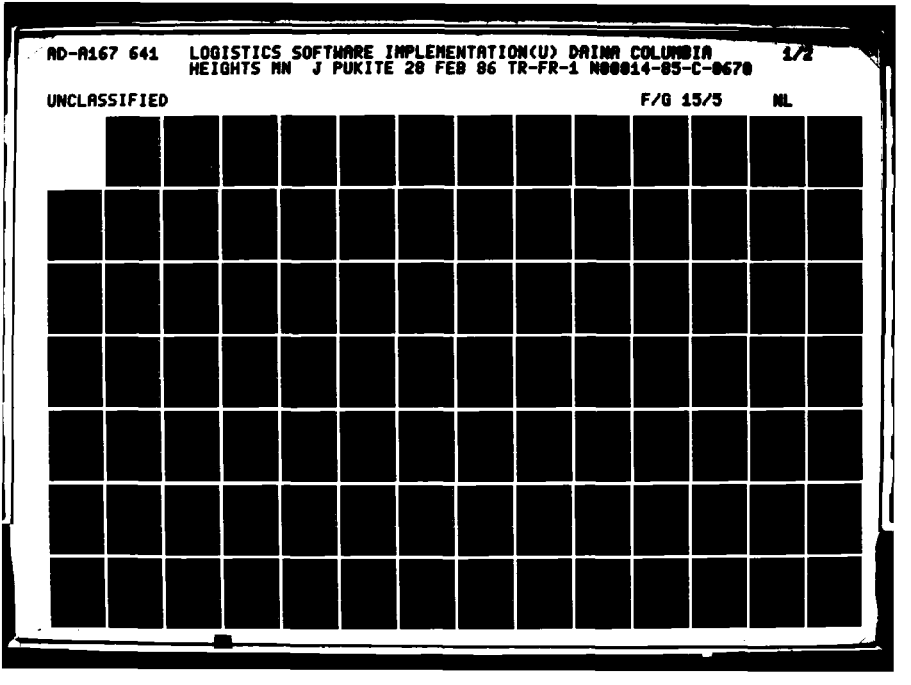
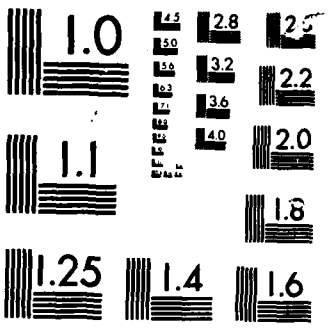
AD-A167 641 LOGISTICS SOFTWARE IMPLEMENTATION(U) DAINA COLUMBIA HEIGHTS MN J PUKITE 28 FEB 86 TR-FR-1 N00014-85-C-0670 1/2

UNCLASSIFIED F/G 15/5 NL





(12)

TECHNICAL REPORT FR-1

LOGISTICS SOFTWARE IMPLEMENTATION

J. PUKITE

DAINA
Columbia Heights, Minnesota 55421

FEBRUARY 1986

FINAL REPORT FOR PERIOD SEPTEMBER 1985 - FEBRUARY 1986

N00014-85-C-0670

DISTRIBUTION STATEMENT

UNCLASSIFIED/UNLIMITED

Prepared for:
Office of Naval Research
Department of the Navy
Arlington, Virginia 22217-5000

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A167641

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	Unclassified/Unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) FR-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION DAINA 6b. OFFICE SYMBOL (If applicable) 1CZ64	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State and ZIP Code) 4960 Fillmore St. N. E. Columbia Heights, MN 55421	7b. ADDRESS (City, State and ZIP Code) 800 North Quincy Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable) N00014	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-85-C-0670
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) Logistics Software Implementation (U)	
12. PERSONAL AUTHOR(S) J. Pukite	
13a. TYPE OF REPORT Final 13b. TIME COVERED FROM 850901 TO 860228	14. DATE OF REPORT (Yr., Mo., Day) 86-02-28 15. PAGE COUNT 178
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
15	05	
09	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Logistics Software Implementation, Logistics Data Base, Logistics Work Station

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

This report discusses the results of a research study dealing with the implementation problems of logistics software on microcomputers. It examines the complex logistics environment, the applicable algorithms, user interface, data base requirements, and the computational needs. It proposes a system-oriented problem solving technique based on hierarchical partitioning and decomposition using data flow as a means for solution control. It also establishes the need for a formal Logistics Description Language to provide a framework for problem statement and data interchange and proposes that these concepts be incorporated into a Logistics Work Station. This concept is then further examined and hardware, software and user interface requirements discussed. An applications example based on ship loading problem is used to illustrate the use of the work station and the key algorithms. It is concluded that the proposed concept is feasible and that a prototype system should be built.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Neal D. Glassman	22b. TELEPHONE NUMBER (Include Area Code) (202)-696-4313 22c. OFFICE SYMBOL N00014

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

SUMMARY

Military logistics systems are large, complex, and expensive to operate. Effective management of these systems involves many difficult decisions. The effective use of microcomputers offers potential for more efficient and less costly logistics management in the future. In this exploratory research project the key objectives and functions of the logistics management are reviewed and the potential for utilizing microcomputers to support the decision making process assessed.

In the past, most of the logistics research has concentrated towards developing new logistics models and more powerful algorithms to solve them. In majority of cases, these investigations have addressed problems at a very detailed level, such as facility location, transportation vehicle routing, inventory control, etc. These techniques have been implemented on large mainframes and have become an important tool in the logistics management process. Unfortunately, these tools are highly specialized and not readily available to the operations research analyst.

This study addresses the logistics problems from a logistics system viewpoint and is concerned with the conceptual development of an interactive, microcomputer based Logistics Work Station (LWS) to support operations research analysts and logistics system operations personnel.

Section I provides an overview of the project and describes its key areas of investigation.

Section II presents the main technical objectives: identification of of practical problem partitioning and decomposition techniques; development of solution control techniques; establishment of logistics problem description, data base and user interface requirements; selection and evaluation of promising OR (Operations Research) algorithms; and selection and demonstration of how these techniques would be applied to a practical logistics problem.

Section III contains a detailed technical discussions of the specific topics investigated during the course of this study. This section is divided into 7 subsections each covering an important logistics topic.

Subsection 1 describes the military logistics environment, logistics measures, and the logistics management objectives. Specific topics covered include material acquisition, distribution and management process, warehouse location, transportation, and scheduling.

Subsection 2 covers the algorithm selection process and describes how the applications problem was used as a starting point. The selected algorithms covered a wide variety of applications and included most of the commonly used techniques. The selected algorithms were chosen from those available in the open literature, well researched, and usually validated by a number of researchers.

Subsection 3 discusses conceptual design of a Logistics Work Station. This discussion covers the proposed architecture, functionality, and an application example illustrating multiple workstation environment.

One of the problems hindering an orderly approach to the logistics problem solution is the inability to describe a logistics problem in a formal manner. In subsection 4 Logistics Description Language is proposed as a solution to this problem.

Subsection 5 covers the unique logistics data base requirements. Specific requirements, such as the ability to handle large amount of data and to support a dynamic environment are discussed in detail.

The acceptance of any computer aided support system depends on its user interface. Subsection 6 covers this very important area and presents very detailed requirements. Use of separate dialog modules is proposed as a solution to support a wide range of system users.

Subsection 7 covers solution control. Concepts of large problem decomposition and partition are discussed and specific computer aided techniques needed to support these tasks explained. An approach based on using data flow concept is proposed and its potential use in the solution control process illustrated.

Finally, subsection 8 covers a specific applications example based on a cargo ship loading situation. The specific logistics functions are identified and algorithms capable of providing support identified.

The next section, IV, presents the conclusions reached at the end of this study. It is concluded that key OR oriented algorithms can be relatively easily implemented on microcomputer based systems. The individual algorithm code modules, when implemented in a higher order language, were compact in size, permitting coresidency of several modules in memory. The processing speed for problems of moderate size was not prohibitive. The solution accuracy using floating point processor was as good or better than those of mainframes. The proposed solution control algorithms and their use in the solution process appears feasible. The data base is complex, but is highly modular and adaptable to a dynamic environment.

It is recommended that the entire logistics environment should be reviewed to identify potential applications of microcomputers. Algorithms needed to solve the specific problems identified should be selected, optimized for use in microcomputer environment, and programmed in a higher level computer language. It is further recommended that the Logistics Description Language design effort be continued, the Logistics Data Base concept extended, and the data flow based solution control approach fully implemented. Finally, it is recommended that all of the concepts presented above be incorporated in a Logistics Work Station prototype.

PREFACE

The work on this program was performed under the direction of Dr. Neal D. Glassman, the Scientific Officer and the contracting officer's designated representative for the U. S. Navy Office of Naval Research. This effort was sponsored under a Small Business Innovative Research (Phase I) Contract N00014-85-C-0670 and was performed by DAINA, Columbia Heights, Minnesota. J. Pukite was the project leader and the principal investigator. He was assisted by Ching-Ping Hong, research assistant.

ACKNOWLEDGMENT

The author wishes to express his appreciation to Dr. Neal Glassman for helpful technical discussions. He would also like to thank Aline Fairbanks for technical editing and Paul Pukite of University of Minnesota for proofreading. Finally, a special thanks to Ethel Sparks of DCAS Twin Cities Office for providing help with the administrative details and to DTIC search section for helping to find poorly identified technical reports.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

The implementation of logistics software on a general purpose microcomputer poses some unique problems. These include the diversity of solution techniques, a large variety of application programs, and a high level of interaction among the various functional areas involved in the logistics operations. All of these factors are important and will be considered in detail to arrive at efficient and practical solutions of the logistics problems.

1. LOGISTICS ENVIRONMENT

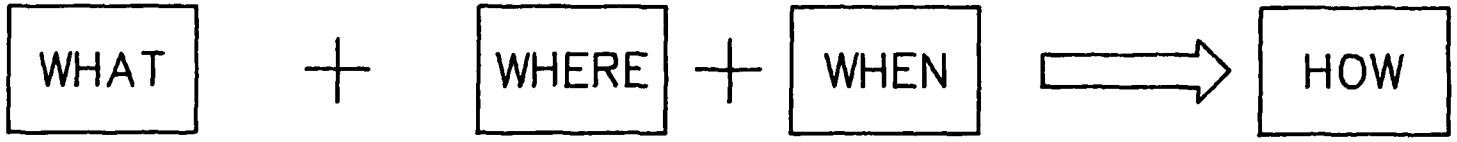
The overall logistics system is shown in a highly simplified form in Figure 1 where the first three major blocks (WHAT, WHERE, WHEN) represent the input to the system and the fourth (HOW) deals with the decision-making process. It is in this latter area where the application of microcomputers will play a major role. This is not to say that computers will not play a major role in other areas where they could easily provide an efficient data transfer needed for the problem definition. In this study, however, primary emphasis will be placed on the 'HOW' problem: the efficient solution of logistics problems using the Logistics Work Station concept.

Typical logistics tasks include resource allocation, assignment, scheduling, routing, selection of carrier, placement, and layout. On the other hand, solution techniques include linear and integer programming, maximum flow, shortest path, packing, covering, and others. Examining the mapping associated with the shiploading problems revealed that in most cases there was a one-to-many mapping from the logistics tasks to the applicable solution algorithms. Attempt will be made to represent this mapping in a decision table format to aid the analyst in the selection process. In the Phase II effort, this support could be further extended by employing the emerging techniques of knowledge engineering.

2. OPERATIONS RESEARCH TECHNIQUE SELECTION

An important question that had to be answered at the beginning dealt with the selection of specific operations research techniques needed by the analyst in the Logistics Tool Kit (LTK). There was no single unique set that could be selected because of the lack of one-to-one mapping between the logistics tasks and the operations research solution algorithms. To solve this problem, a number of iterations were performed. First, typical logistics tasks associated with the shiploading problem were selected and classified, and then appropriate algorithms were sought. Next, another iteration was performed by examining the available solution procedures and then trying to determine where in the logistics process they would be applicable.

To solve the logistics problems associated with the shiploading process will require a number of iterations, possibly with different algorithms. This process will require not only a highly flexible data input interface, but also the capability to store intermediate results and to transfer them to



MATERIAL	MOVEMENT	METHOD
SIZE	DESTINATION	CARRIER
QUANTITY	WHEN NEEDED	ROUTING
CHARACTERISTICS		SCHEDULE
LOCATION		HANDLING

FIGURE 1. LOGISTICS SYSTEM PERSPECTIVE

the next selected procedure. By automating the solution data handling, the need for reentering data will be eliminated and the time needed for each iteration reduced.

Since the logistics operations cover a wide spectrum, a starting point had to be selected which would not only define a practical application, but would also provide the opportunity to apply a large variety of the operations analysis algorithms. The shiploading problem provided such starting basis. In this study, this problem will form the nucleus around which the techniques will be developed. At the same time, the techniques selected for the svstem configuration will make provisions for future extensions.

3. LOGISTICS WORK STATION

The Logistics Work Station concept is illustrated in Figure 2, where the major functional blocks are identified. The block marked Control Subsystem provides the interface to the user. It supports data entry, solution method selection, solution control, and presentation of results.

Data Base contains the problem specification data: variables and constraints. This data base will be designed to be dynamic and extensible. This approach will be necessary because most of the information will be of a transient nature and in a format needed for data transfer to the solution modules.

The Method Bank contains all of the solution procedures available to the analyst. In this area, an open architecture will be selected to permit easy addition of new solution techniques and modification of the existing ones.

The Model Bank holds the simulation techniques needed for investigation of different operational concepts. In the initial version, a queue simulator will be included to permit the investigation of congestion in loading areas and for other similar problems.

The development of the Logistics Work Station thus involves the definition of what specifically must be included in each of the major blocks. Since the Data Base will have to support both the Procedure and Model subsystems, its content will be dictated by the needs of these systems. In turn, the Control Subsystem will be required to control the other blocks in the system. Thus, there will be a hierarchical relationship on one hand, but also a certain amount of distributed communications.

The Work Station concept has been very popular in the design engineering applications, where it now plays a major role. Most of the design automation tools, originally residing on mainframes, have been transferred first to special purpose minicomputers, later to special purpose high-performance microcomputers and more recently to general purpose personal computers with graphics support. Many of the techniques found valuable in the design of the Engineering Work Station will be equally applicable in the design of the of the Logistics Work Station; this transfer of technology will speed the development of the proposed system.

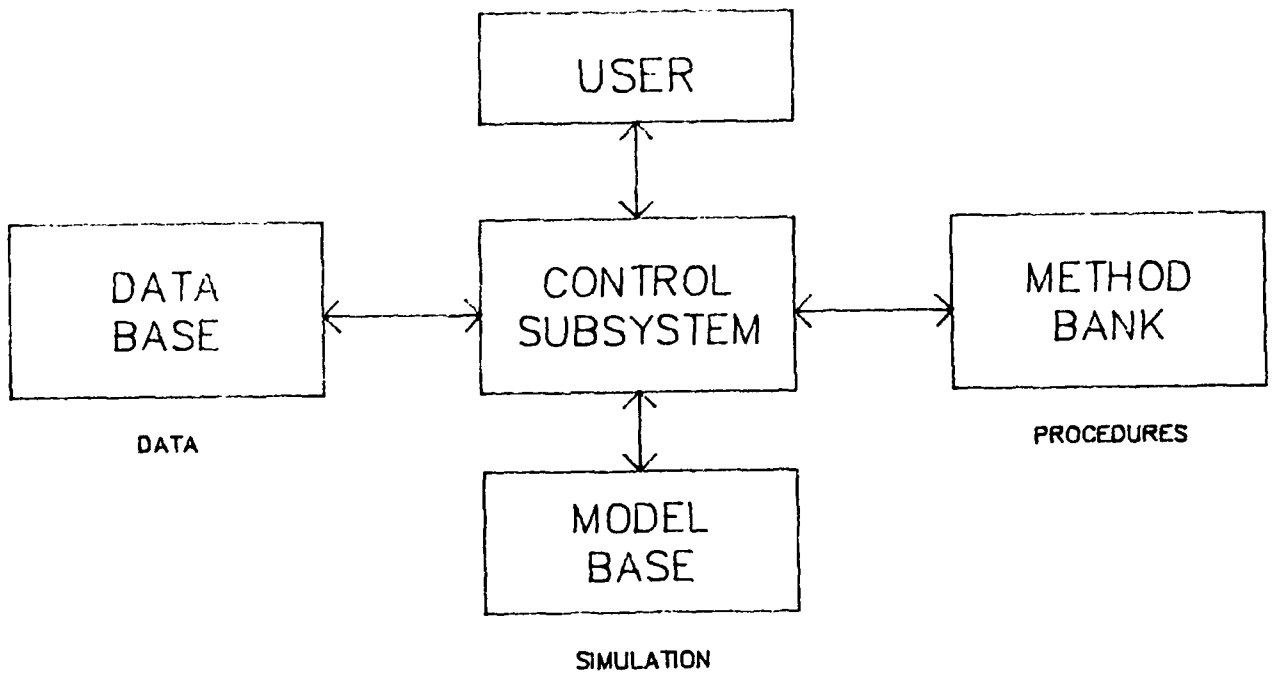


FIGURE 2. LOGISTICS WORKSTATION ARCHITECTURE

Pascal was selected as the official programming language for the project. This language permits the utilization of structured techniques, and also supports those features needed for effective data structure development. It will also permit easier procedure transfer to the official DoD programming language when efficient versions become available on microcomputers.

4. APPLICATION OF THE LOGISTICS WORK STATION

The initial attempt at representing the Logistics Work Station in more detail is shown in Figure 3. This figure shows the major activities in three separate operation areas: depot(warehouse), transfer, and cargo ship. Usually these operations will be performed by different crews directed by their own schedulers. It is apparent, in order to improve efficiency, that good coordination of the tasks will be necessary. This coordination could be provided in a hierarchical fashion by the Master Control and supplemented by distributed communications links between the performing groups.

The breakdown shown in Figure 4 details two major procedure blocks: Scheduler and Loader. Each one of these will contain a number of procedures, possibly modified for the specialized applications. The details of these procedures will be discussed in Section III of this report.

The orderly solution of the logistics problems will require some new ground breaking in this area. Due to time limitations only a general outline of the needed capabilities will be included in the Phase 1 effort.

5. SOLVING LARGE PROBLEMS

In the past, most of the logistics and OR research has focused on the development of specific algorithms. In a few cases decomposition techniques were investigated as part of a specific algorithm development. The more general problem of how to decompose a more complex logistics problem involving a number of different algorithms has received very little attention, although the problem has been identified and discussed previously.

In the past these complex problems were handled by a manual decomposition: the larger problem was partitioned into a number of subtasks and these were then assigned to different teams for detailed solution. When the complex problem is examined in more detail, however, it is apparent that subdivision of this type will not result in an optimum solution because of the interaction of the individual parts.

The proposed approach still relies on partitioning, but the partitioning is done by considering the effects of interaction. Computer techniques will be proposed not only to identify the interacting variables, but also to keep track of the computed variables during the solution process and to control the iterative solution process.

The techniques investigated are not necessarily new, but many of them have never been applied to the solution of logistics and OR problems. For example, data flow analysis, discussed in the technical part of this report, is a familiar tool in computer language design. By recognizing the similarity of the problems, a great number of valuable algorithms available in other

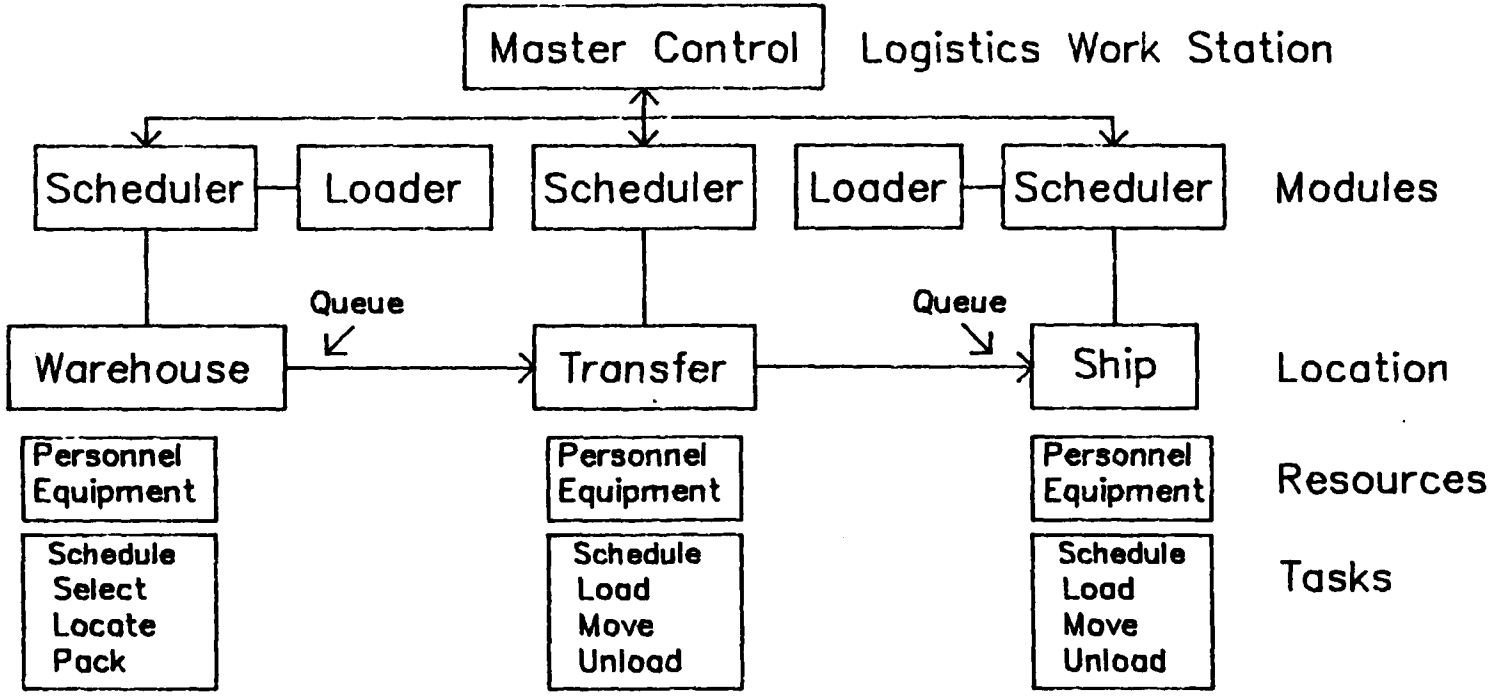


FIGURE 3. LOGISTICS WORK STATION APPLICATION

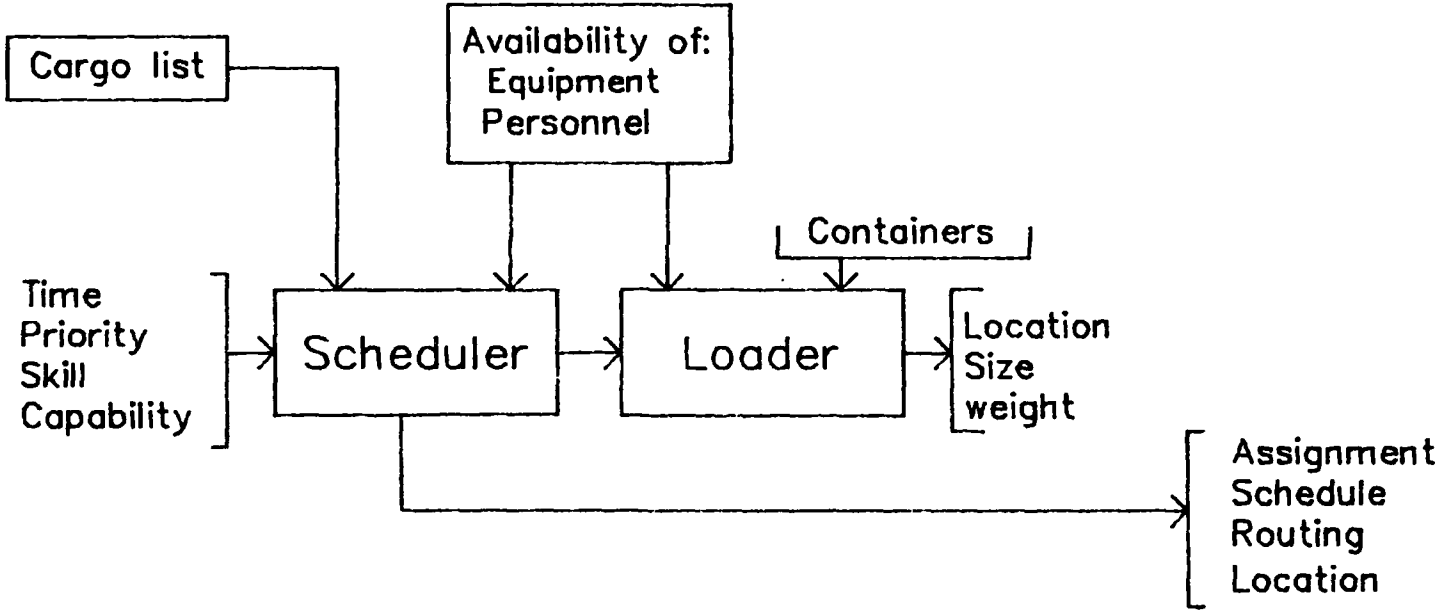


FIGURE 4. DETAIL OF SCHEDULER AND LOADER MODULES

disciplines can be adapted to the solution of logistics problems.

During the investigation, it was also found that some of the techniques developed for manual partition were ill-suited for computerization. The main reasons were that too much emphasis was placed on the human capability of being able to recognize certain patterns and being able to switch between different techniques. Although the area of pattern recognition has received much attention, it has not reached the developmental stage where it could be applied in a routine manner to the problem solution. Fortunately, in every case investigated, better and more efficient algorithms were found capable of solving the same problems. It is interesting to note that these algorithms suitable for computer solution would be rather difficult for humans to apply.

Thus, there are really two distinct areas in the algorithm development: solution control and specific algorithm development. Each one will have to be analyzed in detail and the best computer implementation selected. Blind conversion of the existing manual techniques to computer solution will not always be successful. This observation appears to be well recognized by aeronautical engineers in that they have not yet designed a practical aircraft resembling a real bird in flight.

6. DATA FLOW CONTROL CONCEPT

A search of the applicable computing literature revealed that the data flow concept is used in different contexts and has many different interpretations. The particular data flow concept selected for this application is the one used in the compiler-code-optimization process and has a very formal mathematical basis. Some of the other data flow methods are more of a descriptive nature, and are used to illustrate data base management and data flow in a complex organization environment. The commonly used Information Modeling process falls in this classification.

There is a great need for a formal language suitable for the logistics problem definition. After a thorough literature search of the applicable logistics publications, no references were found. This can be explained by examination of previous efforts which focused on the solution of well defined and and very specific logistics problems. Since these detailed problems did not usually consider interaction with the rest of the system, there was no need to consider the overall logistics process or to develop a general formal system description.

7. LOGISTICS DESCRIPTION LANGUAGE

To improve communications efficiency, standards will be needed to define the key logistics operations. There is already an effort underway to improve some parts of military logistics. This effort, known as the Computer-Aided Logistics Support (CALS), is a joint DoD Industry project, initiated in June 1984. At the present time, this effort includes only parts of logistics activities; namely, those where already there are existing standards, i.e. word processors, graphics, and product definition. Unfortunately, almost every standards effort that has been undertaken has been of the bottom-up type, requiring continuous iterations and modifications. Looking back at Figure 1, the CALS effort represents only a portion of the 'WHAT' block.

The remaining part of this block, and the other blocks shown in this figure, have not yet been considered for standardization.

The initial effort of trying to define a Logistics Description Language capable of describing the larger logistics system environment revealed some rather interesting problems. First, the logistics environment is very complex and involves a high degree of interaction. The key elements are those of the available resources (personnel and moving equipment), material to be moved, and the actual movement. Second, the logistics environment is very context-sensitive; there is a large amount of background information needed to specify any particular problem. Fortunately, this background information is usually of a static nature and is relatively well defined.

Because of the complexity of the logistics environment and the context sensitivy of the descriptions, the logistics description language development will be a complex and difficult process. Nevertheless, without the formal problem description basis, an orderly and efficient solution of logistics problems will not be feasible. These issues are discussed in more depth in the technical discussion section.

8. USER INTERFACE

Since the proposed approach will involve a large variety of different solution techniques and will draw upon an extremely large and complex data base, the acceptance of the proposed concept will depend to a large extent on the ease of the system operation.

The design of the user interface will be one of the more difficult and, at the same time, more challenging tasks. Although there is much talk about how the interactive systems should behave, there are very few generally accepted procedures for the interface design. The guidelines proposed in the technical section have been adapted from a number of sources and modifications proposed to meet the specific logistics requirements. (See References 1 - 7.)

The proposed interface design is based again on using the partition and decomposition principle. The dialog section is separated from the solution modules and is then further decomposed hierachically. Proceeding in this manner leads to a more flexible system that is also easier to maintain.

A specific example of the dialog design is illustrated in Appendix B. This example is based on a real application where the proposed approach resulted in major cost savings and performance improvements.

9. DATA BASE DESIGN

The data base design also poses some challenging problems. Limited storage capabilities and slower processing speed of the microcomputers require the software techniques to compensate for these shortcomings. The data storage formats and file structures are areas where major improvements are possible. These in addition to the Tree oriented data retrieval techniques will permit the proposed system to achieve reasonable operator response.

Partition and hierarchical decomposition together with data abstraction form

the major tools proposed for the logistics data base design. To meet the dynamic data restructuring requirements, a relational data base concept is proposed.

Dynamic data base structures are already finding applications in interactive graphics and Very Large Scale Integrated (VLSI) circuit design areas. A recent background discussion is available in Reference 8. The commercially available data bases are compared in Reference 9. The more theoretical aspects of the relational data base is presented in Reference 10.

10. APPLICATIONS EXAMPLE

The specific applications example considers the shiploading problem in its widest context and covers the movement of the requisitioned material from a depot to the cargo bay area in a ship.

The representation of this complete environment as a dependency graph is one of the key objectives of this specific task. This dependency graph will be used as a formal definition of the logistics problem and will form the road map for the automated solution of the various subproblems.

To lay the groundwork for the formal representation, the logistics functions involved in this process are considered in detail using a bottom-up approach. This particular approach was taken because it is the most familiar to operation research analysts. Later, a simpler top-down approach will be presented, as well as an approach for determining the dependency graph.

Since the proposed approach involves the use of almost all of the known OR techniques and requires the use of a complex data base, emphasis is also placed on discussing the potential aplications and on the definition of an initial data base.

The shiploading problem is divided into six major functions: supply depot selection, container selection and packing, container loading, transportation, container unloading, and cargo loading on the ship. Each of these functional areas is examined in detail, the tasks associated with the corresponding work centers are identified, and the use of the OR algorithms outlined. This information is then presented as a system level functional dependency matrix.

Detailed design of the data base is presented with particular emphasis on the support of the dynamic data base elements. The initial data base design includes a complete set of data elements to support the full shiploading process as defined above. There are data elements to fully describe the material requisition, logistics task structure, available personnel and equipment resources, packaging hierarchy, and the physical warehouse and ship structures.

II. TECHNICAL OBJECTIVES

The principal objective of this program was to conduct research on implementation problems of logistics software on microcomputers. Special attention was placed on decomposition and partitioning of complex problems and on operator interface specifications. The selected techniques were then examined in more detail in a shiploading environment. Due to time limitations, this specific applications example was constrained to the movement of supplies from a single depot to a single cargo ship.

The principal objective stated above was used to establish a series of specific subobjectives:

1. Identification of practical decomposition techniques to partition the large and complex logistics problem into modules or subproblems, and the establishment of the hierarchical structure of the problem.

2. Identification of techniques applicable for solving each of the subproblems created by the partitioning process.

3. Identification of a control structure needed to control the solution of the subproblems and the linking mechanism needed to construct the final solution.

4. Identification of practical techniques for the aggregation and disaggregation of information through the various hierarchical levels.

5. Identification of techniques for handling large and dynamic logistics data bases needed to support the problem solution.

6. Identification of the need for a logistics description language needed for logistics problem structure definition and for specifying the desired solution type.

7. Identification of techniques needed for data presentation and processing.

8. Evaluation of the expected performance of the proposed approach, considering the effects of suboptimization introduced by the decomposition and partitioning process.

9. Demonstration of the concept feasibility by selecting an applications problem from the Navy logistics area and discussing how the proposed techniques would be applied to its solution.

III. TECHNICAL DISCUSSION

1. LOGISTICS ENVIRONMENT

A military logistics system encompasses requirements definition, acquisition, storage and loading of material, and the movement of personnel and material to overseas theaters. Many of these operations involve a single-manager activities (material and transportation managers) which support all of the Military Services.

In the military environment, logistics is managed as an integrated system. The integrated logistics management concept distinguishes four major functional categories: management, systems, operations, and coordination as shown in Figure 5. The key logistics elements are: facility location, transportation, inventory, communication, and handling and storage.

Logistics costs are very high in both military and commercial sectors. It is estimated that the logistics components account for 25 to 30% of the cost of the product, and that system support costs often exceed the original procurement costs by a considerable margin. According to government statistics, in 1980 14% of the total U.S. labor force were involved in logistics or logistics related activities. Recent military sponsored research analyses indicate even higher percentages of personnel assigned to logistics functions. One such research study reports that 78% of the shore-based and 58% of the sea-based U.S. Navy personnel were assigned to logistics functions [11-12].

To obtain the required military readiness within a limited budget or to remain competitive in the commercial market, it is important to identify and to reduce the logistics costs.

The overall objective is to minimize the life cycle cost of a system. The applicable constraints include readiness and sustainability, operational availability, technological feasibility, operational and logistic scenario, manpower/skill availability, order and ship times, etc.

Logistics system capability is measured in terms of readiness and sustainability. In logistics, readiness is the peacetime ability to support the force structure; sustainability is the ability to continue that support at combat rates during military engagements.

Logistics system capability assessment is based on two key questions:

What are the material requirements and can they be met?
Can the material be positioned or moved to where it will be needed?

