listing of program ADD-SLD

```
CLEAR
CLEAR ALL
USE ENLISTED
GOTO TOP
DO WHILE .NOT. EOF()
    STORE ID_NUMBER TO MID_NUMBER
    USE SLD_ADDR  INDEX SLAD
    SEEK MID_NUMBER
    IF FOUND()
       @ 4,5 SAY "ID Number already exists in address file!!!"
       CLEAR ALL
       CLEAR
       RETURN
    ENDIF
    USE ENLISTED
    STORE F_NAME TO MF_NAME
    STORE M_INITIAL  TO MM_INITIAL
    STORE L_NAME TO ML_NAME
    STORE STREET TO MSTREET
    STORE CITY TO MCITY
    STORE STATE TO MSTATE
    STORE ZIP TO MZIP
    STORE PHONE TO MPHONE
    USE SLD_ADDR  INDEX SLAD
    APPEND BLANK
    REPLACE F_NAME WITH MF_NAME, M_INITIAL WITH MM_INITIAL
    REPLACE L_NAME WITH ML_NAME, STREET WITH MSTREET
    REPLACE CITY WITH MCITY, STATE WITH MSTATE, ZIP WITH MZIP
    REPLACE PHONE WITH MPHONE, ID_NUMBER WITH MID_NUMBER
```

```
USE SLD_SERV
SEEK MID_NUMBER
IF FOUND()
   @ 6,5 SAY "ID Number already exists in service file!!!"
   CLEAR ALL
   CLEAR
   RETURN
ENDIF
USE ENLISTED
STORE DATE_ENL TO MDATE_ENL
STORE SERV_DUR TO MSERV_DUR
STORE CLASS TO MCLASS
STORE SERV_DUR * 30   TO MLS
STORE MLS - 120   TO MLE
STORE MDATE_ENL + MLE   TO MDATE_4
USE SLD_SERV   INDEX SLSE
APPEND BLANK
REPLACE ID_NUMBER WITH MID_NUMBER, DATE_ENL WITH MDATE_ENL
REPLACE SERV_DUR WITH MSERV_DUR, CLASS WITH MCLASS
REPLACE DATE_4 WITH MDATE_4, END_TRAIN WITH .F.
USE SLD_PREF   INDEX SLPR
SEEK MID_NUMBER
IF FOUND()
   @ 8,5 SAY " ID Number already exists in preference file"
   CLEAR ALL
   CLEAR
   RETURN
ENDIF
USE ENLISTED
STORE PRF_UNIT1   TO   MPRF_UNIT1
STORE PRF_UNIT2   TO   MPRF_UNIT2
```

176

```
STORE PRF_UNIT3   TO   MPRF_UNIT3
USE SLD_PREF   INDEX SLPR
APPEND BLANK
REPLACE ID_NUMBER WITH MID_NUMBER, PRF_UNIT1 WITH MPRF_UNIT1
REPLACE PRF_UNIT2 WITH MPRF_UNIT2, PRF_UNIT3 WITH MPRF_UNIT3
USE SLD_TRAN   INDEX SLTR
SEEK MID_NUMBER
IF FOUND()
   @ 10,5 SAY "ID Number already exists in transfer file!!"
   CLEAR ALL
   CLEAR
   RETURN
ENDIF
USE ENLISTED
STORE MARIT_STAT TO MMARIT_STAT
STORE NUM_CHILD TO MNUM_CHILD
STORE FINAN_STAT TO MFINAN_STAT
STORE BROTH_SERV TO MBROTH_SERV
STORE FAM_SUPP TO MFAM_SUPP
STORE SPEC_REAS TO MSPEC_REAS
USE SLD_TRAN   INDEX SLTR
APPEND BLANK
REPLACE MARIT_STAT WITH MMARIT_STAT
REPLACE ID_NUMBER WITH MID_NUMBER, NUM_CHILD WITH MNUM_CHILD
REPLACE FINAN_STAT WITH MFINAN_STAT
REPLACE BROTH_SERV WITH MBROTH_SERV
REPLACE FAM_SUPP WITH MFAM_SUPP, SPEC_REAS WITH MSPEC_REAS
USE ENLISTED
IF .NOT. EOF()
   SKIP
ENDIF
```

```
        LOOP
ENDDO
USE ENLISTED
DELETE ALL
PACK
CLEAR ALL
RETURN
```

