A Decision Support System For Integrated Design Analysis

Of A Repairable Item And Its Logistic Support System

by

Roderick J. Reasor, P.E.

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Industrial Engineering and Operations Research

APPROVED:

Marvin H. Agee, Ph.D., Co-chairman

Timothy J. Greene, Ph.D., Co-chairman

Wolter J. Fabrycky, Ph.D.

Loren P. Rees, Ph.D.

Subhash C. Sarin, Ph.D.

April 24, 1990

Blacksburg, Virginia

A Decision Support System For Integrated Design Analysis

Of A Repairable Item And Its Logistic Support System

by

Roderick J. Reasor, P.E.

Marvin H. Agee, Ph.D., Co-chairman

Timothy J. Greene, Ph.D., Co-chairman

Industrial Engineering and Operations Research

(ABSTRACT)

Design of a repairable item and its logistic support system requires consideration of several inter-related decision problems. These decision problems concern the variables, controllable by the design engineer and/or system manager, which affect system performance. This research develops a framework for integration of these decision problems and evaluation of system design tradeoffs. These design decision problems are represented in the model base of a decision support system (DSS). Interrelationships between decision problems are defined using data flow diagrams. Data flows within and between these decision problems are integrated in the DSS database. A simulation capability, imbedded into the DSS permits short-term, accelerated time excursions into possible futures for decision-making purposes. Alternative system designs are evaluated using a multicriteria decision model which considers reliability, maintainability, availability, and life cycle costs.

The logistic support system is modeled as a multilevel inventory system. These inventories include spare repairable items, spare parts, labor, maintenance equipment, and other support resources. Repairable item and logistic support system design decision problems affect the quantity and location of these inventories. Five decision problems identified by Moore [1986] were selected to demonstate the utility of this framework. The selected decision problems are: 1) the equipment design problem; 2) the maintenance configuration problem; 3) the spare equipment problem; 4) the level of repair problem; and 5) the replacement policy problem.

The framework developed supports integration of these decision problems throughout the item's life cycle. A repairable item can be systematically divided into subelements until individual repairable components are identified. This systematic subdivision of the item produces an inverted, tree-like structure. This structure is used as the representational view of the DSS database. As the life cycle progresses and the item design becomes more detailed, the structure expands. The DSS database is designed to accommodate this expansion so that the framework can be used throughout the item's life cycle. The initial fielding and the retirement of the repairable item population produces nonstationary demands on the logistics support system. A multistream model captures the nonstationary aspects of demand, eliminating the need for item-by-item tracking within the model.

The framework developed is illustrated using a comprehensive case study. The case study addresses the design of a Side Loadable Warping Tug (SLWT) and its logistics support system. A population of SLWT's must be deployed to meet demands in two different operating environments. The SLWT is a component of the U.S. Navy's Container Offloading and Transfer System (COTS).

Acknowledgements

Completing this dissertation while employed full time has been one of the most difficult and frustrating experiences of my life. It would have been impossible to complete without the support, encouragement, and guidance of many people. I would like to thank each and every one of them from the bottom of my heart. This document is too short to thank everyone, but a number of people deserve special recognition.

I would first like to thank Dr. Marvin H. Agee and Dr. Timothy J. Greene for their extreme patience and unending encouragement. Their guidance was crucial to the development of this work. I would also like to especially thank Dr. Wolter J. Fabrycky for his encouragement and technical contribution to this work. The advice, support, and suggestions received from Dr. Loren P. Rees and Dr. Subhash C. Sarin were also very helpful.

During my tenure at Virginia Tech, the faculty, staff, and students in the IEOR Department have been very supportive. I especially appreciate Dr. Robert D. Dryden who, as department head, has always been supportive and provided me the resources needed to be successful. Resources were also provided by the U.S. Navy, Naval Electronics Systems Command who provided partial funding for this research on contract N00039-84-C-0346. I appreciate their support and the contribution of the contract technical monitor, Dr. Franz A. P. Frisch.

Last, but not least, I would like to thank my family. Without their support, encouragement, sacrifice, and love, I could not have made it through this process. I would like to especially thank my lovely wife Anita for helping me find the time to complete this research. I also want to thank my children Rebecca, Matthew, and Christopher for tolerating my frequent absences. My parents, Mr. and Mrs. Emmet J. Reasor, have always stressed the importance of education. They have always encouraged me and sought to help in any way possible. I would like to dedicate this work to them in recognition and appreciation of their efforts to provide their children every opportunity to succeed.

Table of Contents

List of Illustrations

List of Tables

1.0 Introduction

The design and/or operation of multilevel inventory systems is one area of management endeavor that offers substantial opportunity for large payoffs. The opportunity for improvement (cost reduction) by U.S. industry alone has been estimated to exceed $40 billion [Kearney, 1984]. Approximately one-third of the assets of the average U.S. business is devoted to inventory [Schwarz, 1981]. The costs to transport, warehouse, inventory, and manage these goods totalled more than $420 billion in 1983 [Kearney, 1984]. These costs do not include the enormous inventories held by local, state, and federal government agencies. The U.S. Air Force alone has estimated spare parts inventories valued in excess of $12 billion [Demmy and Presutti, 1981]. Most of these inventories, both government and industry, are managed within a multilevel inventory system [Schwarz, 1981].

A multilevel (or multiechelon) inventory system contains two or more interrelated activities. An activity is generally defined as an entity that maintains a physical stock of one or more items in order to meet specified demands [Schwarz, 1981]. The entity may contain some production or repair capabilities as well as supply [Clark, 1972]. Example activities include warehouses in a distribution network, production sites, repair facilities, and the vehicles and/or containers used to transport materials between other activities.

Repairable items account for a considerable portion of the money invested in multilevel systems [Nahmias, 1981]. The inventories in a repairable-item system includes spare repairable items, spare parts, labor, maintenance equipment, and other support resources. In addition, the repairable item portion of total inventory investment is growing. For example, 52% of the U.S. Air Force investment in spare parts inventories in 1968 was for repairable items [Sherbrooke, 1968]. By 1975, that percentage had risen to about 65% [Muckstadt, 1979]. Subsequently, that percentage jumped to about 83% [Demmy and Presutti, 1981].

This research addressed integrated design decision problem analysis of a repairable item and its logistic support system. Logistic support is viewed as the composite of all considerations necessary to ensure the effective and economical support of an item throughout its programmed life cycle. The real frontier for research on multilevel inventory systems is in this area of integrated logistics [Clark, 1972]. The interrelationships between item design, acquisition, deployment, operation, maintenance, transportation, and supply need to be considered in the design of the multilevel inventory system. Moore [1986] identified a hierarchy of decision problems which must be addressed during the design of a repairable item and its logistic support system. The ten major decision problems which affect system performance are:

1. **Equipment Design Problem (EDP)** - determine the repairable equipment design, specifications, and/or procurement source.
2. **Maintenance Configuration Problem (MCP)** - determine the configuration of designated maintenance facilities.
3. **Optimal Level of Repair (OLP)** - determine the optimum level and/or location at which each type of equipment failure will be repaired.
4. **Spare Equipment Problem (SEP)** - determine the number of units of repairable, prime mission equipment to be procured to meet mission requirements and the associated procurement profile.
5. **Replacement Policy Problem (RPP)** - determine the conditions under which repairable equipment will be replaced.

6. **Preventive Maintenance Policy (PMP)** - determine the type and frequency of any preventive maintenance to be performed on the repairable equipment.
7. **Inspection and Testing Policy (ITP)** - determine the type and frequency of any inspection and testing to be performed.
8. **Repair Facility Design Problem (RFD)** - determine the repair facility workstation layout; design tools, repair stands, test and diagnosis sets, part storage racks, etc. which may affect repair time distributions.
9. **Mechanic Training Problem (MTP)** - determine the type of training mechanics will receive and its resulting impact on repair time distributions.
10. **Operator Training Problem (OTP)** - determine the level of training required for repairable equipment operators and its effect on equipment design.

These decision problems are interrelated and can not be effectively addressed on an individual basis. They must be integrated to ensure the effective and economical design and support of the repairable item.

This research presents a decision support system approach to integrating the analysis of the design decision problems identified by Moore for a repairable item and its logistic support system. This approach is used because the inherent complexity of the system prohibits formulation of the problem for solution using analytic optimization techniques [Clark, 1972]. A decision support system is an interactive computer-based system that helps decision makers utilize data and models to solve unstructured problems [Sprague and Carlson, 1982]. Using a decision support system, a decision maker can investigate plausible designs for the repairable item and its logistics support system. A simulation capability, imbedded into the decision support system permits short-term, accelerated-time excursions into possible futures for decision-making purposes. Using feedback provided by the decision support system, the plausible design alternatives can be refined and the best system design identified. The decision support system can contain either evaluation-selection procedures for postulated alternatives or adaptive decision-making models [Clark, 1972]. Such an

analytic approach may be more successful than one which tries to simplify the interconnections in the mathematical structure to permit solution using optimization algorithms [Wagner, 1979].

This research emphasizes the life-cycle approach to logistics. This approach allows establishment of logistics requirements early in the life cycle and in the subsequent design activities that precede operation, distribution, and sustaining support during consumer use. The life-cycle approach to logistics is particularly important for repairable-item systems. U.S. Air Force data show that about 80% of the life cycle logistics costs of an aircraft is determined by decisions made before the aircraft begins to fly [Geisler and Murrie, 1981]. These decisions relate to design tradeoffs of performance against reliability and maintainability, with resulting impact on maintenance manning, initial spare parts procurement, location of repair, and ground support equipment investment. Therefore, it is particularly important to consider logistics requirements for repairable-item systems early in the system life cycle.

1.1 Logistics in the System Life Cycle

A system can be defined as a nucleus of elements structured in such a manner as to accomplish a function to satisfy an identified need [Blanchard, 1986]. The elements of a system include all equipment, related facilities, material, software, data, services, and personnel required for its operation and support. The life cycle of a system or product begins with the initial identification of a need and extends through design and development, production and/or construction, use and support, and finally system retirement.

Logistics relates to the support of a system. Blanchard [1986] identifies seven phases in the system life cycle and describes the major facets of logistics related to each of these phases. These seven phases and their logistics components are:

1. **Identification of Need** - The want or desire for systems or products due to deficiencies or problems is made evident by consumer(s).

2. **Advance Planning and Conceptual Design** - The system operational requirements are determined and the system maintenance concept defined. Alternative designs are identified and rank ordered. System operational requirements include:

   - mission definition - identification of what the system is to accomplish and how?
   - performance and physical characteristics - definition of the operating characteristics or functions of the system.
   - operational deployment - identification of the quantity of systems and the expected geographical location.
   - operational life cycle - anticipated time that the system will be in operational use.
   - utilization requirements - anticipated usage of the system and its elements.
   - effectiveness factors - system requirements specified as figures of merit (cost, readiness, availability, supportability, etc.).
   - environment - definition of the environment(s) in which the system is expected to operate.

   The maintenance concept delineates maintenance support levels and repair policies (the extent to which repair of an item will be accomplished). The operational requirements and the maintenance concept are the primary determinants of logistic support resources.

3. **Advance Development and Preliminary System Design** - The major system functions are identified and the system performance measures which have been established are allocated among the various subsystems, units, and assemblies. This process has significant impact on the logistics system as it directly affects system/equipment design. For example, the allocation of system reliability to various subelements of the system determines the expected number and type of maintenance actions the logistics system must support. This impacts spare parts inventories, facilities requirements, labor requirements and other logistic support factors. A

preliminary logistic support analysis determines the logistic support requirements for alternative system designs or configurations. The preferred alternative is then selected.

4. **Detail Design and Development** - During detail design and development, lower-level assemblies and components and associated logistic support resources for the preferred alternative are defined in greater detail. Provisioning and acquisition of logistic support elements for production operations and for consumer life-cycle support is initiated.

5. **Production and/or Construction** - During production and/or construction of the system, requirements are established for material flow, procurement and inventory, packaging, warehousing, physical distribution, transportation and traffic management, communications, data processing, customer service, and logistics management. Provisioning and acquisition of logistic support elements for consumer and for system life-cycle support is performed. Logistic support capability is assessed.

6. **System Use and Life-Cycle Support** - System operation in the field includes sustaining maintenance and logistic support (e.g., supply support, test and support equipment, personnel and training, facilities, transportation and handling, data processing, customer service, and logistics management). Logistic support capability is assessed. Data collection, analysis, and feedback for corrective action is developed.

7. **System Retirement** - System retirement involves material phase-out and recycling and/or disposal, and logistic support requirements.

Therefore, logistics involves planning, designing, analyzing, testing, producing, distributing, and the sustaining support of a system or product throughout the consumer use period. These are an inherent part of the life cycle and must be dealt with on an integrated basis.