The logistics system manager is thus faced with the definition of the proposed system structure and also the evaluation of the operational aspects of the proposed system. To accomplish his task, he must use the available systems analysis and operations research tools as illustrated in Figure 6. System analysis tools help the manager to select the best

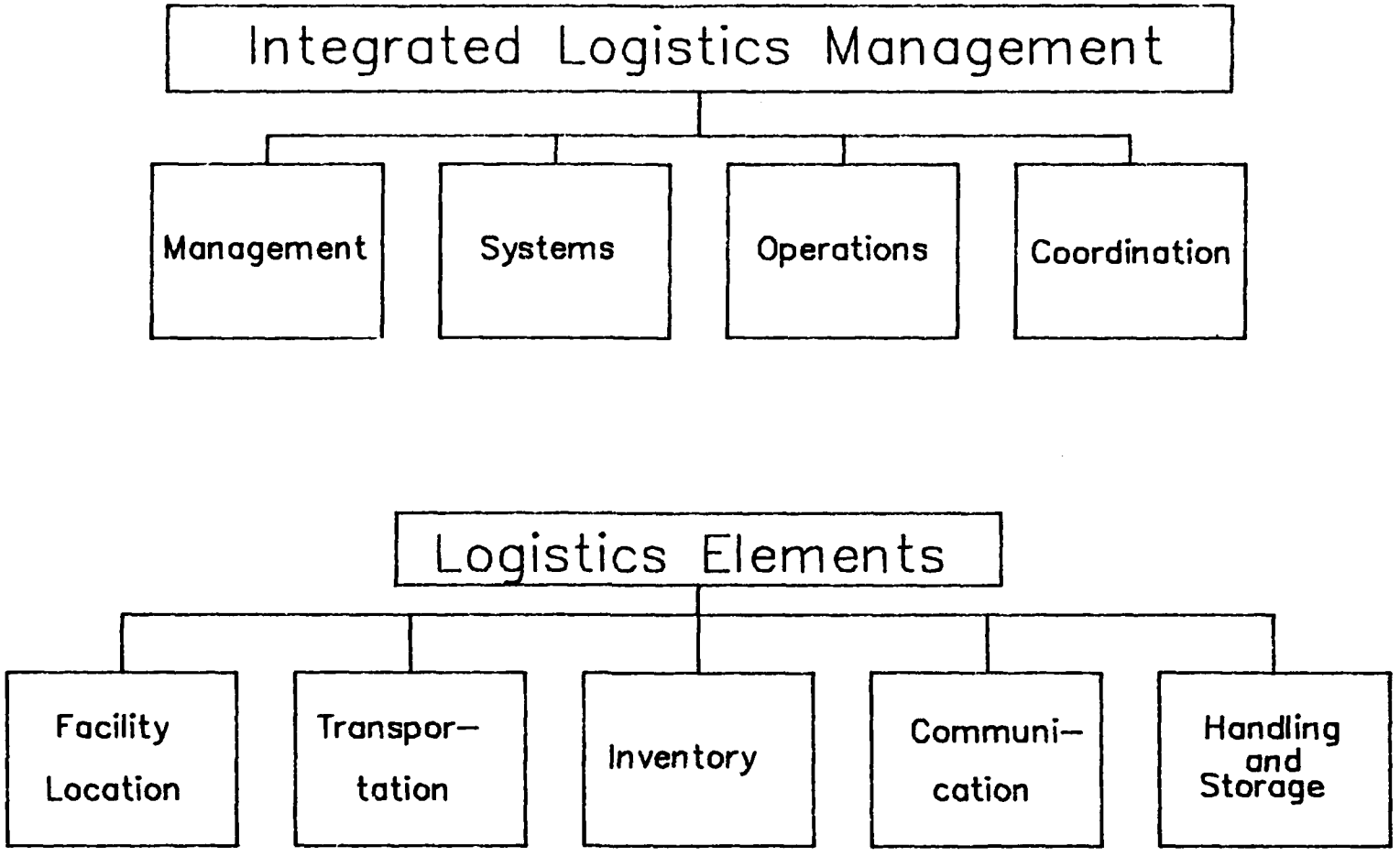


FIGURE 5. INTEGRATED LOGISTICS CONCEPT

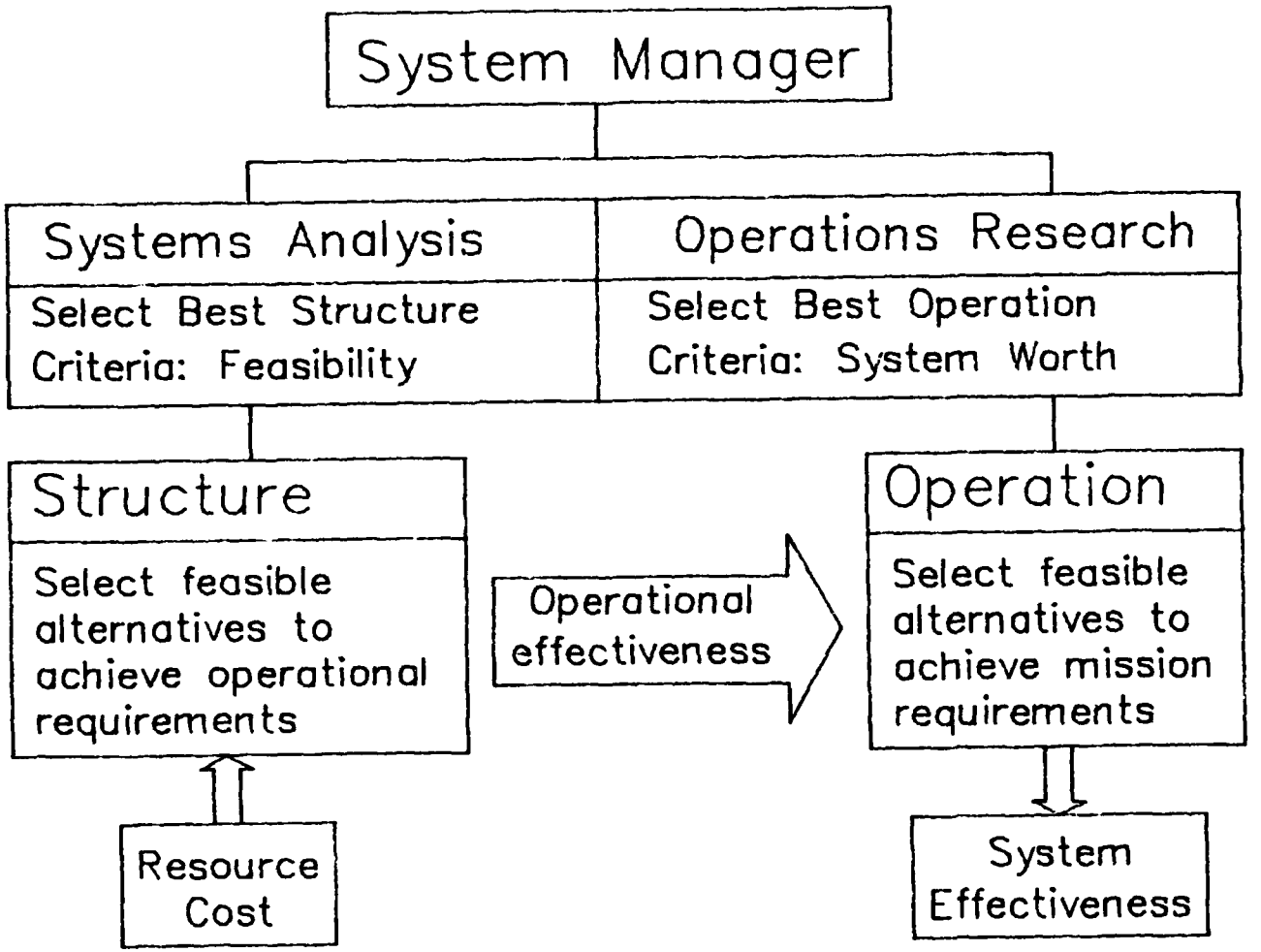


FIGURE 6. SYSTEM MANAGEMENT FUNCTIONS

feasible system structure; operations research tools help him to select the best operational procedures to achieve the given mission requirements.

Although each service is responsible for determining and funding its material requirements, there is also a need to evaluate these requirements on a more global level. Defense system acquisitions require achievement of the best balance between cost, schedule, performance, and supportability. Thus, a number of analyses will be required at the different operational levels to achieve the specified overall requirements.

Unfortunately, much of the currently available information is not complete enough for these assessments, either because of insufficient or incompatible data, or lack of proper analysis tools. Only by establishing logistics data standards, reporting requirements, and by providing the proper analysis tools can we hope that the logistics capability evaluation models will become feasible.

A general system optimization process is illustrated in Figure 7. In this diagram, assumption is made that the proper analysis support tools are available to the designer. There are, however, only a limmited number of practical tools available to support the logistics optimization tasks. [13] Most of these tools are specific and apply only to certain specific problems. The general problem solving process is depicted in Figure 8 detailing the sequence of problem-solving tasks and some of the more familiar logistics models.

To gain a better understanding of the potential microcomputer application to logistics problem solving, some of the key logistics areas will be examined in detail:

a. Production and Material Acquisition

Many different techniques have been proposed to optimize the production and material acquisition process. Some of the better known include design-to-cost technique, value engineering, advanced manufacturing technology, component standardization, life-cycle costing, and others. In all of these approaches, there are many potential applications for microcomputers. There are other areas, however, where the immediate application of computers is not yet feasible without extensive developmental effort. These areas include efforts to reduce procurement lead time, evaluation of requirements, evaluation of incentives, etc.

b. Distribution and Material Management

The distribution objectives are to provide rapid and reliable transportation. Again, there is a need for a distribution system, not just the distribution and material handling equipment. The OR techniques could provide means to determine the optimum mix of inventory investment costs, supply depot operating costs, and transportation costs. Multi-echelon inventory analysis is particularly important in the military environment. Here the objective is to determine ordering, stocking, and distribution policies for two or more

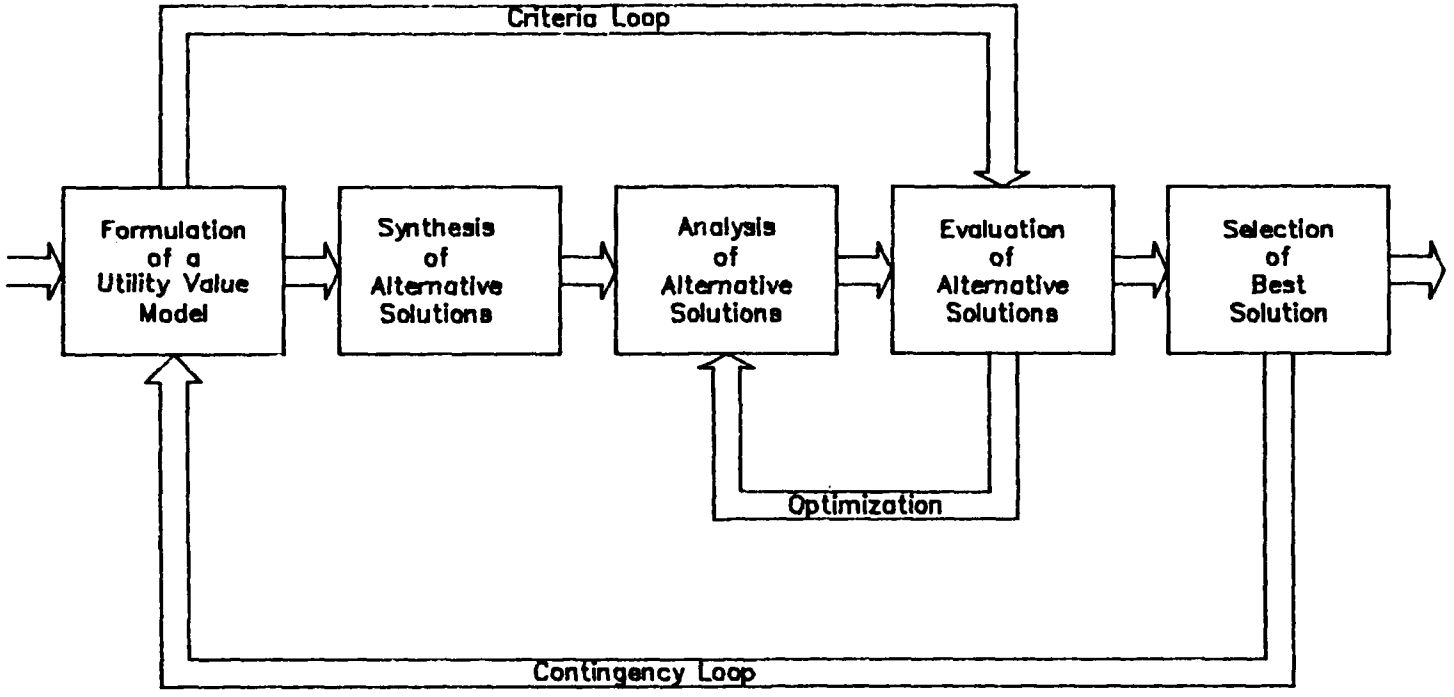


FIGURE 7. SYSTEM OPTIMIZATION PROCESS

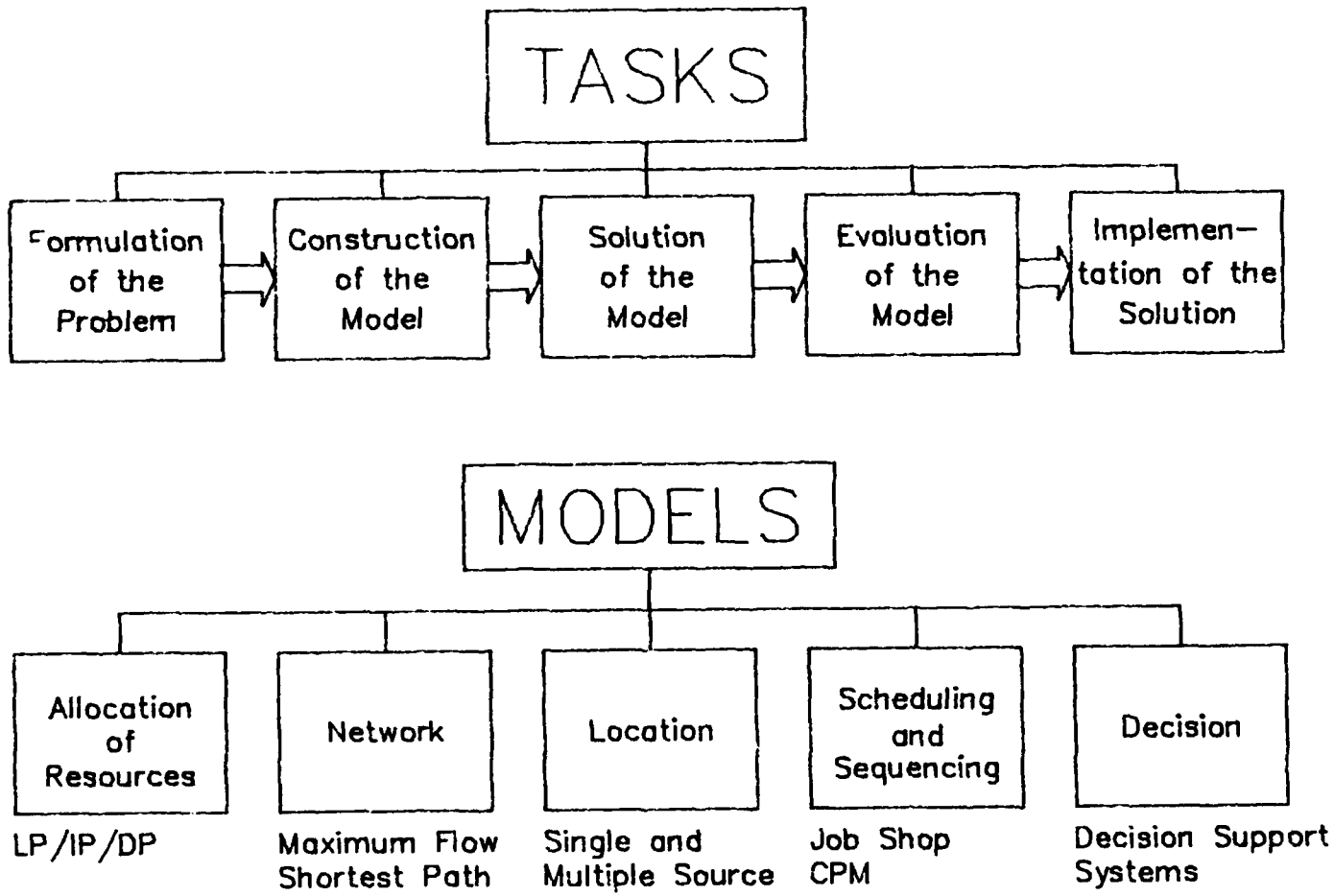


FIGURE 8. PROBLEM SOLVING: TASKS AND MODELS

interrelated supply facilities, such as depot repair, intermediate repair shops, and the associated inventories. In determining these requirements, the effect of component shortages must be included.

Cost information is also needed for maintenance operations. Replacement and inspection policies, reliability, workload scheduling, test equipment availability are some parameters. The key objective is to reduce the frequency of inspection and the false removal rate of components.

c. Transportation

The basic transportation problem is to design a network of routes and carriers to move material and personnel. As shown in Table 1, many options are available for military transportation. The transportation problem is further complicated by the variety of the different supplies carried in a logistics system (see Table 2). Application of computers could provide improved operations by providing a transportation data base and an improved computer based communications system.

A key military requirement is to create a transportation system that can respond in a timely manner to wartime mobilization requirements. The conventional measure of ton-miles per day is not adequate in military environments because it does not consider the effects of delays on the capability of military systems. The military transportation model must consider not only the losses and degradation of the channels, but also incorporate priority transportation for mission critical commodities.

The transportation model must also consider the interaction between transportation and other logistics functions and their effect of the military operations. Evaluation of these more complex models call for considerable computer support.

d. Warehouse Location

Warehouse location is a very important logistics problem. It has to consider not only the area served, but also the amount of goods to be stored, the available transportation routes, etc. This particular area has received a major share of attention in the past. The problem of suboptimization caused by not considering the higher level interaction between other logistics considerations is very seldom considered because of its complexity.

e. Scheduling Process

Effective logistics management presents decision making at all three levels of planning (strategic, tactical, and operational). Decisions related to the locations of facilities (warehouses, depots, or ports) may be viewed as strategic, while problems of transportation fleet size and mix determination could be viewed as tactical. Finally, on the operational level, various decisions concerning the routing and schedul-

TABLE 1. Modes of Shipment

CODE	DESCRIPTION
A	Motor, truckload
B	Motor, less than truckload
C	Van
D	Driveaway, truckaway, towaway
E	Bus
F	MAC
G	Surface parcel post
H	Air parcel post
I	Government truck
J	REA (Railway Express Agency)
K	Rail, carload
L	Rail, less than carload
M	Freight forwarder
N	LOGAIR (Logistics Air Centers)
O	Organic military air
P	TGBL (Through Government Bill of Lading)
Q	Air freight
R	Air express
S	Air charter
T	Air freight forwarder
U	QUICKTRANS (Military Airlift)
V	SEAVAN
W	Water, river, lake, coastal
X	Mixed modes
Y	Intratheater airlift system
Z	MSC
2	Government water craft, barge
3	RORO (Roll on / roll off) service
4	Armed Forces Courier Service (ARFCOS)
5	United Parcel Service
6	Military Official Mail (MOM)
7	Weapon system pouch service
8	Pipeline
9	Local delivery

Reference:
Military Standard Transportation and Movement Procedures (MILSTAMP)

TABLE 2. Military Supply Classes

All items necessary for the equipment, maintenance and operation of a military command, including food, clothing, equipment, arms, ammunition, fuel, materials, and machinery of all kinds. For planning and administrative purposes supplies are divided into classes as shown below:

Class I

Subsistence, including gratuitous health and welfare items. Subclassifications for Class I are: A -- Air (inflight rations); R -- Refrigerated subsistence; S -- Nonrefrigerated subsistence (less combat rations); C -- Combat rations.

Class II

Clothing, individual equipment, tentage, organizational tool sets and tool kits, hand tools, administrative, and housekeeping supplies and equipment. Subclassifications are: B -- Ground support material; E -- General supplies; F -- Clothing and textiles; M -- Weapons; and T -- Industrial supplies.

Class III

Petroleum, oils, and lubricants. Subclassifications are: A -- Air; and W -- Ground (surface).

Class IV

Construction. Construction materials include installed equipment and all fortification/barrier materials. (There are no subclassifications.)

Class V

Ammunition. Subclassifications are: A -- Air; and W -- Ground;

Class VI

Personal demand items (nonmilitary sales items). (No subclassifications.)

TABLE 2. Military Supply Classes (cont.)

Class VII

Major end-items. A final combination of end products which is ready for its intended use; e.g., launchers, tanks, vehicles. Subclassifications are: A -- Air; B -- Ground support material; D -- Administrative vehicles; G -- Electronics; K -- Tactical vehicles; L -- Missiles; M -- Weapons; and N -- Special weapons.

Class VIII

Medical material including medical-peculiar repair parts. (No subclassifications.)

Class IX

Repair parts and components to include kits, assemblies, and subassemblies, reparable and nonreparable, required for maintenance support of all equipment. Subclassifications are the same as for Class VII, with the addition of T -- Industrial supplies.

Class X

Material to support nonmilitary programs, e.g., agricultural and economic development, not included in Classes I - IX. (No subclassifications.)

ing of vehicles and the assignment of crews require inputs on a daily basis. There is very strong interaction between the various levels of planning and the decisions made at one level affect or are influenced by decisions at other levels.

There are two general types of scheduling: aggregate and detailed. The former plans for the overall level of output for a system and specifies the required resources, while the latter allocates, sequences, and obtains time schedules. The aggregate scheduling is usually performed at a higher level in the organization. The detailed scheduling is based on the given aggregate schedule as is performed closer to the operational level.

Scheduling is primarily concerned with the loading and sequencing of orders. Loading, as applied to scheduling, refers to the allocation of tasks to workcenters. Sequencing establishes the order (priority) in which the tasks will be performed by a workcenter. The end product of the scheduling process is the final detailed schedule which specifies starting and completion times for each task at the workcenter. Figure 9 illustrates the scheduling process. It shows not only the normal straight sequence but also illustrates the iterative paths where the scheduling process may have to be repeated because the given requirements were not met. These feedback paths may also be used to optimize the scheduling process if some of the tasks are completed ahead of schedule.

Scheduling, as applied to the shiploading problem, is an intermittent operation because of the variety of tasks associated with the process. The overall objective is to complete each task efficiently so that the time required to transfer material from one location to another is minimized. This objective is accomplished by allocating resources (equipment and personnel) according to the time and place constraints.

Efficient schedule construction requires detailed information about the tasks, as well as information on the availability of material moving equipment and personnel. Each task may require several subtasks usually performed at different workcenter locations. This in turn requires proper sequencing of these subtasks. For example, when moving a container from a warehouse location to its final destination in the cargo area, several subtasks may be easily identified. These include selecting the parts, loading the container, moving the container to the warehouse loading area, moving the container to the dock area, etc. These subtasks will be typically performed by different crews.

Once these tasks are loaded (asigned to the workcenters) they must be sequenced. Sequencing involves consideration both of priorities and task synchronization with other workcenters. Once the sequencing is complete, detailed schedules are developed.

The organizational hierarchy involved in the scheduling process is illustrated in Figure 10. At the top level is the cognizant logistics unit. The aggregate scheduling is usually performed at this level. At the next level, there are a number of workcenters, usually at different physical locations and capable of performing certain specific tasks. A number of tasks are assigned to each workcenter. It is the responsibility of the workcenters to perform the

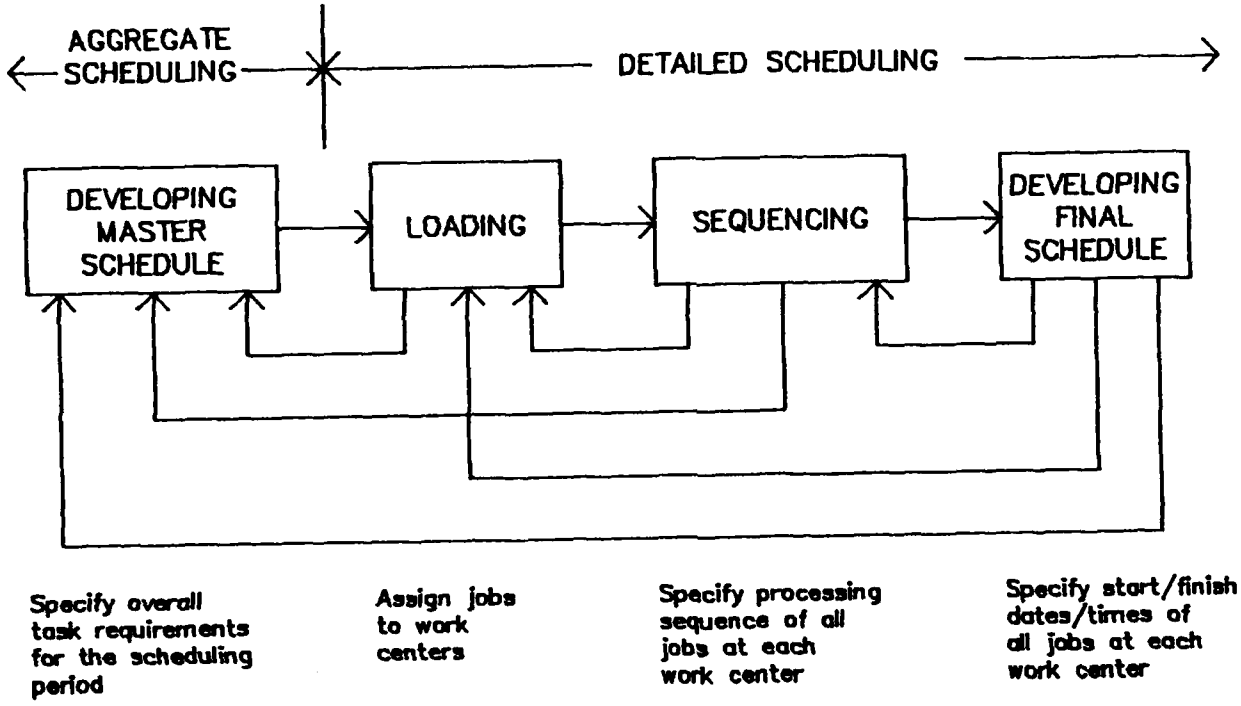


FIGURE 9. SCHEDULING PROCESS

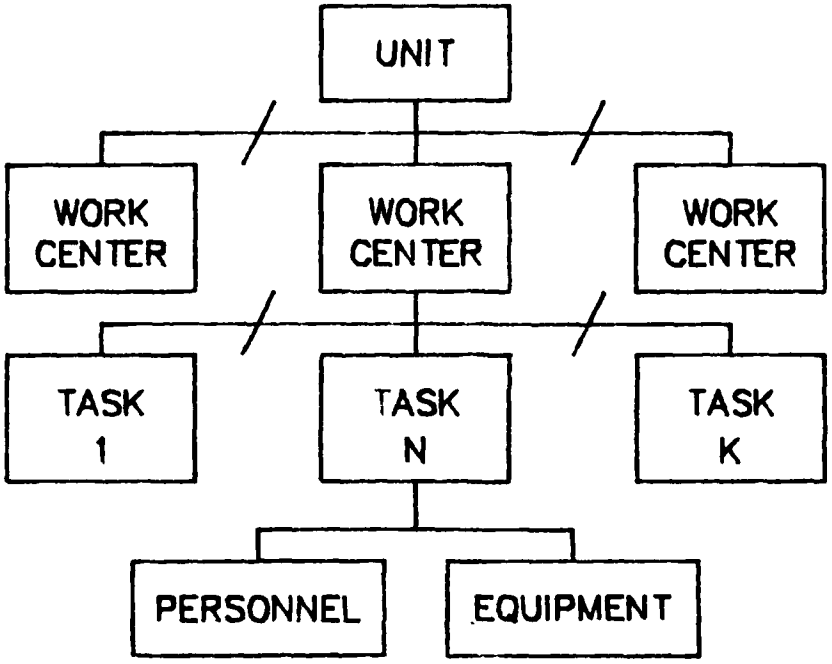


FIGURE 10. ORGANIZATIONAL HIERARCHY

detailed scheduling involving loading of tasks, sequencing, and determining the final time schedule. As mentioned earlier this type of scheduling is highly interactive, and subject to continuous modifications and iterations. These iterations are caused by equipment breakdowns, qualified personnel absence, and other usually unpredictable causes.

The scheduling and routing problems are further complicated by the variety of constraints encountered in realistic situations. A summary of these is presented in Table 3.

f. Future Requirements

In future wars or armed hostilities we can expect a very dynamic environment, highly mobile threats, potential for high vulnerability, and a great deal of uncertainty about the logistics requirements. Therefore, the objectives, constraints, and structures of the logistics model will have to deal more realistically with the dynamics, uncertainty, and changing mission objectives of future warfare.

TABLE 3. Characteristics of Routing and Scheduling

1. Size of Available Fleet:

   One vehicle
   Multiple vehicles

2. Type of Available Fleet:

   Homogeneous (only one vehicle type)
   Heterogeneous (multiple vehicle types)
   Special vehicle types

3. Housing of vehicles:

   Single depot
   Multiple depots

4. Nature of demands:

   Deterministic
   Stochastic
   Partial satisfaction

5. Location of demands:

   At nodes
   On arcs
   Mixed

6. Underlying Network:

   Undirected
   Directed
   Mixed
   Euclidean

7. Vehicle Capacity Restrictions:

   Imposed (all the same)
   Imposed (different vehicle capacities)
   Not imposed (unlimited capacity)

8. Maximum Route Times:

   Imposed (same for all routes)
   Imposed (different for different routes)
   Not imposed

TABLE 3. Characteristics of Routing and Scheduling (cont.)

9. Operations:

   Pickups only
   Drop-offs (deliveries) only
   Mixed (pickups and deliveries)
   Split deliveries

10. Costs:

    Variable or routing costs
    Fixed operating or vehicle acquisition costs
    Common carrier costs

11. Objectives:

    Minimize total routing costs
    Minimize sum of fixed and variable costs
    Minimize number of vehicles required
    Maximize utility function based on service or convenience
    Maximize utility function based on customer priorities

2. SELECTION OF LOGISTICS TOOL KIT

In the past, logistics research has focused on developing algorithms to improve the efficiency of the logistics functions. Most of this research has been spent on basic algorithm development and their implementation on large computers. As a result, there are many efficient algorithms fully described in the open literature and validated by other researchers. Furthermore, many studies have dealt with the comparative evaluation of these algorithms.

There are some unique problems that have delayed the application of micro-computers in the operations research area. Practical logistics problems are usually complex and difficult, causing the complexity of the algorithms to increase and their databases to grow in size. Because of this complexity, little progress has been made in integrating the various solution techniques. Although the speed of the microcomputers has improved by orders of magnitude, they still have less speed and memory capacity than their mainframe counterparts.

Microcomputers,however, are cheap and flexible. They have been accepted as practical and useful tools for solving many home and business problems. Typical applications include spread sheets, word processing, data base management, small business accounting, etc. Microcomputer applications in scientific and engineering applications have been lagging, although there are some important exceptions, such as statistics packages and Computer Aided Design (CAD). Most of the important engineering applications have made use of the more powerfull 32-bit microprocessors configured as engineering work stations with considerable graphics support.

a. First Generation Software

In the Operations Research (OR) area we already find a number of programs that run on microcomputers. Mathematical programming applications include linear programming (LP), mixed-integer, quadratic programming, nonlinear integer optimization, and shortest path through a network. Operations analysis oriented programs include personnel assignment, cost improvement tracking, decision tree analysis, queueing analysis, flow analysis, and solution of transportation problem. The majority of these applications are construction management oriented. There are also a number of integrated project management programs complete with CPM and PERT support fully developed and documented.

These programs represent the first generation of microcomputer based software and, with a few exceptions, have rather severe limitations:

1. Programs are stand-alone packages
2. Programs are of restricted applicability (i.e., not clearly stated)
3. Programs are able to handle only a small number of variables
4. No provisions are made for integration with other programs
5. Programs do not support effectively the overall decision process
6. Programs perform suboptimization and do not permit easy iteration
7. Programs are excessively rigid; they cannot be modified by the user
8. Programs support only primitive file and data management capabilities

9. Most programs are written in BASIC language and are slow

In their present form, these programs are more suitable for familiarization with the available algorithms and solution techniques than for use in practical military or commercial environments.

b. Algorithm Selection Criteria

The Ship Loading Problem was used as the starting point in identifying the operations research algorithms to be included in the initial Logistics Tool Kit (LTK). As mentioned in the introduction, this selection process was relatively difficult but was accomplished by first defining the logistics operations and then identifying the applicable techniques.

The primary objective in the Ship Loading Problem is the efficient movement of goods from a depot to a cargo ship within the constraints of the available manpower and equipment. The primary input to the loading problem is a requisition list of goods to be moved with their destinations and priorities identified. Since only a limited space will be available in the cargo bay areas, selection of the goods to be moved will be in the order of established priorities. The objective will be the optimal utilization of the available cargo space subject to weight and other restrictions.

Modern transportation methods depend heavily on the use of standard containers. Thus, in the Ship Loading Problem we are not only concerned with the loading of the containers in the cargo bay, but also with the loading of the individual containers. It is apparent that these two cases are similar except for different size and weight constraints. This analogy could be carried down one or more levels to account for lower level packaging. The result is a hierarchy of containers, with the ship's cargo area being at the vertex. At any given level in the loading process we are concerned with only the visible container. Data at the lower level will not be needed at this point, but will be necessary later on for cargo distribution.

This example illustrates the interaction of the hierarchical data levels and the information needed for the loading algorithm. It is also apparent that the same algorithm, if properly designed, could be used at the various levels of packing and loading.

Another area of major importance is the allocation of resources. In the specific shiploading situation we are concerned with the allocation of material moving equipment and of operators to specific tasks. Again, there is a certain amount of parallelism. The allocation problem occurs in the warehouse where teams are working on picking the goods and packing them into containers. It also occurs at the loading areas where truck drivers and lift operators are to be assigned to their equipment, and where the ship is being loaded. These are different teams, usually under the control of different dispatchers, but the problems are similar. Again, there is considerable interaction. If the operations are not properly coordinated, delays will be experienced and time will be lost.

In addition to the allocation problem, assignment of specific operators

to specific equipment, constrained by equipment capabilities and operator skills, plays a major role. The solution of this problem also involves task priorities and availabilities. A related problem deals with the scheduling of the specific tasks -- again constrained by the overall system requirements.

Closely related to the scheduling problem is the routing problem. This problem also appears in different areas: in the warehouse, on the road to the cargo ship, and movement of material within the ship. Again we find that the same algorithm will be needed in different areas.

There are no available analytical methods capable of solving the general task scheduling problem. Nevertheless, there are a number of OR techniques useful for solving parts of the scheduling problem. These techniques coupled with heuristic methods and queueing simulation in an interactive environment can provide better solutions than conventional manual simulation. Most routing and scheduling problems of interest may be formulated as network problems. Special cases include:

Shortest path from node to node
Shortest path from node to all other nodes
Shortest paths between all nodes
K shortest paths
Minimal spanning tree
Capacitated minimal spanning tree
Transportation problem
Maximum flow problem
Minimum cost flow

Other routing and scheduling problems can be formulated as instances of a special class of 0-1 integer programs known as set partitioning or set covering problems. While set covering and partitioning provides a valid conceptual framework for the formulation of many logistics routing and scheduling problems, its practicality is limited to smaller problems.

At the aggregate sheduling level, linear programming algorithms may be used to select optimum resource allocations. At the detailed scheduling level, there are a number of applicable techniques, including Knapsack, maximum cardinality matching, and graph coloring in addition to some specialized network scheduling algorithms.

Since these algorithms are applicable only to parts of the scheduling process, it is highly desirable to be able to utilize a number of them in an iterative manner without the need for reentering existing information. The following data requirements are associated with a wide variety of routing and scheduling problems:

Distance and/or travel times
Transportation costs
Service requirements

Many of these data elements are fixed in a given environment and can be kept in a permanent data base. The data flow techniques discussed later in this report provide help in analyzing the creation and modification of data in

an interactive configuration where a number of solution modules are employed in the solution process. The use of man-machine interaction in the data base input is vital to the correct preparation and validation of data. The interactive capability also facilitates the incorporation of problem-dependent parameters and side constraints, and guides the algorithmic procedures.