LIST OF REFERENCES

1. Naur, Peter and Randell, Brian, _Software Engineering, Report on a conference sponsored by the NATO Science Committee, Garmish, Germany, 7-11 October 1968_ NATO Scientific Affairs Division, Brussels, 1969.

2. Fairley, Richard, _Software Engineering Education: Status and Prospects_, Proceedings of the 12th Hawaii International Conference on System Sciences, Pt. I, pp. 140-146, Western Periodicals Ltd, North Hollywood, CA, 1979.

3. Yourdon, Edward, _Managing the Structured Techniques_, Yourdon Inc., New York, 1986.

4. Mc Clure, Carma, _Managing Software Development and Maintenance_, Litton Educational Publishing Inc., 1981.

5. U.S. Bureau of the Census, _Statistical abstract of the United States: 1979_, Washington, D.C., 1979.

6. Reifer, Donald, _Tutorial: Software Management_, IEEE Computer Society, 1984.

7. Clifton, David and Fyffe, _Project Feasibility Analysis: A Guide to Profitable New Ventures_, John Wiley and Sons, Inc., New York, 1977.

8. Fitzerald, J., Fitzerald and Stallings, _Fundamentals of Systems Analysis_, John Wiley and Sons Inc, New York, 1981.

9. Kroenke, David, _Database processing_, Science Research Associates, Inc., 1983.

10. Davis, William, Systems Analysis and Design, Addison
    Wesley Publishing Co., 1983.

11. Boehm, Barry, Software Engineering, IEEE Transactions on
    Computers, vol. C-25, no 12, December 1976.

12. De Marco, Tony, Structured Analysis and System Specifica-
    tion, Yourdon Press, New York, 1978.

13. ISDOS Project, PSL/PSA User's Reference Manual, University
    of Michigan, Ann Arbor, Michigan.

14. Connor, M., F., SADT. Structured Analysis and Design
    Technique Introduction, SofTech report 9595-7, SofTech
    Inc., Waltham, Mass, 1980.

15. Mullery, G., P., CORE. A method for controlled requirement
    specification, Proceedings of the 4th International Confe-
    rence on Software Engineering, pp. 126-35, 1979.

16. Alford, M, W, Software Requirements Engineering Methodolo-
    gy (SREM) at the age of four, Proceedings of the Inter-
    national Computer Software and Applications Conference,
    pp. 866-74, 1980.

17. Salter, K., G., A methodology for decomposing system
    requirements into data processing requirements, Procee-
    dings of the 2nd International Conference on Software
    Engineering, pp. 91-101, 1976.

18. Peterson, J., L., Petri Net Theory and the Modeling of
    Systems, Prentice Hall, Englewood Cliffs, N.J., 1981.

19. Jackson, M., System Development, Prentice Hall, Englewood
    Cliffs, N.J., 1983.

20. SSL 1982a, Introduction to SDS, Software Sciences Limited, Macclesfield, 1982.

21. Gane, C. and Sarson, T., Structured Systems Analysis: Tools and Techniques, Prentice Hall, Englewood Cliffs, N.J., 1979.

22. Everest, Gordon, Database Management, McGraw Hill, Inc., 1986.

23. Campbell, Sally, Microcomputer Software Design, Prentice Hall, Inc., 1984.

INITIAL DISTRIBUTION LIST

Thesis
K14275   Karatasios
c.1   Software engineering
          with database management
          systems.