1.2 Statement of the Problem

The problem facing logistics managers is how to analyze the multifaceted nature of life-cycle logistics requirements in an integrated manner. Logistics costs associated with inventories, warehousing, transportation, repair facilities, and personnel are interrelated. They are affected by repairable-item design trade-offs for reliability, maintainability, and availability. Logistics costs are further affected by operational considerations like operating environment, level of repair decisions, and procurement or mode-of-fielding strategies. The body of research applicable to the individual design problems associated with a repairable item and its logistic support system is extensive. However, the portion that takes a global, systems view which considers the interrelated nature of decision problems over the system life cycle is very limited. Some of the reasons for this limited treatment are the:

1. complexity of a repairable item and its logistics support system,
2. absence of the relatively inexpensive computing capability needed to address the logistics system from a global, systems perspective, and
3. application of systems engineering concepts has only recently been embraced by the logistics engineering community.

Integration of the design problems associated with a repairable item and its logistics support system requires development of a framework for integration. The framework provides a basic structure for problem definition. Imbedded in the framework is a set of tools to address individual design problems, and a methodology for design analysis. A tool is something used to perform an operation (e.g., to determine the average inventory level). A methodology is a body of methods, rules, and postulates. The methodology specifies how the tools will be used to analyze alternative system designs. Past research has focused on tool development. Therefore, a framework for integration which uses existing tools to analyze alternative system designs is needed.

The framework must support design analysis at each phase of the life cycle. Logistic support requirements are frequently updated as the repairable item progresses through its life cycle. During the early phases of the life cycle, the logistics manager is working with conceptual designs where logistics can only be considered on a more aggregated or macroscopic level. As the system design progresses through the life cycle, the design becomes better defined and logistic support requirements can also be described in greater detail. The framework must be a flexible, easily expandable structure to allow definition of system parameters based upon data availability. These system parameters include repairable-item parameters (e.g., Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR)) and logistics support parameters (e.g., the number of maintenance and repair levels). As the system design evolves, the analysis of alternative designs within such a framework will minimize duplication of previous efforts.

As a system is placed into service, there are many modes of fielding the population of end items. The population of end items will not exhibit steady state failure characteristics during start-up as items are placed in service and during retirement as items are removed from service. Also, transient (nonsteady state) conditions exist as the number of units assigned to different operating environments varies during the system operational phase. These transient conditions can have a significant impact on logistics requirements and should be considered within the framework.

Many measures of performance have been recognized to evaluate system effectiveness. These include reliability, maintainability, availability, and life cycle costs. Life cycle costs, when considered by themselves, are an inadequate measure of system effectiveness. The methodology used to analyze alternative designs must contain a tool which will allow the simultaneous consideration of these many factors and allow the weighting of those identified as most important to a particular system design.

1.3 Research Objectives

The primary objective of this research is to develop a framework for integrating the analysis of design decision problems for a repairable item and its logistic support system. Solution of the described problem requires simultaneous consideration of all the design decisions identified by Moore. These decisions concern equipment design, repair policy, maintenance support levels, failure profiles, level of repair, number of repair channels, spare parts policy, retirement age, and procurement policy. A framework for analysis of logistic support requirements throughout the system life cycle will ensure the effective and efficient provision and management of logistic resources. Specifically, this research:

1. develops a framework in which interrelated system design decisions can be analyzed in an integrated manner.
2. defines the interrelationships which exist between system design decisions.
3. models the multi-indenture relationship between a repairable item and its component parts.
4. models the three levels of maintenance and repair commonly used in repairable-item logistic support systems.
5. models the nonstationary nature of maintenance and repair facility demands during the operational and retirement phases of the life cycle.
6. captures the stochastic nature of system operation and performance.
7. considers several measures of system performance and simultaneously determines their collective impact on system effectiveness.
8. demonstrates the application of the framework developed.

Accomplishment of these objectives is discussed in detail in Chapter 3, **Approach to the Problem.** However, to briefly summarize, this research demonstrates how a DSS can be used to provide a framework in which system design decisions can be integrated. The individual decision problems

are imbedded in the DSS model base and integrated using the DSS data base. The interrelationships between these system design decisions are documented using data flow diagramming techniques. A maintenance tree is used to systematically subdivide a repairable item into lower level components. The maintenance tree models the multi-indenture relationship between a repairable item and its component parts. The logistic support system is modeled as a multilevel inventory system. A simulation capability imbedded in the DSS permits more complex multilevel inventory systems (three or more levels of maintenance and repair) to be modeled. In addition, a simulation model supports specification of probabilistic parameters thereby capturing the stochastic nature of system operation and performance. The nonstationary nature of maintenance and repair facility demands during the operational and retirement phases of the life cycle is modeled by representing the population of repairable items as a family of nested Markov chains. A multicriteria decision model is used to combine different aspects of system performance into a single measure of system effectiveness. Finally, the methodology developed is illustrated using a case study from the U.S. Navy.

1.4 System Description

There are six distinguishing features which are commonly used to characterize multilevel inventory systems [Clark, 1972]. These features can be expressed using the following dichotomies:

1. single-product versus multiproduct systems,
2. consumable versus repairable products,
3. deterministic versus stochastic demands,
4. stationary versus nonstationary demand parameters,
5. backlogging versus no-backlogging for unsatisfied demands, and
6. continuous versus periodic inventory review.

Using these dichotomies, the multilevel inventory system considered by this research is for a single, repairable product. The activities within the system experience stochastic demands. These demands are nonstationary and unsatisfied demands are backlogged. The inventory system operates under periodic review.

The multilevel inventory system considered in this research has an arborescent structure. This structure provides modeling flexibility since it can also be used to model less general structures by imposing additional structural restrictions. For example, a serial structure is an arborescent structure with the additional restriction that each node have at most one predecessor node. The activities in the multilevel inventory system are facilities where spare parts inventories and other logistic support resources like personnel, maintenance equipment, and warehouse space are available to accomplish end item maintenance and repair. At each activity there is also an inventory of end items from which failed end items are replaced while they undergo repair.

The multilevel, repairable-item inventory models reported in the literature generally consider a maximum of two levels [Nahmias, 1981]. The multilevel inventory system considered in this research has three levels, as shown in Figure 1 on page 12. These three levels represent the three levels of maintenance described by Blanchard [1986]. Organizational maintenance is performed at the lowest level (level 3), at the operational site (e.g., on the airplane, vehicle, or other operational facility). Intermediate maintenance is performed at level 2, a base or regional facility. The depot level, level 1, constitutes the highest type of maintenance. It supports the accomplishment of tasks above and beyond the capabilities available at the intermediate level.

An end item is defined to be an individual repairable item (e.g., a vehicle, a machine, a computer, an aircraft, etc.). End items are designed to accomplish some mission, meet some need, or provide some service. These end items are assigned to various operating environments in which they perform as functionally designed. An operating environment is a unique set of circumstances or conditions in which an end item is deployed. For example, a lift truck may be deployed in a variety of operating environments. Two such operating environments are a warehouse and a production

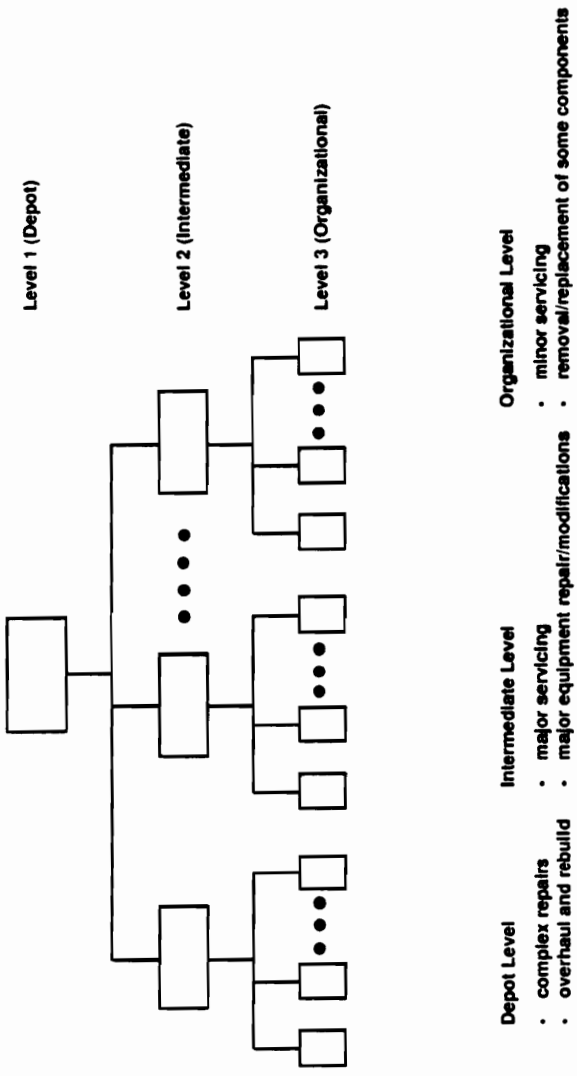


Figure 1. Maintenance and Repair Levels Modeled

department. Climate differences (temperature, humidity, etc.), travel distances, stack heights, unit load weights, operator training and proficiency, frequency of use, etc. are all conditions which may make each of these operating environments unique. However, for modeling purposes, these environments will be treated as unique operating environments only if these differences have some effect on the end item failure and/or repair characteristics. Each operating environment is assumed to have some known, deterministic demand for end items over a given planning horizon. The population of end items is the sum of the end items assigned to all operating environments plus the spare end items inventoried at the various maintenance and repair facilities.

It is assumed the end item can be systematically divided into subelements until individual repairable components are identified. For example, an automobile can be subdivided into an electrical subsystem, a transmission subsystem, a cooling subsystem, etc.. The electrical subsystem can be further subdivided into individual repairable components like the alternator, the ignition, etc.. Each subdivision of the end item represents a level of indenture. At some level of indenture, it is not desirable and/or feasible to subdivide the item any further. At this point, failure and repair time distributions are specified. In addition, a stockable parts list, labor requirements, and other logistics support resources needed to repair the component upon failure must be identified. The failure of individual repairable components produces stochastic demands for spare parts, labor, equipment, and other logistic support resources at the maintenance and repair facilities.

1.5 Dissertation Overview

Chapter 1.0 has presented an overview of this research. The significance of the problem was established, the problem was defined, the research objectives were stated, and the system addressed by this research was described. Chapter 2.0 presents a review of the modeling approaches reported in the literature which have influenced and/or guided this research. The decision problems which

must be addressed in the design of a repairable item and its logistics support system are defined. The literature review continues with a discussion of planning and evaluation models which are targeted more at systems design and evaluation. The literature review concludes with a short tutorial on decision support systems.

Chapter 3.0 is a detailed description of the decision support system (DSS) approach used to accomplish the objectives of this research. Each of the design decision problems included in the DSS is discussed and the General Decision Evaluation Function [Fabrycky, 1989 and Blanchard and Fabrycky, 1990] values for each are defined. The modeling constructs employed in this research are discussed in detail and equations for all associated calculations are developed. The multicriteria measure of system effectiveness is established and its optimization is discussed.

The implementation of the DSS is described in Chapter 4.0 and Appendix C. The DSS developed by this research is named CODAS, for Concurrent Design Analysis System. Appendix C describes the 46 interactive input/output screens contained in CODAS, their function, record layouts, a program listing, and other details of the DSS operation. Most of Chapter 4.0 is a case study which demonstrates the application of the developed framework. Appendix A and Appendix B contain data used by the case study. The final chapter, Chapter 5.0, provides a critique of this research. Strengths and weaknesses of the approach are identified. These naturally lead to identification of opportunities for future research.

2.0 Literature Review

The body of research applicable to the analysis of repairable equipment logistics systems is extensive. A wide variety of modelling techniques have been used to address individual logistics decision problems. However, research which takes a global, systems view, which considers the interrelated nature of logistics decisions over the systems life cycle is extremely limited. The following is a review of the modelling approaches reported in the literature which have influenced and/or guided this research.

This review begins by defining the decision problems which must be addressed in the design of a repairable item and its logistics support system. Moore [1986] presents a hierarchy of decision problems which must be considered, selects four of these to model, and develops two optimization methods. Next the various mathematical models for determining stocking levels for repairable item inventory systems are reviewed. These models include continuous review models, periodic review models, and models based on cyclic queueing systems. These mathematical models are directed more towards managing the everyday flow of material in a multiactivity inventory system.

The literature review continues with a discussion of planning and evaluation models which are targeted more at system design and evaluation. This group of models includes some of the more comprehensive logistics design and evaluation models reported in the literature. The literature re-

view concludes with a short tutorial on decision support systems. An important aspect of decision support systems is the ability to integrate data access and decision models. By embedding the decision models in an information system, the data base can be used as the integration and communication mechanism between models.