At this point, the major logistics functions (allocation, assignment, scheduling, routing, and packing) have been identified. The next task is to find the applicable algorithms. As mentioned earlier, this task is not a simple one because there is not a one-to-one mapping from the functions to the algorithm as the case is in many other application areas. One approach that was found to be useful in this study was to look at the available OR algorithms and their potential use in assisting in the logistics support functions.

c. Initial Selection

Based on these considerations, an initial list of 28 algorithms were selected covering the various application areas. As expected, there was considerable overlap, but also differences in the problem definition and in the results available. Selection was made to obtain a wide spectrum of solution techniques, covering not only exact methods, but also approximations and heuristics. Some of the selected algorithms are relatively unsophisticated in comparison to the packages available on mainframes, but nevertheless will play a major role in demonstrating the integrated systems approach to solving logistics problems.

A brief description of each of the selected algorithms is included in Appendix A. This description also includes their potential applications in the logistics area and to the specific Ship Loading Problem. Some of the selected algorithms will be modified later to streamline their interface and change their operation. For example, the Traveling Salesman Problem in its unmodified form is not directly applicable to solving the routing problem. It was included as a starting point for modifications and to obtain benchmark performance results. The list presented in Appendix A is not final and should be considered only as the initial worksheet.

d. Exclusions

A number of important OR techniques were excluded from the list. These were decision analysis, search analysis, value theory, optimal control, nonlinear and geometric programming, and game theory and gaming. These were excluded because they either were not directly applicable to the Ship Loading Problem or were too complex to be incorporated because of the limited time available for Phase I effort. This set of techniques will be reviewed in more detail during Phase II and incorporated as needed.

e. New Developments

To solve the complexity problems, decomposition or aggregation techniques will be developed and implemented on microcomputers. Computer aids will

assist the operations research analyst in the decomposition process and in controlling the subsystem and system solutions.

To compensate for the slower computational speed of the microcomputers, faster algorithms will be selected, floating point processors used, and the need to access secondary storage will be reduced.

To compensate for the reduced data storage capacity, compact data storage techniques will be identified and special encoding used. Faster retrieval techniques will be used to improve response. Large programs will be divided into modules and used as program overlays.

To facilitate communications with other systems and research team members, a Logistics Description Language (LDL) will be developed to permit and unambiguous statement of the problem in a concise and precise manner. A neutral file format will be used for data interchange.

f. Extensions

Although the formulation of the problems may differ, similar problems are encountered in other related fields. For example, efficient techniques have been developed to aid in the design process of digital computers. Here, decomposition techniques, modelling and simulation are used on a routine basis. Powerful hardware description and simulation languages help to specify the structure and operation of the system. Other areas where decomposition techniques have been used include econometrics, sociology, chemical process design, civil engineering, electrical circuit analysis, data management, computer science, etc. Many of these techniques have potential application in the logistics area after suitable modifications and extensions.

The development of new or modified techniques to facilitate the implementation of logistics software on microcomputers will pay dividends in many areas. In the military area, the optimization of logistics operations will result in lower operating costs and increased combat readiness. In the business logistics area, we can also expect lower operating costs that in turn will reduce the product cost and improve competitive advantages.

3. LOGISTICS WORK STATION ARCHITECTURE

In many engineering areas, the interactive design approach is already well defined, and Engineering Work Stations serve as useful and efficient design tools. It is, therefore, important to examine some of the presently used techniques and discuss how they could be applied in the design of a Logistics Work Station.

In electrical, mechanical, and computer engineering, network diagrams form a convenient shorthand description of some very complex systems. Examples are: circuit diagrams, logic diagrams, mechanical system diagrams, etc. These network diagrams are exact in the sense that they formally describe a certain system configuration and usually give the specific parameter values associated with the specific components. These diagrams can be easily captured in digital form and the resulting database used to conduct complex analysis and simulation tasks.

A network diagram, which is a form of a network language, consists of a collection of symbols with topological relations (interconnections) among them, and alphanumeric annotations identifying elements by name and their values. A model to be solved, on the other hand, is usually expressed in terms of mathematical equations.

Since the network diagrams provide an exact description, they can be used with special translator programs to obtain the system equation set governing the given configuration. In other words, a translator program is used to derive all of the applicable equations. This is a time-consuming task if performed manually. Using this approach, a relatively inexperienced person can easily perform complex circuit analysis and simulation, provided that the input description is correct.

Although the present workstations differ in their implementation aspects, their structures are relatively standard. To adapt the Engineering Work Station concept to logistics problems, a detailed examination and evaluation of the logistics requirements is needed.

Figure 11 represents a very general and somewhat simplified representation of an interactive workstation. Only the major functional blocks are shown without any specific representation of the integral data base elements which are residing in Blocks 3, 5, 9, and 12. This was done to simplify the discussion.

A user interacts with the workstation by issuing a sequence of commands from keyboard, mouse, trackball, digitizer, etc. This sequence of commands is part of a very formal language that is well defined and usually tailored for a specific application. For example, SPICE language definition is part of an electronic circuit analysis program used by electronic engineers. In some cases, the same type of language is used not only to support the interactive but also the batch environment. This language is simple, yet powerful, and can be used to describe a variety of problems. It consists of problem description, and solution and output control commands.

The complete system is complicated, but easy to understand from a functional

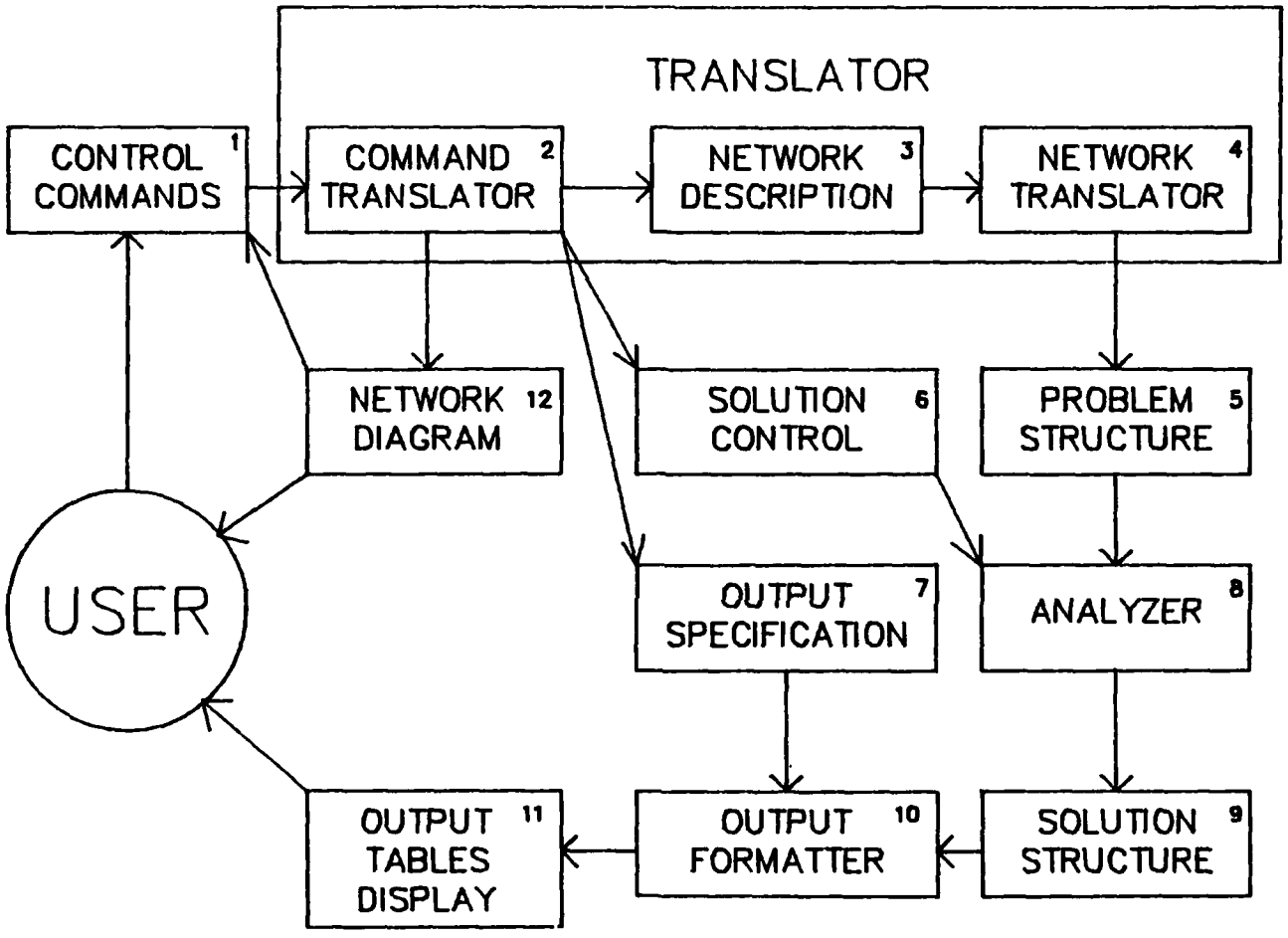


FIGURE 11. WORK STATION FUNCTIONAL FLOW DIAGRAM

viewpoint. It basically translates the user commands to form a graphical representation of the network diagram, sets up a data structure which describes the model to be solved (Problem Structure), and controls the solution and the output format of the solution so that it can be easily acted upon by the user.

The Command Translator (Block 2), determines the type of command (network description, solution control, or output formatting), translates it and transmits it to one of the processing blocks (Blocks 3, 6 and 7). It also keeps track of the network description entered (Block 12), and provides information feedback to the user and to the Command Processor (Block 1) for a consistency check.

The Network Description (Block 3) cannot be used directly because it is a net structure description and not a set of analytical equations. Therefore it is processed by another translator, Network Translator (Block 4), which converts the structure to a system of equations defining the Problem Mathematical Structure (Block 5).

The Analyzer (Block 8) contains the analysis or simulation procedures. These procedures are controlled by the Solution Control (Block 6) which contains the translated solution control commands issued by the user. These commands typically include the constraints, stopping criteria, etc.

Output results of the Analyzer are held in Solution Structure (Block 9), which the user can selectively examine using output specification commands that are processed by the Command and Output Specification (Block 7) translators.

The selected output is presented to the user in either tabular form or as a graphics display. Block 11 handles these display tasks by receiving output from Block 10.

The user feedback comes from two sources: problem definition and solution. By having the problem definition available, the user is able to review the existing definitions and to make any necessary modifications without having to reinput all of the problem definition data.

There are a number of problems faced in designing a Logistics Work Station, the most severe being the lack of a good problem-description language. Some of the techniques developed in the past, such as Markov process diagrams, PERT, GERT, activity diagrams, etc., already have some of the features needed for a formal network description, but considerable work will be required to develop a Logistics Description Language capable of handling a majority of the logistics problems. Another problem is that of designing the needed translators capable of taking the shorthand network descriptions and converting them to a set of equations to be processed by the computer solution procedure modules.

In Phase I, emphasis was placed on developing a subset of this language needed for the shiploading problem. Phase II work will involve the initial development of a general purpose descriptive language and a full implementation of the Logistics Work Station similar in structure to that shown in Figure 11.

The translators used in engineering workstations are similar to the traditional language compilers in that they contain both syntax parsing and semantic interpretation (code generation). They differ, however, in that they also have to parse solution control and output specification commands. The Logistics Work Station, in addition, will contain additional capabilities to facilitate the data transfer. These techniques will be discussed later in this report.

A formal framework is needed for the translator programs to answer four basic questions:

1. What useful information is directly conveyed by the network diagram?
2. What information is needed by the analysis program?
3. What useful information is indirectly conveyed by the symbols?
4. What is the algebraic procedure corresponding to the transformation from a network diagram to a problem structure?

Again, these questions are similar to those defining the programming languages. The first concerns the syntactic description of the input language, the second calls for syntactic definition of the output language, the third calls for semantic definition of the symbolic operators, and the fourth calls for specification of the translation procedure.

For the logistics applications, the network diagram must be be able to represent the basic operations inherent in a logistics process in a short-hand notation. The analysis programs will receive their inputs from the network diagram translator. The selection of symbols to represent logistics operations will be a difficult one. For the purpose of this study only those symbols needed for the shiploading process will be proposed. The ease of the translator design will depend greatly upon the symbol set.

4. LOGISTICS DESCRIPTION LANGUAGE

a. General

Logistics Description Language (LDL) is a system of symbols needed to describe logistics data and logistics activities. It is needed to acquire knowledge and to communicate information, and to control logistics operations. Since the logistics information environment includes humans and computers, the description language must serve a dual purpose and be both human and machine readable.

In any language, we can distinguish certain basic elements. These are vocabulary (words), grammar (syntax), and associated meaning (semantics).

Every language is based on a vocabulary. In the conventional language, its elements are called words. In the formal language, the basic elements are called symbols. In every language, therefore, we can form sentences out of these basic elements.

Syntax is a system of rules of grammar that controls the order and harmonious arrangement of parts or elements of sentences, phrases, and clauses. This system governs how a sentence is formed and, further, controls its structure so as to permit recognition of its meaning (semantics).

The basic character of a language is determined by the data types on which it operates and how changes in their values are made. Natural languages are very rich in their variety of words since they have to express all that we have to communicate. At the same time, some words may have a number of different meanings and their interpretation may depend on the context and on the environment. Because of the lack of well-enforced formalism, natural languages are not well suited to problems involving man-computer type environment.

To communicate with computers, we have to use artificial languages that have strict rules and unique meanings. To communicate this information to the users, the language should be easy to understand, yet flexible enough to express everything in that particular environment.

Since the beginning of computers, major efforts have been spent on language development. These computer-based languages range in complexity from very simple ones (machine language) to the very complex (simulation languages). One characteristic they all share in common is their formalism and their suitability to a particular environment. Although initially there were attempts to find the one truly universal language, these efforts were not successful and today we select a language to fit the problem.

A particularly important area in language development is that which deals with Higher Order Languages (also referred to as Problem Oriented Languages). These languages have been developed to assist the user in specifying the problem for a computer solution. In this area expressibility and efficiency of representation is of major importance. Today these languages can be found in almost every field. Examples range from word processing and spread sheet command languages to the complex computer hardware description and simulation

languages.

An important factor to remember is that all of these languages are highly specialized and fine tuned for a particular application. Usually they cannot be efficiently used outside their originally intended use. For example, one would not normally use a programming language to write a progress report or a word processing program to solve a mathematical equation. There are some exceptions; for example, some word processing programs may include facilities for simple mathematical computations. More recently, a number of integrated program packages combining word processing, spread sheets,and graphics under one structure have emerged.

In the proposed implementation, the LDL will serve as an intermediate language. As such, it will have to be processed and translated to another language which will be used to describe the computational process. The latter will be a conventional programming language, such as Pascal.

b. Logistics Description Language Processing

The LDL processing model is illustrated in Figure 12. The individual processors are further detailed in Tables 4-7. The overall process is very similar to that of a typical programming language preprocessor. Throughout the process, care will be taken to capture errors and to provide user diagnostics. Since the system will be designed to be used in an analytical and operational environment, diagnostics and recovery procedures will be more user-oriented than those in a programming environment.

The development of an LDL does pose some unique problems. First, the present day logistics environment is very much paper-form oriented. Although the actual forms are usually prepared on a computer system, many of the intermediate logistics tasks are performed manually. As a result, there are many manual breaks in the logistics order-processing sequence and considerable manual effort is required to interface with the various programs.

The most commonly used formalism for expressing grammar rules is the Backus-Naur-Form (BNF). This formalism was originally used in the definition of ALGOL 60 language. It can be best illustrated by considering a simple sentence structure in natural English.

Taking a simple sentence such as

The analyst solved the difficult problem

we can analyze it by breaking it into its components: subject phrase, verb, and object phrase. Further decomposition of the subject phrase and object phrase can be obtained.

This process can be represented in the BNF notation as follows:

<sentence> ::= <subject phrase><verb><object phrase>
<subject phrase> ::= <definite article><noun>
<object phrase> ::= <definite article><adjective><noun>
<definite article> ::= the

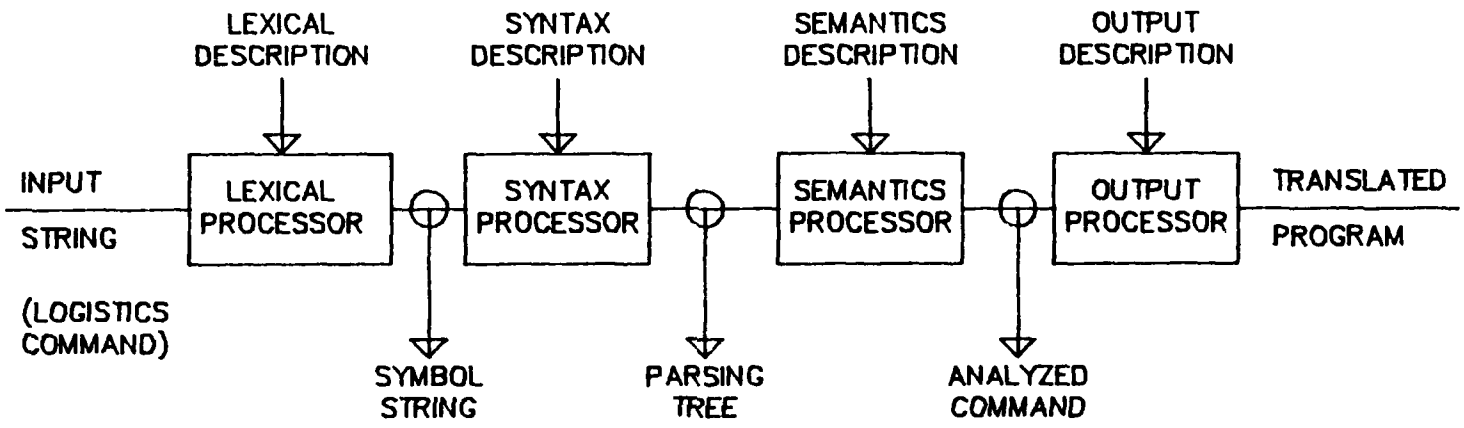


FIGURE 12.

LOGISTICS DESCRIPTION LANGUAGE PROCESSING MODEL

TABLE 4. LEXICAL PROCESSOR

Processor Type:

LEXICAL

Purpose:

Key interface to the language processors. Separates incoming character string into symbols. Has the capability to distinguish symbol separators: spaces, punctuation marks, etc.

Input:

Character stream representing Logistics Description Language commands from an interactive user terminal or text file.

Output:

Separated symbol stream representing valid Logistics Description Language symbols.

Error diagnostics: invalid text characters, invalid symbols, etc.

Control Input:

Regular expressions describing how the characters are to be assembled into symbols. May contain some preliminary conversion capability: conversion to upper case, elimination of text file control characters, etc.

TABLE 5. SYNTAX PROCESSOR

Processor Type:

SYNTAX

Purpose:

Parses the incoming symbol stream to identify the structure of the command statements. Creates parsing tree of the command containing a structural representation of the command.

Input:

Symbol stream representing valid Logistics Description Language symbols.

Output:

Structural representation of the Logistics Description Language command.

Error diagnostics: invalid or incomplete tree structure.

Control Input:

Valid syntax description for each Logistics Description Language command using standard notation.

TABLE 6. SEMANTIC PROCESSOR

Processor Type:

SEMANTIC

Purpose:

Interprets the structural representation of the command statement. Identifies the need for processing module calls. Sets up the necessary call linkage.

Input:

Parsing tree identifying the specific command to be processed in standard tree format.

Output:

Call statements to processing modules and specific data statements, as needed.

Error diagnostics: incomplete command structure or invalid command description.

Control Input:

Command processors for interpreting the specific command. These processors contain linkages to other processing modules needed to further expand the command.

TABLE 7. OUTPUT PROCESSOR

Processor Type:

OUTPUT

Purpose:

Interfaces the Logistics Description Language compiler to the external world. Extracts and converts solution to specified output format. Presents results.
Identifies problems associated with the solution: lack of convergence, incomplete specification, etc.

Input:

Symbolic logistics model from the semantic processor specifying the necessary computations to be performed.

Output:

Results of the specific logistics problem.

Error diagnostics: identifies invalid problem.

Control Input:

Links to the logistics processing modules, such as resource allocation, assignment, scheduling, etc.

<noun> ::= analyst
<noun> ::= problem
<adjective> ::= difficult
<verb> ::= solved

The nonterminals (those subject to further decomposition) are enclosed in angle brackets to distinguish them from the terminals (words or symbols).

Each of the rules in the above list is called a production and it describes the composition (or decomposition) of the nonterminal (sentence).

The above sentence can also be represented in a parsing tree (graph) form as shown in Figure 13.

The above example illustrates only one of the many possible natural language sentence structures. Because of the infinite variations in the natural language it cannot be used directly as a Logistics Description Language. It does, however, provide a starting point because the artificial language we will be developing has to be human readable. The closer it relates to the natural language, the easier it will be to understand it.

c. Logistics Description Language Implementation

As indicated earlier, any language development effort is a major undertaking and cannot be expected to be completed within the constraints of this study. Since LDL will play a major part in the development of the Logistics Work Station, examples of typical LDL sentences will be shown.

The key areas where LDL will play a major role will be in logistics resource specification, task definition and control, and material-oriented operations. Typical examples from these areas follow:

(1) Resource Description:

<pers_skill> ::= PERSON [personnel_id> HAS SKILLS <skill_list>.

<skill_list> ::= <skill> <skill_list>.

<equip_capability> ::=
EQUIPMENT <equipment_id> HAS CAPABILITY <capability_list>.

<capability_list> ::= <capability> <capability_list>.

<assigned_pers> ::= ASSIGNED_PERSONS <personnel_list>.

<assigned_equip> ::= ASSIGNED_EQUIPMENT <equipment_list>.

<idle_pers> ::= AVAILABLE_PERSONS <personnel_list>.

<idle_equip> ::= AVAILABLE_EQUIPMENT <equipment_list>.

<unavailable_pers> ::= UNAVAILABLE_PERSONS <personnel_list>.

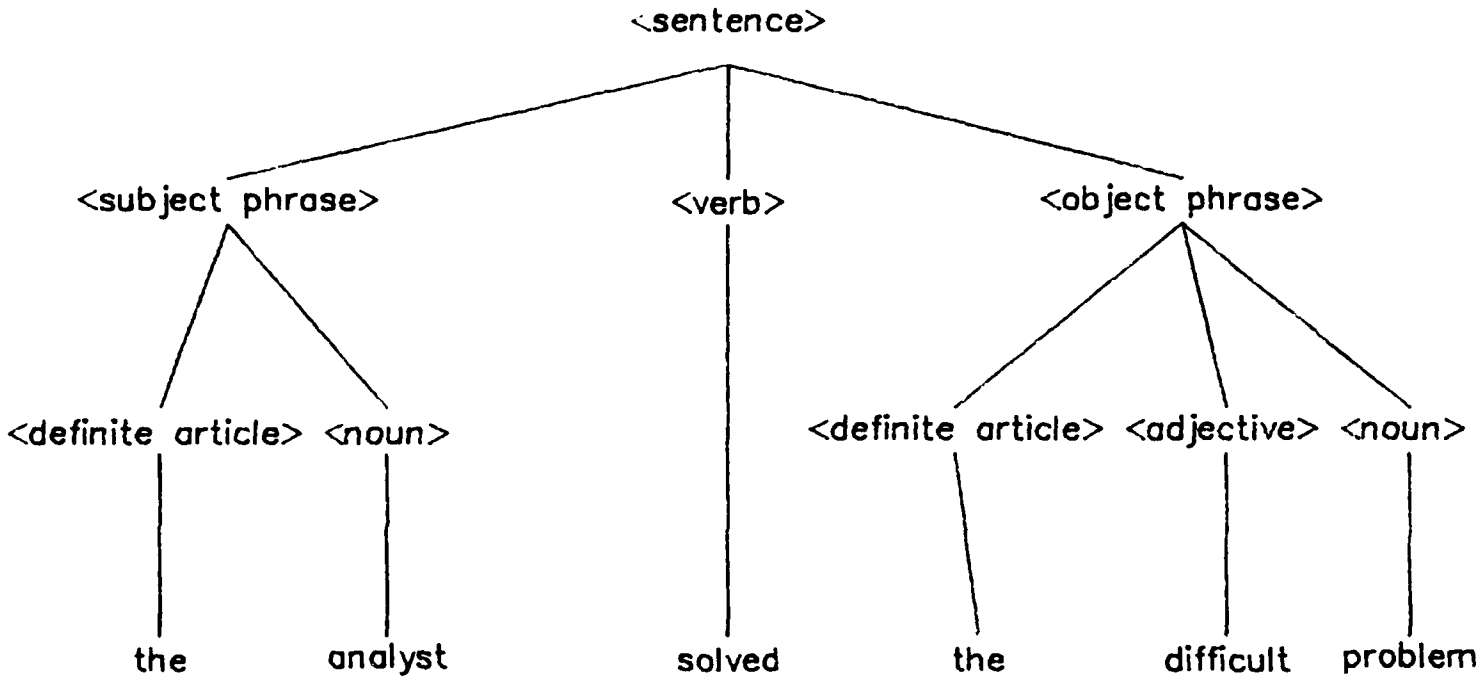


FIGURE 13. ENGLISH SENTENCE PARSING TREE

<unavailable_equip> ::= UNAVAILABLE_EQUIPMENT <equipment_list>.

<personnel_list> ::= <pers_id> <personnel_list>.

<equipment_list> ::= <equipment_id> <equipment_list>.

(2) Material oriented operations:

<start_symbol> ::= <ship_order>.

<ship_order> ::= SHIP <item_list> TO <destination_id> .

<select_order> ::= SELECT <item_list>.

<move_order> ::=
MOVE <item_list> FROM <source_id> TO <destination_id> BY <carrier_id>.

<packing_order> ::= PACK <item_list> INTO <container_id>.

<hold_order> ::= HOLD <item_list> AT <location_id>.

<item_location> ::= ITEM_LOCATION <item_id> AT <location_id>.

<item_list> ::= <item_id> <item_list>.

(3) Task Oriented Operations:

<task> ::= DEFINE_TASK <task_id> AS <operation_list>.

<assignment> ::= ASSIGN <personnel_list> AND <equipment_list> TO <task_id>.

<start_task> ::= INITIATE_TASK <task_id>.

<suspend_task> ::= SUSPEND_TASK <task_id>.

<release_task> ::= RELEASE <task_id>.

<operation_list> ::= <operation> <operation_list>.

(4) Parsing Tree

A sample move order parsing tree is shown in Figure 14. Here we can distinguish the keywords (MOVE, FROM, etc.) and the terminals (WHSE1, SHIP1, etc.). The keywords improve the command readability and at the same time will provide more reliable decoding because of their uniqueness.

As mentioned above, the overall parsing tree is more complex because it may include certain processing between sentences. As an example, consider a typical ship order. The order will carry its associated material list and the destination where this material will be needed. To process the ship order, a locator module will be needed to locate the warehouse where the specific material ordered is located. The locator module then will in turn create a series of move orders if the material is in different warehouses. An example

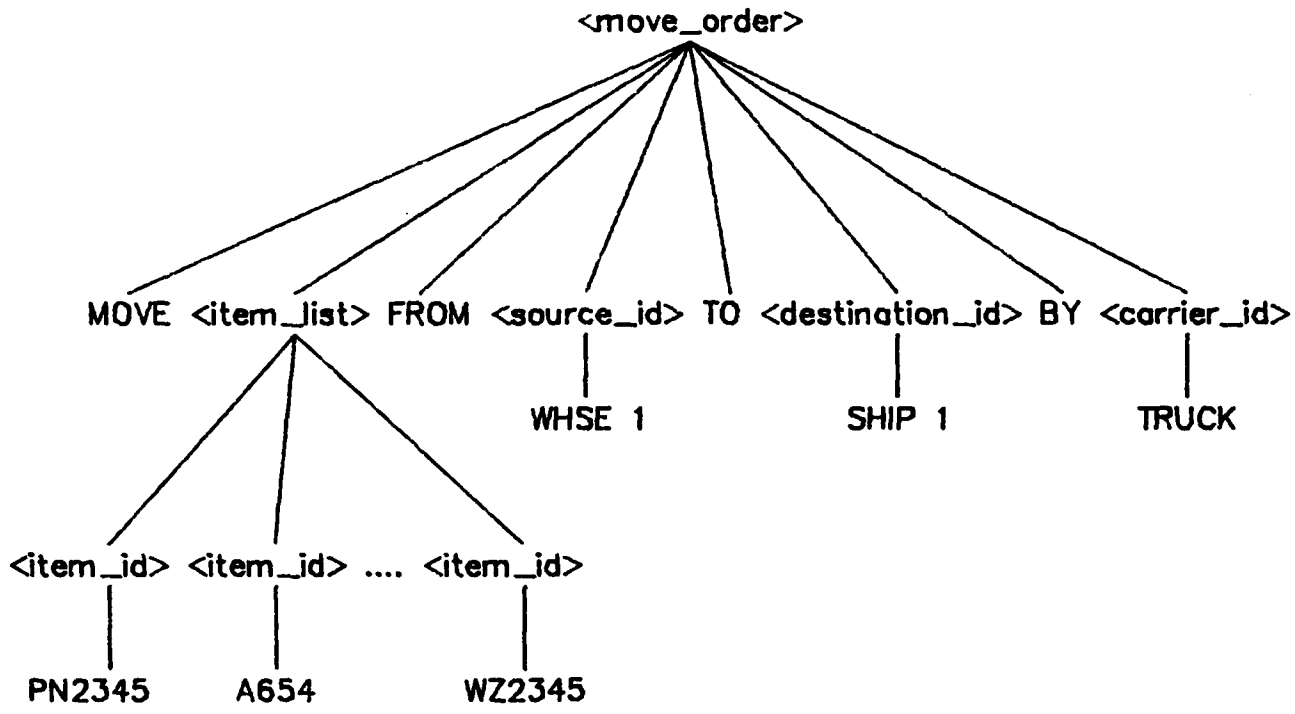


FIGURE 14. MOVE ORDER PARSING TREE

of this process is illustrated in Figure 15.

Other processes may exhibit even more complexity. Consider the task-assignment process. As shown in Figure 16, there are complex interrelationships between logistics resources and material move requirements. It is in this area where some of the recent developments in knowledge engineering could offer new solutions.

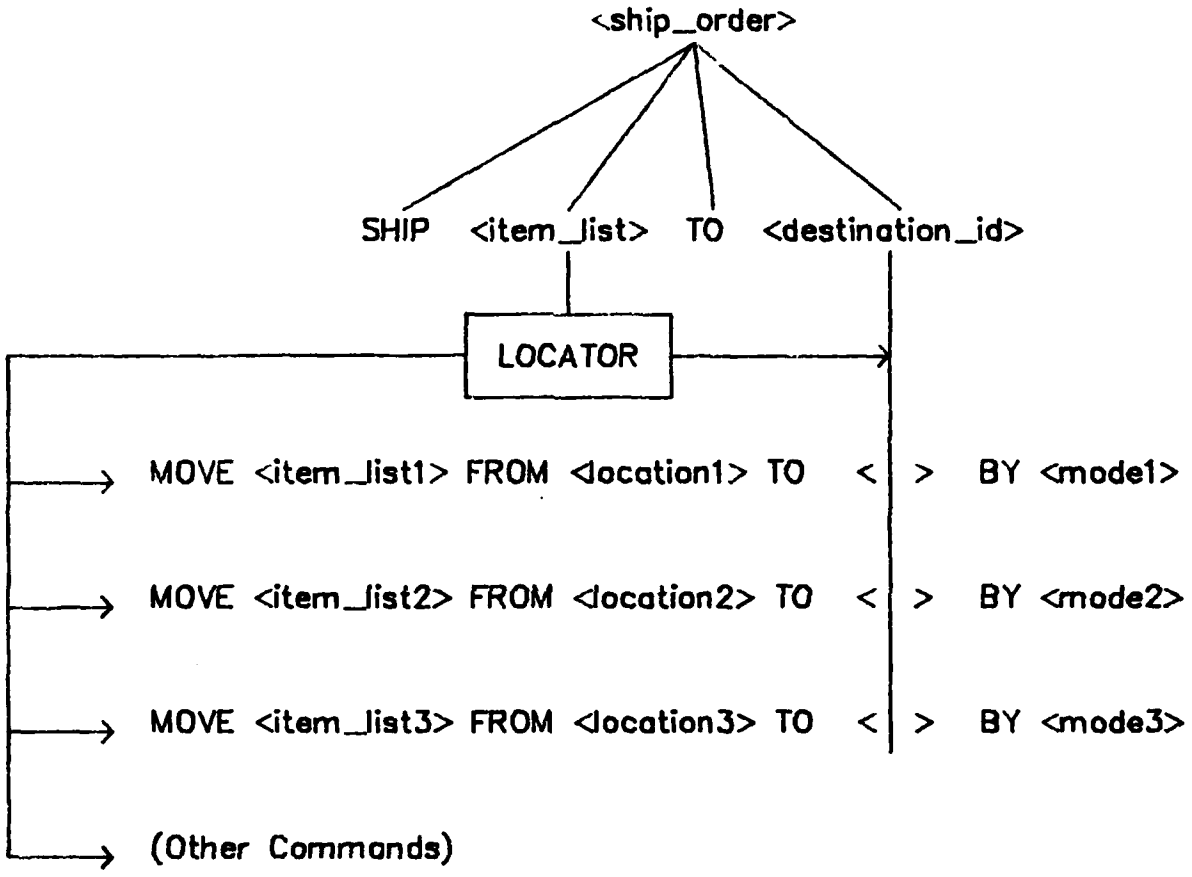


FIGURE 15. SHIPPING ORDER DECOMPOSITION

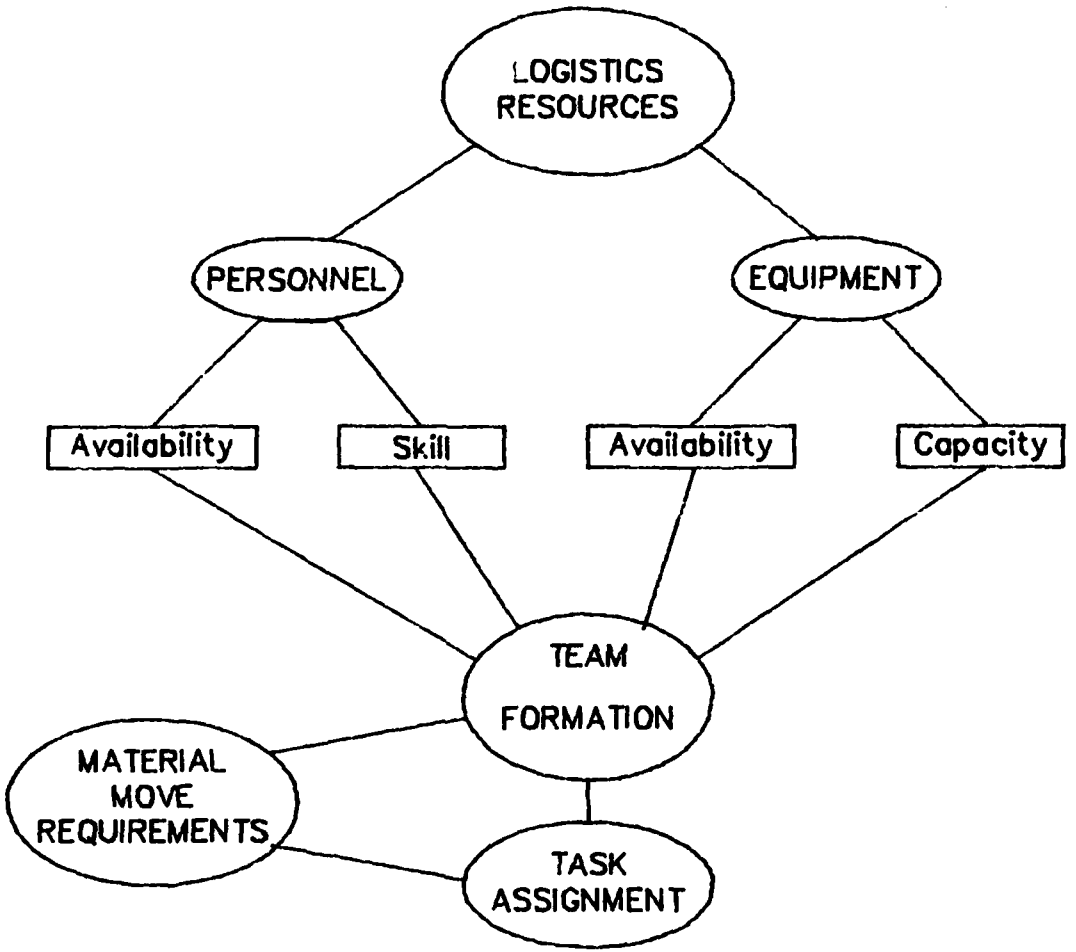


FIGURE 16. TASK ASSIGNMENT PROCESS

5. LOGISTICS DATA BASE REQUIREMENTS

a. General Concepts

Logistics problems are not only complex but also need a large amount of data for the specification of variables and constraints. Compact data storage and fast retrieval techniques will be needed to provide the required support in a microcomputer-based environment instead of the "brute force" techniques used in mainframes.

Efficient solution of logistics and operations research related problems need a variety of specialized data structures usually not available in the conventional data base management systems. These structures include:

Graphs (directed and undirected)
Trees
Linked Lists (including bidirectional)
Stacks
Queues

Although it would be possible to use generalized arrays to simulate the above data structures, the loss in computing speed and the increased complexity of algorithm formulation prohibit this approach in practical situations.

Furthermore, the data base orientation should be dynamic in nature, not static. This means that the data base should be capable of supporting changing record structure requirements. In the conventional data base systems, each change in the record structure is accompanied by unloading the data base, incorporating the new changes in the record structure, translating the data to the new format, and then reloading the data base. As the size of the data base increases, this process can become very time consuming and impractical.

Other important data base factors include fast record update processing time and fast query access to the data base. These are needed because in the proposed system configuration the majority of the accesses to the data base will be from computer programs while only a relatively few requests will be directly user-initiated. This requirement again calls for a design different from the conventional commercially available data base systems developed for microcomputer applications.

Each of the special data base requirements identified above will be discussed in more detail and solutions proposed to reach the desired goal.

The general approach used in the data base development will be based on the same principles used in other areas of the project: i.e., partition and decomposition.

b. Efficient Data Storage

Most of the input files used in microcomputer applications are of the text-file type containing ASCII character strings. Although these files are easily read by the user, they are inefficient in storage density and require translation to the binary form for processing numerical information. Promising solutions in this area include the use of binary format for numerical values and use of data compression techniques for text.

There are a number of problems associated with this approach. First, if nonstandard binary representation is used for numerical files, then the interchange of data among the users will become a major problem. A simple solution to this problem would be to utilize the available standards, such as the IEEE standard for floating point representation. Second, the data compression techniques for text have not yet reached a mature state where widely accepted standards are available. Therefore, a full ASCII-type text transfer may be preferable to eliminate transfer problems.

c. Fast Retrieval Techniques

As mentioned above, retrieval speed is of vital importance in solving the logistics problems. There are two approaches to this problem: the first approach is based on using faster hardware, and the second depends on software techniques.

Microcomputer storage peripheral performance is continuously improving. This includes more capacity, faster access time, and lower prices. Faster retrieval can be obtained by using hard disk drives or random memory access files for frequently needed data. With decreasing hardware costs both approaches are practical.

Even more improvement in retrieval speed can be obtained using software based access algorithms. If a conventional sequential file is used to store information and if this file is scanned from the beginning to the end to find the needed information, then access time will be proportional to the file size: O(n), where n is the number of records. Using the more recent techniques, such as the B-Tree storage concept, search time shows considerable improvement: O(log n). Of course, this improvement does require additional storage needed for handling the more complex structure.

The use of dynamic data base structures has received major attention in the electronics design area, particularly with VLSI design where the data bases typically contain close to a million data elements. At the present time there are about 20 different search tree storage schemes that have been proposed. Many of these have a direct application to the logistics area and will be further evaluated during the Phase II effort.

During the course of this project, only the B-Tree storage concept was investigated because it has been used in more applications and its implementation is better understood. All of the preliminary tests have indicated excellent performance with relatively small overhead for the additional

storage structure.

d. Data File Formats

Present microcomputer operating systems, such as PCDOS, MS-DOS or CP/M, support two different file concepts: sequential and random file. The specific selection of a particular format is dictated by the appplication. The sequential format is more appropriate where the complete range of data is being generated, whereas the random file format is preferred where selective retrieval and updating is needed.

Again, it is important to realize that the processing speed can be influenced by reducing the data translation requirements. For example, by storing numbers in their binary representation instead of using ASCII, or by storing complete records instead of storing separate elements, data transfer will be greatly improved.

Especially efficient formats can be used to further speed up processing such as the use of the general file concept in Pascal.

e. Operating System File Interface

The performance of the operating system file interface with current systems, such as PCDOS or CP/M, is very poor. There is a large system overhead and minimum flexibility. Problems include: small buffer size and buffer handling, inability to use reentrant system procedures, minimal user control over the I/O process, etc. The available error recovery procedures are minimal.

There are no simple solutions in this area. Most of the users have accepted these limitations as a fact of life, and the operating system developers have shown little interest in improving the operating system performance. Bypassing the operating system and developing special file drivers would cause incompatibility problems with other programs; this has to be ruled out.

The only hope is that in the future these problems will be better understood and that the operating system developers will provide the needed solutions.

f. Data Structures

As indicated above, an efficient solution to the majority of logistics problems requires the use of specialized data structures. Some of these structures and their means of implementation are discussed below.

Undirected and directed graphs find wide application in transportation-type problems where they are used to indicate feasible points between different locations. They also play an important part in depicting the variable interdependency in large and complex situations. Other applications include flow problems, matching, scheduling, and others. Although it is possible to use matrix formulation, such as incidence and adjacency matrices, it

is very inefficient from a the computational standpoint because these matrices are usually very sparse. Using the linked-graph formulation, the evaluation of the algorithm is speeded up considerably because no time is spent processing zero elements. The implementation of graphs is very simple in Pascal language because of the availability of records and pointers. The same implementation in FORTRAN would be more difficult because pointers would have to be implemented in arrays.

Undirected and directed trees form another set of important data structures in logistics problems. A familiar application involves spanning trees used to solve optimal communications network configurations. The implementation problems of tree structures are very similar to those of graphs discussed above. Again, Pascal has a definite advantage as an implementation language.

The implementation of linked lists, data queues and stacks form another similar structure group. Again all of these structures are easily implemented in Pascal. Since ADA data structures are similar to those of Pascal, the future program translation will be much easier than the equivalent conversion effort from FORTRAN.

Review of the various solution modules revealed that additional savings in response time could be gained by using these specialized data structures in intermediate files. (The current implementation uses standard text files for data interchange between modules.) This would involve setting up data structure standards for the specialized forms and would require a relatively major effort to arrive at a set of general standards. No data standards are available for these forms.

The efficient solution of LP programs requires the use of better matrix processing techniques. A number of practical approaches based on sparse matrix techniques are available for use on mainframes. It appears that some of these could be converted without major modifications for use on micro-computers. Time limitations prevented further investigations in this area. During Phase II implementation effort, these techniques will be given further consideration.

g. Data Abstraction

The approach used in developing large and complex programs is usually based on the "divide and conquer" principle: the program is divided into a set of modules each of which can be viewed as a simple abstraction. The same concept can also be applied to the data base development where modules of data are defined and given abstract names.

There are many advantages in data abstraction; the key ones include easier understanding of the data structures, faster development, and easier maintenance. The modern programming languages, such as Pascal, Modula-2, and ADA, provide many features to make data abstraction implementation easy and practical. These features are not available in the older languages, such as FORTAN and BASIC.

Typical implementation examples include records and data typing. Both of these

features were used to the maximum extent during the Phase I effort and will form the basis for Phase II work.

h. Dynamic Record Structure

The key difference between business and scientific problems is that the business data structure is static, and the data structure for scientific problems is dynamic in nature. Business examples include payroll structure, organizational structure, etc. Scientific examples include task allocation, scheduling, assignment, etc., where the task structure may continuously change depending on the complexity of the task.

Conventional data base management systems developed for business applications are ill-suited in scientific and engineering problems. Lack of understanding of the basic capabilities of these conventional data bases has resulted in many aborted application programs in engineering and manufacturing areas with considerable loss of money and wasted manpower. The same lack of understanding, unfortunately, exists today with the same predictable results.

The data base support needs in logistics are dynamic and user-defined. The only form of data base organization capable of supporting this environment is the relational one permitting many different forms of records to coexist in a dynamic environment. The fifth-generation languages, such as PROLOG, also provide excellent dynamic data base support.

The proposed approach for the Logistics Work Station will therefore be based on the relational concept.

i. Data Base Interface with Other Subsystems

Data base interface with dialog and control subsystems is greatly simplified by the use of the modular approach. Separation of the dialog functions from the data base will permit the development of highly flexible user interface to deal with the data base. Use of the data flow concept will permit the linking of data base status with the main solution control.

j. Data Presentation

In a practical situation, the user will require different data presentation techniques. These will range from the simple graphical displays of bar charts to the more complex and comprehensive printed records.

Data presentation will be coordinated via the dialog session where the data conversion modules will form the link between the data base and the actual display of the data.

The clarity and understanding of the results will be of key importance. Presenting the results in a well understood format, such as used by microcomputer based spreadsheets, will in general be preferable to the more

difficult-to-interpret data listings.

k. Other Considerations

A number of other considerations will affect the final data base implementation. These include multi-user and distributed-file environment, remote access capability, local area networking, etc. Although these factors are very important, the current operating system support is not yet able to provide the needed capabilities for the implementation of an efficient and practical system. Work in developing extensions in operating system support is proceeding at an accelerated rate and some of the needed features will be probably available in MS-DOS Version 4.0, or later.

The separation of the control, processing, data base, and dialog modules anticipates future needs and will provide the basis for the needed extensions without a complete redesign of the system.

l. Conclusions

The logistics problem data base has to be compact, capable of supporting fast retrieval, and must be capable of supporting dynamic user-defined data structures.

The proposed data base configuration is based on the relational data base concept to support the dynamic structuring; it uses specialized data structures to reduce storage requirements and has a tree-oriented retrieval structure to permit fast retrieval.

6. USER INTERFACE REQUIREMENTS

a. General Concepts

The basic objective of this research project is the efficient integration of microcomputers within the logistics environment.

In the common process of system design, the user is often not taken into account, thus making these systems difficult to use, increasing training costs, causing worker stress, and usually leading to system early retirement.

In the end-user concept of design, the user is regarded as the most important factor in the man-computer system. Systems created using this concept are usually known as user-friendly and, in most cases, are well accepted.

A system is user-friendly if the individual user feels that the subjective complexity (his external view of the system) is low. It also means that the interface must be flexible and adaptable to different system users and their needs. To adapt to different users, the system must be able to distinguish among them and to support their individual requirements. In every case, the special skills needed to operate the system should be minimized.

Flexibility of use means that the user has flexibility in the delegation of tasks to the computer. He has the choice to select only those features of the program that are needed for his application. Flexibility also means that the user does not have to act in a strictly prescribed way.

Man-machine dialog should resemble human communication insofar as possible. The proposed Logistics Description Language will be designed to meet this requirement. The fully developed language will include, in addition to commands, facilities to incorporate user-supplied descriptions, capability to ask questions, and to make comparisons.

The language will be layered in two dimensions: it will provide layers or functions so that users need learn only those relevant to their problems, and, within the sets of functions, there will be layers of complexity and power so that users can progress from the basic levels to the more complex as their skills increase.

The multilevel approach will permit the user to switch to a more terse input format as he gains familiarity with the system operation and will provide faster operation.

The advantage of computer-guided interaction is that the language used can be natural (or almost natural) language, as opposed to artificial language; thus, easily understood by the user. Further, inputs by the user to the computer, using natural language, will be easy to interpret with a simple computer program.

The system will behave in a consistant way so that the user can learn to

anticipate its functions. Similar functions will be invoked in a similar way with the results presented in similar format. This will help in the habit formation and will enable the user to learn the system quickly.

b. Design Philosophy

(1) General

The techniques used for the user-interface design will follow those proposed earlier for the system design: partition and hierarchical decomposition.

The partition process will be applied to separate the dialog part from the process algorithms. The dialog part will be further decomposed in a hierarchical manner to introduce another level of structuring. There are a number of advantages in using this approach:

The algorithmic part will contain only computational procedures. All the specific details associated with input/output and data base interface will be handled by the dialog modules. These will include checking the input, determining if sufficient information is available to start a solution, and providing the associated help functions.

The dialog modules can be altered to adapt to new hardware or to a different system configuration without the need of changing the algorithm module. This will result in decreased maintenance costs.

Separation of dialog from the algorithm will also help in the system development by having experienced dialog specialists concentrate on the dialog part without the need for understanding the intricacies of the solution algorithms. On the other hand, the programmer experienced in computational algorithm design will be able to concentrate in his area of expertise and be relieved from concern about the specific dialog.

Many of the dialog modules will be similar in form. This will permit the establishment of dialog libraries and further reduce the development effort.

The dialog modules can be further extended to provide additional facilities for user help, error recovery, and audit trails.

The programming method for the dialog will require detailed, readable, and interpretable specifications. Information for this task will be obtained from the algorithm module specifications which will also contain the input/output requirements needed to establish the necessary interface and to specify the needed peripheral devices.

(2) System Users

To further define the dialog module requirements, it will be necessary to examine the potential users of the system and their needs. Here we can distinguish five major groups of users:

System Programmers. This group will be concerned with the structural development of the system. They will have a good understanding of the system structure and will provide for the incorporation of dialog modules. During the development phase, they will use prototype modules for checkout. Their needs for specialized dialogs will be minimal.

Application Programmers. This group will include dialog and algorithm developers. They will work closely with the system programmers during the development effort. Again, prototypes will be used during the initial phase. Specialized dialogs will be provided for this area to aid in module integration (system generation).

Operations Analysts and Researchers. This group will need the greatest flexibility in dialog types because their tasks will cover a wide variety of problems. Initial dialog developmental effort will concentrate on this group.

Data Base Technicians. Their tasks will be associated with updating and modifying the data base. Most of these tasks will be of a routine nature and will use short dialog formats.

Casual Users. This group includes those system users who will not use the the full system capabilities, but will need access to parts of the system. Task schedulers and dispatchers, for example, will use scheduling modules. Dialog requirements for this group will stress ease of use and clear presentation of output.

Thus, for each of the user groups described above, we will need concrete knowledge about their tasks, their needs, and their behavior. We will also have to establish a description of the possible dialog concepts and their impact on the user.

In general, it will not always be possible to attach a certain dialog concept to a specific group, although it may appear to be the simplest. For some of the groups, particularly analysts and researchers, a variety of different dialog concepts will have to be developed to suit their specific needs. At the same time, by introducing additional flexibility, the task of learning the system will become more difficult.

(3) Task Structures

Next we will examine some typical dialog task structures:

Simple inquiry. This is one of the simplest dialog types and requires a simple response to a well-posed question.

Decision making following complex inquiry. In this dialog, a sequence of questions is followed by computer response. For example, in a linear programming problem, the dialog would be used to establish the data structure and the objective function. After the problem specification, the solution module would be activated and the results presented to the user.

Collecting data systematically. As mentioned above, this is one of the simplest dialog formats since it will require a minimum of computer-user interaction. A possible option could be the online checking of the data range.

Continuous design and redesign on the basis of rules, constraints and work already completed. This is the most complex dialog structure and involves full access to the existing data base. The modular design of the dialog modules will make this task easier to implement.

Discovery of relationships. This category includes many of the statistical techniques used in forecasting. The dialog considerations for this category will be a mix of the last three groups above.

c. Dialog Modules

(1) General

The language construct that will be presented here for specifying a dialog procedure is called a dialog module. A dialog module is a unit which can completely specify one step in the dialog. Each step contains the following parts:

Action from the user

Corresponding external reaction from the system

Effect on the internal state of the system

Environment and conditions in which the action takes place

Dialog modules will be organized in hierarchies. These hierarchies will allow for the specifications for dialog syntax (sentence specification), semantics (system response), and subdialog module interface.

(2) Module Activities

The four groups of dialog module activities are specified in four separate submodules that make up the dialog module:

Prompt initializes the subdialog and activates all required submodules. It also informs the user that the system is ready to accept an input. Prompt may range from a simple continue cue to a full screen setup. It specifies who asks the question and who provides the answer.

Symbol or input sentence. It will specify how the input will be read and when the module input is completed.

Echo provides a message to be displayed to the user after the acceptance of a symbol or input sentence.

Value. Each accepted symbol has an associated value. This is interpreted

by the system. The global and local value schemes are controlled through the module hierarchy. The module will usually include input validity checks.

d. The Basic Cycle of a Dialog Call

When a dialog module is called up, its four components become active. Each section in turn may contain names of submodules. The restriction here is that the submodule calls refer only to the specific component of that submodule. For example, if prompt names another module, then only the prompt part of that module is referenced.

Input (symbols or sentence) can only be read when the module is active, i.e., prompt section must be executed before parsing. This guarantees that the value expressions are evaluated after all values needed have become available.

After the delivery of the external results, the module is ready for a new complete basic cycle. Whether this will happen or not is controlled by the surrounding module.

A dialog module is either activated by the algorithmic part as a dialog procedure (root module), or it is activated as a submodule of a dialog module closer to the root module. This activation can be either synchronous or asynchronous. In the synchronous case a module is activated just prior to the action by the symbol part to accept this symbol. The activating dialog module is forced to wait until the symbol asked for has been produced. The module is deactivated immediately after its symbol has been accepted.

The symbol part of a dialog consists of a grammar rule. This rule directly controls the input parser.

A dialog is called self-descriptive when the purpose and method of usage can be explained during the dialog at the request of the user. The type and extent of the explanation will be geared to the user, and defined by the actual application and the user classification.

e. Interface Design Requirements

(1) General

This subsection considers the most important requirements to be followed during the system interface design process.

(2) Dialog Flexibility

(a) General

The dialog must be designed to be flexible and adaptable to the different needs and skills of the potential users. It must also support a flexible task structure which can be extended if needed.

The user will influence the dialog by choosing the functions to be performed, switching between different dialog types, and selecting or rejecting options offered by the system.

The adaptation of the system to the user will be realized through different dialog types and through different layers of detail, both in the input and output sections.

(b) Objectives

The following specific objectives shall apply:

The user's influence on the dialog:

System activities should be determined by the user and by his objectives. For example, the user should be able to select a major function and then a specific technique associated with this function.

In every dialog state, the user shall have at least two alternatives of action.

The user should be able to modify the set of selected system activities (extension functions). Usually the system will provide a default set of activities fitting the general problem. Sometimes the user may want to exercise only a single activity. The system should not force him to proceed through the general sequence. (This requirement does not apply to the internal modification of algorithms, only the selection of activities.)

System's adaptation to user abilities and needs:

The system shall support different users. Special emphasis should be placed on supporting analysts and researchers.

It should not restrict the user to a certain type of use and behavior.

The system should offer different modes of dialog ranging from detailed to abbreviated.

It should offer system messages in different detail according to user's wishes and qualifications.

(3) Transparency

(a) General

The system behavior should be transparent to the user. Then the user will be able to develop a consistent model of the system while using it and the system behavior will become predictable and capable of being influenced by him.

The more flexible a system, the more complex it will appear to the user and the more difficult it will be to understand. Therefore an optimum compromise will have to be selected between dialog flexibility and system transparency.

There are several methods to support the development of a system model in the mind of the user:

The system can make detailed information available about the system in different layers. The user gets to know only those layers which are suitable to his needs and current abilities.

The complexity of the model can be reduced if the system adapts to the actual behavior of the user (e.g., by withholding information about complicated functions or methods that the user would not be able to use in his present state). This can be accomplished by examining the user profile contained in the data base and initially presenting only information that is consistent with his profile.

Special states (states that occur frequently), should be referred by characteristic names (e.g., basic state, help state, etc.). These states can be indicated by system prompts, different cursors, or special icons.

The internal construction of a system model should be supported by uniform system behavior; i.e., similar actions should be performed in similar ways and there should be a standard way of handling errors, etc. This approach will lead to a more predictable system behavior.

If the system uses different dialog types, the vocabulary for these types should be consistent, and follow the rules established for the Logistics Description Language and the data base conventions.

(b) Objectives

The specific objectives shall be:

Support the user to develop a consistent system model by:

Offering a well-structured list of functions. This list of functions should be organized hierarchically.

Indicating current system state. This includes not only the present mode, but also an indication if the system is performing an extended computation.

Requiring that similar tasks use similar actions. Examples include entering numbers, space characters, input format, etc.

Making the organization of the dialog transparent to the user. This can be accomplished by presenting only essential information to the user.

Predictable system behavior:

Different systems have standardized interfaces. This applies not only to the data but also to the control interface.

The system should produce no surprise effects; it should show similar behavior in similar situations.

The system should exhibit a predictable response time. If there is an extended computation in effect, then the system should so indicate.

User options to cause modifications should be included:

In the structuring of the dialog, specific implementation should consider the current environment.

In the mode of the dialog, mode options available will depend upon the current state.

(4) Ease of Use

(a) General

The system will be a practical tool for the user if it does not frustrate him. It should support his needs, should provide help if necessary, and react tolerantly to his errors.

(b) Objectives

This is an important and broad class of system objectives. They can be divided into three subclasses:

Metadialog

Many input/output functions. Current microcomputers support a wide variety of input and output devices other than keyboard and CRT. These include mouse, digitizers, graphics plotters, voice, etc. Future dialog design should provide options to support these devices.

Extension functions. A typical example would be the use of macro-capabilities to create compound commands.

Communications functions. In a distributed environment, a wide range of communications support will be needed. These functions should be considered when designing a full dialog system.

Help functions

There is a wide variety of help functions. These may include general or specific help functions at the system level, related to a unique function, and support for situation-dependent problems.

The system may not require extra explanations. The extensive use of help functions can be avoided if extra care is taken in designing a clear dialog.

The system never leaves the user alone with his problems. Help or encouragement is always available!

(c) User Adaptability

Consideration of the user's properties, abilities, knowledge, and their variability:

Dialog design should be as similar to human needs and characteristics as possible:

The system should allow a kind of communication which is as close as possible to human communication applicable to the specific environment.

The system should not force the user to hasty actions. Each step should be clearly presented. Whenever possible, the step should be reversible.

It should consider the application background of the user. Information on the user should be available in the profile file.

It should react appropriately to human weaknesses (errors, stress, forgetfulness, etc.) within reason.

Consideration of changes in user behavior:

The system should consider that the user gains experience working with it and should provide less explanation associated with specific activities. This option should be user controlled. The specific level of expertise could be stored in the user profile file.

It should consider that user objectives might change during the dialog and should, therefore, provide means for the user to interrupt the current sequence of activities.

It should consider that the user tends to make more errors after after working for longer periods with the system. This could be an adaptive process based on user time on the sytem.

(d) Error tolerance

The system should react to and take precautions against disturbances of the interaction produced by typical human nature, such as fallability, forgetfulness, or varying degrees of concentration. This approach may not be as difficult as it sounds. It is relatively easy in the system to keep track of user input error rate and provide additional help.

Systems should be designed to minimize human errors. The key factor in preventing errors is limiting the amount of memorization required to use the sytem. Other promising approaches are to use selection instead of entry, names instead of numbers, allow for predictable behavior, use good and complete error messages, include some redundancy, and provide for reversible actions and sound data structures.

Error protection:

The system should prevent the user from unwanted consequences. Actions in a specific environment should be questioned by the system if they could result in a situation where recovery is not possible.

The system should prevent the user from deleting important data. This includes file deletion.

Error diagnosis:

The system should offer clear and understandable error messages. Error messages should be clear and easy to understand. Numerical error messages shall not be used.

Error messages should contain hints for correction. This could include an automatic transfer to edit mode.

Easy error corrections:

Error correction actions should be specified in respect to those parts which contained errors. If an error is discovered during a lengthy input process, then the user should not be forced to reenter the complete sequence.

The system should prompt the user for error correction. After the error has been corrected, the system should return to the point where the error occurred (if in data entry mode) or to the restart point (if system is in a computational mode).

The system should ignore common typing errors. This would apply only to the most frequent errors, provided that their interpretation would not result in an unintended action.

(5) Ease of Learning

(a) General

The system should be designed in such a manner that it is easy to learn and should require no additional resources. Furthermore, execution of simple tasks should not require special training of the user.

Two problems should be considered:

How can the information for the user be structured and distributed in a way that is easy to understand, and,

How can our conception be based on learning principles which are characteristic of the user.

To introduce the sytem to the novice user:

Instruction on how to use the terminal, especially those facilities that enable him to correct his input, and to ask for help should be included in a tutorial file.

The introduction to the system itself should be done in a top-down order; that is, the user should first get to know about the main functions of the system: what they do, in which context they are usable, etc. After that, he should learn about the specific functions he can use and how he can get more detailed information about the system.

Generally, it is better to give less information than too much; the user can always ask for more. If an output is long, the main parts should be highlighted.

To support the learning process of the user, there are a variety of techniques:

First, the system should be transparent to the user; all responses should be clear and the response time should be in accordance with the complexity of the given task.

Second, the user should be assisted to utilize user-initiated dialog types as much as possible, because activity supports learning by doing. The reaction to an input should show whether the user was successful or not, or whether he has instant control over his activities. Fears by the user should be overcome by providing help facilities in every possible situation; to be avoided are input loops from which he cannot escape without specific knowledge.

(b) User Support

Support for the user when using the system shall include:

A system usable without specialized computer knowledge.

A system that supports learning by doing.

A system that offers the user manual via terminal upon request.

(c) Learning Support

Additional learning support shall include:

Introductory courses for operating the system should be provided.

Experts should be available for help if the user demands it.

(6) Reliability

(a) General

The system should be designed in a way that the user can do his work in a reliable manner.

The system should be accessible at any time when the user needs it. Therefore, a well designed organization and reliable hardware is paramount.

To do his job correctly, the user must get to know all the effects of his input to the system, including side effects. The system must react promptly and react adequately to user errors (i.e., the user must get the chance to continue his tasks without difficulties).

(b) Availability:

The system should be in operation when necessary.

System crash occurrence should be minimized.

(c) System user support:

Side or hidden effects in any of the operations shall be eliminated.

The system response time should be optimized.

The system should offer facilities for data protection and security.

f. Problems of Realization

There are three major tradeoffs to be considered in determining the optimum system implementation:

Conflict between high dialog flexibility and high reliability can be circumvented by careful design and implementation. The higher system flexibility will result in higher development costs.

High dialog flexibility and ease of use is a conflict between two opposite objectives: a system with few functions and modes of dialog is easier to operate than a complex system with large sets of functions and modes. The selection here will depend on user needs and training.

Dialog flexibility and transparency interfere with each other because the more modes there are, the less transparent the system tends to be for the user. The user needs more time and experience to form a system model in his imagination, requiring more training and patience in using the system. If the system is too complex then it also may face rejection by the user.

g. Conceptual Recommendations

(1) Basic Concepts

There are two aspects to consider when designing man-computer interfaces:

What contents should be represented by computer input/output; i.e., which functions should be provided and what information should the user obtain from the system.

Which dialog modes can the user use and how can he switch between different ones.

(2) Dialog Contents

Dialog contents include:

(a) Functions.

The set of possible functions represents the flexibility of use of a given system. Common functions are those associated with dialog, input/output, help, and control.

From the dialog functions, dialog interruption and restart require careful implementation, as do functions which switch between different types of dialog. These functions will normally be used by more experienced users. Other dialog functions support the user by providing information on who has the initiative, what kind of input is expected, what kind of format is required, what kind of default values are in the system, and how to switch to a different dialog mode.

Input/output functions form the most important class. They are the primary means by which a user interacts with the system. They should be well designed and contain extensive error checking capabilities.

Design of help functions is also important. These functions should support all classes of users. The general help functions give information about the system behavior and operation. The specific help functions answer questions about the actual situation in a given dialog state. In addition there are additional help functions which explain the available system algorithms. For each of these, help functions answer questions as to what the algorithm does, what kind of input is required, and what side effects the algorithm has.

In the control function area, the cancellation of tasks and activities are the most important ones. If a cancellation function is issued, then it is important to reset the system to its restart point. Another important control function is the 'undo' which permits the user to go backwards if he makes an error. General implementation of this

function is extremely difficult, and normally only one reverse step will be permitted.

(b) Dialog Modes.

There are four possible layers in an interface with an interactive system:

In every state, the system can propose task alternatives to the user. If there is a limited set of alternatives, then the tasks can be chosen by menu, otherwise the system can ask for choice by question. This mode is more suitable to the beginning user.

Another approach is to have the system ask the user to make an input and provide the proper syntax. This approach is more suitable for experienced users. Logistics Description Language will be supported in this mode.

Still more experienced users may prefer a method where they can formulate the task with their own commands. If the input is not fully specified, then the system will ask for the missing information. The Logistics Description Language will be supported in this mode for processing multiline inputs.

The most complex dialog structure uses a quasi-natural language approach. At the present time, this technique can be practically implemented only in very limited applications because of the ambiguity associated with the language.

These four alternatives cater to different kinds of users with different needs and skills. They show that in a large system with large inhomogeneous user groups it is sensible to provide the user with different kinds of dialog layers.

h. Realization of Abstract Dialog Types.

(1) General

When developing the actual dialog, the following guidelines should be considered:

The vocabulary should be as close as possible to natural or professional language used in the logistics environment. There should be options for using abbreviations as well as mnemonic and symbolic codes accepted in the user community. The vocabulary should be consistent and should minimize typing effort.

The syntactical rules for input should be restricted as far as possible and be easy to apply. There should be as few formatting rules as possible. They are of value only if the user can understand their purpose.

Formal redundancy, such as fillers and blanks, should be tolerated by the

system whenever possible.

Overdetermination of the task specification should be tolerated.

All ambiguous input should be questioned by the system.

User vocabulary should be restricted to the set specified by the Logistics Description Language.

(2) Output Design Requirements

The output should be designed as follows:

It should be self-descriptive and concise.

The vocabulary should consist of natural language words including common abbreviations. It should be consistent with input vocabulary and the Logistics Description Language.

The output syntax and format should be chosen according to the contents of the actual output. Text, keyword, or graphics should be selected so that the important information is easily recognizable.

The interpretation instructions of the output should be as concise as possible.

There should be a choice between different kinds of representation: *short or long.* This choice should be under user control.

i. Conclusions

The user interface of the Logistics Work Station must be user-friendly, efficient, and modifiable to meet the various user requirements.

The proposed approach regards user and computer having equal status, according to their own characteristics, in a systems operation. This approach is based on system partition and hierarchical decomposition techniques which will permit evolutionary development of the user interface and have minimum effect on the solution modules. This approach also will not only reduce the initial programming effort but will result in a more reliable and easily maintainable system.

Effort will be made to humanize the applications, to make allowances for behavior differences, and to introduce the human as the final decision maker.

An application example of the dialog development procedure is presented in Appendix B.

7. SOLUTION CONTROL

a. Introduction

There are two aspects to solution control. The first involves decomposition techniques needed to partition and decompose the very large problems occurring in logistics applications. The second deals with the automation of solution control. Both of these topics will be discussed below.

b. Decomposition Techniques

Decomposition is the process used to break up large-scale problems into subproblems of lower dimensionality. These subproblems are then solved independently and individual solutions are then combined to yield the solution to the original problem.

Partitioning of the problem is the rearrangement of the components of a problem into a hierarchical structure that follows the information flow in that problem. After partitioning, it is possible to identify disjointed and irreducible subproblems. The latter contain loops (cycles/circuits) that must be solved by tearing and then using iterative techniques.

The shiploading problem in its full context is very complex. As discussed earlier, it involves many functional tasks, some occurring simultaneously, with a large number of variables and constraints. Many of these tasks are highly interdependent.

The complex interdependency of these factors is best represented in the form of a graph. Vertices are used to represent the factors and directed edges the dependency of these factors. If the graph contains no loops, then it is possible to solve sequentially the resulting system of equations. If, however, it contains loops, then some form of an iterative solution is desired. As the system becomes more complex, the task of identifying the loops becomes more difficult and computer techniques must be used.

Task 1 of the project was planned to investigate and evaluate the currently used graph and matrix decomposition techniques. The most familiar of these techniques is that originally proposed by Steward [13]. This technique has been used to decompose complex problems in occurring power plant design, econometrics, chemical process design, missile control, and sociological systems. Application examples given in his book deal with large system analysis and management [14].

The approach used by Steward employs matrix representation and reduction. His method is well suited for manual decomposition, but when computerized involves complex bookkeeping of intermediate results. Since decomposition plays such an important part in solution control, better methods were sought. An area rich in decomposition and optimization techniques is that of compiler optimization. Although different in application, the compiler problem is very similar to some of the logistics problems. It involves allocation of registers, sequencing of computations, and keeping track of computed variables.

One of the key methods used for decomposition is based on Tarjan's [16] algorithm. With minor modifications, this algorithm could be adapted for logistics problem decomposition -- a hope that was expressed in the proposal. The hope expressed in the proposal, that with minor modifications The algorithm was slightly modified to allow for more flexible representation, was coded in PASCAL, and tested. It has minimal program and intermediate storage requirements, has a minimal amount of backtracking, and is very fast in performance.

Another promising algorithm has been developed by Sharir [17]. This algorithm was also programmed and tested. (See Appendix A for data summary.) Other applicable work is contained in References 18 - 22.

Further extension of these algorithms is planned during the Phase II effort in order to incorporate the capability to reference data elements associated with the specific nodes and to establish links to the data base. Provisions for these extensions were already made in the basic algorithms.

The decomposition algorithm will be of major importance in solving logistics problems. First, it will be applied at the main system level to identify those areas where loops exist requiring an iterative approach. Second, it will be used to keep track of information flow in the system. Since the algorithm determines the existance of the loops by tracing out the functional dependence, it can be used as an information flow analysis tool.

c. Data Flow Analysis

The discussion of data flow analysis will begin with a short description of the concepts involved and will conclude with a discussion of its potential application to solving logistics problems.

Data flow analysis is a new form of analysis developed in the last fifteen years. It has found applications in compiler design where it is used to optimize the final object code. The concept of data flow analysis is general and is slowly finding applications in other related fields. The data flow analysis has powerful techniques for keeping track of data usage in the program. It is capable of handling very complex program structures and the more recently developed algorithms are fast and efficient.