2.1 A Classification Hierarchy

Before a model which considers the interrelated nature of logistics decisions can be developed, it is necessary to define the multitude of design decisions the logistics manager must address. Moore [1986] presents a classification hierarchy of decision problems for MREAL (Multiple Repairable Equipment And Logistic) systems. This hierarchy presents a representation of the many decision problems associated with MREAL systems and suggests a holistic structure to the analysis of MREAL systems. It encompasses, as subproblems, much of the existing logistics modelling work, and many of the functional areas of logistics. The classification hierarchy is shown in Figure 2 on page 17. It consists of five levels of systems and problems which the logistics manager must address. The first level of this hierarchy deals with the optimal allocation of resources for collections of REAL (Repairable Equipment And Logistics) systems. The second level consists of the individual REAL systems, each representing a different major end item of repairable equipment.

At the third level of the hierarchy, Moore identifies ten major elements which can affect system performance and which cannot be modelled separately from the rest of the system. A holistic model is an important tool for the allocation of resources between such elements as spare end items, maintenance channels, physical design, and operator training. The ten elements identified by Moore for consideration during logistics planning are:

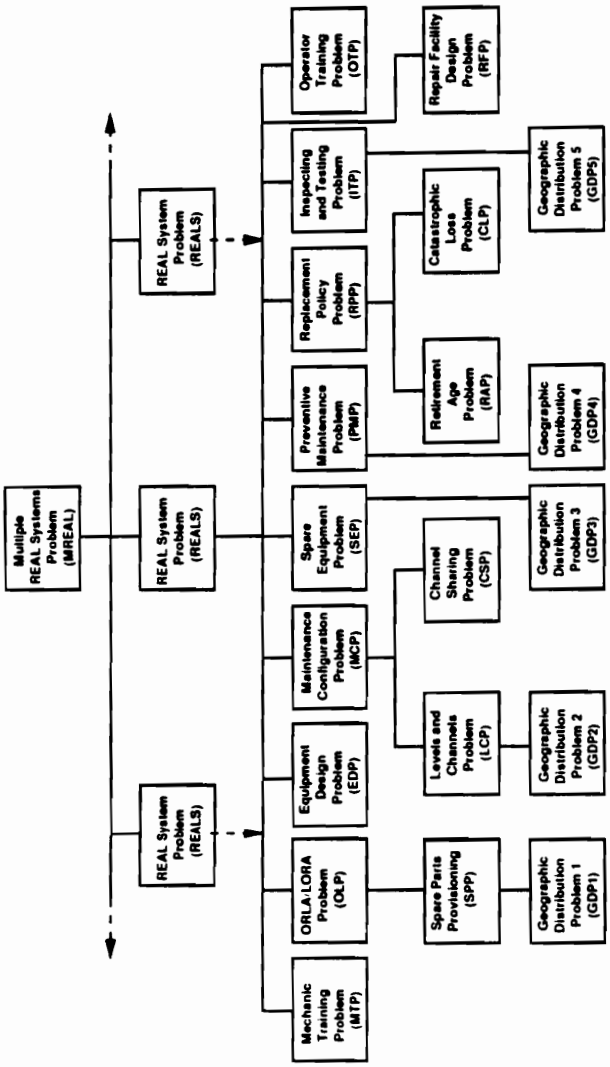


Figure 2. Moore's Hierarchy of Logistics Decision Problems

1. **Mechanic Training Problem (MTP)** - determine the type of training mechanics will receive and its resulting impact on repair time distributions.

2. **Optimal Level of Repair (OLP)** - determine the optimum level and/or location at which each type of equipment failure will be repaired.

3. **Equipment Design Problem (EDP)** - determine the repairable equipment design, specifications, and/or procurement source.

4. **Maintenance Configuration Problem (MCP)** - determine the configuration of designated maintenance facilities.

5. **Spare Equipment Problem (SEP)** - determine the number of units of repairable, prime mission equipment to be procured to meet mission requirements and the associated procurement profile.

6. **Preventive Maintenance Policy (PMP)** - determine the type and frequency of any preventive maintenance to be performed on the repairable equipment.

7. **Replacement Policy Problem (RPP)** - determine the conditions under which repairable equipment will be replaced.

8. **Inspection and Testing Policy (ITP)** - determine the type and frequency of any inspection and testing to be performed.

9. **Repair Facility Design Problem (RFD)** - determine the repair facility workstation layout; design tools, repair stands, test and diagnosis sets, part storage racks, etc. which may affect repair time distributions.

10. **Operator Training Problem (OTP)** - determine the level of training required for repairable equipment operators and its effect on equipment design.

Level four in the hierarchy contains logistics system design issues which are subelements of those identified in level three. These level four issues are either problems whose solutions depend upon the policies established for their respective level three element, or they are significant problems which must be resolved before the policies for their respective level three element can be established. Moore identifies five, level four subelements. These are:

- **Spare Parts Provisioning (SPP)** - determine inventory levels, reorder rules, replenishment quantities, and the storage locations for each spare part.

- **Levels and Channels Problem (LCP)** - determine the number of levels of maintenance, the number of repair channels to operate at each level, the number of mechanics to assign to each repair channel, and the queueing and service disciplines to use at each repair channel.

- **Channel Sharing Problem (CSP)** - determine the extent of conditions under which equipment from each REAL system will be maintained at the repair channels of the other REAL systems.

- **Retirement Age Problem (RAP)** - determine the point in time that equipment will be retired from service.

- **Catastropic Loss Problem (CLP)** - determine which types of catastropic damage will lead to repair of the equipment and which will lead to replacement.

The fifth level of the hierarchy contains the elements of the REAL system which deal with location problems. Moore defines five location problems, called Geographic Distribution Problems (GDP1 through GDP5) which address the geographic distribution or location of spare or repair parts (GDP1), repair channels (GDP2), the repairable equipment itself, especially the spare units

(GDP3), where each type of preventive maintenance will be performed (GDP4), and where inspection and testing will be performed (GDP5).

Moore's hierarchy is extremely valuable as a conceptual aid in the identification of the policies and trade-offs which affect the cost and performance of repairable equipment logistics systems. It should stimulate the logistics manager to generate "what if" type questions which lead to a better understanding of the system interractions. It also provides a convenient mechanism to remind the logistics manager of the interrelationships that exist between the many logistics system design decisions. The classification leads to the visualization of a number of possible holistic models of repairable equipment logistics systems. It also shows that a single, truly holistic model, one that encompasses all of the problems and subproblems, would be extremely complex. The current practice for dealing with repairable equipment logistic systems in most businesses and organizations is to separate the various components of the system and attempt to optimize each by itself.

Recognizing that existing models do not come close to encompassing all of the components of a repairable equipment logistics system, Moore selected four elements and subelements of REAL systems to model. He calls his model MREAL1 to indicate that it is a first effort at holistically modelling a MREAL system. Included in MREAL1 are representations of the equipment reliability and maintainability design problem, the maintenance capacity problem, the retirement age problem, and the population size problem, for each of the multiple populations. MREAL1 models the steady state stochastic behavior of the equipment repair facilities using an approximation based upon the finite source, multiple-server queueing system. System performance measures included in the objective function are system life cycle cost, the steady state number of shortages, the probability of catastrophic failure in the equipment, and two budget-based measures of effectiveness. Two optimization methods for this problem are presented. One optimization method involves computation of the objective function and the constraints for a specified subset of the solution space. The second optimization procedure utilizes a sequential, unconstrained minimization technique based upon an augmented Lagrangian penalty function adapted to the integer nature of the MREAL1 problem.

While recognizing the importance of models, Moore concludes that models do not make decisions. They can only provide a human system manager with insights which can be used to aid decision making. However, most decision makers are not willing to learn the intricacies of building and optimizing models. Therefore, they need a decision support system which is very user friendly. He states the MREAL1 model is far from being user friendly or a complete DSS.

2.2 Repairable-Item Inventory Models

The management and design of repairable equipment logistics systems present problems which parallel the management and design of multiechelon inventory systems. Moore [1986] states that "the steady state modelling of the logistics of repairable equipment may be ultimately recognized as a second branch of inventory theory." Repairable equipment logically represents an extension of classical inventory theory, in which the assumption that an inventory item may satisfy a demand exactly one time is relaxed. Repairable items are held to satisfy customer demands repeatedly for a certain period of time. Nahmias [1981] published a comprehensive review of the various mathematical models that have appeared in the literature for determining stocking levels for repairable item inventory systems. Nahmias classified existing models into three general classes: continuous review, periodic review, and models based on cyclic queueing systems.

2.2.1 Continuous Review Models

Most of the continuous review models employ one-for-one (S-1,S) ordering policies [Nahmias, 1981]. Many of these models describe two-echelon systems where items fail at the base level (the 2nd echelon). The item may be repaired at the base level or, if the repair is too complex, at the depot level (echelon 1). The base replaces the failed item from base-level stock (if it is available)

and immediately places an order for a replenishment from the depot. The inventory position at the base is defined as the total number of units on hand, plus units due in from base and depot repair, minus backorders. The base maintains its inventory position at a fixed level S. According to the (S-1, S) policy, every time one or more units is demanded, the inventory position drops below S and an order for an equal number of units is placed.

One of the earliest works in this area was by Scarf [1958] who identified the analogy between the two-echelon system described in the previous paragraph and the theory of infinite server queues. The base-level orders for replacement units from depot stocks can be viewed as customers entering a queueing system with an infinite number of servers. The number of busy servers corresponds to the number of outstanding orders (units in repair). Palm's theorem [1938] states that if customers arrive according to a stationary Poisson process and service times are independent, identically distributed random variables with a finite mean, then the steady-state probability distribution of the number of busy servers is Poisson, independent of the form of the service distribution. In the inventory problem, this implies that if customer demands are generated by a stationary Poisson process, the number of outstanding orders (units in repair) is a random variable having a Poisson distribution, regardless of the distribution of lead times (resupply times for units sent to repair) as long as lead times are independent. That is,

$$\begin{aligned} P\{\text{x orders outstanding}\} &= P\{\text{net inventory} = S - x\} \\ &= e^{-\lambda\tau}\frac{(\lambda\tau)^x}{x} \qquad , x = 0,1,2,..., \end{aligned}$$

where λ = mean number of units demanded per unit time and τ = expected lead time. This result is based on the assumption that excess demand is backordered. A similar result holds in the case of lost sales, except that the Poisson distribution is truncated at S.

An important generalization of Palm's theorem was obtained by Feeney and Sherbrooke [1966] who showed that the distribution of outstanding orders is still Poisson when demands are generated by a stationary compound Poisson process. That is, the times of demand occurrences still follows

a Poisson process with rate λ, but at each demand occurrence the number of units demanded is a random variable with an arbitrary discrete distribution {0, 1, 2,...}. This result is only true, however, when each of the units in a demand is repaired at the same time (i.e., infinite repair channel capacity exists).

More traditional type inventory models involving steady-state cost minimization have been developed by Galliher, et al. [1959] and Hadley and Whitin [1963]. However, in the context of a repairable item system, it is more common to use one of two service level criteria rather than cost minimization. These two criteria are fill rate and the number of backorders. The fill rate is $1 - P_{out}$, where P_{out} is the probability of being out of stock at a random point in time. However, a number of studies of (S-1,S) policies which use these service level criteria have not been incorporated into a multiechelon model. These include Gross and Harris [1971], Rose [1972], Das [1977], Simon [1971], Higa et al [1975] and Kruse [1977].

An important class of continuous review models is based on the METRIC model developed over a period of years by a research group at the Rand Corporation as a tool for managing Air Force inventories [Sherbrooke, 1968]. METRIC is an acronym for Multi-Echelon Technique for Recoverable Item Control. It is important in that it appears to capture many of the significant features of the problem of determining suitable spares levels in a large scale repairable item inventory system and thus is one of the few multiechelon models to be implemented.

The METRIC model provides a methodology for computing optimal stock levels in a two echelon inventory and repair system consisting of a depot and several bases. Each location in the system posseses a supply of serviceable spares and a capability for returning unserviceable assets to a serviceable condition. Demand for spare part i at each base j is assumed to follow a compound Poisson process with known mean rate λ_{ij} . Item i can be repaired at base j with probability r_{ij}, while $(1 - r_{ij})$ is the probability it must be shipped to the depot for repair. Each base and depot use a continuous review, one-for-one (S-1, S) inventory policy. The model determines the stock levels

at each base such that the expected number of backorders is minimized subject to a constraint on the total dollar investment available.

The METRIC model assumes the expected repair times and replenishment lead times are known constants. All items can be repaired and there is no lateral resupply among bases. Successive repair times are assumed to be independent, identically distributed random variables. This last assumption essentially states that there is an infinite number of servers at the repair channels. The expression for the expected number of backorders is developed using Feeney and Sherbrooke's [1966] extension of Palm's theorem. The minimization is performed using a two-phase, marginal allocation algorithm developed by Sherbrooke. Demmy and Presutti [1981] report this model has proven useful in three basic areas. These are:

1. evaluation of the support effectiveness associated with a given set of stock levels,
2. distribution of a given number of assets among several locations, and
3. calculation of optimal stock levels consistent with a given investment constraint.