Given a definition of a data item in a program, it is frequently desirable to find what uses might be affected by that particular definition. The inverse is also true: for a given use the definitions of data items of data items which can potentially supply values to it are also of interest.

Another data flow relationship which is of interest is the following: what data definitions are live at a particular point in the program; that is, what data definitions given before this point are used after this point. This information is of interest, for example, when working with a large number of blocks where we need to know if certain data already have been computed and are available, or if we have to compute these data items (or estimate them) before we can proceed with the solution.

In order to define more precisely the relationships derived by the procedure, a number of basic concepts and constructs must be defined. Data flow relationships are usually expressed in terms of the control flow graph of the computer procedures. For example, the scheduling process diagram (Figure 9) discussed in Section III.2 is a form of a control flow graph. This graph shows the sequence of operations performed and how they relate to each other. Assuming that the scheduling process starts from scratch (no information is yet available) the initial block tells us that the first step is the development of the master schedule and that no other information is available. On the other hand, if the system is already in operation, then in updating the master schedule we would also have to consider the information from the detailed scheduling blocks (schedule slippage, changes in equipment and personnel, etc.).

Data flow analysis can handle both of these cases and thus provide a control scheme tying together valid information with the specified processing sequence.

Data definition, as used in the data flow analysis, refers to that part in the process which modifies or creates a data item. In programming language terms, it is the left side of the equation. In terms of a procedure, data definition refers to data created within that module.

Data use is that part of an equation which references a data item without modifying it. In terms of procedures, data elements which are referenced within the module, but not modified, could be considered data usage elements.

In order to determine which definitions affect which uses, two types of expression relationships can be distinguished: those relationships which exist in a straight line sequence, and those which exist in the context of control flow. Methods of finding and codifying relationships in a straight line sequence are relatively easy, but if there are feedback paths then the computation is more complex.

Because of this complexity, control paths are often completely ignored. Two excellent examples from the OR field are PERT and Critical Path Method. Both of these techniques deny the existance of feedback paths and assume that the project can only proceed in the forward direction without any surprises or setbacks. If only the real world would act the same way!

Since most computer programs contain a relatively large number of conditional switches, data flow analysis must be capable of identifying and handling these loops (or circuits) in the program. A number of fast and efficient techniques have been developed for finding these loops and their associated blocks. The elements comprising the loop are called Strongly Connected Components (SCC), and are handled by special procedures.

Two of the procedures used for this task, based on the algorithms developed by Tarjan and Sharir, have been converted to PASCAL programs and tested using some relatively complex loops. Procedure description sheets for both of them are included in the appendix. (Note that the same algorithms were also used in partitioning and decomposition process.)

As hinted earlier, the successful solution of complex logistics problems containing loops will depend on the use of the data flow analysis concept. By using the control flow graph as a road map, detailed data flow analysis indicates if the data are available or if a new computation is required. Thus, an orderly procedure can be established capable of directing the execution of the solution modules.

One of the key concepts in computer science is the duality of data and control. Every computer program can be examined from either viewpoint. These two viewpoints are not contradictory but provide complementary views of the same computer program. As the complexity of the program increases and the need arises for partitioning of the problem and solution by parts, there is a definite advantage in utilizing both viewpoints concurrently.

In the current literature, these two viewpoints have not been clearly identified, and there is considerable variation in the meaning associated with either flow concept. To clarify this important concept, a very simple example will be disussed. Figure 17A shows a segment from a structured program containing two separate branch structures. In the control-flow representation (Figure 17B), the program statements are expressed in a a flow graph format. The individual nodes represent the elements in the statement sequence, and the arcs represent feasible branch points. For example, Node 1 represents the boolean expression (A>B). If this expression is true, then the control proceeds to Node 2 where the assignment X = 7 is made. At Node 4 another Boolean expression is encountered and evaluated. The rest of the flow graph is processed in a similar manner.

The control flow graph thus represents the sequence of statements, or in the more complex case, the sequence of processing modules. In the latter case, the variables could be arrays. The procedure developed for the logistics problem solver will be based on the module concept.

The data flow diagram in its pure form is depicted in Figure 17C. Here the interpretation is as follows:

Variable X change depends on the values of variables A and B.
Variable Y change depends on the values of variables C and D.

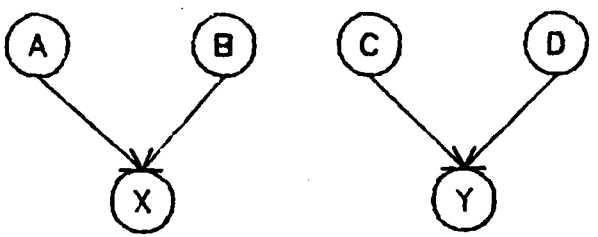
Both of these expressions represent static relationships between the variables.

The above representations are of the control-flow and data-flow in their pure form. It is also possible to use a hybrid representation where both control flow and data flow are merged. This is the case with the data flow approach used in compiler optimization and initially discussed earlier. In this representation the basic graph used is that of the control flow, but there is set of associated vectors with each node representing the status of all the program variables. These vectors represent those variables used, defined, or killed during the execution of the statement sequence. In the later discussions the data flow method will refer to this hybrid model.

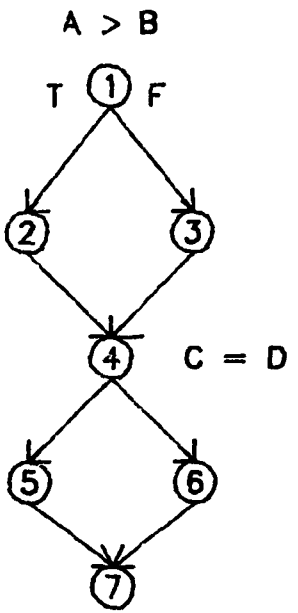
There are well defined techniques available for computing the various data

```
IF A>B                    (1)
   THEN
     X = 7                (2)
   ELSE
      X = B + 5           (3)
IF C = D                  (4)
   THEN
      Y = C + 3           (5)
   ELSE
      Y = 7               (6)
- - - - -                  (7)
```

(A) PROGRAM STATEMENTS



(C) DATA–FLOW DIAGRAM



(B) CONTROL–FLOW DIAGRAM

FIGURE 17. CONTROL–FLOW VERSUS DATA–FLOW

sets associated with the graph nodes. One of the commonly used techniques is based on the concept of an interval and sequential reduction of these intervals until only one node remains.

To illustrate this approach, we use Figure 18. The initial graph, G1, contains a number of loops. The division of the flow graph into intervals serves to put a hierarchical structure on the flow graph. The basic interval is either a natural loop and an acyclical structure associated with it, or just a single node. Every interval has a header node which dominates all the other nodes in the interval. In the example, Interval(3) consists of nodes 3, 4, 5, 6. Node 3 is the header node because the control path to any of the remaining nodes in the interval has to pass through the header node.

The formal definition of the interval I(n) with header n is as follows:

1. n is in I(n)
2. If all the predecessors of some node m<>n0 are in I(n), then m is in I(n)
3. Nothing else is in I(n)

In the above representation, n0 represents the entry node.

In the sequential reduction process, after each pass each interval is represented by a single node and the process repeated. For example in G2, Node 9 represents I(3) in G1 consisting of initial nodes 3, 4, 5, 6. In the final pass, G4, there is only one remaining node and the process terminates.

The interval method provides an orderly approach to the reduction process needed to compute the associated data set vectors. Also note, that if the graph were evaluated directly by expansion, there would be an infinite sequence of steps because of the loops inherent in the basic program definition (Node 6 to Node 3; Node 7 to Node 2).

Although the formal definition for the interval appears to be very simple, the actual algorithm is somewhat complex since it also involves recognition of all loops.

A form of this algorithm was programmed in Pascal and tested. The summary sheet for the program (INTERVAL) is included in the Appendix A.

As discussed earlier, the conventional data flow analysis technique required few modifications to make it more useful in logistics applications. First, the technique would have to be extended to handle the solution on the algorithmic level. Second, a flexible interface was needed for the interface to the data base.

Both of these modifications, although time consuming, are feasible and are planned for full implementation during Phase II of the project. The separation of the data base from the algorithms will further improve the computational efficiency because a single data base does not require translations from one solution format to another.

The application of the data flow concept to the logistics problems is best

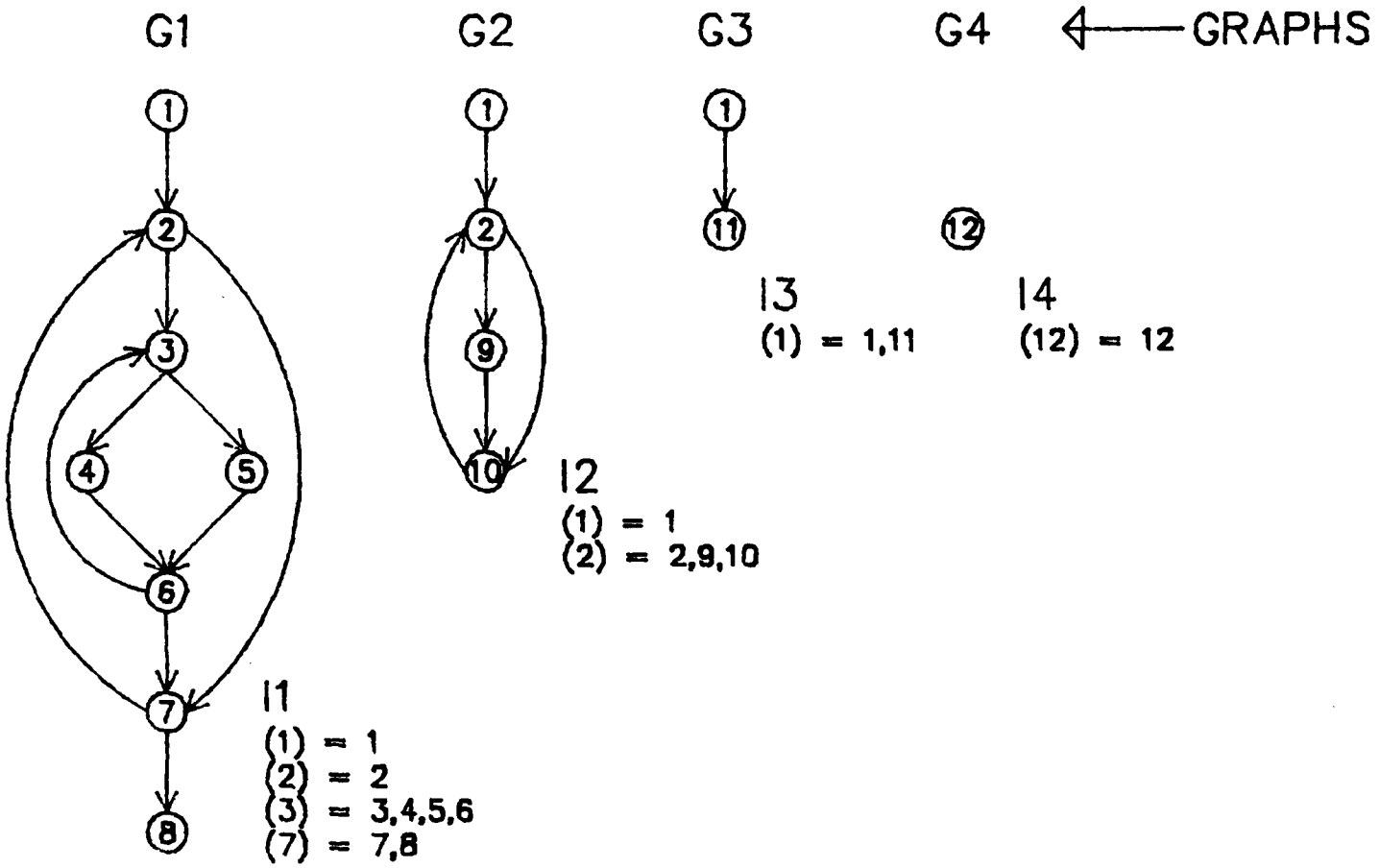


FIGURE 18. FLOW GRAPH REDUCTION USING INTERVALS

illustrated by considering a real example such as the shiploading problem. The basic system formulation is based on the functional dependency chart such as the one presented in Table 9. This matrix actually represents the applicable functional dependencies occurring in the logistics system. If this graph is processed using one of the strongly connected component solution algorithms developed earlier, then all of the major feedback loops will be identified, in addition to the independent variables.

Independent variables, usually representing available resources and other constraints, will have to be initialized before actual computations can begin. To keep track of these variables, a Boolean type table will be created and maintained by the data flow subsystem. In this table, a specific bit setting will indicate the availability of that variable. As the solution proceeds additional bits representing the output variables will be set. Again, it is important to realize that these bits do not represent the values of the variables but only the availability of data. Using this approach, it will be possible to block a computational process where all of the variables have not yet been specified.

In addition to this fail-safety feature, the data flow approach has a number of other features to facilitate the solution process. For example, a reference to the dependency graph will display the specific variables entering a given computation. This will enable a simpler interface to the help system. Furthermore, the dependency graph can also be used to derive the sequential data processing requirements. Thus, using the topological algorithms, the sequence of solution steps can be determined.

In the case of the ship loading problem, the individual functions making up the total solution system can be easily extracted from the dependency matrix. The same matrix also provides information needed to separate the independent variables for each of the functional tasks.

Once the logistics description language is developed, automatic generation of the dependency graph from the language statements will become feasible. Until that time the old time-consuming manual process will have to be used for the derivation of this important information.

8. SHIP LOADING PROBLEM

a. Introduction

This section contains technical discussions relating to the naval logistics applications problem, the underlying data base, and the applicable solution techniques.

b. Problem Definition.

The specific problem selected to illustrate the application of the proposed techniques to solving naval logistics problems involves the consideration of the ship loading problem in its widest context, starting with the receipt of the supply requisition and ending with the placement of the shipping container in the cargo bay area.

This particular problem is typical of those encountered in the logistics area. It involves determining the best methods for moving supplies within the constraints of the available resources, and at the lowest logistics system cost.

The top level logistics system perspective is shown in Figure 19 and identifies the key considerations. The key logistics elements involve the definition of material (WHAT?) to be moved, time (WHEN?) of the move, destination (WHERE?) of the move, the selected method (HOW? and WHO?), and the assigned resources.

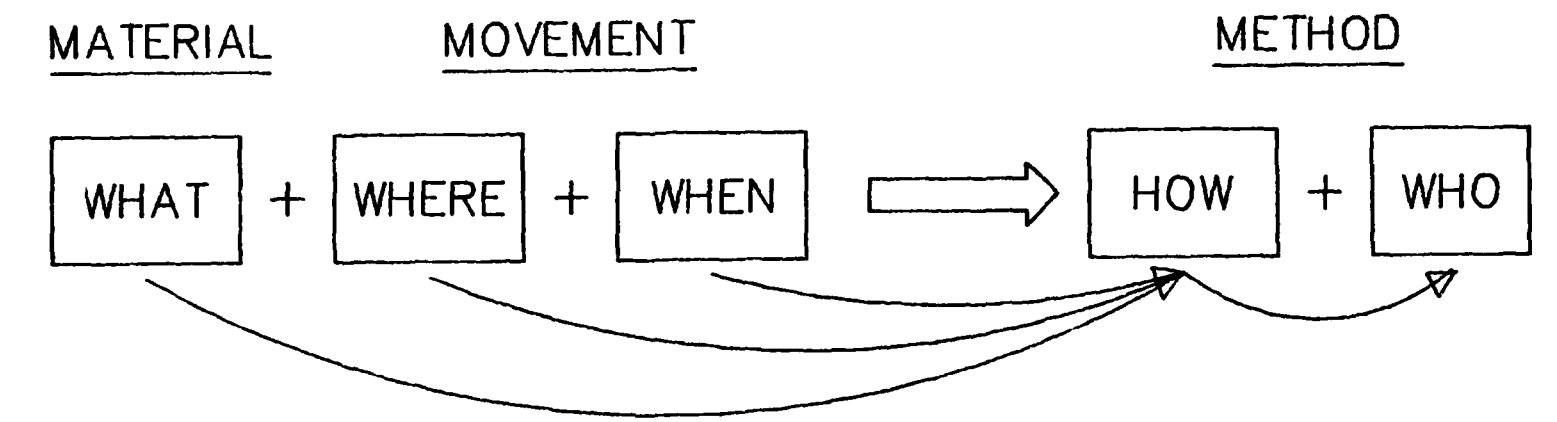
The material (WHAT?) is further defined by its characteristics (weight, size, handling precautions), quantity needed, and transportation restrictions. The location of the requisitioner (WHO?), the supply depot and the intermediate shipping points determine the extent of the movement (WHERE?). The time constraints (WHEN?) on the supply requirements, such as priority and shipment dates help to further limit the available solution options.

The determination of the shipping method (HOW? and WHO?) is further constrained by the availability of shipping containers, transportation equipment, and operational environment. The determination of personnel support needs is based on the selected transportation environment and amount of goods to be shipped.

Once a transportation method has been selected, it is possible to determine if the given system constraints can be met; then the resulting logistics operational costs can be computed.

Throughout this process, a wide variety of operations research algorithms are available to support the selecting, sequencing, scheduling, routing, and assigning tasks. The specific selection of specific algorithms will depend on the given constraints and the decision maker's goals.

c. Functional Description



OR SUPPORT:

SELECT SEQUENCE SCHEDULE ROUTE ASSIGN

FIGURE 19. LOGISTICS SYSTEM PERSPECTIVE

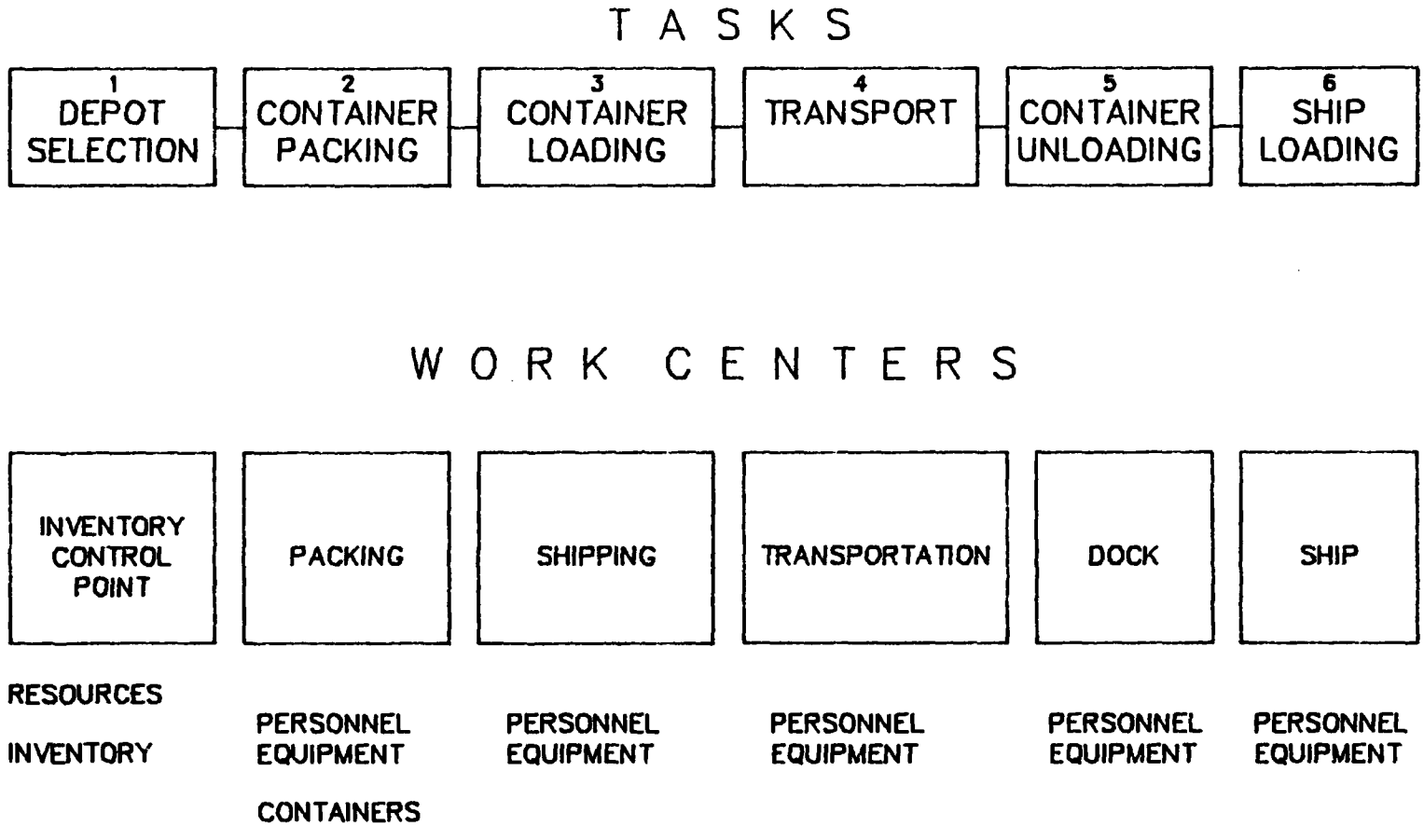


FIGURE 20. KEY SHIPLOADING TASKS AND WORK CENTERS

The ship loading problem can be further decomposed into a series of functional tasks as shown in Figure 20. The decomposition used in this particular case is based on the physical separation of tasks by the performing work centers. Each of these tasks is performed in a different location and by different functional task groups. Each of these six areas as shown in Figure 20 will be discussed in detail.

(1) Depot Selection

The depot selection starts with the receipt of supply requisitions from different base locations as illustrated in Figure 21. The selection of a specific depot location involves the consideration of the supply availability, depot and shipping point locations, quantities required, and the requested priority levels. A number of operations research algorithms are available to support this selection process. These include linear and integer programming, transportation, and goal programming algorithms. The use of these algorithms with applicable constraints will yield the initial selection. (To obtain a system level optimization, a series of iterations over all of the applicable tasks may be required.) The initial selection consists of fill-orders issued to the selected supply depots.

(2) Container Packing

Once these orders have been selected at the supply depot (warehouse), they are further processed as shown in Figure 22. Specific subtasks involve the location and identification of the requested supplies, and the selection and packing of containers. Again a number of OR algorithms may be used: linear and integer programming, transportation, and bin packing and covering. The latter two are applicable to the packaging. The applicable constraints are based on the availability of containers, personnel, and material moving equipment. This specific task ends with the filling of the shipping containers which will carry priority and destination designations. The contents of the containers will be listed in the packing lists.

(3) Container Loading

The next functional task, detailed in Figure 23, involves moving the shipping containers from the packing area to the depot loading dock and then to the selected loading area. Specific subtasks deal with equipment and operating crew selection, task scheduling, and material moving. These tasks are constrained by container size, weight, and moving resource availability. The applicable OR algorithms include linear, integer and goal programming, transportation, assignment and scheduling. At the completion of this task, material is ready for shipment on the loading dock.

(4) Transportation to Port

The transportation task, shown in Figure 24, deals with the movement of material from the suppply depot to the unloading dock at the shipping port. The specific subtasks involve transportation equipment and crew selection, and the scheduling and routing of the vehicles. These subtasks are constrained by the container priority, quantity, size, weight, handling limitations, and resource availability. The applicable OR algorithms include linear, inte-

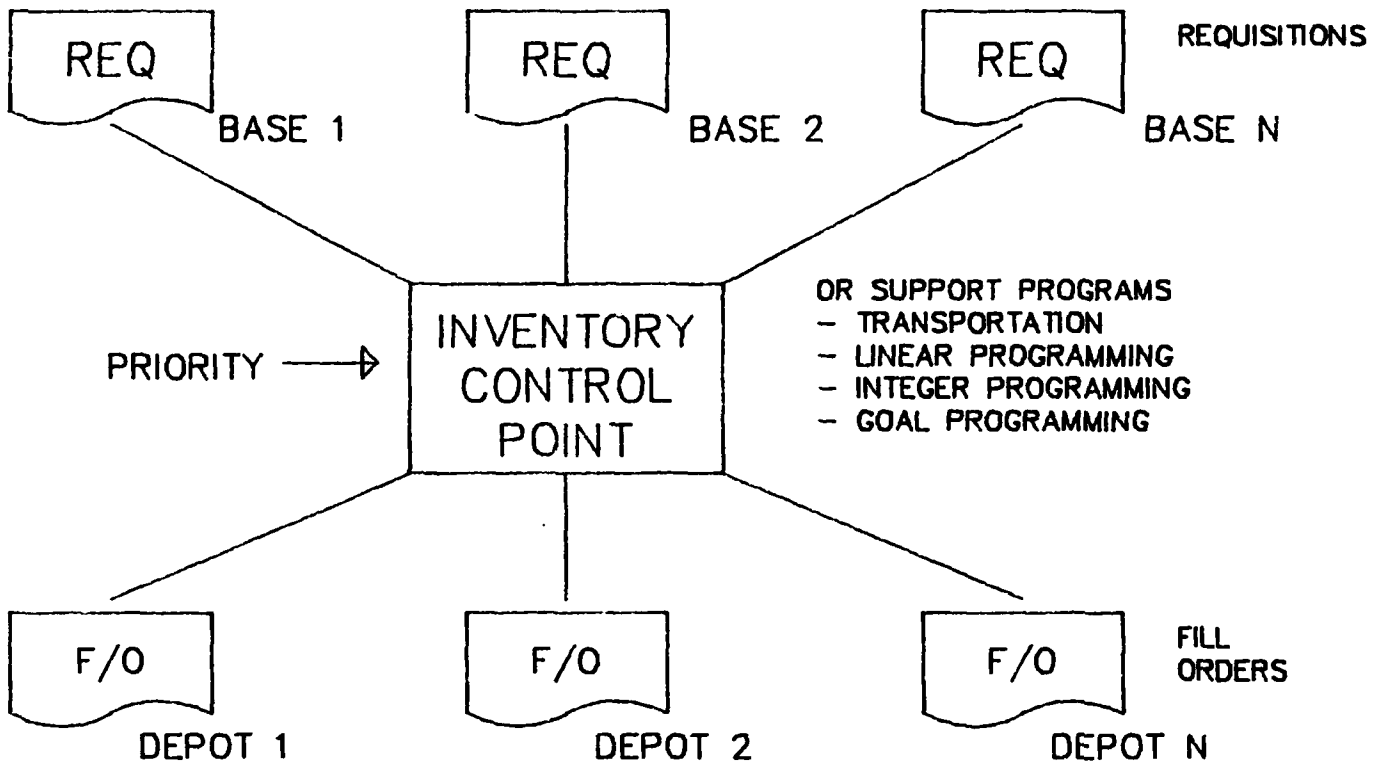


FIGURE 21. DEPOT SELECTION

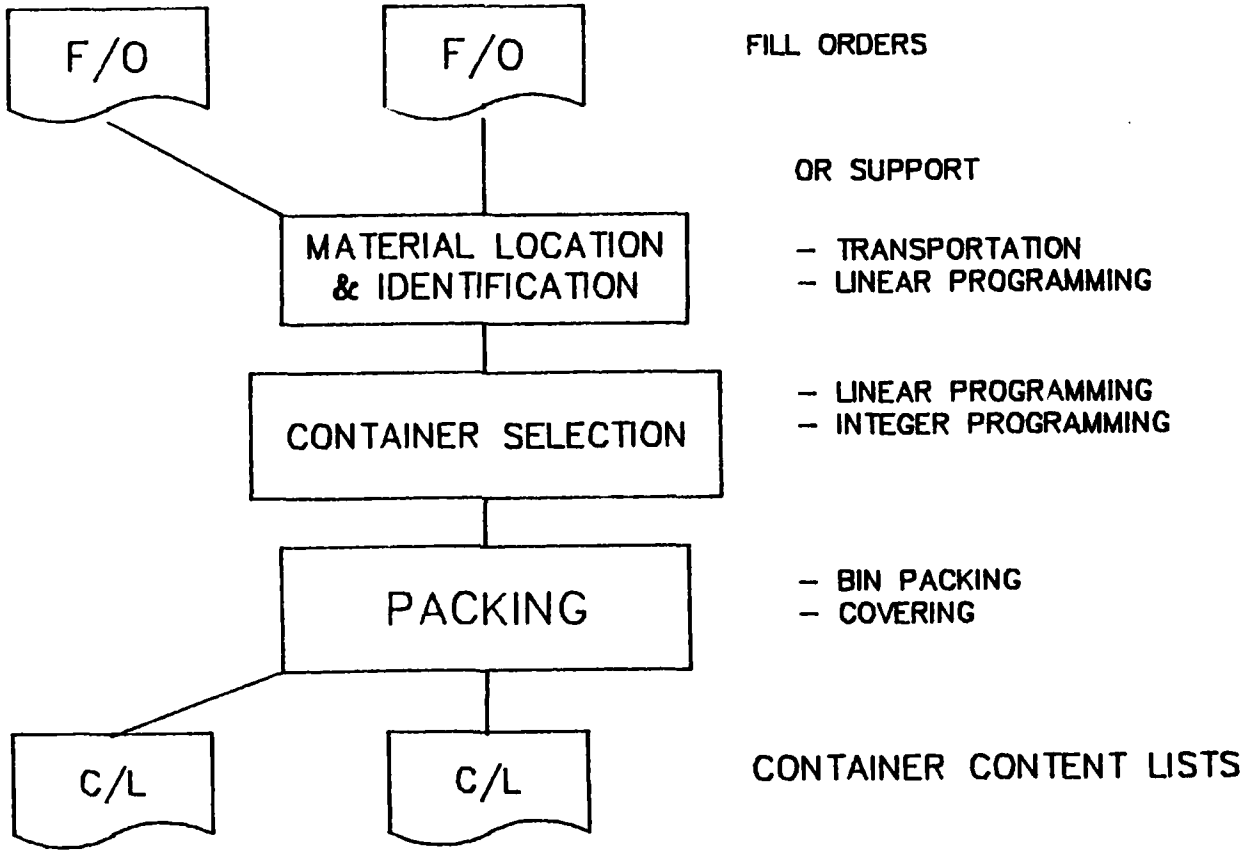


FIGURE 22. MATERIAL SELECTION AND PACKING

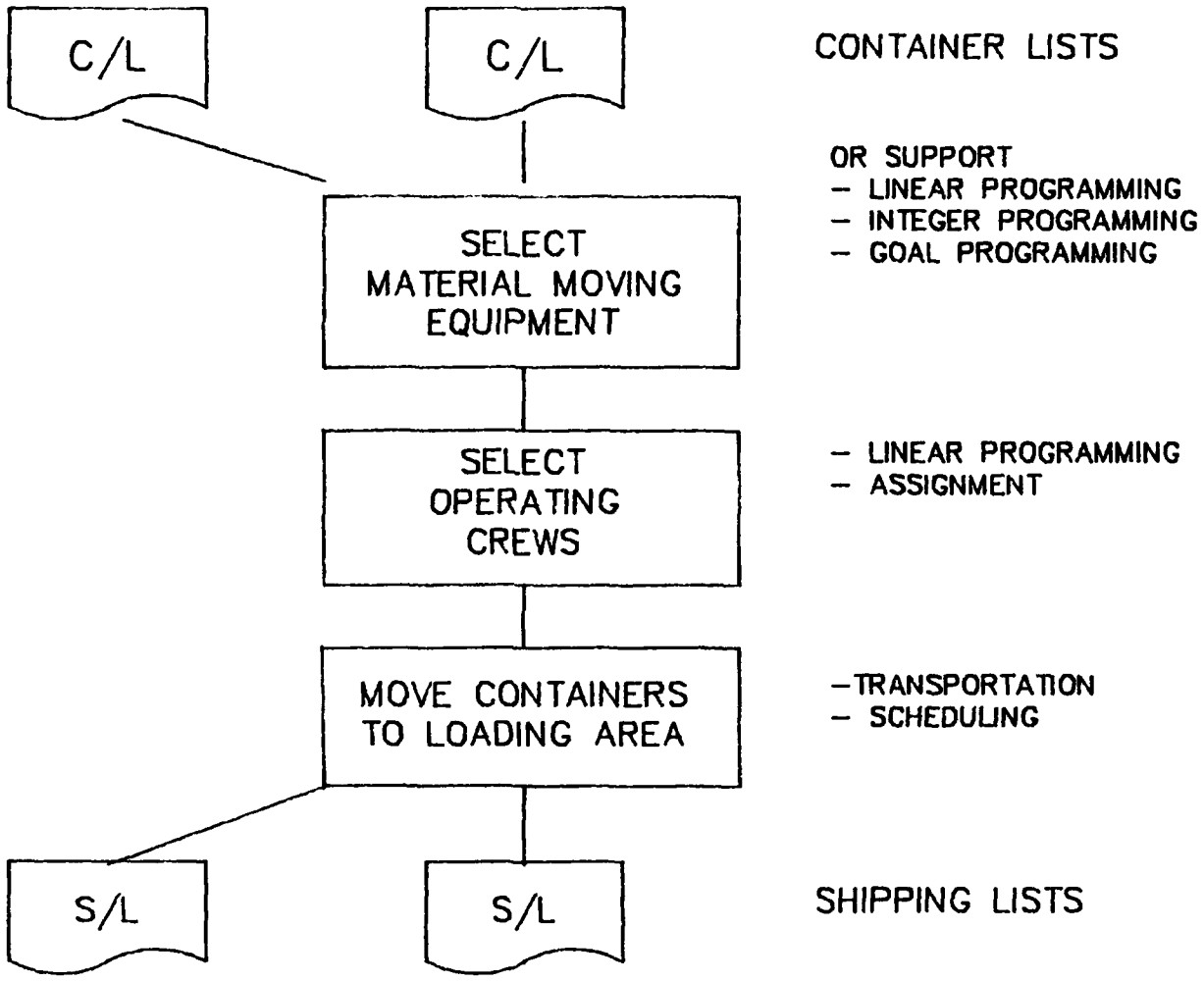


FIGURE 23. DEPOT LOADING FOR TRANSPORTATION

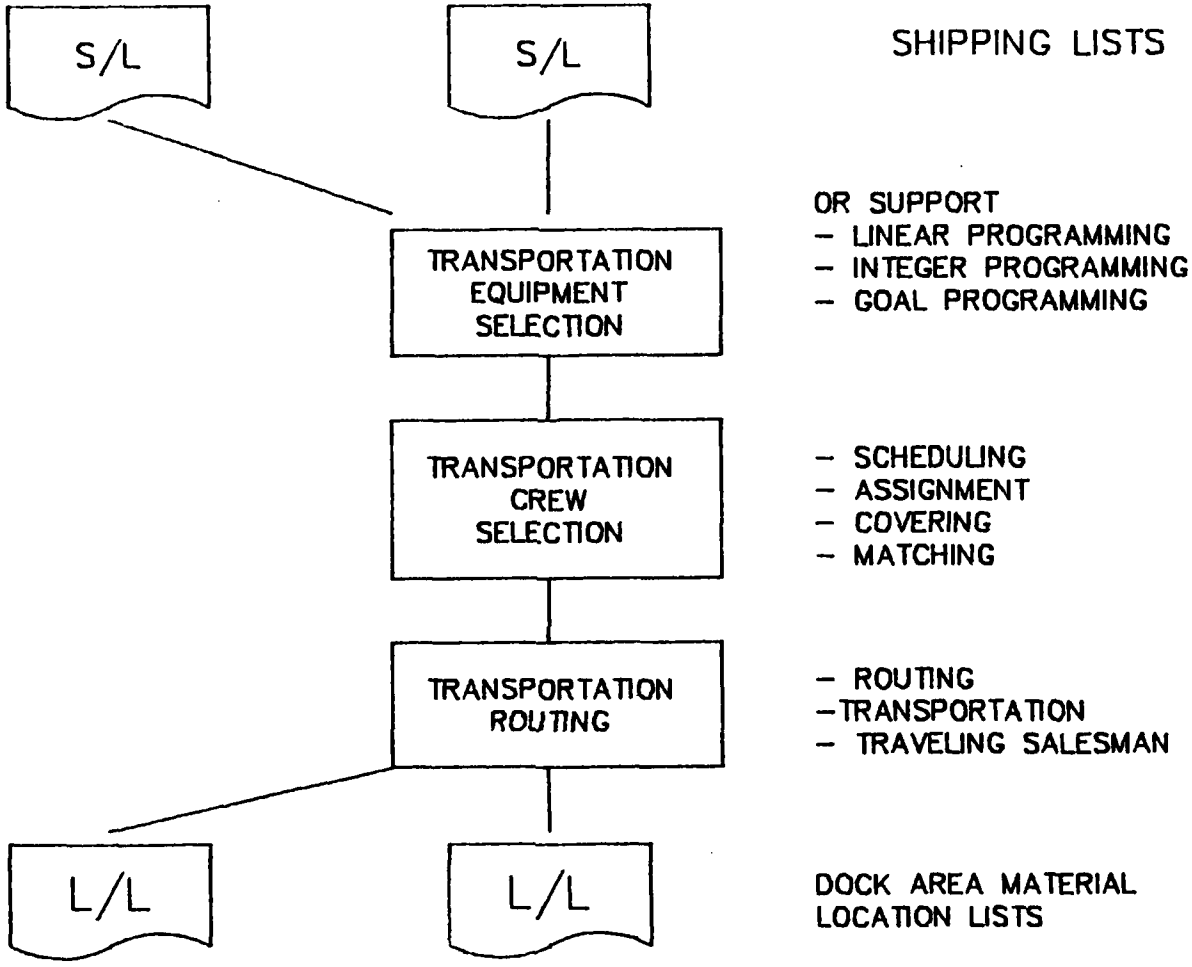


FIGURE 24. TRANSPORTATION TO SHIP

ger and goal programming, transportation, scheduling, and routing. At the completion of this task, material moving equipment has arrived at the dock-side unloading areas.

(5) Container Unloading

The container unloading task, shown in Figure 25, deals with selection of unloading equipment and crews. The constraints and the applicable techniques are similar to the loading task discussed above.

(6) Cargo Bay Loading

The sixth and final task, shown in Figure 26, covers the container movement from the dock area to the ship and placement of containers in the cargo bays. The applicable constraints include container specifics, cargo bay limitations, and the available resources. The OR techniques include linear, integer and goal programming, assignment, scheduling, packing, and matching.

d. Applicable Solution Techniques

A summary of the applicable OR techniques to the functional tasks is presented in Table 8. Examination of this table reveals the wide applicability of the OR techniques in the different areas, and the need for an integrated solution control and data management.

The next set of illustrations, Figures 27 through 30 represents the functional tasks from a process point of view and shows the various considerations that enter into the decision process.

Figure 31 illustrates the data aggregation as it occurs during the performance of the functional tasks. Starting with the requisition line items, warehouse selection partitions them into fill-orders. The packing process further aggregates the data within packages and then within the containers. During the transportation process, a load consists of a number of containers, and in the final stage the containers become part of the cargo bay contents.

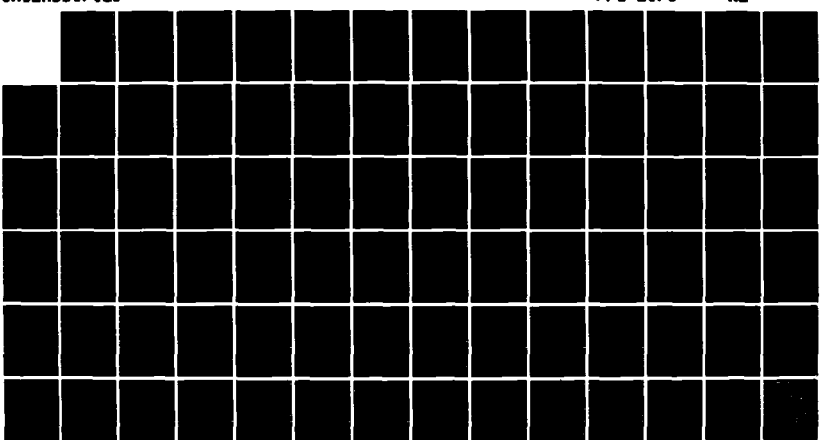
This data aggregation process is very important from the standpoint of the information processing, because it enables the hierarchical handling of data and the reduction of the data processing requirements. For example, during the transport loading process the containers can be considered as the basic data entities without the need to know the specific contents of the container.

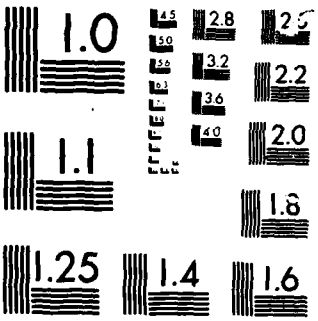
The data aggregation process will be further discussed in the next section dealing with the data base design.

The discussion up to this point has dealt with the functionality of the individual subtasks of the ship loading process. To provide the necessary system level control and to integrate the data base it is necessary to establish a system level function dependency matrix. This matrix, shown in Table 9, represents the different variables, constraints, and functions entering in the overall problem. Entries marked by "X" represent the functional dependencies of the entities in question. For example, row 9 (shipping distance) contains entries in columns 7 (supply depot location) and 32 (ship location), indica-

AD-A167 641 LOGISTICS SOFTWARE IMPLEMENTATION(U) DRINA COLUMBIA HEIGHTS MN J PUKITE 28 FEB 86 TR-FR-1 N00014-85-C-0670 2/2

UNCLASSIFIED F/G 15/5 NL





MICROCOPY CHART

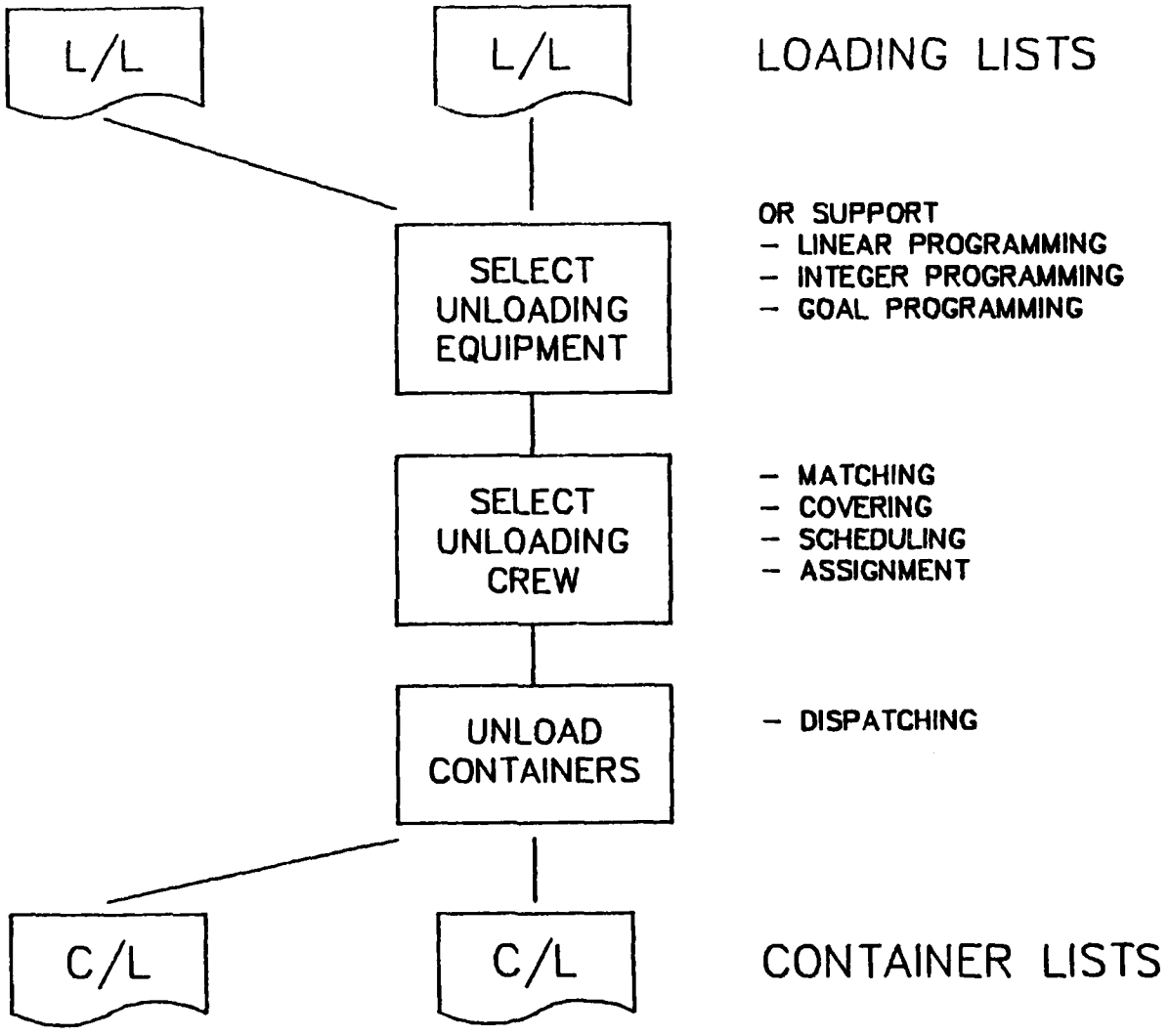


FIGURE 25. CONTAINER UNLOADING

FUNCTIONS	PROGRAMS	LINEAR PROGRAMMING	INTEGER PROGRAMMING	TRANSPORTATION	KNAPSACK	SHORTEST PATH	SET COVERING	MATCHING	NETWORK	MINIMUM COST	TRAVELING SALESMAN	GOAL PROGRAMMING
SELECT	WAREHOUSE	X	X	X								X
PACK	SELECT CONTAINER		X									
	ASSIGN PERSONNEL		X					X				X
	SELECT ITEMS			X	X							
LOAD	SELECT LOAD		X		X							
	SELECT EQUIPMENT		X									X
	ASSIGN PERSONNEL		X	X				X				
MOVE	SELECT EQUIPMENT		X	X								X
	ASSIGN OPERATOR		X	X				X				
	SCHEDULE		X				X		X			
	ROUTE					X				X	X	
UNLOAD	ASSIGN PERSONNEL		X	X				X				X
	SELECT EQUIPMENT		X									
	SELECT AREA		X		X		X					
LOAD SHIP	SELECT CONTAINERS		X		X							
	ASSIGN PERSONNEL		X	X				X	X			X
	SELECT EQUIPMENT		X									
	ROUTE					X				X	X	

TABLE 9. PROGRAM SELECTION CHART

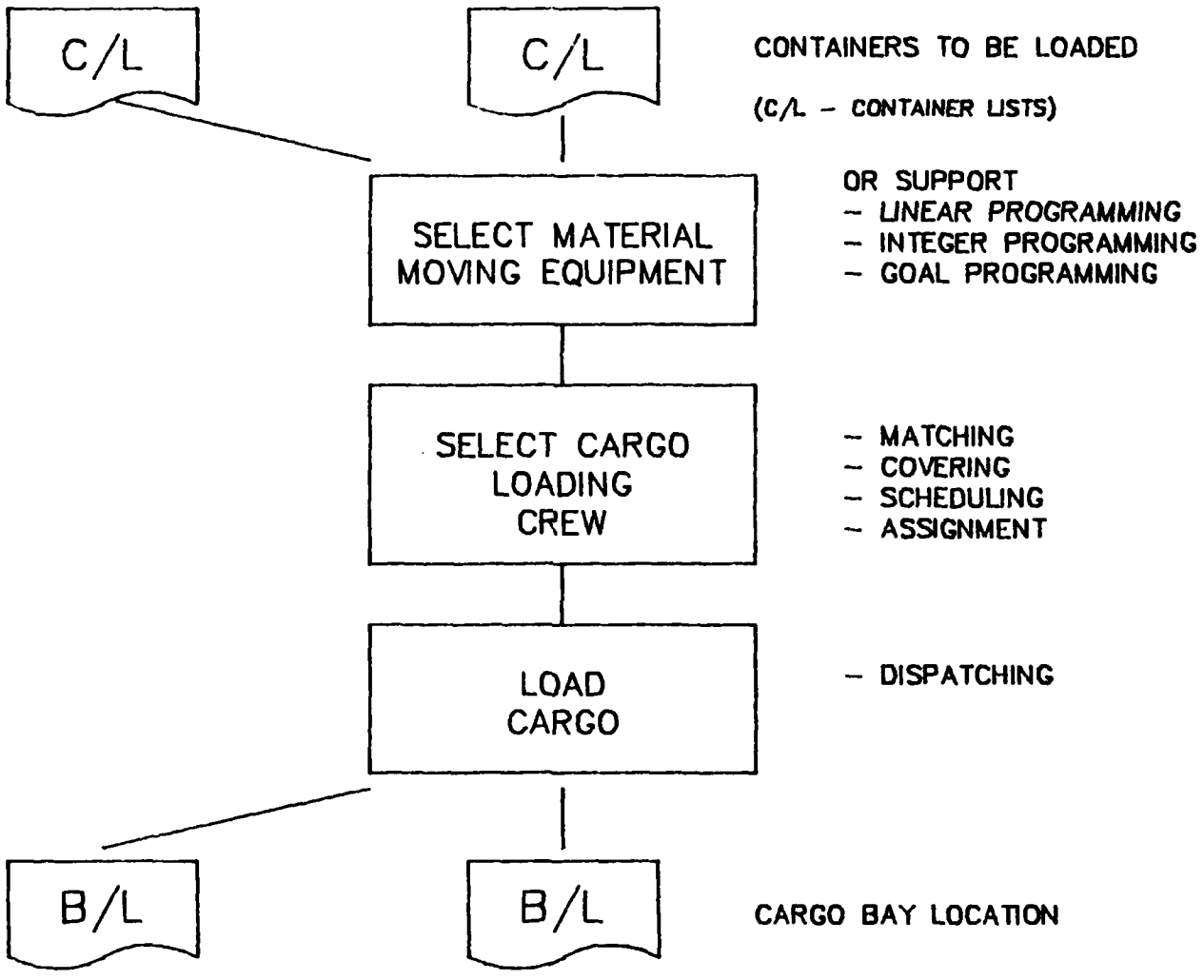


FIGURE 26. SHIP CARGO LOADING

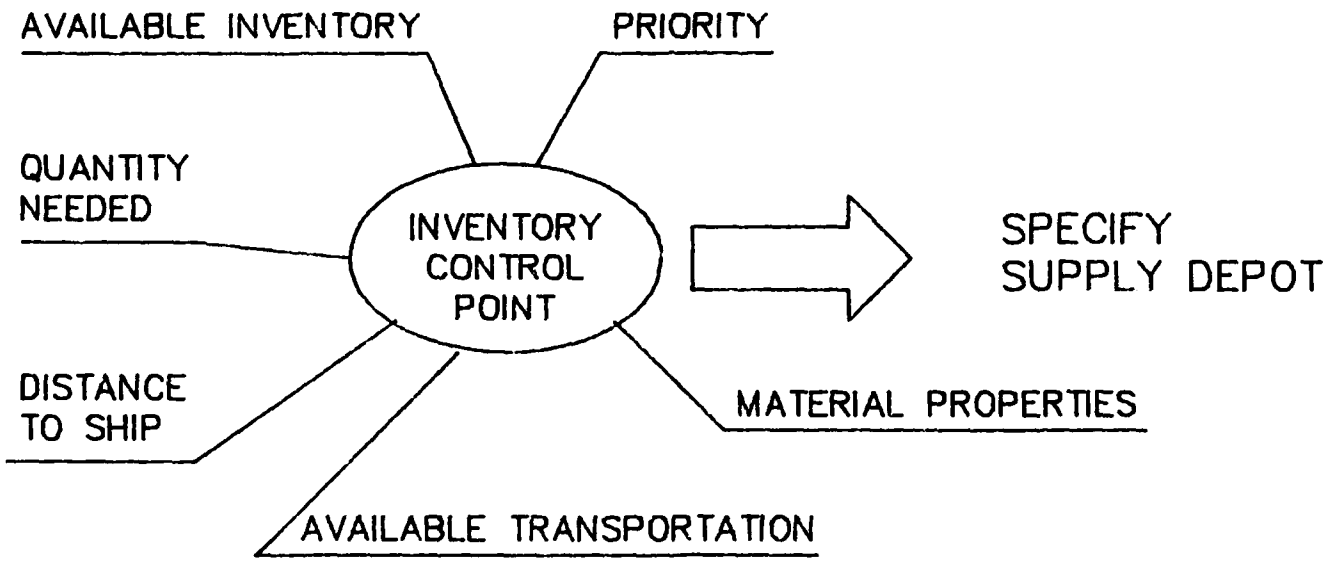


FIGURE 27. SUPPLY DEPOT SELECTION PROCESS

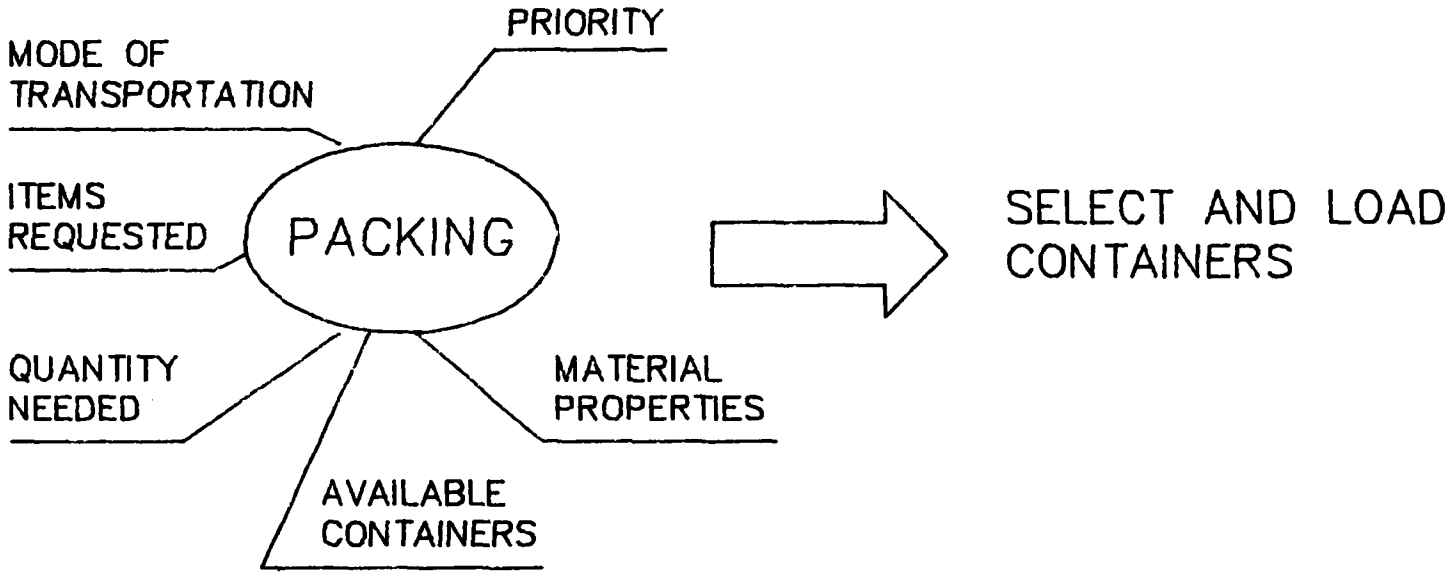


FIGURE 28. PACKING PROCESS

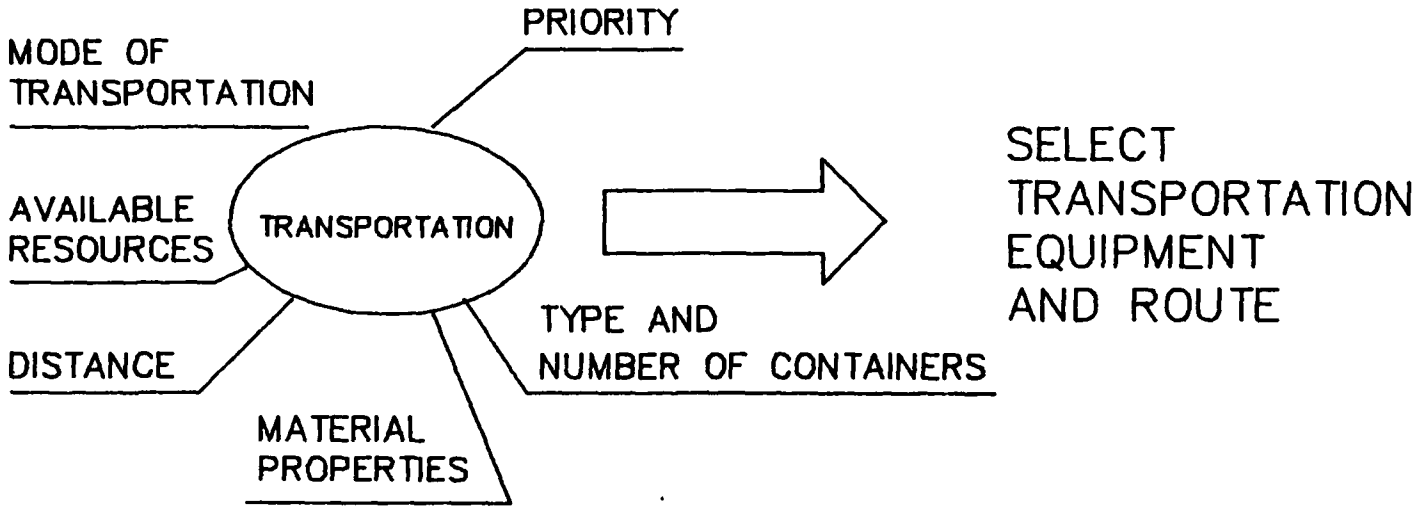


FIGURE 29. TRANSPORTATION PROCESS

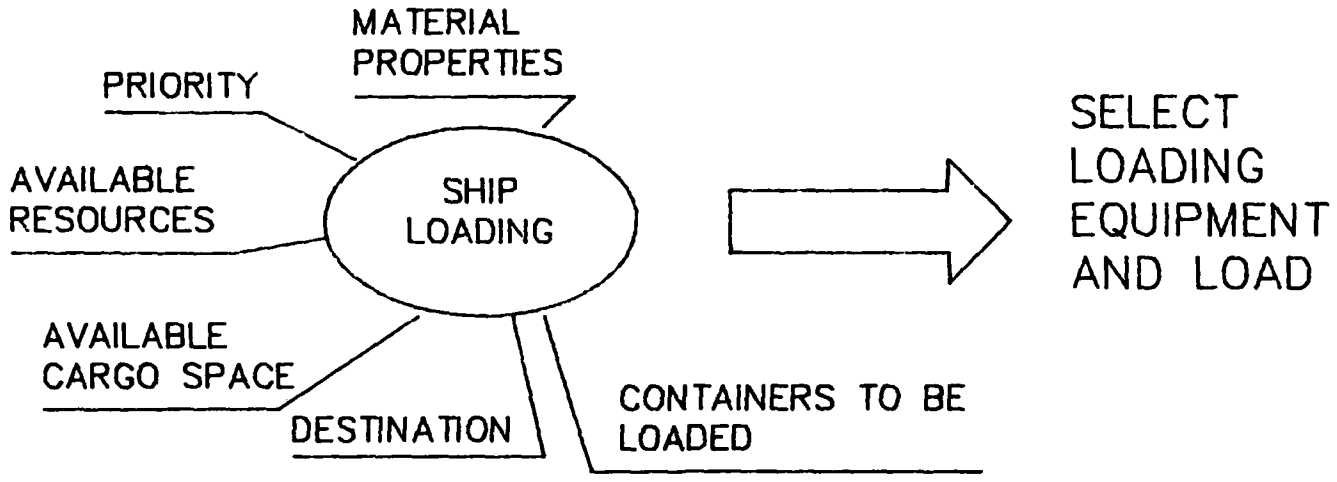


FIGURE 30. SHIP LOADING PROCESS

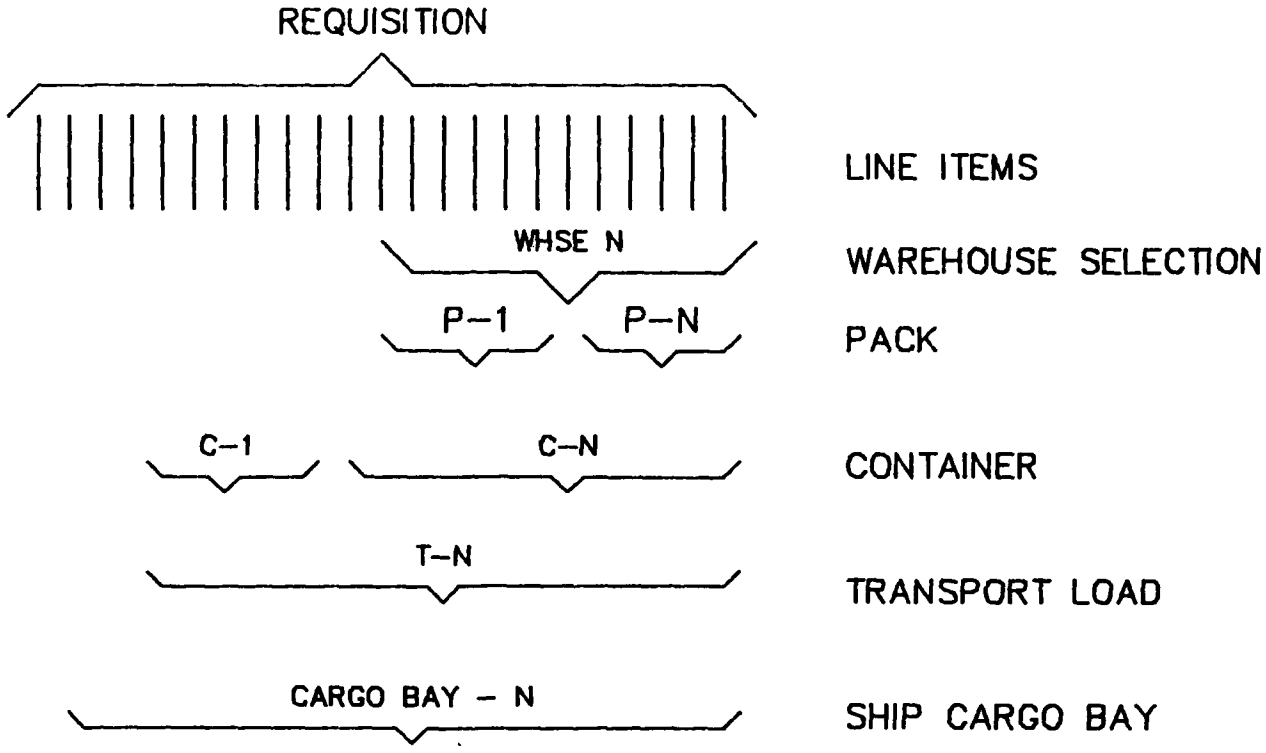


FIGURE 31. MATERIAL DATA AGGREGATION

TABLE 10. Functional Dependency Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
1 CUSTOMER	O																																						
2 CUSTOMER LOCATION	X	O																																					
3 ITEMS NEEDED	X		O																																				
4 QUANTITY NEEDED	X			O																																			
5 PRIORITY	X				O																																		
6 REQUISITION	X					O																																	
7 SUPPLY DEPOT LOCATION							O																																
8 ITEM AVAILABILITY								O																															
9 SHIPPING DISTANCE							X		O																								X						
10 DEPOT SELECTION					X			X	X	O																													
11 EQUIPMENT CAPACITY											O																												
12 PERSONNEL SKILL												O																											
13 MATERIAL CHARACTERISTICS													O																										
14 PERSONNEL AVAILABILITY														O																									
15 EQUIPMENT AVAILABILITY															O																								
16 CONTAINER - AVAILABLE																O																							
17 CONTAINER - FILLED																	O	X			X																		
18 CONTAINER - SELECTION																X		O																					
19 PACKING TASK SCHEDULING											X	X		X	X				O																				
20 PACKING RESRC ASSIGNMENT																				O																			
21 PACKING																					O																		
22 LOADING TASK SCHEDULING											X	X		X	X							O																	
23 LOADING RESRC ASSIGNMENT																							O																
24 LOADING																								O															
25 TRANSPORT EQUIP SELECTION									X					X											O														
26 TRANSPORT TASK SCHEDULING											X	X		X	X											O													
27 TRANSPORT EQUIP ROUTING									X					X												X	O												
28 TRANSPORTATION																											X	O											
29 UNLOADING SCHEDULING											X	X		X	X														X	O									
30 UNLOADING RESRC ASSIGN																														X	O								
31 UNLOADING																															X	O							
32 SHIP LOCATION																																O							
33 SHIP CARGO CAPACITY																																	O						
34 SHIP LOADING SCHEDULING											X	X																				X		O					
35 CARGO ASSIGNMENT																																	X		O				
36 SHIP LOADING																																				O			
37 PERSONNEL UTILIZATION																						X			X											O	X		
38 EQUIPMENT UTILIZATION																						X			X											X	O		
39 LOGISTICS COSTS					X				X													X			X											X		O	

ting that the shipping distance is a function of the supply depot location and the ship location. (In this matrix the row and column matrix designations are identical and the columns are referenced by numbers.)

The functional dependency matrix thus provides a road map for the control of the solution and the means to reference the entities to the data base. In general appearance, the dependency matrix is similar to the popular spreadsheets. In their interpretation, however, there is considerable difference because the functional dependencies are usually complex and their solution require the use of complex OR optimization algorithms.

There are a number of rows that do not contain any marked entries. These are considered to be the given variables (such as item availability, personnel skill, etc.). In a more complex logistics problem environment, these variables could carry further functional dependency and would contain marked entries.

There are additional lower level functional dependencies besides the system level dependency shown. These will be established during the more detailed subsystem analysis process and will be represented again as functional dependencies in subsystem dependency matrices.

The functional dependency matrix for a specific ship loading environment will probably be different and will be adjusted for the local differences.

d. Ship Loading Problem Data Base

In the conventional approach to logistics problem solving, the data base design did not receive a major share of attention. Each specific algorithm was usually developed for a general solution and the data elements were represented as generalized matrices. This approach not only required considerable effort in preparing the input data, but also required translation of the results in a form suitable for the end user. As a result, an efficient iterative solution involving a number of different algorithms was generally not feasible because of the extra work required from the analyst.

The data base design approach suggested for the proposed implementation is different in several respects. First, the common data elements are identified and data records defined. Second, a dynamic structure is established needed to support the problem solution requirements. Third, a linkage is provided to the solution control, enabling determination of the status of currently available information.

The initial format of the proposed data base is included in Appendix C. A detailed discussion of this format follows.

The data base format presented in Appendix A is based on PASCAL programming language data type definition. In developing the initial data base description, emphasis was placed on those data elements which play a major part in the solution algorithms. A number of data fields were left as general descriptors. These normally will be extended to match the specific system requirements.

The initial section contains data formats for the general descriptors used to

identify a unique entity, such as material, package, warehouse, etc. These identifiers were expressed as string variables to provide flexibility in handling different length data. Another alternative would be to use fixed length character arrays. This approach would be typically used in established, well-defined logistics environment.

The next section contains general records which are used throughout the rest of the data base. A general record labeled "box" is used to identify three-dimensional entities, such as packages, cargo bays, etc. Again, in a fully-developed system, a more complex three-dimensional structure will be needed. Two additional records define date and time in DoD format.

The personnel section contains record definitions to identify and structure the available manpower resources. The specific details include identification of a specific person (an identification or Social Security number), skill specification, current status, and a structure for creating task teams. (Note that the data representation is given bottom-up. This is required by the PASCAL programming structure.)

A similar set of records are used to identify the material moving equipment. These records help to identify the specific units and provide fields to describe equipment capacity and capabilities, operator skill requirements, etc. Again, provisions are available to group the equipment in resource pools.

A set of records are used to identity the various tasks. In addition to facilities for describing the current status of the task, other fields enable the chaining of assigned task personnel and equipment. This particular task structure is needed in the dynamic logistics environment. Highly structured languages, such as PASCAL, are particularly suitable for representing the constantly changing environment. In older languages, such as FORTRAN, the dynamic task representation is rather difficult and inefficient.

The next group of data records are supply oriented. These include data records for describing the requisition, inventory and packaged goods ready for transportation. All of these records are of dynamic structure. This enables one to easily handle variable length requisitions and the hierarchical data structures res needed to describe contents of packages and containers. As such, they easily support the data aggregation concept discussed previously and depicted in Figure 31.

The following group represents the key logistics functional tasks, such as packing, loading, transporting, unloading, and finally loading the ship. All of these records are work-center oriented and fully support the dynamic chaining of personnel and equipment data.

The last set of records deal with the description of the physical facilities, such as warehouses, docks, and ships. Here the hierarchical description capability is particularly important. These records support both the fixed and the dynamic data representations. The fixed data represent the more permanent physical features of the facilities, and the dynamic part provides a linkage to assign the dynamic data that flow in the system. For example, a loaded container is assigned to the loading dock, transportation vehicle, shipside dock, and finally to a cargo bay within the ship. During this time the contents of

the container remain fixed, but the assignment undergoes a change from one center to another.

Reviewing the data base design it is apparent that we are dealing with a relatively complex structure which, nevertheless, can be represented in a highly structured and hierarchical manner. It is also obvious using the integrated approach that the data handling in the system can be simplified and the data transfer made more reliable because there is less translation required from one data format to another.

In a fully developed system the logistics problem description language and the standard data elements will play a major role in improving the overall efficiency of the logistics system. The initial design presented in this report should not be considered as final but only as an initial starting point for the Phase II design.

CONCLUSIONS AND RECOMMENDATIONS

CONCLUSIONS

In this feasibility investigation emphasis was placed on developing a system oriented approach to implementing logistics software on microcomputers. A summary of conclusions reached at the completion of the Phase I effort follows.

Complexity

Military logistics systems are complex and costly to operate. The complexity is caused in part by the large number of supplies carried and the availability of different modes of transportation. It is further complicated by the multi-echelon organization and the inherent communications problems. To provide the required microcomputer support in this environment both centralized and decentralized operation is needed. Thus, the analysts and support personnel can operate independently and also have data communications access to other logistics units. To deal with the overall complexity it is necessary to use computer aided problem partition and decomposition techniques.

Algorithms and Logistics Models

Many logistics oriented algorithms have been developed, validated, and implemented on mainframe computers. A very recent catalog of the various logistics models used in DoD environment is available from Defense Logistics Information Exchange (DLSIE).[13] Majority of the models in this catalog have been developed for mainframe computers, are batch oriented, and do not have graphics support capabilities. Only a few special purpose logistics models specifically developed for microcomputers were listed in the DLSIE collection.