The METRIC model fails to recognize the indenture relationship between end items and their subassemblies or other components which are also repairable. Several authors, including Sherbrooke [1971], Demmy [1970], Fawcett [1967], and Porteous and Landsdowne [1974] have developed models which expand the basic METRIC model to consider these relationships. However, Demmy and Presutti [1981] report that Muckstadt's [1973] MOD-METRIC model has had the most influence on Air Force applications. MOD-METRIC extends the two-echelon METRIC model to include the multi-indenture relationship between end items and their subassemblies. The significant computational differences between METRIC and MOD-METRIC are two-fold. One is the manner in which the mean base repair time for the end item at a given base is computed. Second, the resulting optimization problem is considerably more complex due to incorporation of the multi-indenture aspects of the problem.

2.2.2 Periodic Review Models

Periodic review models assess inventories at discrete points in time with a given periodicity. The total demand in a period is a random variable with a specified distribution function, F(t), and density f(t). The methods of analysis and the results obtained are quite different than continuous review models. Most of the periodic review models are dynamic programming formulations that address a limited number of logistics decisions.

One of the first periodic review models for a repairable item inventory was developed by Phelps [1962]. The model is based on a two-dimensional dynamic program whose state variables at each stage represent: (1) x_s , the amount of serviceable stock (items repaired or procured from outside) and (2) x_r, the amount of repairable stock on hand at the start of the current period. The model is formulated to maximize total expected return where the return function allows for procurement, repair, and salvage. The decisions that must be made at each stage are: (1) the number of items to procure from outside, (2) the number of repairables to repair, and (3) the number of repairables and/or serviceables to scrap for salvage. Phelps analyzes both the single period problem and the infinite horizon problem and shows that the optimal policy can be specified by seven regions in the (x_s, x_r) plane. These regions are very difficult to establish and therefore it is unlikely this model will ever be implemented.

Veinott [1966] and Simpson [1978] considered a similar two-dimensional dynamic programming formulation. The most significant difference between their formulations and Phelps' formulation is that they assume there is a joint probability density, $f(t,u)$, of the random variables giving the number of serviceables demanded, t, and the number of repairables returned, u. Another difference is that Phelps' model assumes lost sales while Veinott and Simpson allow backordering of excess demand. Veinott and Simpson each developed different expressions for shortage and holding costs. Simpson's model addresses the optimal policy for an n-period planning horizon while Veinott formulates the infinite horizon problem. Simpson shows that the optimal policy for an n-period

problem has a structure similar to that obtained by Phelps. He identified seven regions which describe the optimal number of items to procure, repair, and scrap. The n-period solution is completely determined by three constants whose values depend on n. Veinott's formulation yields a relatively simple optimal policy. Veinott defines a one period cost function, $G(y,z)$, where the decision variables y and z are the total number of serviceables on hand after procurement and repair and the total number of both serviceables and repairables, respectively. Veinott shows that the period cost function G is minimized when $z = y = y^x$. The optimal policy is a function of x_s , the number of serviceable items. If $x_s < y^x$, then all repairables should be repaired and $y^x - x_t$ should be procured. Here, x_t is defined to be the number of serviceables plus repairables. If $x_s > y^x$, then one either repairs a portion of the available repairables (determined from a nonlinear function) or takes no action.

Prawda and Wright [1972] developed a model similar to Veinott's except they add considerations for procurement and repair lead times. Cohen, Nahmias, and Pierskalla [1980] consider a model which assumes inventory decisions are based on knowledge of on-hand inventory only; the number of units in repair is assumed unknown. Haber and Sitgreaves [1975] consider a model where items fail during some fixed time, T_1, and are then replaced by repair or external procurement in a subsequent period, T_2. Based on assumed costs of overage and shortage and probability distribution for failure and repairs, a simple Newsboy solution is developed to determine the optimum level of outside procurements.

2.2.3 Cyclic Queueing Models

Most queueing models reported in the literature address spares provisioning. A frequently encountered modeling approach is the machine repair model. Using this approach, a population of operational units (or machines) are supported by a given number of spares. When a failure occurs, the item enters a repair facility containing a number of parallel servers. The failed item is imme-

diately replaced by a spare if one is available. Failed items are generally repaired on a first come first served basis. State probabilities can be derived by analyzing the underlying birth and death process. Models using this basic approach include those reported by Gross and Harris [1971], Lureau [1974], Gross et al. [1977], and Fabrycky et al. [1984]. Gross and Ince [1978] extended the classical machine repair model to allow for more than a single stage in the repair phase. In their model, the queueing system consists of a number of stages in series.

These queueing models have two major advantages over the inventory models. The queueing models do not require the usual assumption that there is an infinite number of servers at the repair facility. In addition, the number of service channels can be treated as a decision variable. However, all of these models only address a single level of repair.

2.3 Planning and Evaluation Models

Inventory control models are directed towards managing the everyday flow of material in a multi-activity inventory system. Planning and evaluation models are targeted more at system design and evaluation problems than everyday management. Models have been developed to address planning and evaluation decisions like inventory positioning and level of repair.

The inventory positioning problem involves determining the particular activities at which inventories for each product should be carried such that total system costs are minimized, or some other system performance objective is satisfied. Brown, Silverman, and Perlman [1971] considered a parallel system of activities (aircraft carriers) supported by a forward supply point where all included activities carry all products of interest. This system is supported by a higher level system which is represented in terms of alternative resupply processes. The length of time required to resupply the lower system is dependent upon which combination of higher-level processes are incurred. This

length of time affects the amounts and costs of inventories required in the lower system. Since each higher level resupply activity has associated costs, the problem is one of selecting a combination of the resupply activities, from among those available, for which the sum of inventory costs and resupply process costs are minimized. A multistage solution process which draws upon dynamic programming and integer programming is applied to each product independently, ignoring possible economies of scale.

A study by Pincus [1971] deals directly with the inventory positioning problem where all activities in the multiactivity system represent different stockage facilities. In this study, a basic multiactivity structure is assumed, from which a substructure is to be selected for each product in order to minimize expected costs over all products and facilities. For each possible substructure, inventory costs will result for each product and each included activity will have various fixed and recurring costs. The problem is formulated as a 0-1 integer program and solved using a branch and bound algorithm. The solution yields a substructure for each product such that total inventory and facility costs are minimized.

An important problem is that of deciding where a repairable component is to be repaired. Solving this problem controls the mix of inventories at each activity in the system, as well as the mix of other resources such as repair equipment and personnel. One of the more general investigations of this problem is the RANGE model developed by Williams [1969]. The hierarchical structure of the end item(s), factors describing the multiactivity repair and supply system, reliability factors for parts, and various cost functions relating to the repair and supply operations under stationary assumptions are inputs to the model. The model uses a dynamic programming solution method to find combinations of repair actions to be taken which minimize overall support costs. As outputs, the model specifies whether or not a component should be repaired when it fails and, if repaired, which facility in the structure should do the repair. These outputs determine for each item the particular activities in the system that will experience demand because of the repair decisions and, hence, where inventories of the item should be established. The outputs also control the positioning of maintenance resources in an analogous fashion.

A variant of this problem is addressed more recently by the "OATMEAL" model developed by Kaplan and Orr [1985] for the U.S. Army. OATMEAL is an acronym for Optimum Allocation of Test Equipment and Manpower Evaluated Against Logistics. The model is designed to determine where a repair is to be performed and the quantity and placement of test equipment and special repair skills. The model considers four echelons of maintenance and supply where repairable items (weapon systems) may have three levels of indenture. However, failure rates are defined at the "module" level (level two) of the product indenture. Mixed integer programming combined with a Lagrangian approach is used to perform the constrained cost minimization. The objective is to minimize all costs dependent on the maintenance and stockage policies while achieving a target system operational availability.

Multi-activity models typically suffer from constraining assumptions concerning the environment within which the system operates. Simulation models have been developed to allow more robust representations of the environment for planning and evaluation purposes. For example, Haber [1971] developed a multiproduct simulation model for the operation of a particular three level, arborescence activity system (the Polaris submarine support system). The model was used to study alternative inventory policies. The simulation model was used to estimate expected costs stratified by the type of cost and type of policy, and to estimate various inventory performance statistics for each type of policy. From these results, judgements concerning the behavior and relative effectiveness of the alternative policies for various classes of products were made.

Another simulation model was developed by Geisler and Murrie [1981] to examine the operation of a single Air Force base. Their model, called the Logistics Composite Model (LCOM), simulates the interactions between maintenance, supply, and aircraft operations. LCOM simulates failure and repair activities on aircraft components. The lowest level of indenture handled by LCOM is components which can be removed by first level maintenance personnel. Information on a number of performance measures including equipment utilization, manpower utilization, maintenance activity, and mission performance (in terms of aircraft availability) are produced.

2.3.1 Frisch Multistream Model

Frisch [1983] represents a population of repairable items as a nested multiple stream model. Each stream represents a group of items that exhibit similar failure characteristics. The model is based on the generic mortality behavior of any population of end items. He uses the model to investigate how a population can be built-up and how it can be maintained within the total system's life cycle. In Frisch's model, each stream consists of similarly aged items. Frisch examines the deterioration of the population over discrete time periods to determine the impact of two variables, quality and mode of fielding, on logistics requirements. He defines quality, in terms of mortality, as "the deterioration of performance over time, measured against the design performance of the new product." Performance is defined as the output of a machine at its design point. Mode of fielding refers to the method by which the population of end items is procured and placed in service. The mode of fielding determines whether the population of end items will be placed in service all at one time (block procurement), at some uniform rate, at some increasing or decreasing rate, or at some other rate specified by the logistics manager.

Frisch reached a number of conclusions about transient and steady state logistics support requirements based upon a series of trial calculations representing a variety of combinations of mortality rates and assumptions about mode of fielding. He concluded that for block procured systems,

- the logistics support requirements will fluctuate, exhibiting peaks and valleys, until some time when support requirements stabilize,
- the span of fluctuations between peaks and valleys increases with increased quality of the system and decreases with decreased quality of the system, and
- the support requirements for high quality items stabilize very slowly at a low steady state level after many fluctuations. Low quality systems stabilize quickly but at a high steady state level.

He further concluded that if block procurement is not required, then a population of any desired quality can be fielded in such a manner that a steady state logistics support requirement exists from the very beginning. Also, support requirements decrease as the quality of the system increases.

Frisch suggests using the model to determine logistics support requirements during each discrete time period over some defined planning horizon. These logistics support requirements can then be costed to determine life cycle costs. He suggests coupling this model with a dynamic program to optimize period costs. The intent of the model is to serve as a "what-if" tool by which a logistics manager can try different combinations of the two variables (quality and mode of fielding) until an acceptable cost alternative is identified.

Brammer [1985] developed a microcomputer-based version of Frisch's multiple stream model. Brammer's model computes period by period maintenance requirements for a population of repairable items. The model estimates spare parts requirements, manpower requirements, warehouse space requirements, population availability, and period cost summaries for each period of a user defined planning horizon. The model supports four activity levels and allows failure and repair data to be defined for up to five levels of indenture.

2.4 Decision Support Systems

The concepts which embody a decision support system (DSS) were first articulated by Michael S. Scott Morton in the early 1970's under the term management decision system [Sprague and Carlson, 1982]. This forerunner to DSS was defined by Scott Morton [1971] to be an **interactive** computer-based system that **helps** decision makers utilize **data** and **models** to solve **unstructured** problems. The unique contribution of DSS resulted from the key highlighted words. Researchers and developers have struggled with this definition and the absolutes that it imposed.

In the late 1970's, Alter [1977], Keen [1978], and others began to describe the characteristics of decision support systems. The ***characteristics approach*** seems to hold more promise for an understanding of DSS and their potential than definitions. A decision support system may be defined by its capabilities in several critical areas. The four major characteristics of DSS that have evolved from the work of Alter, Keen, and others are that they:

1. tend to be aimed at the less well structured, underspecified problems,

2. attempt to combine the use of models or analytic techniques with traditional data access and retrieval functions,

3. focus on features that make them easy to use by noncomputer people in an interactive mode, and

4. emphasize flexibility and adaptability to accomodate changes in the environment and decision-making approach of the user.