A representative sample of OR algorithms useful in logistics applications were selected and converted to microcomputer based PASCAL programs. The resulting code was tested and evaluated with respect to code size, solution accuracy, and execution time.

The conversion time to PASCAL was reasonable (1-2 days per well-documented and validated algorithm). There were no conversion problems, except for slight differences in some of PASCAL I/O statements and supporting utilities. Addition of interactive capabilities to the basic algorithms also posed no special problems. All of the basic algorithms were found to be reasonably fast and flexible.

Computer code sizes for the individual algorithms used in the evaluation process are listed in Table 10. From an examination of this table it can be concluded that it will be possible to have a number of basic algorithms residing in memory at the same time. This capability will help to speed up the optimization process and greatly improve the system response time in complex iterative situations.

TABLE 10. Representative OR Algorithm Code Size

Algorithm	Code Size in Bytes
Revised Simplex	2,608
Dual Simplex	2,896
Ford-Fulkerson Transportation	2,032
Gomory Integer Programming	1,808
Balas 0-1 Integer Programming	2,432
Knapsack Reduction	2,832
Knapsack Approximation	1,840
Martello-Toth Knapsack Backtracking	3,792
Set Partition Reduction	5,216
Set Partition Implicit Enumeration	6,112
Dijkstra's Shortest Path	976
Pape, d'Esopo and Moore Shortest Path	816
Floyd's Shortest Path	1,072
Kruskal Minimum Spanning Tree	1,600
Prim Nearest Neighbor Spanning Tree	944
Maximum Flow	5,584
Busacker's Minimum Cost Flow	1,952
Maximum Cardinality Matching (Pape-Conradt)	1,424
Branch and Bound Traveling Salesman	3,488
Traveling Salesman - Approximate Insertion	960
Traveling Salesman - Two-Edge Optimization	1,040
Traveling Salesman - Three Edge Optimization	1,632
Graph Coloring Ordering	2,480
Graph Coloring - Sequential Ordering with Insertion	5,776
Balas Network Scheduling	3,792
Tarjan's Strongly Connected Components	3,616
Sharir's Strongly Connected Components	2,256
Interval Reduction	2,736
Queue Simulator (Numerical Output)	3,488
Queue Simulator (Color Graphics Output)	9,104

Note: Turbo Pascal 3.0 Compiler with 8087 support.
Above code sizes do not include data arrays.

The accuracy obtained was as good as those reported previously for the original mainframe based evaluations. This can be explained by the inherent computational accuracy of the floating point processor (80-bit internal data representation).

Time limitations prevented exhaustive testing and evaluation of the actual algorithm solution times. The initial tests revealed that for problems containing up to 20 variables (using the examples reported in literature), the response time usually was under one minute. Since many of the OR algorithms selected are of an iterative type, the actual running time is not only a function of the number of variables, but also of the actual values of these variables. A more realistic running time evaluation will be obained during the Phase II effort using some real logistics data.

The initial evaluation proved that the transfer of well known algorithms from a mainframe formulation to a microcomputer is feasible, and that the solution times are acceptable for interactive environment for small or medium size problems with no sacrifice in solution accuracy.

Further improvements can be expected after optimizing the algorithms for microcomputer environment. No such optimization was attempted in the initial effort.

Logistics Work Station (LWS)

The concept of a Logistics Workstation was modeled after the existing Engineering Work Station (EWS). EWS concept has been used for a number of years in engineering environment, particularly in computer design. An extensive set developmenal tools are available to the user and provide fast and reliable computational capability. Without these tools the design of the complex VLSI circuits would not be possible.

The proposed LWS would provide a similar capability for the operations research analyst and others involved in making complex logistics tradeoff decisions. It must be noted that the proposed concept is in its initial conceptual stage and that the actual LWS configuration will be somewhat different to take advantage of the rapid developments in the microcomputer industry.

It is possible, however, to sketch general microcomputer requirements for a LWS:

286 or 68020 Microprocessor (386 in future)
8 Mhz (or Higher) Clock Speed
High Speed Floating Point Processor
1 MByte Memory
Floppy Disk
20 Mbyte Hard Disk
Color Graphics Support
High Resolution Color Monitor

Printer
Communications Interface
Local Area Network Capability

To support computing-intensive tasks, special purpose hardware accelelators could be developed. Again, in concept this would be similar to the EWS approach.

The supporting software, in addition to Operating System and utilities, would include the general purpose logistics software, as described earlier. Present day Operating Systems do not have the capability to provide fast and efficient multitasking capability. It is expected that these capabilities will be available within a year or two.

Although the proposed concept is new in logistics environment, its success in other fields make it particularly promising as a tool to reduce the escalating logistics system operating costs.

Logistics Description Language (LDL)

The need for a formal language has been identified. This specific area has not received any real attention and development effort in the past. As a result logistics problems have to be described in an informal way, that is subject to misinterpretation, and then converted manually to an input data file. This process is not only error prone but also very labor intensive. The proposed LDL concept would provide a formal way to describe the problem and also permit the use of a compiler to convert the LDL description automatically to an input file. Using this approach the error rate would be reduced and a considerable saving in manual processing time obtained. An additional benefit would be the capability to communicate the logistics information in a formal way to other areas (design, procurement, etc.). The CALS effort mentioned before is focused on the design and technical data interface and covers only part of the logistics environment.

Again, this same concept has been applied very successfully in other areas. Typical examples include computer language compilers, silicon compilers (used in VLSI circuit layout), numerical machining languages, etc. The opportunity to apply this concept to logistics is here.

Logistics Data Base Requirements

Military logistics data bases are very large and dynamic in nature. To avoid costly data base search and data conversion, it is important that these data bases be structured and standards established for individual data records and elements. The same situation is experienced in other areas dealing with high complexity, where a number of data base support tools have been developed to deal with the complex environment. Again, transfer and adaptation of these tools to the logistics environment is feasible. An illustration of the data base design for the Ship Loading Problem was discussed earlier in this report. The sample data base presented in Appendix B serves to illustrate how the dynamic part of the

data base can be created using dynamic records linked via pointers.

Concepts such as data encoding and the use of differential files can further reduce the actual storage requirements at some minimal sacrifice in processing time.

User Interface Requirements

User interface is without doubt the most important issue in the logistics software implementation on microcomputers. Without an efficient and flexible interface the whole implementation project is doomed to a failure.

The key issues here is to provide a flexible support to a multitude of users with different experience levels. The proposed approach is based on separating the dialog part from the algorithm, so that the dialog can be modified without having to modify the algorithm internally. This approach has been implemented partially on a number of projects of a highly interactive nature. The results have been very promising; no apparent decrease in computer response was experienced, program development time was decreased and the maintenance of the final program simplified.

The specific details of the user interface received an indepth coverage. It was hoped that by specifying the interface requirements early and by stressing their importance the problems caused by the conventional approach, where the user interface is usually an afterthought, could be eliminated.

Solution Control

A feasible concept based on data flow analysis has been presented. This particular concept has been used before in a different environment and will require major modifications to adapt it to the logistics problem solving. Specific modifications include interfacing with data base, solution algorithms, and user dialog. Fortunately, the basic algorithm is flexible, fast, and easy to extend to meet the solution control requirements.

The advantages of using this approach are numerous. Not only can it provide problem iteration control and check on data availability but it can also keep track of the individual program steps and control data transfer between modules. As a result, the user is no longer required to manually select the different algorithms and to continuously transribe data from one format to another.

Applications Example

The Ship Loading Problem was used to illustrate the variety of OR techniques needed in a practical logistics area. It also helped to show that the same OR algorithms are applicable in different areas after suitable interpretation of the variables. Although the Ship Loading Problem covers only a small part of the logistics operations, nevertheless, it demonstrates that the

basic problems, such as resource allocation, scheduling, assignment, etc., occur in different environments and at different times. It is expected that in the full LWS implementation a number of other algorithms will be added to provide even more flexibility and computing power.

RECOMMENDATIONS

Based on the Phase I findings the development of a LWS appears feasible. It is recommended that the following steps be taken:

1. The full logistics environment should be reviewed to identify potential applications of microcomputers, and algorithms needed to solve the specific problems. (Table 10 shows some of the potential applications.)

2. The identified algorithms should be converted and optimized for use on microcomputers.

3. The Logistics Description Language effort should continue to obtain a complete description needed to describe all common logistics problems. This task should be closely coordinated with the CALS effort to provide efficient transfer of data from and to the design environment.

4. Logistics Data Base concept should be further extended to permit the incorporation of other data elements needed in a full system.

5. The Solution Control concept should be further extended to include capabilities needed to interface with the data base, solution algorithms, and user dialog and fully implemented in an operational environment.

6. A LWS prototype developed and demonstrated.

Accomplishment of the above will be the first step towards providing the logistics analysts with similar work station capabilities enjoyed by computer system analysts and designers.

TABLE 11. Areas of Computer Application in Logistics

a. Conceptual phase

Mission scenarios and profiles
Feasibility studies
Operational and maintenance concepts
Operations and support effectiveness factors
Establishing environmental criteria
Logistics planning

b. Advanced development phase

System analysis
System optimization
Synthesis and definition
Allocation of logistics requirements
Development of criteria for logistics supportability
Preliminary logistics support analysis

c. Detailed design and development phase

Detailed equipment design
Design liaison and logistics support services
Reliability and maintainability prediction
Utilizations of design aids
Logistic support analysis
Logistic provisioning data
Design review
Test and evaluation of equipment
Feedback and corrective action

d. Operational use phase

Operation and maintenance of equipment in the field
Accomplishment of special tests
Assessment of logistics support capability
Feedback and corrective action processing

REFERENCES

1. Dehnig, W., Essig, H., and Maass, S., The Adaptation of Virtual Man-Computer Interfaces to User Requirements in Dialogs, Springer-Verlag, Berlin, 1981.

2. Blackman, M., The Design of Real Time Applications, Wiley, New York, NY, 1975.

3. Gaines, B.R., Shaw, M.L.G., The Art of Computer Conversation, Prentice Hall, Englewood Cliffs, NJ, 1984.

4. Hayes, R.M. and Becker, J. (eds.), Man-Machine Communication, Wiley-Interscience, New York, NY, 1970.

5. Monk, A. (ed.), Fundamentals of Human-Computer Interaction, Academic Press, London, 1985.

6. Barstow, D.R. et al (eds.), Interactive Programming Environments, McGraw-Hill, New York, NY, 1984.

7. Kim, W. (ed.), Query Processing in Database Systems, Springer-Verlag, Berlin, 1985.

8. Overman, M.H., The Design of Dynamic Data Structures, Springer-Verlag, Berlin, 1983.

9. Schmidt, J.W., Brodie, M.L. (eds.), Relational Database Systems, Springer-Verlag, Berlin, 1983.

10. Maier, D., The Theory of Relational Databases, Computer Science Press, Rockville, Maryland, 1983.

11. Monahan, R.H. and Schubert, W., An Analysis of Naval Personnel Resource Allocation to Logistics, Volume I - Navy Sea-Based Personnel Resource Allocations to Logistics Functions, SRI International, Menlo Park, California, August 1982. (AD A122 763)

12. Monahan, R.H., An Analysis of Naval Personnel Resource Allocation to Logistics, Volume II - Methodology for Examining Effects of Personnel Shortfalls on Navy Logistics Personnel Readiness, SRI International, Menlo Park, CA, August, 1982. (AD A122 764)

13. -, Department of Defense Catalog of Logistics Models, Defense Logistics Studies Information Exchange, Fort Lee, VA, January 1985. (AD B089 161)

14. Steward, D.V., "On an Approach to Techniques for the Analysis of the Structure of Large Systems of Equations," SIAM Review, Vol. 4, No. 4, p. 321, 1962.

15. Steward, D.V., Systems Analysis and Management, Petrocelli Books, New York, NY, 1981.

16. Tarjan, R.E., "Space Efficient Implementation of Graph Search Methods," ACM Transactions on Mathematical Software, Vol.9, No.3, pp. 32-36, September 1983.

17. Sharir, M., "A Strong-Connectivity Algorithm and its Application in Data Flow Analysis,", Computers and Mathematics with Applications, Vol. 7, pp. 67-72, 1981.

18. Aho, A.U. et al, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.

19. Hecht, M.S. and Ullman, J.D., "A Simple Algorithm for Global Data Flow Analysis," SIAM Journal on Computing, pp. 519-532, 1975.

20. Kam, J.B. and Ullman, J.D., "Global Data Flow Analysis and Iterative Algorithms," Journal of the ACM, Vol. 23, pp.158-171, 1976.

21. Himmelblau, D.M. (ed.), Decomposition of Large-Scale Problems, Elsevier North-Holland, 1973.

22. Syslo, M.M. et al, Discrete Optimization Algorithms, Prentice-Hall, Englewood Cliffs, NJ, 1983.

BIBLIOGRAPHY

-, Material Distribution Study, Volume II Technical Report, Department of Defense, July 1978.

-, System Engineering Management Guide, Defense Systems Management College, Fort Belvoir, VA, October 1983. (AD A136 020)

-, The Impact of Low Cost Computing Technologies on the Department of Defense, Arthur Young & Company, Washington, DC, April 1983. (AD A128 489)

-, Analytical and Computational Issues in Logistics R&D, Army Research Office, Research Triangle Park, NC, May 1984. (AD A153 829)

Adam, E.E. and Ebert, R.J., Production and Operations Management, Prentice-Hall, Englewood Cliffs, NJ, 1982.

Aho, A.A. et al, Data Structures and Algorithms, Addison-Wesley, Reading, MA, 1983.

Balas, E., "Machine Sequencing: Disjunctive Graphs and Degree-Constrained Subgraphs," Naval Research Logistics Quarterly, Vol. 17, pp. 1-10, 1970.

Ballou, R.H., Basic Business Logistics, Prentice-Hall, Englewood Cliffs, NJ, 1978.

Ballou, R.H., Business Logistics Management, Prentice-Hall, Englewood Cliffs, NJ, 1979.

Barnett, A.I., "Control Strategies for Transport Systems with Nonlinear Waiting Costs," Transportation Science, Vol.12, pp. 119-136, 1978.

Baron, R.J., Data Structures and their Implementation, Van Norstrand Reinhold, New York, NY, 1980.

Battersby, A., Network Analysis for Planning and Scheduling, Wiley, New York, NY, 1970.

Bellmore, M., "A Decomposable Transshipment Algorithm for a Multi-Period Transportation Problem," Naval Research Logistics Quarterly, Vol. 16, pp. 517-524, 1969.

Becker, H.B., Functional Analysis of Information Networks, Wiley-Interscience, New York, NY, 1973.

Bennington, G. and Lubore, S., "Resource Allocation for Transportation," Naval Research Logistics Quarterly, Vol. 17, pp. 471-484, 1970.

Best, M.J., Linear Programming: Active Set Analysis and Computer Programs, Prentice-Hall, Englewood Cliffs, NJ, 1985.

Blanchard, B.S., Logistics Engineering and Management, Prentice-Hall,

Englewoood Cliffs, NJ, 1981.

Brown, P.J. (ed.), Software Portability, Cambridge University Press, London, 1977.

Budde, R. (ed.), Approaches to Prototyping, Springer-Verlag, Berlin, 1984.

Buffa, E.S. and Dyer, J.S., Essentials of Management Science/Operations Research, Wiley, New York, NY, 1978.

Chofaras, D.N., Designing and Implementing Local Area Networks, McGraw-Hill, New York, NY, 1984.

Clark, A.J., and Scarf, H., "Optimal Policies for a Multi-Echelon Inventory Problem," Management Science, Vol. 6, pp. 475-490, 1960.

Conway, R.W., Maxwell, W.L., and Miller, L.W., Theory of Scheduling, Addison-Wesley Publishing Company, Reading, MA, 1967.

Coppola, A., Artificial Intelligence Applications to Maintenance Technology Working Group Report, Institute for Defense Analysis, Alexandria, Virginia, August 1983. (AD A137 329)

Coyle, J.J. and Bardi, E.J., The Management of Business Logistics, West Publishing Company, St. Paul, MN, 1980.

Crowston, W.B., "Decision CPM: Network Reduction and Solution," Operations Research Quarterly, Vol. 21, pp. 435-452, 1970.

Davidson, D.G. and Fraser J.J., Adapting Logistics Models to a Microcomputer for Interface with Computer-Aided Design Systems, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, September 1984. (AD A147 666)

Dinkel, J.J. et al, Management Science: Text and Applications, Richard D. Irwin, Homewood, IL, 1978.

Dixon, E.J, The Database Management Module of the SPLICE (Stock Point Logistics Integrated Communications Environment) System, Naval Postgraduate School, Monterey, CA, June 1983. (AD A132 795)

Dretske, F.I., Knowledge and the Flow of Information, MIT Press, Cambridge, MA, 1981.

Drezner, S.M. and Hillestad, R.J., Logistics Models: Evolution and Future Trends, Rand Corporation Report P-6748, March 1982. (AD A118 808)

Eilon, S. et al, Distribution Management: Mathematical Modelling and Practical Analysis, Hafner, New York, NY, 1971.

Elmaghraby, S.E., Activity Networks: Project Planning and Control by Network Models, Wiley, New York, NY, 1977.

Fair, M.L. and Williams, E.W. Jr., Economics of Transportation and Logistics, Business Publications, Plano, TX, 1975.

Fiorello, M., Defense Integrated Data System (DIDS) Computer System Analysis, Logistics Management Institute, Washington, DC, February 1977. (AD A040 354)

Fox, M.S., The Intelligent Management System: An Overview, Carnegie-Mellon University, Pittsburgh, PA, December 1982. (AD A126 345)

Fox, M.S., Constraint-Directed Search: A Case Study of Job-Shop Scheduling, Carnegie-Mellon University, Pittsburgh, PA, December 1983. (AD A138 307)

Frank, H. et al, "Optimal Design of Centralized Computer Networks," Networks, Vol. 1, pp. 43-57, 1971.

Frank, H. et al, Communication, Transmission, and Transportation Networks, Addison-Wesley, Reading, MA, 1971.

Garfinkel, R.S. and Nemhauser, G.L., Integer Programming, Wiley, New York, NY, 1972.

Garfinkel, R.S. and Rao, M.R., "The Bottleneck Transportation Problem," Naval Research Logistics Quarterly, Vol. 18, pp. 465-472, 1971.

Gaver, D.P. and Thompson, G.L., Programming and Probability Models in Operations Research, Brooks/Cole, Monterey, CA, 1973.

Goheen, L.C. et al, An Expansion of BALFRAM (Balanced Force Requirements Analysis Model) to Include Representation of Logistics Functions, SRI International, Menlo Park, CA, June 1982. (AD A116 590)

Golden, B.L., "Implementing Vehicle Routing Algorithms," Networks, Vol. 7, pp. 113-148, 1977.

Gonnet, G.H., Handbook of Algorithms and Data Structures, Addison_Wesley, London, 1984.

Hammer, P.L. and Rudeanu, S., Boolean Methods in Operations Research, Springer-Verlag, New York, NY, 1968.

Hammer, P.L., "Time-minimizing Transportation Problems," Naval Research Logistics Quarterly, Vol. 16, No. 3, pp. 345-357, 1969.

Handler, G.Y. and Mirchandani, Location on Networks: Theory and Algorithms, MIT Press, Cambridge, MA, 1979.

Highfill, K.G., A Decision Model for Selection of Microcomputers and Operating Systems, Naval Postgraduate School, Monterey, CA, June 1984. (AD A149 076)

House, W.C. (ed.), Decision Support Systems: A Data-Based, Model-Oriented, User-Developed Discipline, Petrocelli Books, New York, NY, 1983.

Ignall, E.J. et al, "Using Simulation to Develop and Validate Analytical Models," Operations Research, Vol. 26, pp. 237-253, 1978.

Jackson, E.B. (ed.), Industrial Information Systems, Dowden, Hutchinson and Ross, Stroudsburg, Pennsylvania, 1978.

Kalaba, R.E. and Juncosa, M.L., "Optimal Design and Utilization of Communications Networks," Management Science, Vol. 3, pp. 233-44, 1956.

Keen, P.G.W. and Morton, M.S.S., Decision Support Systems: An Organizational Perspective, Addison-Wesley, Reading, MA, 1978.

Kiebler, K.K. and Rozycki, R.F., Assessing Department of Defense Logistics System Capability, Logistics Management Institute, Washington D.C., August 1983. (AD A141 491)

Kleindorfer, P.R., Miller, L.W., and Chang M.G., System Design Procedures for Improved Effectiveness of Military Sea Transportation Service Operations, University of Pennsylvania, Philadelphia, PA, September 1983. (AD A 133 231)

Kleinrock, L., Queueing Systems, Vols. I and II, Wiley, New York, NY, 1976.

Laderman, J., "Vessel Allocation by Linear Programming," Naval Research Logistics Quarterly, Vol. 13, pp. 315-320, 1966.

Ladson, L.S., "An Efficient Algorithm for Multi-Item Scheduling," Operations Research, Vol. 19, pp. 946-969, 1971.

Lampson, B.W. et al (eds.), Distributed Systems: Architecture and Implementation, Springer-Verlag, New York, NY, 1981.

Larson, R.C. and Odoni A.R., Urban Operations Research, Prentice-Hall, Englewood Cliffs, NJ, 1981.

Lee, N., An Account of the Heuristics in the Conceptual Design of Data Processing System, Army Electronics Command, Fort Monmouth, NJ, April 1975. (AD A008 891)

Leong-Hong, B. and Plagman, B.K., Data Dictionary/Directory Systems: Administration, Implementation and Usage, Wiley-Interscience, New York, NY, 1982.

Makridakis, S. and Wheelwright, S.C., Forecasting Methods and Applications, Wiley, New York, NY, 1978.

Marlow, W.H., Editor, Modern Trends in Logistics Research, MIT Press, Cambridge, MA, 1976.

Moder, J.J. (ed.), Handbook of Operations Research, Van Norstrand Reinhold, New York, NY, 1978.

Muckstadt, J.A. et al, "A Model for Multi-Item Multi-Echelon Inventory

System," Management Science, Vol. 20, No.4, pp. 472-481, December 1973.

O'Neill, R.R. and Weinstock J.K., "On Cargo Handling System," Naval Research Logistics Quarterly, Vol. 1, pp. 282-288, 1954.

O'Neill, R.R., "Analysis and Monte Carlo Simulation of Cargo Handling," Naval Research Logistics Quarterly, Vol. 4, pp. 223-236, 1957.

O'Neill, R.R., "Scheduling of Cargo Containers," Naval Research Logistics Quarterly, Vol. 7., pp. 577-584, 1960.

Pearl, J., Heuristics - Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley, Reading, MA, 1984.

Perkinson, R.C., Data Analysis: The Key to Data Base Design, QED Information Sciences, Wellesley, MA, 1984.

Price, P.L., Data Processing Workstation for Computer Aided Warehouse Design and Evaluation, David Taylor Naval Ship Research and Development Center, Bethesda, Maryland, December, 1984. (AD A151 487)

Pritsker, A.A. and Hopp, W.W., "GERT: Graphical Evaluation and Review Techniques," Journal of Industrial Engineering, Vol. 17, No.5, 1966.

Pritsker, A.A., Modeling and Analysis Using Q-GERT Networks, Halsted Press, New York, NY, 1977.

Pruzan, P.M. and Jackson, J.T., "The Many Product Cargo Loading Problem," Naval Research Logistics Quarterly, Vol. 14, pp. 381-390; 1967.

Rhees, T.R., Kuskey, K.P., and Kraft, R.N., Research on Management Concepts for Large-Scale Simulation Naval Warfare, Decisions and Designs, McLean, Virginia, July 1981. (AD A103 089)

Rhees, T.R., Kuskey, K.P., and Kraft, R.N., Research on Management for Simulation Programs in the U.S. Navy, Decisions and Designs, McLean, Virginia, November 1981. (AD A109 030)

Rose, W., Logistics Management, Wm. C. Brown Company, Dubuque, IA, 1979.

Sanderson, G.F., Opportune Rescheduling for Military Airlift Command Cargo and Passenger Missions, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, December 1978. (AD A064 065)

Scheer, A. -W., Computer: A Challenge for Business Administration, Springer-Verlag, Berlin, 1985.

Schniederjans, M.J., Linear Goal Programming, Petrocelli Books, Princeton, NJ, 1984.

Sherif, Y.S. and Smith, M.L., "Optimal Maintenance Models for Systems Subject to Failure," Naval Research Logistics Quarterly, Vol. 28, No. 1, 1981.

Sherif, Y.S. and Kheir, A.H., Weapons System Analysis, University of Alabama, Huntsville, Alabama, October 1981. (AD A 111 637)

Sowa, J.F., Conceptual Structures, Addison_Wesley, Reading, MA, 1984.

Srinivasan, V. and Thompson, G.L., "Determining Optimal Growth Paths in Logistics Operations," Naval Research Logistics Quarterly, Vol. 19, pp. 575-599, 1972.

Stanton, R. et al, A Model of the Uniform Material Movement and Issue Priority System Within the DOD Logistics Process, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, June 1978. (AD A059 180)

Steed, R.F., The Evolution of the Department of Defense Transportation System: Current Problems and Trends, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, September 1982. (AD A122 811)

Sullivan, N.M., An Analysis of Demand Forecasting Emphasizing Inventory Effectiveness, Naval Postgraduate School, Monterey, CA, September 1983. (AD A138 615)

Sutherland, J.W., Systems: Analysis, Administration and Architecture, Van Norstrand Reinhold, New York, NY, 1975.

Swarc, W., "The Truck Assignment Problem," Naval Research Logistics Quarterly, Vol. 14, pp. 529-557, 1967.

Synott, W.R. and Gruber, W.H., Information Resource Management, Wiley-Interscience, New York, NY, 1981.

Tersine, R.J., Production/Operations Management: Concepts, Structure, and Analysis, Elsevier, North-Holland, New York, NY, 1980.

Thierauf, R.J., Systems Analysis and Design of Real-Time Management Information Systems, Prentice-Hall, Englewood Cliffs, NJ,1975.

Townsend, W.B, Artificial Intelligence Techniques for Industrial Applications in Job Shop Scheduling, Naval Postgraduate School, Monterey, CA, June 1983. (AD A132 164)

Tummala, V.M.R., Decision Analysis with Business Applications, Intext Educational Publishers, New York, NY, 1973.

Voich, J., Jr. et al, Information Systems for Operations and Management, South-Western, Cincinnati, OH, 1975.

Wagner, H.M., Principles of Operations Research, Prentice-Hall, Englewood Cliffs, NJ, 1975.

Wenocur, M.L., A Production Network Model and its Diffusion Approximation, Stanford Univerity, Stanford, CA, September 1982. (AD A120 685)

Williamson, S.G., Combinatorics for Computer Science, Computer Science

Press, Rockville, MD, 1985.

Wise, D.W. et al, Micro-Computers: A Technology Forecast and Assessment to the Year 2000, Wiley-Interscience, New York, NY, 1980.

Wood, W.T., The Use of Machine Aids in Dynamic Multi-Task Environments, Massachussetts Institute of Technology, Cambridge, MA, June 1982. (AD A118 419)

Woods, W. et al, Research in Knowledge Representation for Natural Language Understanding, Bolt, Beranek and Newman Inc., Cambridge, MA, September 1982. (AD A127 323)

Wright, K.J. and Hamilton, W.P., Manufacturing Technology Program, Information System: Functional Description, Logistics Management Institute, Washington, DC, February 1983. (AD A127 293)

Yeh, R.T. et al, Decision Support Systems, University of Texas, Austin, TX, September 1977. (AD A108 104)

GLOSSARY

Constraints - Restrictions or key boundary conditions that impact the overall capability, priority, and resources in system acquisition.

Design Parameters - Qualitative, quantitative, physical, and functional value characteristics that are inputs to the design process, for use in design tradeoffs, risk analyses, and development of a system that is responsive to system requirements.

Facilities - The permanent or semipermanent real property assets required to support the material system, including conducting studies to define types of facilities or facility improvements, locations, space needs, environmental requirements, and equipment. One of the principal elements of ILS.

Goals - Values, or a range of values, apportioned to the various design, operational, and support elements of a system which are established to optimize system requirements.

Integrated Logistic Support (ILS) - A disciplined approach to the activities necessary to: (a) cause support considerations to be integrated into system and equipment design, (b) develop support requirements that are consistently related to design and to each other, (c) acquire the required support; and (d) provide the required support during the operational phase at minimum cost.

Logistic Support Analysis (LSA) - The selective application of scientific and engineering efforts undertaken during the acquisition process, as part of the system engineering and design process, to assist in complying with supportability and other ILS objectives.

Manpower - The total demand, expressed in terms of the number of individuals, associated with the system. Manpower is indexed by manpower requirements, which consist of quantified lists of jobs, slots, or billets that are characterized by the descriptions of the required number of individuals who fill the job, slots, or billets.

Manpower and Personnel - The identification and acquisition of military and civilian personnel with the skills and the grade required to operate and support a material system at peacetime and wartime rates. One of the principal elements of ILS.

Objectives - Qualitative or quantitative values, or range of values, apportioned to the various design, operational, and support elements of a system which represent the desirable levels of performance. Objectives are subject to tradeoffs to optimize system requirements.

Optimization Models - Models which accurately describe a given system and which can be used, through sensitivity analysis, to determine the best operation of the system being modeled.

Personnel - The supply of individuals, identified by specialty or classification, skill, skill level, and rate or rank, required to satisfy the manpower demand associated with the system. This supply includes both those individuals who support the system directly (i.e., operate and maintain the system), and those individuals who support the system indiorectly by performing those functions necessary to produce and maintain the personnel required to support the system directly. Indirect support functions include recruitment, training, retention, and development.

Provisioning - The process of determining and acquiring the range and quantity (depth) of spares and repair parts, and support and test equipment required to operate and maintain an end item of material for an initial period of service.

Supply Support - All management actions, procedures, and techniques required to determine requirements for, acquire, catalog, receive, store, transfer, issue, and dispose of secondary items. This includes provisioning for initial support as well as replenishment supply support. One of the principal elements of ILS.

Supportability - The degree to which system design characteristics and planned logistics resources including manpower meet system peacetime operational and wartime utilization requirements.

Support System - A composite of all the resources that must be acquired for operating and maintaining a system or equipment throughout its life cycle.

Tradeoff - The determination of the optimum balance between system characteristics (cost, schedule, performance, and supportability).