A DSS contains three key subsystems as illustrated in Figure 3 on page 33. These are the Dialog Subsystem, the Data Subsystem, and the Models Subsystem. Much of the power, flexibility, and usability characteristics of a DSS are derived from capabilities in the interaction between the system and the user. The Dialog Subsystem is the medium through which the decision maker interacts with data and models. The dialog subsystem uses a combination of dialog styles to provide an interactive, user friendly decision support environment. Commonly used dialog styles include menus, question-answer dialog, command language, input form / output form, and input in context of output. The Data Subsystem contains all the data necessary to support the decision maker. Most DSS data subsystems use one of five data models. A data model is a method of representing, organizing, storing and handling data in a computer. These five data models are a record model, a relational model, a hierarchic model, a network model, and a rule model. Finally, the Models Subsystem contains the decision models and analytic techniques needed to analyze data. An im-

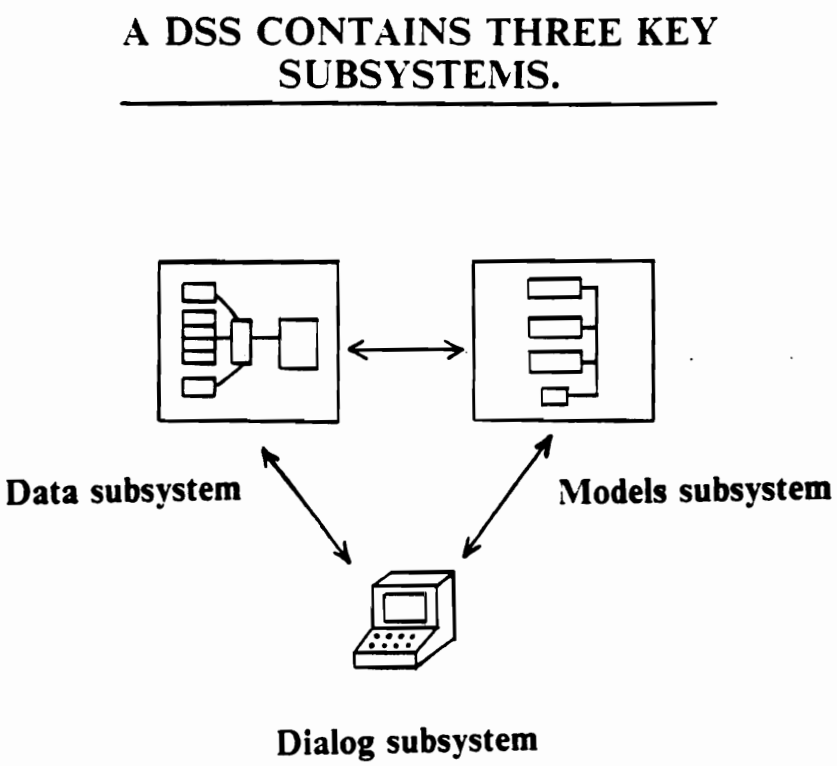


Figure 3. DSS System Structure

portant aspect of DSS is the ability to integrate data access and decision models. They do so by embedding the decision models in an information system that use the data base as the integration and communication mechanism between models.

The misuse and disuse of models has been a problem for many years [Benjamin, 1988]. One major problem has been that model builders were frequently preoccupied with the structure of the model. The existence of the correct input data and the proper delivery of the output to the user was assumed. In addition, models have suffered from inadequacy because of the difficulty of developing an integrated model to handle a realistic set of interrelated decisions. The solution has typically been to develop a collection of separate models, each dealing with a distinct part of the problem. Communication between these related models and integration of the system effects was left to the decision maker as a manual and/or intellectual process. Improvements in information technology have permitted new and different forms of integration [Benjamin, 1988]. DSS seeks to embed these models in an information system with the data base as the integration and communication mechanism between them.

2.5 Summary

The complex nature of logistics analysis has impeded the development of integrated, holistic models which encompass the policy decisions required to design and manage logistics support throughout the system life cycle. Logisticians such as Geisler and Murrie [1981], Kline [1982], and others have called for the development of models that are useful during all phases of the life cycle and are robust in that they address a broad range of design decisions. Geisler and Murrie [1981] called for the introduction of comprehensive logistics models that are analytic in nature to permit optimization of alternative performance measures. The model should be dynamic in order to treat nonsteady state (transient) as well as steady state conditions. The model should also be able to contend with dif-

ferent levels of detail so that useful planning can be done over the entire life cycle of the system. Kline [1982] recognized that, "It would be desirable to have logistic support models which are useful during all phases of the system life cycle. Unfortunately, most models are useful only beginning with the Full-Scale Development Phase."

In his review of multiechelon inventory theory, Clark [1972] states that, "the real frontier for research on multi[echelon] inventory systems lies in the area of integrated logistics, where the surface so far has barely been scratched." He recognizes that an appreciation of the interplay among operations, maintenance, transportation, and supply, tends to relegate pure inventory models to a less relevant status. Even though the full problem is exceedingly complex, there are several avenues worth exploring by analytic techniques. Clark suggests a decision-making approach which he calls micro-simulation. The micro-simulation technique is based upon recognition of an advanced real-time management system as being, in effect, a model of the real-world system being managed. Since such a management system contains status records in varying degrees of detail, supported by data flows to keep the records current, it is possible to add components that would simulate the external stresses upon the system as caused by the real-world operations. It would then be possible to make short-term accelerated-time, simulated excursions into possible futures of the system for decision-making purposes. Such micro-simulations might employ evaluation-selection procedures for postulated alternatives, or they might contain adaptive decision-making models.

The approach described in the next chapter provides a framework for the integrated design analysis of a repairable item and its logistics support system. The developed approach provides the capabilities identified by Geisler and Murrie, Kline, and Clark. The decision problems identified by Moore are embedded in a decision support system where the data base is used as the integration and communication mechanism between them. Because of the structure of the simulation routine included in the DSS, the modelling approach uses a periodic review inventory model. This permits development of more complex planning and evaluation models than have been previously reported.

3.0 Approach to the Problem

The objectives of this research were accomplished through the development of a decision support system (DSS) for the integrated design analysis of a repairable item and its logistic support system. Through integration of data and models, the DSS allows the decision maker to analyze the inter-related nature of design decisions throughout the system life cycle. The structure of the DSS is defined by describing each of its three key subsystems. The research database utilizes a record data model due to the limited size of the demonstration database. The research DSS uses primarily menus and question-answer dialog. The dialog subsystem was written in PL/1. Interactive input/output was programmed using IBM's Display Management System. The system was designed to run on an IBM mainframe computer in a VM/CMS environment.

The models subsystem includes a number of representative models to demonstrate the integration and evaluation of the interrelated decision problems which affect repairable items and their logistics support system design. The logistics decision problems addressed in the models subsystem are the:

1. equipment design problem,
2. maintenance configuration problem,
3. level of repair analysis problem,
4. spare equipment problem, and

5. replacement policy problem.

These decision problems were selected because they demonstrate the interrelated nature and inherent complexity of the decision problems which must be addressed in the design of a repairable-item and its logistics support system.

The integration of these decision problems is accomplished in the DSS data base. The data base contains all inputs needed for the decision problems. The inputs may be provided by the DSS user through the dialog subsystem, or they may be an output of one of the other decision problems. Therefore, the data flows shared by decision problems and the data flows between decision problems provide the necessary integration. These data flows were identified and documented using the data flow diagramming techniques described by Yourdon [1982].

A DSS provides decision making support. It is often used to aid in the understanding of relationships between a set of controllable variables and a set of uncontrollable variables [Sharda, 1988]. A necessary first step to the optimization of the design and operation of any system is specification of these controllable and uncontrollable variables. Fabrycky [1989, 1990] has developed a General Design Evaluation Function (GDEF) which mathematically links the design and operation of a system. This function can be expressed as:

$$E = f(X, Y_d, Y_i), \text{ where}$$

X = design variables (e.g., number of units deployed, retirement age, repair channels, etc.)

Y_d = design dependent parameters (e.g., weight, capacity, reliability, maintainability, etc.)

Y_i = design independent parameters (e.g., time value of money, labor rates, end item demand, etc.)

The controllable variables in this function are the decision variables and the design dependant parameters. Design alternatives are specified by establishing values for the set of design dependent parameters. For each set of these values, optimal values are sought for the decision variables. The resulting function value, $E = f(X, Y_d, Y_i)$, allows comparison of the design alternatives. Specification of each design decision problem requires definition of these variables and parameters. These values have been identified for each of the decision problems addressed in this research. Identification of these values provides the foundation for subsequent design optimization.

Research objective one was to provide a framework to support the investigation of interrelated design decisions. The DSS contains an imbedded simulation capability to analyze the effect of different design decisions on system performance over a defined life cycle. This approach is used because the inherent complexity of repairable item and logistics support design problems prohibits formulation of the integrated problem for solution using analytic optimization techniques [Clark, 1972]. The DSS supports the analysis of design alternatives. The approach developed by this research relies upon the expertise of the design analyst to search a set of feasible alternatives. For a given set of design dependent parameters, the designer must experimentally search the solution space for each decision variable. This approach does not preclude the addition of optimization procedures to search the solution space for each decision variable in the future. To take full advantage of Fabrycky's GDEF, optimization of the decision variables associated with a given set of design dependent parameters is required. Although this DSS does not currently employ analytic optimization techniques to perform such a search, the DSS is an effective decision making tool. Decision support systems in general do not employ analytic optimization procedures. Yet, studies show that a DSS can significantly enhance decision making performance [Sharda,1988]. Using a DSS allows investigation of more alternatives, produces better decisions, reduces performance variability, and increases user confidence in their decisions.

The DSS is basically a real-time, management system model of the repairable item and its logistics support system. The DSS is used to perform a logistic support analysis for plausible design alternatives. Logistic Support Analysis (LSA) is a process employed on an iterative basis throughout

the repairable item's life cycle. LSA determines the logistic support requirements for a given design (repairable item and its logistics support system). LSA aids in the evaluation of:

1. operational requirements for the repairable item and its logistics support resources,
2. alternative repair policies allowable within the constraints dictated by the maintenance concept and the allocated logistic support factors (reliability, maintainability, etc.),
3. specific characteristics in the equipment design (e.g., inherent reliability and maintainability features),
4. specific logistic support resource requirements based upon a fixed or assumed design configuration, and
5. overall system effectiveness (the repairable item and its logistics support resources).

As the system (the repairable item and its logistics support resources) progresses through the latter stages of its life cycle (production and/or construction, operational use, and retirement), logistic support resource requirements are updated and revised as necessary.

3.1 Logistics Design Decisions

There are several decision problems which must be addressed during the design of a repairable item and its logistics support system. Moore [1986] developed a hierarchy of logistics decision problems which must be considered. This research considers five of the ten primary decision problems identified by Moore. These are described below.

1. **Equipment Design Problem -** Determine the design, design specifiction, and/or procurement source for the repairable item which yields the best overall performance. This includes the allocation of various factors affecting logistics support (e.g., reliability, maintainability, etc.) to

the subelements of the system. In addition, the extent to which repair of an item will be accomplished must be determined (fully repairable, partially repairable, etc.).

2. **Maintenance Configuration Problem -** Determine how the maintenance and repair facilities will be configured. The number of maintenance levels to operate, the number of maintenance and repair facilities to operate at each level, the labor requirements at each facility, and the service disciplines to be used at each facility must be specified.
3. **Optimal Level of Repair/Level of Repair Analysis Problem -** Determine the maintenance level and the activity within that level at which each type of repairable-item failure will be repaired. Imbedded in this decision problem is the determination of spare parts inventories, reorder rules, replenishment quantities, and stockage locations for each spare part.
4. **Spare Equipment Problem -** Determine the number of repairable items to be procured, the timing of these procurements, and the deployment of these items into the various operating environments to satisfy demands and achieve availability objectives.
5. **Replacement Policy Problem -** Determine the circumstances under which a repairable item will be replaced. Inherent to this problem is the determination of the age at which items will be retired and replaced with a new item of the same design.

Five design problems identified by Moore are not addressed in this research. These are: 1) the mechanic training problem; 2) inspection and testing policy; 3) preventative maintenance policy; 4) the repair facility design problem; and 5) the operator training problem.

3.1.1 Equipment Design Problem

The equipment design problem concerns the design, design specifications, and/or procurement source for the repairable item which yields the best overall performance. This includes the allocation of various factors affecting logistics support (e.g., reliability, maintainability, etc.) to the

subelements of the system. In addition, the anticipated extent to which repair of an item will be accomplished is determined. This research addressed five aspects of this problem. Specifically, the:

- maintenance concept,
- reliability characteristics (MTBF and distribution),
- repair characteristics,
- maintainability characteristics, and
- end item cost.

The maintenance concept specifies the extent to which repair of an item will be accomplished. Three repair policies are commonly considered. These repair policies specify whether an item will be nonrepairable, partially repairable, or fully repairable. The repair policy chosen determines the extent to which the maintenance tree is defined (levels of indenture and number of nodes). In addition, a repair policy must be specified for each bottom-of-the-tree node in the maintenance tree. The policy chosen affects the spare parts required and the mean time to repair (MTTR). The maintenance concept selected is specified through definition of the:

1. maintenance tree,
2. mean time to repair (MTTR), and
3. spare parts and labor required to repair each bottom-of-the-tree node.