Source: MIL-STD-1388-1A Logistic Support Analysis

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

ADA	DoD High Level Programming Language (R)
ALGOL	High Level Programming Language
ASCII	American Standards Code for Information Interchange
BASIC	Beginner's All-purpose Symbolic Instruction Code
BNF	Backus-Naur Form
B-TREE	Data Storage Structure
CAD	Computer Aided Design
CALS	Computer Aided Logistics Support
CPM	Critical Path Method
CP/M	Digital Research Operating System for Microcomputers (R)
CRT	Cathode Ray Tube
DLSIE	Defense Logistics Studies Information Exchange
DP	Dynamic Programming
EWS	Engineering Work Station
FORTRAN	FORmula TRANslator (Programming Language)
GERT	Graphical Evaluation and Review Technique
HOL	Higher Order Language
I/O	Input/Output
ILS	Integrated Logistic Support
IP	Integer Programming
LDL	Logistics Description Language
LGP	Linear Goal Programming
LP	Linear Programming
LSA	Logistic Support Analysis
LTK	Logistics Tool Kit
LWS	Logistics Work Station
MILSTAMP	Military Standard Transportation and Movement Procedures
MSDOS	Microsoft Operating System (R)
PASCAL	Programming Language
PCDOS	IBM Personal Computer Operating System (R)
PERT	Program Evaluation and Review Technique
POL	Problem Oriented Language
PROLOG	PROgramming in LOGic (Programming Language)
SCC	Strongly Connected Component (Topology)
SPICE	Electronic Circuit Analysis Program
VLSI	Very Large Scale Integration

APPENDIX A

PROCEDURE SUMMARIES

OR Classification: LINEAR PROGRAMMING

Program Name:

PSIMPLEX

Type:

General purpose

Algorithm:

Revised Simplex Algorithm

Global Constants:

Number of constraints
Number of variables

Input:

A - Left-hand-side coefficient array
B - Right-hand-side coefficient array (constraints)
C - Cost array

Output:

Nonfeasible (yes/no)
Unbounded (yes/no)
Optimal basic variables
Optimal value of objective function

Logistics Applications:

Optimal allocation of resources
Investment planning
Inventory control
Staffing

Ship Loading Applications:

Assignment of material moving equipment and operators
Selection of goods for shipment

OR Classification: LINEAR PROGRAMMING

Program Name:

DSIMPLEX

Type:

General purpose linear programming

Algorithm:

Dual Simplex Algorithm

Global Constants:

Number of constraints
Number of variables

Input:

A - Left-hand-side coefficient array
B - Right-hand-side coefficient array (constraints)
C - Cost array

Output:

Nonfeasible (Yes/No)
Unbounded (Yes/No)
Optimal basic variables
Optimal value of objective function

Logistics Applications:

Optimal allocation of resources
Inventory control
Sensitivity analysis

Ship Loading Applications:

Assignment of material moving equipment and operators
Selection of goods for shipment

OR Classification: TRANSPORTATION

Program Name:

TRANSPORT

Type:

Transportation cost minimization

Algorithm:

Ford and Fulkerson Primal Dual Algorithm

Global Constants:

Number of supply nodes
Number of demand nodes

Input:

A - Array of supplies
B - Array of demands
C - Cost array (unit transportation costs)

Output:

Array of optimal shipping
Optimal shipping cost

Logistics Applications:

Transportation
Distribution
Allocation of resources
Assignment of personnel and equipment
Communication network planning
Plant location

Ship Loading Applications:

Warehouse, transportation, and ship loading crew selection
Assignment of material moving equipment
Assignment of operators to equipment

OR Classification: GOAL PROGRAMMING

Program Name:

LGP

Type:

Linear goal programming

Algorithm:

Dual Simplex Goal Programming Algorithm

Global Constants:

Number of goal constraints
Number of decision variables
Number of priority levels

Input:

Left-hand-side coefficient array
Right-hand-side coefficient array (constraints)
Sign symbols (+,-)
Cost coefficient array

Output:

Nonfeasible (yes/no)
Unbounded (yes/no)
Optimal basic variables
Optimal value of objective function

Logistics Applications:

Optimal allocation of resources with different priorities
Multiple criteria decision making
Manpower planning

Ship Loading Applications:

Assignment of material moving equipment and operators
Selection of goods for shipment

OR Classification: INTEGER PROGRAMMING

Program Name:

GOMORY

Type:

General purpose integer programming

Algorithm:

Gomory Cutting Plane Method

Global Constants:

Number of constraints
Number of variables

Input:

A - Left-hand-side coefficient array
B - Right-hand-side coefficient array (constraints)
C - Cost array

Output:

Non-feasible (yes/no)
Optimal solution vector

Logistics Applications:

Optimal resource allocation where variables are constrained to constants
Budgeting
Warehouse location
Line balancing

Ship Loading Applications:

Loading task scheduling
Operator scheduling

OR Classification: INTEGER PROGRAMMING

Program Name:

BALAS

Type:

Zero - one integer programming

Algorithm:

Balas Additive Algorithm

Global Constants:

Number of constraints
Number of variables

Input:

A - Left-hand-side coefficient array
B - Right-hand-side coefficient array (constraints)
C - Cost array

Output:

Nonfeasible (yes/no)
Optimal solution vector
Optimum value of the objective function

Logistics Applications:

Resource allocation where variables are constrained to zero-one values
Facility location

Ship Loading Applications:

Resource and task assignment

OR Classification: KNAPSACK PROBLEM

Program Name:

KNAPRED

Type:

Bounded knapsack problem

Algorithm:

Knapsack Reduction Algorithm

Global Constants:

Number of variables

Input:

P - Array of object profits
W - Array of object weights
T - Total weight limit of the knapsack

Output:

Array of reducible arrays
Profit of the partial assignment
Optimum solution flag - indicates if the partial solution is optimal

Logistics Applications:

Selection and loading of goods
Budget control

Ship Loading Applications:

Container loading
Cargo loading
Cargo selection

OR Classification: KNAPSACK PROBLEM

Program Name:

KNAPAPPROX

Type:

Approximate heuristic solution

Algorithm:

Knapsack Approximation Algorithm

Global Constants:

Number of variables (objects) in the knapsack problem

Input:

P - Array of object profits
W - Array of object weights
T - Total weight limit of the knapsack

Output:

Solution vector
Profit of the solution obtained

Logistics Applications:

Selection and loading of goods
Budget control

Ship Loading Applications:

Container loading
Cargo loading
Cargo selection

OR Classification: KNAPSACK PROBLEM

Program Name:

KNAPBACKTRACK

Type:

Exact enumeration

Algorithm:

Martello-Toth Algorithm

Global Constants:

Number of variables (objects) in the knapsack problem

Input:

P - Array of object profits
W - Array of object weights
T - Total weight limit of the knapsack

Output:

Solution vector
Profit of the solution obtained

Logistics Applications:

Selection and loading of goods
Budget control

Ship Loading Applications:

Container loading
Cargo loading
Cargo selection

OR Classification: SET COVERING PROBLEM

Program Name:

SETPARTRED

Algorithm:

Set Partition Reduction Algorithm

Global Constants:

Number of elements in the base set
Number of subsets

Input:

A - Array of 0-1 constraint matrix
S - Array of family of sets

Output:

Nonfeasible flag (yes/no)
Optimality flag (yes/no)
Array of the reduced problem
Cost of the partial solution

Logistics Applications:

Crew scheduling
Delivery scheduling
Facility location

Ship Loading Applications:

Loading crew assignment and scheduling
Transportation crew assignment and scheduling
Transportation route scheduling
Pickup and delivery of goods

OR Classification: SET COVERING PROBLEM

Program Name:

SETPARTBACKTRACK

Algorithm:

Implicit Enumeration Algorithm

Global Constants:

Number of elements in the base set
Number of subsets

Input:

A - Array of 0-1 constraint matrix
S - Array of family of sets

Output:

Nonfeasible flag (yes/no)
Optimality flag (yes/no)
Array of the reduced problem
Cost of the partial solution

Logistics Applications:

Facility location
Loading crew assignment and scheduling
Transportation crew assignment and scheduling
Transportation route scheduling
Pickup and delivery of goods

Ship Loading Applications:

Loading and transportation crew assignment and scheduling

OR Classification: NETWORK

Program Name:

DIJKSTRA

Type:

Shortest path algorithm

Algorithm:

Dijkstra's Algorithm

Global Constants:

Number of nodes in the network

Input:

S - Source node
T - Sink node
W - Weight matrix of the network

Output:

Path flag (yes/no)
Shortest path distance array

Logistics Applications:

Shortest path selection in transportation network
Fastest, most economical routing
Personnel scheduling

Ship Loading Applications:

Material moving equipment routing
Operating crew scheduling

OR Classification: NETWORK

Program Name:

PDM

Type:

Shortest path algorithm

Algorithm:

Pape, d'Esopo, and Moore Algorithm

Global Constants:

Number of nodes in the network
Number of edges in the network

Input:

S - Source node
T - Sink node
W - Weight matrix of the network

Output:

Path flag (*yes/no*)
Shortest path distance array

Logistics Applications:

Shortest path selection in transportation network
Fastest, most economical routing
Personnel scheduling

Ship Loading Applications:

Material moving equipment routing
Operating crew scheduling

OR Classification: Network

Program Name:

FLOYD

Type:

Shortest path algorithm

Algorithm:

Floyd's Algorithm

Global Constants:

Number of nodes in the network

Input:

W - Weight matrix of the network

Output:

Negacycle flag (yes/no)
Shortest path distance matrix from every node to every other node
Path matrix

Logistics Applications:

Shortest path selection in transportation network
Fastest, most economical routing
Personnel scheduling

Ship Loading Applications:

Material moving equipment routing
Operating crew scheduling

OR Classification: NETWORK

Program Name:

KRUSKAL

Type:

Minimum spanning tree

Algorithm:

Kruskal's algorithm

Global Constants:

Number of nodes in the network
Number of edges in the network

Input:

E - Edge matrix
W - Weight matrix of the network

Output:

Connect flag (yes/no)
Minimum spanning tree matrix
Total weight of minimum spanning tree

Logistics Applications:

Communications network design
Facility layout

Ship Loading Applications:

Cabling layout for communications network
Packaging area layout in a warehouse

OR Classification: NETWORK

Program Name:

PRIM

Type:

Minimum spanning tree

Algorithm:

Prim's Nearest-Neighbor Algorithm

Global Constants:

Number of nodes in the network

Input:

E - Edge matrix
W - Weight matrix of the network

Output:

Connect flag (yes/no - indicates if the network is disconnected)
Minimum spanning tree matrix
Total weight of minimum spanning tree

Logistics Applications:

Commmunications network design
Facility layout

Ship Loading Applications:

Cabling layout for communications system
Packaging area layout

OR Classification: NETWORK

Program Name:

MAXFLOW

Type:

Maximum flow problem

Algorithm:

Malhotra, Kumar, and Maheshwari Algorithm

Global Constants:

Number of nodes in the network

Input:

Source nodes
Sink nodes
Array capacities of the edge

Output:

Array of max-flow pattern

Logistics Applications:

Transportation network optimization

Ship Loading Applications:

Transmission of information
Shipment of goods
Movement of people

OR Classification: NETWORK

Program Name:

BUSACKACKER

Type:

Minimum cost flow problem

Algorithm:

Busacker and Gowen Algorithm

Global Constants:

Number of nodes in the network

Input:

Source nodes
Sink nodes
Cost matrix of the given network
Capacity matrix of the given network

Output:

Minimum cost flow pattern for the given network
Total cost

Logistics Applications:

Routing optimization: minimum driving time, minimum fuel consumption

Ship Loading Applications:

Material handling vehicle routing with constraints

OR Classification: NETWORK

Program Name:

MATCH

Type:

Maximum cardinality matching

Algorithm:

Pape-Conradt Algorithm

Global Constants:

Number of nodes in the network
Number of edges in the network

Input:

Array of edges

Output:

Matched nodes
Number of unmatched nodes

Logistics Applications:

Assignment

Ship Loading Applications:

Assignment of personnel to tasks
Assignment of operators to equipment

OR Classification: NETWORK

Program Name:

BABTSP

Type:

Traveling Salesman Problem

Algorithm:

Branch-and-Bound Algorithm

Global Constants:

Number of nodes in the network

Input:

Array of nodes
Weight matrix of the given network

Output:

Array of optimal route
Total weight of the route

Logistics Applications:

Transport routing

Ship Loading Applications:

Vehicle routing

OR Classification: NETWORK

Program Name:

FITSP

Type:

Traveling Salesman Problem

Algorithm:

Approximate Farthest Insertion Algorithm

Global Constants:

Number of nodes in the network

Input:

Starting node
Array of nodes
Weight matrix of the given network

Output:

Traveling salesman route
Total weight of the route

Logistics Applications:

Transport routing

Ship Loading Applications:

Vehicle routing

OR Classification: NETWORK

Program Name:

TWOOPT

Type:

Traveling Salesman Problem

Algorithm:

Two-Edge Optimization Algorithm

Global Constants:

Number of nodes in the network

Input:

Starting node
Array specifying the initial traveling salesman route
Weight matrix of the given network

Output:

Traveling salesman optimal local route
Total weight of the route

Logistics Applications:

Routing improvement

Ship Loading Applications:

Vehicle routing improvement

OR Classification: NETWORK

Program Name:

THREEOPT

Type:

Traveling salesman problem

Algorithm:

Three-Edge Optimization Algorithm

Global Constants:

Number of nodes in the network

Input:

Starting node
Array specifying the initial traveling salesman route
Weight matrix of the given network

Output:

Traveling salesman optimal local route
Total weight of the route

Logistics Applications:

Routing improvement

Ship Loading Applications:

Vehicle routing improvement

OR Classification: GRAPH COLORING

Program Name:

ORDER

Algorithm:

Color Ordering Algorithm

Global Constants:

Number of vertices to be colored

Input:

Graph
Smallest-first or largest-first ordering flag

Output:

Ordering of vertices of the graph

Logistics Applications:

Assignment and scheduling

Ship Loading Applications:

Optimal schedule of tasks in man-machine environment

OR Classification: GRAPH COLORING

Program Name:

SEQCOL

Algorithm:

Simple Sequential Ordering Algorithm

Global Constants:

Number of vertices to be colored

Input:

Graph
Array of vertice ordering in the graph

Output:

Array of vertex colors

Logistics Applications:

Assignment and scheduling

Ship Loading Applications:

Optimal schedule of tasks in man-machine environment

OR Classification: GRAPH COLORING

Program Name:

SEQINT

Algorithm:

Sequential Ordering Algorithm with Interchange

Global Constants:

Number of vertices to be colored

Input:

Graph
Array of vertice ordering in the graph

Output:

Array of vertex colors

Logistics Applications:

Assignment and scheduling

Ship Loading Applications:

Optimal schedule of tasks in man-machine environment

OR Classification: SCHEDULING

Program Name:

NSCHED

Type:

Network scheduling

Algorithm:

Balas Algorithm

Global Constants:

Number of activities
Number of arcs in the network

Input:

Initial ordering of nodes
Array of processing times

Output:

Topological ordering of nodes
Array of earliest possible starting time
Number of feasible networks generated by the procedure

Logistics Applications:

Task scheduling

Ship Loading Applications:

Loading task scheduling where precedence contraints exist

OR Classification: SCHEDULING

Program Name:

LEVELSCHED

Type:

Network scheduling

Algorithm:

Hu's Modified Level Scheduling Algorithm

Global Constants:

Number of machines which can be used
Number of tasks in the system

Input:

Precedence constraints

Output:

Array of schedule times

Logistics Applications:

Flow job shop scheduling

Ship Loading Applications:

Loading task scheduling

OR Classification: GRAPH

Program Name:

TARJAN

Type:

Strongly connected components

Algorithm:

Tarjan's Algorithm for Directed Graphs

Global Constants:

Number of vertices
Number of edges

Input:

Graph of system components

Output:

Array of all strongly connected components

Logistics Applications:

Large problem analysis to identify loops (circuits)

Ship Loading Applications:

Problem partition and control

OR Classification: GRAPH

Program Name:

SHARIR

Type:

Strongly connected components

Algorithm:

Sharir's Algorithm for Directed Graphs

Global Constants:

Number of vertices
Number of edges

Input:

Graph
Entry node

Output:

List of roots of strongly connected components in their reverse postorder and a map of nodes for each strongly connected component in reverse postorder

Logistics Applications:

Large problem analysis

Ship Loading Applications:

Data flow analysis
Problem solution control

OR Classification: QUEUEING

Program Name:

QUESIM

Type:

Queueing simulation

Algorithm:

Numerical solution of differential equations using Runge-Kutta algorithm

Global Constants:

Number of system states
Number of transition arrows

Input:

Transition graph
Intial state probabilities

Output:

State probabilities as functions of time

Logistics Applications:

Waiting line analysis
Service efficiency

Ship Loading Applications:

Congestion at loading docks
Service efficiency in warehouse, transportation, and loading operations

OR Classification: GRAPH

Program Name:

INTERVAL

Type:

Interval search

Algorithm:

Modified Tarjan's Algorithm

Global Constants:

Number of vertices
Number of edges

Input:

Graph
Entry header

Output:

List of intervals in processing sequence

Logistics Applications:

Large problem solution control

Ship Loading Applications:

Data flow analysis
Problem solution control

APPENDIX B

SHIPLOADING DATA BASE

```
                    { *** DATA BASE DEFINITION *** }
TYPE

                              {*** DESCRIPTORS ***}

    p_ident  = string[20];                      {part/material identifier}
    pk_ident = string[20];                      {package identifier}
    c_ident  = string[10];                      {container identifier}
    t_ident  = string[10];                      {task identifier}
    s_ident  = string[10];                      {ship identifier}
    wh_ident = string[10];                      {warehouse identifier}
    wc_ident = string[5];                       {work center identifier}
    l_ident  = string[5];                       {location designator - A/N}
    descr    = string[20];                      {general descriptor}
    unit_id  = string[5];                       {material unit}
    cust_id  = string[20];                      {customer identifier}


                              {*** GENERAL ***}

    box = record                                {package physical size}
            width    : integer;
            height   : integer;
            depth    : integer;
          end;

    coord = record                              {location coordinates}
              x,y    : integer;
            end;

    m_date =
          record                                {military date record}
            yy,mm,dd : array[1..2] of char;
          end;

    m_time =
          record
            hh,mm,ss : array[1..2] of char      {24-hour military clock}
          end;


                              {*** PERSONNEL ***}

    pers_ident  = string[20];                   {personnel identification}

    p_status = (p_avail,p_busy,p_nonavail);     {personnel status}

    skill_ptr = ^p_skill;

    p_skill  =
          record                                {personnel skill}
            skill_id : string[10];
```

```
        next     : skill_ptr;
      end;

person = record                                  {basic personnel record}
           pers_id  : pers_ident;
           status   : p_status;
           skill    : ^p_skill;
         end;

pers_resources =
      record                                     {total assigned personnel}
        assign   : ^person
      end;


                    {*** EQUIPMENT ***}

equip_ident = string[20];                        {equipment identification}

e_status = (e_avail,e_busy,e_repair);            {equipment status}

e_type   = (truck, lift, crane);                 {equipment type}

e_capab  =
      record                                     {equipment capacity}
        volume   : box;
        weight   : real;
        speed    : real;
        range    : real;
      end;

e_ptr = ^equipment;

equipment =
      record                                     {basic equipment record}
        equip_id : equip_ident;
        kind_of  : e_type;
        operator : p_skill;                      {operator skill requirements}
        status   : e_status;
        capab    : e_capab;
        next     : e_ptr
      end;

eq_resources =
      record                                     {total assigned equipment}
        assign   : ^equipment;
      end;


                    {*** TASK ***}

t_status = (active, suspended, completed);
```

```
clock = record
          date      : m_date;
          time      : m_time
        end;

t_ptr = ^task;                                  {task pointer - queue link}

task = record                                   {basic task record}
         task_id    : t_ident;
         t_start    : clock;                    {date/time}
         t_stat     : t_status;                 {current task status}
         t_descr    : descr;
         t_loc      : l_ident;                  {work center}
         t_pers     : ^person;                  {assigned personnel}
         t_equip    : ^equipment;               {assigned equipment}
         next_task  : ^t_ptr;
       end;

                    {*** REQUISITION ****}

p_rating = 0..15;                               {priority rating}

item_ptr = ^item;                               {item link}

item = record                                   {line item in requisition}
         line_no   : integer;
         part_id   : p_ident;
         qty       : integer;
         unit      : unit_id;
         next      : item_ptr;
       end;

requisition =
       record                                   {basic requisition}
         items     : ^item;
         priority  : p_rating;
         date      : m_date;
         to_loc    : coord;
         customer  : cust_id;
       end;

                    {*** INVENTORY ***}

inv_rec =
       record                                   {basic inventory record}
         part_id   : p_ident;
         qty       : integer;
         unit      : unit_id;
         whse_id   : l_ident;
       end;
```

```
icp = record                                    {inventory control point}
        inv      : ^inv_rec
      end;


part = record                                   {part specification}
         part_id  : p_ident;
         weight   : real;
         size     : box;
       end;

whse_inv =
       record                                   {locator}
         part_id  : p_ident;
         loc_id   : coord;
         qty      : integer;
       end;


                    {*** PACKAGING ***}

pack_ptr = ^pack;

pack = record                                   {package}
         pack_id  : pk_ident;
         p_part   : part;
         qty      : integer;
       end;

container_capacity =
      record                                    {container specification}
        cont_id  : c_ident;
        size     : box;
        weight   : real;                        {max weight limit}
      end;

cont_ptr = ^filled_container;

filled_container =
      record
        cont_id  : c_ident;
        priority : p_rating;
        size     : box;
        weight   : real;                        {actual container weight}
        customer : cust_id;
        next_pack      : ^pack_ptr;
        next_container : ^cont_ptr;
      end;

packaging_area =
      record                                    {packaging area specification}
        ident    : l_ident;
        where    : coord
```

```
      end;

packaging_pool =
      record                                    {packaging resources}
        pool_id  : wc_ident;
        area     : ^packaging_area;
        wp_pers  : ^person;
        wp_equip : ^equipment;
      end;

packaging_tasks =
      record                                    {active tasks in warehouse}
        p_task   : ^task
      end;


                    {*** LOADING ***}

loading_pool =
      record                                    {loading resources}
        pool_id  : wc_ident;
        pers     : ^person;
        equip    : ^equipment;
      end;

loading_tasks =
      record                                    {active tasks in warehouse}
        l_task       : ^task;
        l_containers : ^filled_container;
      end;


                    {*** TRANSPORTATION ***}

transport_pool =
      record                                    {transportation resources}
        pool_id  : wc_ident;
        tp_pers  : ^person;
        tp_equip : ^equipment;
      end;

transport_tasks =
      record                                    {active transportation tasks}
        t_tasks  : ^task;
        t_load   : ^filled_container;
      end;


                    {*** DOCK UNLOAD ***}

dock_pool =
      record                                    {dock resources}
        dock_id  : wc_ident;
```

```
        d_pers   : ^person;
        d_equip  : ^equipment;
      end;

dock_tasks =
      record                                    {active unloading tasks}
        d_task   : ^task;
        containers : ^filled_container;
      end;


                    {*** SHIP LOAD ***}

ship_pool =
      record                                    {ship loading resources}
        ship_id  : wc_ident;
        s_pers   : ^person;
        s_equip  : ^equipment;
      end;

ship_tasks =
      record                                    {active ship loading tasks}
        s_task   : ^task
      end;


                    {*** WAREHOUSE ***}

whse_bay =
      record                                    {warehouse bay location}
        ident    : l_ident;
        location : coord
      end;

warehouse =
      record                                    {warehouse specification}
        whse_id  : wh_ident;
        location : coord;
        w_bays   : ^whse_bay;
      end;

warehouses =
      record                                    {available warehouses}
        whses    : ^warehouse
      end;


                    {*** DOCK ***}

dock_area =
      record                                    {dock area specification}
        ident    : l_ident;
        size     : box;
```

```
        area_loc : coord
      end;

dock = record                                   {dock specification}
         ident    : l_ident;
         dock_loc : coord;
         d_areas  : ^dock_area
       end;

loading_docks =
      record                                    {loading dock specification}
        dock     : ^dock
      end;

unloading_docks =
      record                                    {unloading dock specification}
        dock     : ^dock
      end;


                    {*** SHIP ***}

cargo_bay =
      record                                    {cargo bay specification}
        location : coord;
        volume   : box;
        contents : ^filled_container;           {cargo bay contents}
      end;

ship_ptr = ^ship;                               {ship link}

ship = record                                   {ship specification}
         ship_loc : coord;
         load     : real;                       {actual weight}
         c_areas  : ^cargo_bay
       end;

cargo_fleet =
      record                                    {cargo fleet specification}
        ships    : ^ship
      end;
```

APPENDIX C

DIALOG MODULE IMPLEMENTATION

DIALOG MODULE IMPLEMENTATION

This appendix contains an illustration of the details of implementation associated with the practical tasks of planning and programming dialog modules in an interactive enviroment.

The specific illustration is taken from the Teleorder Program developed by DAINA for use in telemarketing applications. Since this application is highly communications oriented, it will serve to illustrate many of the concepts mentioned in Section III.6.

This specific example deals with the operations associated with a customer record. The record contains 5 fields; its format is shown below:

Name (25 characters)
Street Address (30 characters)
City (20 characters)
State (2 characters)
ZIP (5 integers)

In the actual implementation, the name field is further subdivided with the record containing additional information. These details are not necessary to illustrate the approach and will not be further explained here.

There are a number of dialog oriented activities associated with this record:

1. Creating the original record
2. Displaying the record
3. Editing the record
4. Confirming the accuracy of the record

Each of these cases will be considered separately.

1. CREATING THE ORIGINAL RECORD

Using the proposed concept of partition and hierarchical decomposition this case is handled as follows:

First, the processing module is examined to determine the best partitioning. The objective is to separate all of the dialog oriented actions from the process module. In this particular case the simplest partition can be obtained by issuing from the process module a Get_Customer dialog procedure call. The dialog module then handles all of the interactive details and after completion returns to the process module the results of the dialog -- either a new customer record or an indication that the call was not successful.

Next, a detailed dialog tree is prepared for the particular module. This tree is illustrated in Figure B-1. The top line contains the name of the main dialog module, in this case Get_Customer. The top level of the dialog module may also issue information to the user; for example, Creating Customer

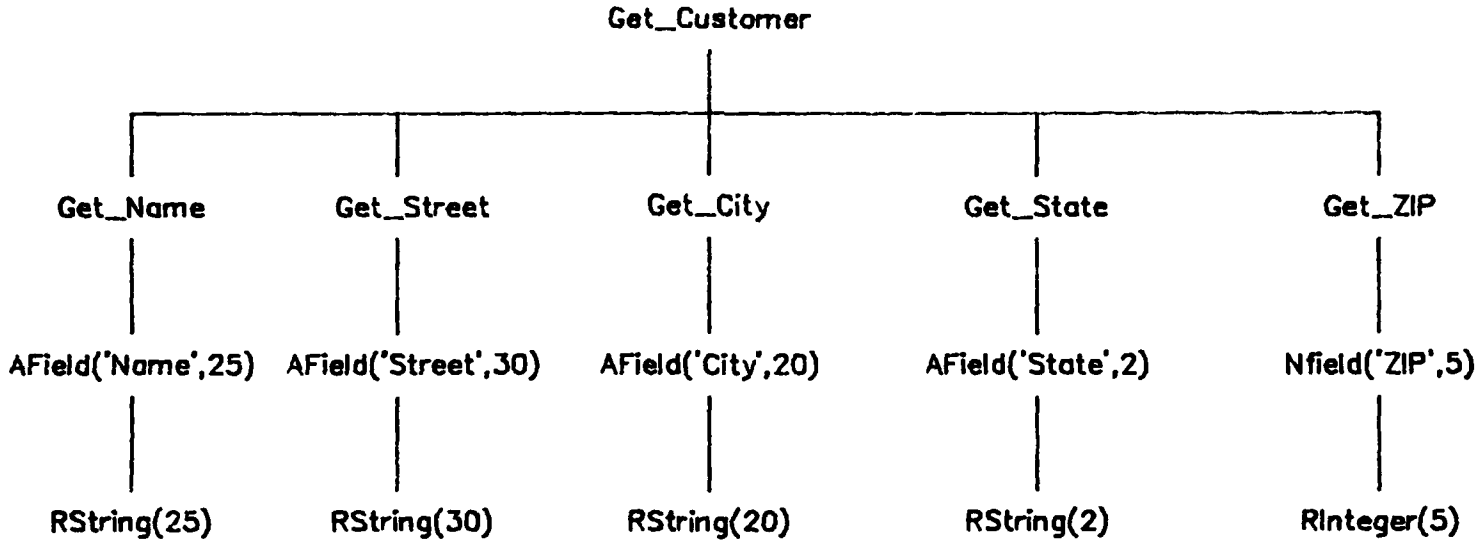


Figure BI. Customer Record Input Dialog Tree

Record.

The actual data record is then examined and further decomposed. In this particular case, it is subdivided into five submodule calls, one for each of the fields. This decomposition results in the second level hierarchy - Get_Name, Get_Street, etc. Further examination of the record structure reveals the type of the field (alpha, numeric, etc) and the specific field dimensions. This decomposition yields the third level where calls to record field processing procedures are issued.

The procedure call, AField('Name',25), is further processed. The first variable in the parameter list is used to create an informative prompt for the user, and the second variable is used to issue a call to read a character string not exceeding 25 characters in length.

Other fields are then processed in a similar manner. In the tree diagram, the branches are processed from top to bottom and from left to right. Each field is completed before the next field.

At the initial examination this process may appear overly complex and possibly not very efficient. There are, however, advantages to this approach:

Programming is considerably simplified. Procedures at the bottom level will often be used. For example, the Read String (RString) and Read Integer (RInteger) will appear in almost every record. These procedures are not simple; they have to process the character strings originating from the user that may exceed the specified data field length or contain characters (backspace, control characters, etc.) which must be further edited to obtain a clean field value. These read instructions must be able to know from where the information is originating and to select the proper communications channel. They must also know the characteristics of the originating source in order to translate these characters, if necessary, to the system format.

Thus, on further examination, it soon becomes apparent that the proposed approach can handle interactive computing situations without the need for loading different versions of the main program.

A further explanation may be in order. In a multiuser environment, it would be naive to expect that every user would have the same type of terminal or personal computer. Since these terminals have varying characteristics, considerable amout of reformatting is necessary since the line length may be 24, 40, 64, 80, 128, or some other number of characters. The same also applies to the number of lines per screen. The number of possible combinations soon becomes astronomical. It is, therefore, important to do all the necessary conversion on the fly by reference to the user profile file which is normally used to store user terminal characteristics.

2. DISPLAYING CUSTOMER RECORD

This case is illustrated in Figure B-2. The sequence of operations follows the same ground rules as discussed above. The field name information is used

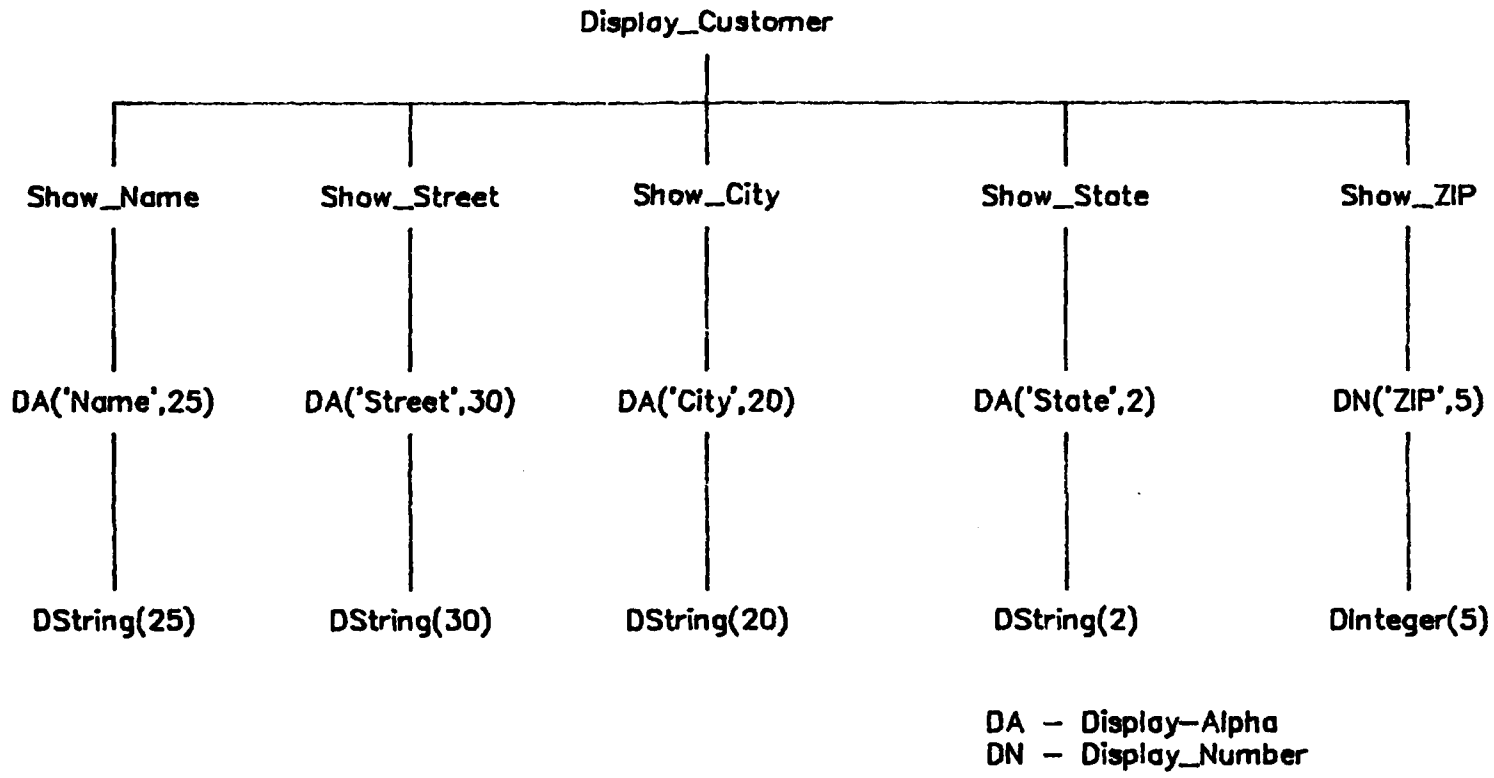


Figure B2. Customer Record Display Tree

to make the display more readable.

Again, at the lowest level, there is considerable repetition in procedure calls meaning that the variety of the lower level procedures will not increase exponentially.

3. EDITING CUSTOMER RECORD

This case is illustrated in Figure B-3. The Display_Customer procedure detailed above is used to display current information. Select_Record_Field procedure displays the options. User can select a specific field at a time and supply the new values. Note that the individual field input procedure calls are the same that were illustrated in Figure B-1.

4. CONFIRMING CUSTOMER RECORD ACCURACY

Another dialog level can be easily introduced where confirmation of record accuracy is needed. In this scheme, illustrated in Figure B-4, the current data is displayed and then the user is queried about the accuracy. If there are errors, then the editing dialog is called.

5. COMMENTS

Initially it may seem that there is a paradox in the proposed approach because of the dialog module separation and the multilevel structure. In reality there are major savings not only in interactive program development time, but even greater savings in program maintenance time. The separation of the dialog from the process modules also permits fine tuning the dialog without affecting the operation of the process modules.

In the development of the Teleorder Program use of partitioning permitted the development of a single program able to support operation in local, remote, and in test mode by controlling only the i/o operations.

There are numerous applications of this approach in the proposed Logistics Work Station. These include the dialogs not only for input, output, and editing operations, but also for controlling the display modes, selecting the solution options, and specifying the report formats.

It is also easy to incorporate other modes of dialog. In the above example, the input sequence was linear, of the type normally encountered in simple system environments. A different dialog could be developed, for example, to support full screen operation. Since the dialog modules are separate from the main program, the switching of the dialog modes could be either under user control or it could be controlled by the user profile file.

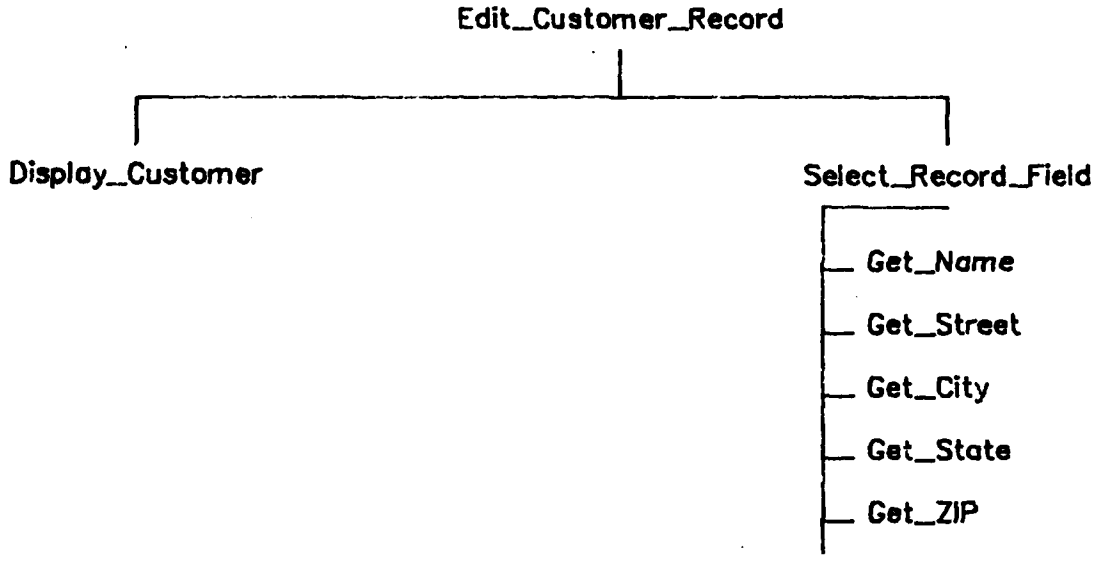


Figure B3. Customer Record Edit Tree

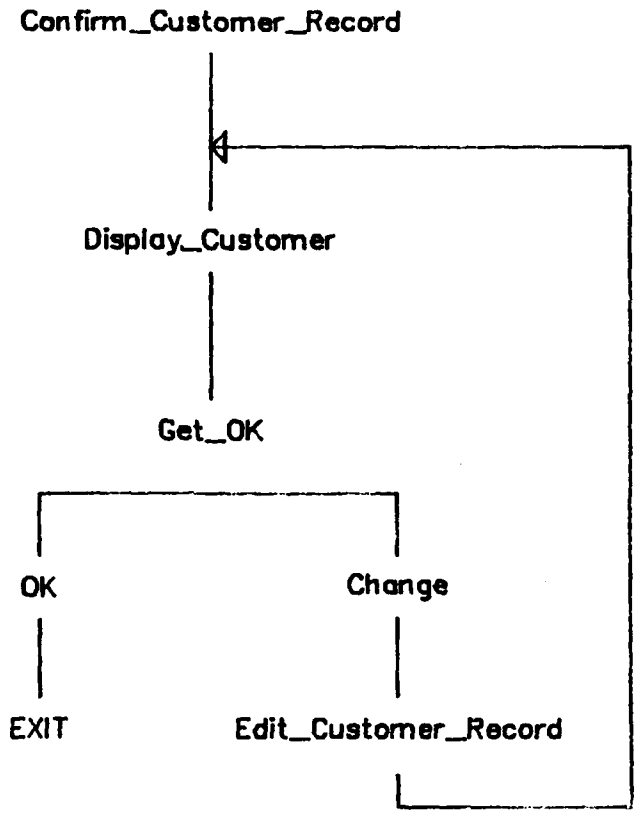


Figure B4. Customer Record Editing Process

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Dr. Neal D. Glassman, Code 411 Office of Naval Research, 800 North Quincy Street Arlington, Virginia 22217-5000	1
2.	Defense Contract Administration Services Management Area/Twin Cities 2305 Ford Parkway St. Paul, Minnesota 55116-1893	1
3.	Director, Naval Research Laboratory, ATTN: Code 2627 4555 Overlook Avenue SW Washington, DC 20032	1
4.	Defense Technical Information Center, Bldg. 5, Cameron Station Alexandria, Virginia 22314	12

END

DTIC

6-86