The design of the repairable item affects its cost, its failure characteristics, and its maintainability characteristics. The failure characteristics include the mean time between failure (MTBF) and the failure distribution. The frequency of failure may be reduced, thereby reducing the frequency of repair, by increasing the mean time between failure (MTBF) or by changing the failure distribution. This can be accomplished by using parallel redundant components or by selecting a higher quality, more reliable component. These increase the cost of the item but they should cause lower costs for labor and inventories. Parallel redundant components may increase repair time (MTTR) because

several components must be replaced rather than just one. The design cost is dependent upon the maintenance concept chosen and the failure characteristics specified.

In summary, the GDEF values considered by this research for this decision problem are:

Decision Variable

- number of parallel redundant components

Design Dependent Parameters

- maintenance concept
- reliability characteristics (MTBF and failure distribution)
- repair characteristics (MTTR)
- maintainability characteristics (spare parts and labor hours required to repair)
- end item costs

Each design alternative considered would reflect a different specification for the design dependent parameters.

3.1.2 Maintenance Configuration Problem

The maintenance configuration problem determines the structure of the maintenance and repair facilities. Specifically, this research addressed three aspects of this problem. These are:

- the number of repair levels,
- the number of repair channels at each level, and
- the labor requirements at each channel.

The number of repair levels specified determines the extent to which maintenance and repair facilities are decentralized. The DSS developed by this research accommodates a maximum of three levels of repair. This is strictly a programming restriction. Additional levels of repair could be easily accomodated. The number of repair channels at each level must also be specified. Failed items seeking repair at a given level are assumed to be equally distributed among the repair channels at that level. Therefore, if 50 failed items are seeking repair at the second level where there are five repair channels, the assumed demand at each channel is 50/5 or 10 failed items.

The trade-offs associated with this problem involve repair time versus repair costs. The MTTR includes repair time, transportation time to and from the repair facility, and wait time at the repair facility. As the number of levels and channels increases, those elements of MTTR other than repair time tend to decrease. This improves the availability measure of system effectiveness. However, the increased number of levels and channels increases facility costs and inventory costs. Therefore, life cycle costs tend to increase.

The labor requirements at each channel depend upon design decisions made in the equipment design problem and the level of repair analysis problem. The type and quantity of labor depends upon the maintenance concept chosen and the failure characteristics specified in the equipment design problem. The distribution of this labor depends upon the level of repair specified in the level of repair analysis problem.

In summary, the GDEF values considered by this research for this decision problem are:

Decision Variable

- number of repair channels at each level

Design Dependent Parameters

- number of repair levels and
- repair costs per day per channel
- labor hours required to repair

3.1.3 Level of Repair Analysis Problem

The level of repair analysis problem determines the maintenance level at which each type of repairable-item failure will be repaired. Imbedded in this decision problem is the determination of spare parts inventories, reorder rules, replenishment quantities, and stockage locations for each spare part. This research addresses four aspects of this problem. Specifically, the:

- level at which each repair is accomplished,
- spare parts inventory requirements at each level,
- periodic inventory review policy to be followed, and
- warehouse space requirements at each level.

The number of levels in the logistics support system is established by the maintenance configuration problem. At each level, it is assumed that the logistics support resources needed to accomplish repair of a failed item are available. The level at which each repair is accomplished determines the type and number of logistics support resources required at each level. The maintenance configuration problem addresses personnel and equipment resources. This decision problem focuses on spare parts inventory and warehousing requirements. The level at which each bottom-of-the-tree node is repaired is a user input. By changing the repair level specified for each node, the spare parts demand at each level is altered. Spare parts inventories are maintained according to a periodic review inventory policy. Warehouse space requirements are based upon average inventory levels.

Spare parts demand is based upon component failure rates. The failure rate for a given component is a function of its age, its MTBF, and its failure distribution. The spare parts required to support the population of end items are calculated for each stream and then summed for all streams.

$$D_{jt} = \sum_{s} NODE_{it} \times \overline{N}_{st} \times n_j$$

where:

D_{jt} = the demand for spare part j during period t.

$NODE_{it}$ = the probability of failure for node i in period t.

$\overline{N}_{st}$ = the average number of end items in operation in stream s during period t.

n_j = the number of spare part j required to repair node i.

The sum of the D_{jt} required for each stream gives the number of spare part j needed to support the total population for one time period. The D_{jt} are calculated for each time period simulated, t = 1, 2, 3, ..., run length.

Each spare part is assumed to have an initial inventory equal to the lead time demand. Therefore, if the lead time for part j, L_j , is 2 periods, the initial inventory, I_{j0}, is $D_{j1} + D_{j2}$. Inventory replenishments are then calculated using one of four reorder strategies. The strategies available are: 1) lot for lot, 2) economic order quantity, 3) period order quantity, and 4) the Silver-Meal heuristic.

With the lot for lot strategy, only the spare parts needed for the next period $(t + L_j)$, are ordered. Therefore, an order is placed every period. The order quantity in any period t for spare part j is $Q_{jt} = D_{j,t+L_j}$. Assuming the replenishment orders arrive at the beginning of the period and the de-

mand for spare parts occurs uniformly throughout the period, the average inventory in period t for spare part j is:

$$\bar{I}_{jt} = \frac{D_{jt}}{2}$$

The total inventory costs include the purchase cost of the spare parts, the procurement costs associated with order processing, and the inventory carrying costs. The total inventory costs, TC_j, associated with the lot for lot strategy are:

$$TC_j = (C_j \times D_j) + (T \times PC) + \sum_{t=1}^{T} \bar{I}_{jt} \times HC_j$$

where:

C_j = the unit cost of spare part j

D_j = total demand for spare part j during time simulated.

PC = procurement cost incurred every time a procurement of spare part j is initiated.

HC_j = the holding cost incurred to warehouse one unit of spare part j for one period.

T = number of time periods simulated

The reorder quantity for an economic order quantity (EOQ) policy seeks to balance order costs and inventory carrying cost. Whenever the lead time demand exceeds the available inventory, an order for a constant number of units is placed. Otherwise, the quantity ordered is zero. For an EOQ policy, the optimal order quantity is:

$$Q_{jt} = \sqrt{\frac{2 \times PC \times \overline{D_j}}{HC_j}}$$

The average demand per period for spare part j is:

$$\bar{D}_j = \sum_t \frac{D_{jt}}{T} \quad \text{, and}$$

The average inventory for an EOQ strategy is:

$$\bar{I}_{jt} = \frac{Q_{jt}}{2} \quad \text{for all t.}$$

The total inventory costs, TC_j, associated with the EOQ strategy are:

$$TC_j = (C_j \times D_j) + \frac{D_j}{Q_j} \times PC + \frac{Q_j}{2} \times HC_j$$

The period order quantity is based upon the economic order quantity. Rather than maintain a constant order quantity, it seeks to maintain a constant time between orders. The time between orders, T_j, is:

$$T_j = \frac{EOQ_j}{\bar{D}_j}$$

When the lead time demand exceeds the available inventory, an order for T_j periods demand is placed. If an order is placed, the order quantity, Q_{jt} is:

$$Q_{jt} = \sum_{k=t+L_j}^{t+L_j+T_j} D_{jk}$$

The quantity ordered will be zero otherwise. The average inventory during these T_j periods will be $\bar{I}_{jt} = \frac{Q_{jt}}{2}$. The total inventory costs associated with the period order quantity are:

$$TC_j = \sum_{t=1}^{T} (C_j \times Q_{jt}) + (P_t \times PC) + (\bar{I}_{jt} \times HC_j)$$

where:

P_t = 1 if an order was released in period t (zero otherwise)

$\bar{I}_{jt}$ = average inventory for spare part j during period t

The Silver-Meal heuristic seeks to minimize the average cost (order cost plus holding cost) per period. It does this by trading the cost of an additional procurement against the cost of holding inventory. Each procurement seeks to minimize the average cost per time unit (AC/TU) for the next k periods. The order quantity is established by determining that value of k that minimizes:

$$AC/TU = \frac{PC + [(1-1)D_{j1} + (2-1)D_{j2} + \cdots + (k-1)D_{jk}](HC_j)}{k}$$

The order quantity is then:

$$Q_{jt} = \sum_{m=t+L_j}^{t+L_j+k} D_{jm}$$

if an order is placed in period t. As in the previous reorder strategies, an order is placed if the lead time demand exceeds the available inventory. Otherwise, $Q_{jt} = 0$. The optimal value of k must be determined every time an order is placed. The average inventory during each of these k periods can be estimated as $\frac{Q_{jt}}{2}$.

The total inventory costs associated with the Silver-Meal heuristic are:

$$TC_j = \sum_{t=1}^{T} (C_j \times Q_{jt}) + (P_t \times PC) + (\bar{I}_{jt} \times HC_j)$$

The reorder strategy chosen is used to replenish all spare parts inventories. Warehouse space requirements can simply be determined by dividing the average inventory levels by the warehouse space requirements per unit stored (units per ft^2) input as a design independent parameter. In summary, the GDEF values considered by this research for this decision problem are:

Decision Variable

- level at which repair is accomplished

Design Dependent Parameter

- periodic inventory review policy

Design Independent Parameters

- spare parts order processing cost ($/order)
- warehousing costs ($/$ft^2$/$yr$)

3.1.4 Spare Equipment Problem

The spare equipment problem determines the number of repairable items to be procured, the timing of these procurements, and the deployment of these items into the various operating environments

to satisfy demands and achieve availability objectives. Specifically, this research addressed two aspects of this problem. These are:

- the number of spare end items to procure, and
- the mode of fielding (procurement profile).

The number of end items to procure is treated as a decision variable. As such, it is an input provided by the system designer. The initial allocation of these end items to the various operating environments must also be specified. Demand for end items in each operating environment is deterministic. Any end items procured, over and above the number required to meet this demand, are treated as spare items. These spare items increase population availability by helping meet demand while failed items are being repaired. Spare items are assigned to operating environments according to the ratio of environment demand to total demand or based upon a designer specified deployment.

The system designer may select from one of five basic procurement policies to follow when fielding the population of end items. Besides the option of procuring based on the period by period inputs of the user, the model supports procurement in uniform amounts, block amounts, increasing amounts, or decreasing amounts. Variations of these basic policies involving constraints on the population size provide the model user a total of nine different procurement policies from which to select. These nine policies are:

1. Continuous uniform policy
2. Uniform procurement until upper limit is reached
3. One initial block procurement
4. Block procurement with replenishment as units fail fatally
5. Continuous increasing procurement amounts
6. Increasing procurement until upper limit is reached
7. Continuous decreasing procurement amounts

8. Decreasing procurement until upper limit is reached
9. User defined policy

The model must calculate the amount to procure if an increasing or decreasing policy is selected. Based upon a specified percentage increase or decrease, the amount procured is given by:

$$P_p = P_{p-1} \pm [P_{p-1}(\frac{g}{100})]$$

where:

P_p = amount to be procured on the pth procurement,

P_{p-1} = amount procured on the p-1 procurement,

g = percentage increase or decrease in procurement amount.

Under the block procurement policy there is only one, initial block procurement. Under the uniform procurement options, P_p remains constant over time except as noted below. In the remaining procurement policies, an upper limit on the number of end items in the population can be maintained. In this case, the model checks the difference between the upper limit and the number of units in operation at the end of the period and compares this to P_p before determining the actual amount to procure.

$$\Delta = UPLIMIT - [N_t \times (1 - NODE_{6t})]$$

where:

N_t = total number of end items in operation in period t,

$UPLIMIT$ = upper population limit, and

$NODE_{6t}$ = probability of fatal failure during period t.

If the Δ is greater than P_p, then the amount procured is P_p. If P_p is greater than Δ , then Δ is the amount procured. The amount procured is rounded to the nearest integer value.

In summary, the GDEF values considered by this research for this decision problem are:

Decision Variables

- number of end items procured
- operating environment deployment

Design Dependent Parameter

- mode of fielding (procurement profile)

Design Independent Parameter

- cost of a procurement action

3.1.5 Replacement Policy Problem

The replacement policy problem determines the circumstances under which a repairable item will be replaced. Inherent to this problem is determination of the age at which items will be retired and replaced with a new item of the same design. This research specifically addressed the end item retirement age.

The end item retirement age primarily affects the period of time over which the initial cost of the end item can be capitalized. The longer this period of time, the lower the equivalent annual capital cost. However, corrective maintenance frequently tends to increase as the item ages, thereby increasing logistic support costs. These two cost components must be weighed against each other. Also, since the frequency of repair tends to increase with age, the availability measure will tend to deteriorate as the population ages. This will require an increase in the spare end items available to meet operating environment demands while these older items are undergoing repair.

The salvage value of a repairable item is assumed to be a function of age. The system designer must specify the repairable item first cost (in the equipment design problem), the retirement age, and the salvage value upon retirement. The salvage value is assumed to decrease at a decreasing rate as the item ages according to the following function.

$$SV_t = FC \times \beta^t$$

where:

SV_t = salvage value at time t

FC = repairable item first cost, and

β = fraction of value retained each period.

The value of β is determined based upon the first cost, salvage value, and retirement age input by the system designer.

$$\beta = e^{\frac{\ln(SV_n/FC)}{n}}$$

Therefore, if $FC = 50000$, $SV_n = 15000$, and $n = 10$ periods, then:

$$\beta = e^{\frac{\ln(15000/50000)}{10}}$$
$$= 0.8866$$

An item retired at the end of only three periods usage would have a salvage value of:

$$SV_3 = (\$50{,}000) \times (0.8866^3)$$
$$= \$34{,}846$$

At the end of each design simulation, all repairable items currently deployed are retired and salvage values determined based upon the age of the stream. In summary, the GDEF values considered by this research for this decision problem are:

Decision Variable

- retirement age

Design Dependent Parameters

- salvage value upon retirement
- salvage value upon fatal failure.

3.2 Multi-Indenture Relationships

A repairable item can be systematically divided into subelements until individual repairable components are identified. Each subdivision of the repairable item represents a level of indenture. At some level of indenture, it is not desirable to subdivide the item any further. At this point, either failure data are available or can be estimated. Once this level of indenture is reached, a list of stockable parts needed to repair the item upon failure is identified. This systematic subdivision of the end item into individual repairable components produces a tree-like structure [Brammer, 1985] referred to as a maintenance tree (see Figure 4 on page 56). At each level of indenture, the identi-

fied subelements are represented as nodes on the maintenance tree. The maintenance tree is used to identify the minimal information requirements necessary to determine logistic support requirements. The maintenance tree also provides a framework for expansion of this data base as the system design process progresses and additional data become available. The maintenance tree is the representational view of the data base from the user's perspective.

The maintenance tree also supports any of the repair policies a decision maker may choose. A fatal failure node is tied to each maintenance tree. The decision maker need not define all subelements of the end item to the same level of indenture. Therefore, an end item may be nonrepairable, partially repairable, or fully repairable. The maintenance tree also provides flexibility in analysis of logistics requirements over the system life cycle. In the early stages of the life cycle when detailed data are not available, the maintenance tree can be defined with few or no levels of indenture. As the design progresses and more detailed data become available, progressive levels of indenture can be created and additional data utilized. The maintenance tree also allows the decision maker to allocate reliability, maintainability, and other factors to the various subelements of the system and consider their effects on logistics requirements.

3.2.1 Maintenance Tree Terminology

Certain terms are used frequently in describing the maintenance tree. Definitions of key terms include:

- **End Item** - A single piece of repairable equipment (e.g., an airplane, a ship, a computer, etc.).

- **Component** - Refers to subassemblies, modules, or parts of the end item. Each component defined is assigned a node on the maintenance tree.

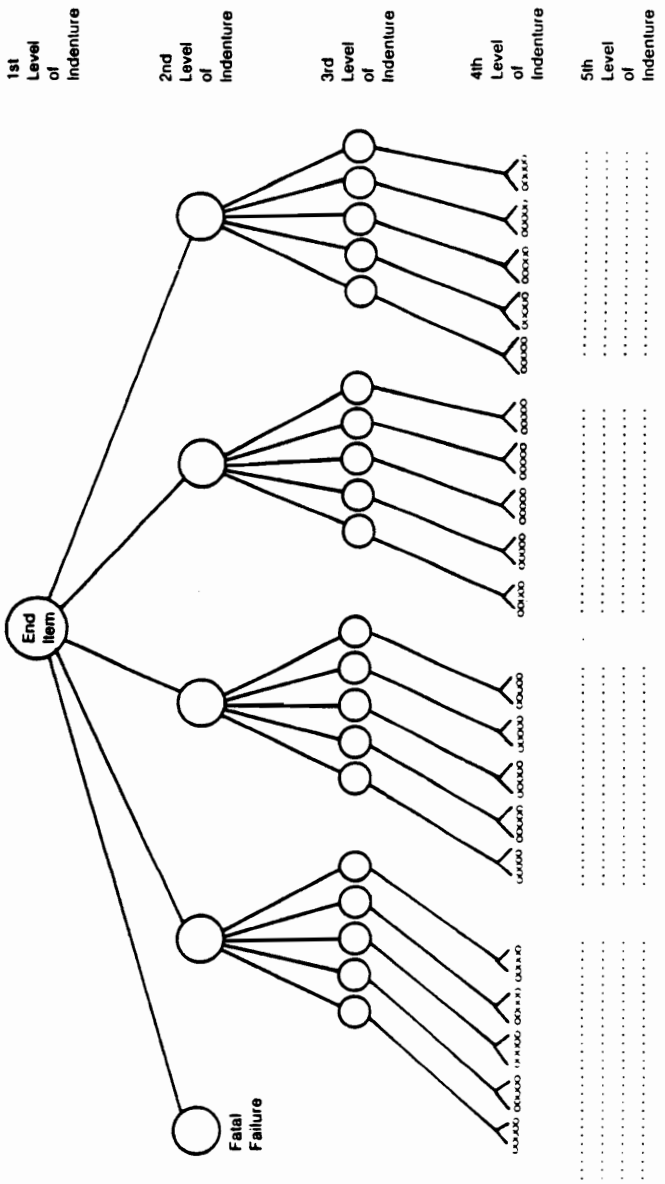


Figure 4. Maintenance Tree Model of Multi-Indenture Relationships

- **Level of Indenture** - A decomposition of the end item into its components. The first level of indenture is the end item itself. The second level of indenture includes major components (subassemblies) that make up the end item. The third level of indenture includes components (modules) that are components of the second level of indenture and so forth. At the last level of indenture, the stockable items required to accomplish repair are defined.

- **Node** - A node is a point on the maintenance tree from which subsidiary components originate. Node 1 represents the end item. Nodes 2 through 5 at the second level of indenture, represent major assemblies or divisions of the end item. Node 6 is used to represent the probability of a fatal failure of the end item. Failures at any other node are assumed to be repairable. However, a failure at Node 6 is not repairable and causes the end item to be removed from service. Nodes 7 through 26 at the third level of indenture, represent subassemblies or subdivisions for Nodes 2 through 5. Other nodes represent a further decomposition of the end item into its component parts.

- **Stockable Item** - A spare part required to accomplish repair of the bottom-of-the-tree nodes.

3.2.2 Assumptions

Several assumptions about the end item and the interrelationships between its component parts are necessary to support the modeling approach employed in this research. These are:

1. Failure of any critical component or a parallel network of redundant components represented by a node on the maintenance tree results in an immediate failure of the end item.
2. Component failure probabilities are based upon the time an end item is initially fielded, not when it was last repaired. Component repairs at time t-1 do not affect the probability of failure for that component at time t. This avoids the necessity of having to track each individual component repair.

3. Component failure probabilities at any level of indenture are independent of the failure of other components at that same level of indenture. Only one component can fail at a time.
4. Failure of a component at one level of indenture is dependent on the failure of components at lower levels of indenture that are on a connected branch of the maintenance tree.
5. When a component at the bottom of the tree fails, all stockable items associated with that node are required to accomplish a repair.

3.2.3 Probability of Failure

Demands are placed upon the logistics support system by failure of a component at one of the bottom-of-the-tree nodes. Therefore, a probability of failure for each period must be determined for each component represented by a bottom-of-the-tree node. These failure probabilities may be input directly by the user or may be generated using one of the predefined failure distributions. There are five predefined failure distributions, see Figure 5 on page 59, that may be used to compute failure probabilities. These are the uniform, exponential, increasing exponential, normal, and bathtub distributions.

These five failure distributions represent the failure characteristics frequently exhibited in practice. The uniform distribution represents those situations where the component has a constant probability of failure from period to period. The exponential distribution reflects those situations where the probability of failure decreases exponentially over time. The increasing exponential distribution represents the opposite scenario where the probability of failure increases exponentially over time. The last two distributions also represent opposite situations. The approximate normal distribution reflects lower failure rates in the early and late stages of a component's life. The bathtub distribution represents those situations where there are higher failure rates early in the life of a new component, lower failure rates during the majority of the life of the component and then higher rates again during the later stages of component life.

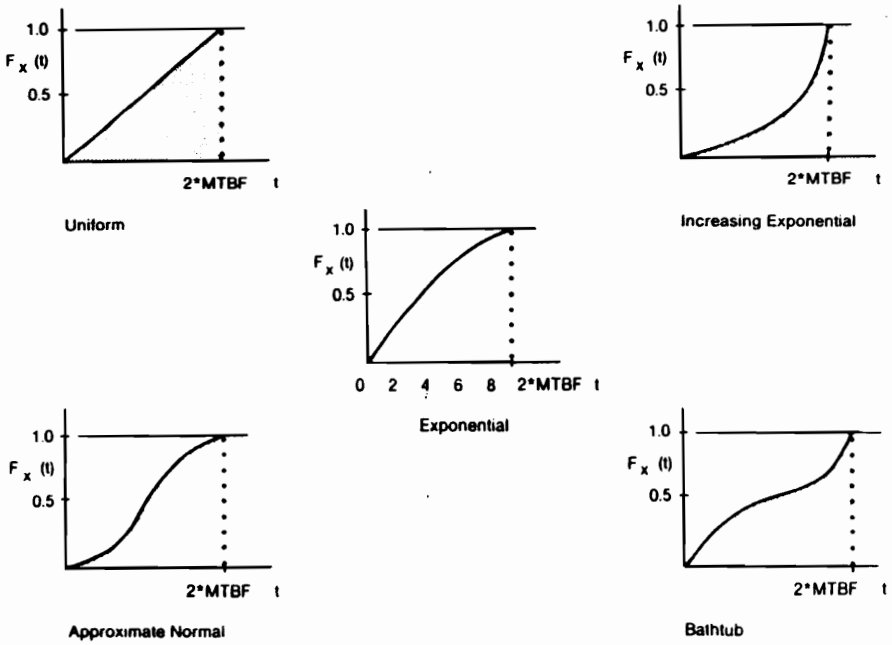


Figure 5. Cumulative Failure Probability Distributions

The equations used to compute these failure probabilities are well known and widely published in the literature. The exponential distribution is a tangential function that approaches a failure probability of 1.0 as time approaches infinity. The approximate normal shape produced by the Weibull distribution with a shape parameter of three also has a cumulative distribution function which is tangential to 1.0 as time approaches infinity. However, it is assumed that the probability of failure equals 1.0 at the end of a component's expected life span. A component's expected life span is assumed to be equal to twice the MTBF for the component. For these distributions, $F_x(t) \cong 1$ when $t = 2 \times MTBF$. Therefore, this appears to be a reasonable assumption.

In the Uniform, Exponential Increase, and Bathtub distributions, the coefficients have also been chosen such that $F_x(t) = 1$ at the end of the component's assumed life span. After the end item has been in operation a period equal to twice the component's MTBF (see an example given in Figure 6 on page 61), it is assumed that the component fails and the probability distribution for this component regenerates. This assumption is necessary because as t approaches (2 * MTBF), components fail every period.

Using the stated assumptions about these five distributions, the cumulative probability of failure, $F_x(t)$, for each is:

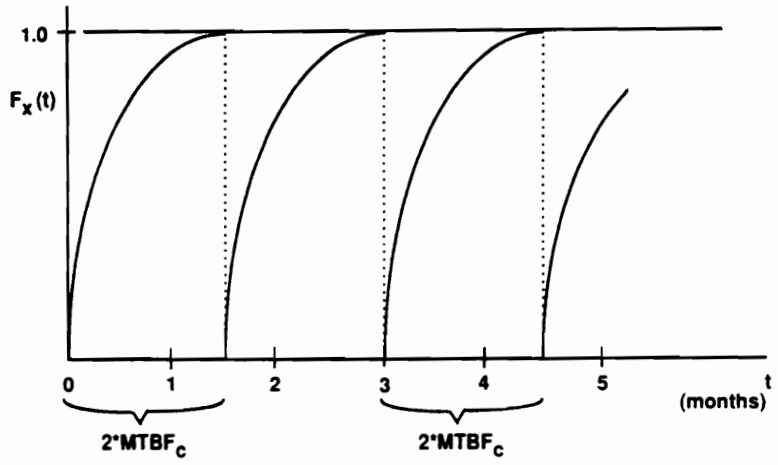
Uniform Distribution:

$$F_x(t) = \frac{\lambda t}{2}$$

Exponential Distribution:

$$F_x(t) = 1 - \frac{1}{e^{\lambda t}}$$

Increasing Exponential Distribution:



$MTBF_c$ = .75t = .75 months

$MTBF_c$ is the mean time between failures for component c.

t = 0 is the point in time at which the end item, of which component c is a part, first went into operation.

The cumulative failure probability distribution for component c regenerates after reaching an upper limit of 2^*MTBF_c.

Figure 6. Failure Distribution Regeneration

$$F_x(t) = \frac{1}{e^{1/\lambda}} e^{t/2}$$

Approximate Normal Distribution: (Weibull with shape parameter = 3)

$$F_x(t) = 1 - \frac{1}{e^{(\lambda t)^3}}$$

Bathtub Distribution:

$$F_x(t) = \frac{1}{e^2} e^{\lambda t}$$

where:

λ = failure rate (1/MTBF)

e = irrational number = 2.718282.

t = number of periods that the end item has been in operation.

3.2.4 Probability of Failure of End Items

The probability of failure of the end item is conditional on the probability of failure of all of the "bottom-of-the-tree" nodes plus the fatal failure node, $NODE_6$. The probability of failure for nodes located at intermediate levels of the tree (between the bottom-of-the-tree nodes and the end item node) are conditional on the probability of failure of nodes connected directly beneath them. In this limited research model, each node can be decomposed into at most five nodes at the next lower level of indenture (see Figure 4 on page 56). Therefore, a given node at one level in the maintenance tree can be the "parent" of at most five "children" at the next lower level. In addition, that

same node will be a child of a node at the next higher level of indenture. This is a self-imposed programming limitation, not a limitation of the framework developed by this research nor a limitation of the programming language used to implement the framework. This restriction can be eliminated if a more extensive decomposition of the end item is needed.

The probability of failure at the parent node is dependent upon the failure probabilities for its five children at the next lower level of indenture. Since components at a given level of indenture fail independently, the failure probability for each parent node is given by the union of the failure probabilities for its five children. For example, $NODE_1$, the probability of failure for the end item, is given by:

$$NODE_1 = NODE_2 \cup NODE_3 \cup NODE_4 \cup NODE_5 \cup NODE_6$$

The Venn Diagrams in Figure 7 on page 64 illustrate these calculations graphically. Each ellipse in the Venn Diagram represents the probability of failure at a given node in the maintenance tree.

These calculations are recursively applied to the maintenance tree, beginning with the bottom of the tree nodes, until the entire tree has been resolved. The steps required to compute $NODE_1$ (probability of failure of an end item) are:

Step 1: Compute $NODE_i$ for each bottom of the tree node as described in the previous section. In a fully defined maintenance tree, these nodes will be at the 4th level of indenture.

Step 2: Compute the failure probability for any bottom of the tree nodes at the third level of indenture as described in Step 1. Using conditional probabilities, and assuming horizontal independence of failure in the maintenance tree, compute the failure probabilities for each parent node at the 3rd level of indenture based upon the failure probabilities for those nodes directly below it.

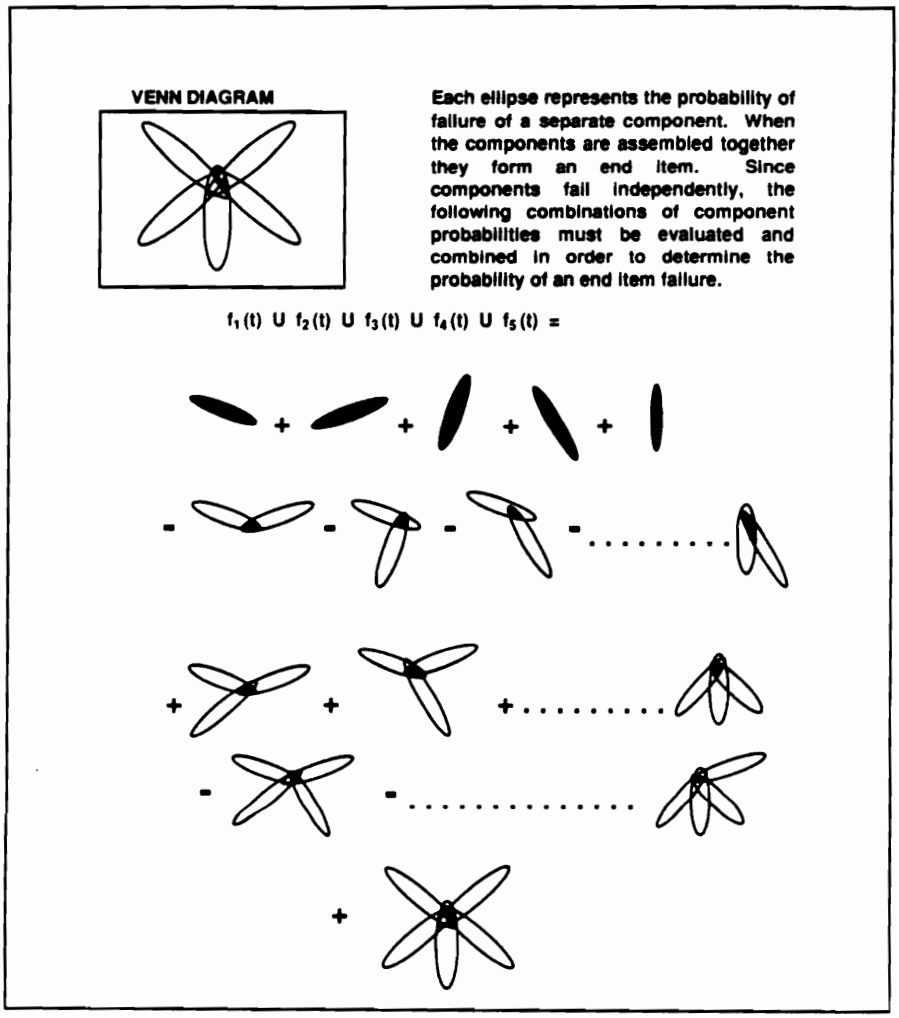


Figure 7. Computation of Conditional Probabilities

The basic expression used to calculate the failure probability of a parent node at any level of indenture in the maintenance tree is illustrated for $NODE_{26}$ below. Other parent node failure probabilities can be calculated by simply changing the subscript values. The children of $NODE_{26}$ are $NODE_{122-126}$. To compute the failure probability for $NODE_{26}$, the expression would be:

$$NODE_{26} = \sum_{i=122}^{126} NODE_i - \sum_{i=122}^{125}\sum_{j=i+1}^{126} NODE_i \times NODE_j + \sum_{i=122}^{124}\sum_{j=i+1}^{125}\sum_{k=j+1}^{126} NODE_i \times NODE_j \times NODE_k - \sum_{i=122}^{123}\sum_{j=i+1}^{124}\sum_{k=j+1}^{125}\sum_{m=k+1}^{126} NODE_i \times NODE_j \times NODE_k \times NODE_m + \prod_{i=122}^{126} NODE_i$$

If a given node is not used (e.g., $NODE_{122}$ in Figure 12 on page 98), the probability of failure at that node will obviously be equal to zero. Therefore, the above expression could be simplified on a case by case basis. However, the more general form of this expression is used to recursively calculate the failure probability at each parent node in the maintenance tree.

Step 3: Repeat Step 2 for the 2nd level of indenture.

Step 4: Use the following formula to calculate $NODE_1$, once the conditional probability calculations have proceeded to that level of indenture in the tree. In this calculation, $NODE_{2-6}$ are children of $NODE_1$. Therefore, the subscript values for i, j, k, and m are 2 to 6. This is the same expression given in Step 2 with a different set of subscripts.

$$NODE_1 = \sum_{i=2}^{6} NODE_i - \sum_{i=2}^{5}\sum_{j=i+1}^{6} NODE_i \times NODE_j + \sum_{i=2}^{4}\sum_{j=i+1}^{5}\sum_{k=j+1}^{6} NODE_i \times NODE_j \times NODE_k - \sum_{i=2}^{3}\sum_{j=i+1}^{4}\sum_{k=j+1}^{5}\sum_{m=k+1}^{6} NODE_i \times NODE_j \times NODE_k \times NODE_m + \prod_{i=2}^{6} NODE_i$$

3.2.5 End Item MTTR

The mean time to repair an end item in any time period is computed by multiplying the "bottom of the tree" node failure probabilities by the MTTR for each node. The resulting value is the expected MTTR for that item. End items in different streams may have different expected MTTR's in a given time period. The expected MTTR is given by:

$$\overline{MTTR}_{st} = \frac{\sum_{i=1}^{B} [MTTR_i \times NODE_{it}]}{B}$$

where:

$MTTR_i$ = MTTR of the component represented by node i.

$NODE_{it}$ = failure probability for bottom-of-the-tree node i in period t.

B = total number of bottom-of-the-tree nodes.

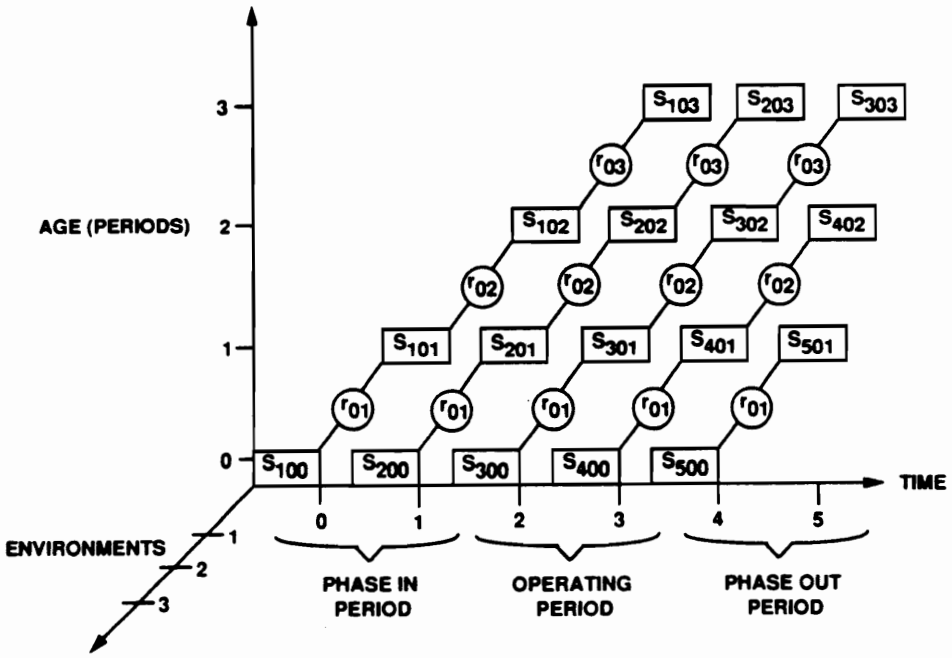
$\overline{MTTR_{st}}$ = expected time spent in repair during period t by an end item from stream s.

3.3 Nonstationary Demands

The transient nature of logistics requirements during system operation and retirement can be captured using the multistream concept described by Frisch [1983]. In the multistream concept illustrated in Figure 8 on page 68, a family of nested Markov chains is used to model a population of repairable items. The population is divided into streams depending upon such parameters as age of the end item, the number of hours that the end item has operated, or the environment in which the end item operates. Because of the parameter used to divide the end items into streams, each stream represents a homogeneous group of end items. The homegeneity within each group or stream is maintained by applying the same failure distribution and repair characteristics to each item within a particular stream. In other words, items within a stream are similarly aged, are at the same stage of deterioration, and experience similar failure characteristics. Grouping end items into streams avoids the need for item by item tracking.

A stream begins when the first item in that grouping is procured and the stream ends when the last item in that grouping is removed from operation through retirement or fatal failure. Each stream is in effect a Markov chain and all of the streams grouped together form a model representing the failure characteristics of a population of repairable items. Each step of the individual streams represents a period specified by the user of the system.

The multistream concept also captures the stochastic nature of system operation and performance. Defining probalistic parameters on the maintenance tree, allows modeling of the various failure and



S_{ijk} = number of end items in stream i, environment j after k periods of operation.

S_{ij0} = initial procurement (fielding) for stream i, environment j (k = 0).

r_{jk} = end item survival rate in environment j after k periods of operation [1-P (Failure)].

Figure 8. Multistream Model of Nonstationary Demands

repair characteristics of subelements within the end item. Data established on the maintenance tree is used at each step of the Markov chain to determine the logistics support requirements of the population during operation.

3.4 Measures of System Effectiveness

There are many measures of performance appropriate to analyze logistics support systems. These include reliability, maintainability, availability, and life cycle cost. Optimization of one measure does not necessarily optimize the other measures and trade-offs between performance and cost must be considered. Therefore, a multicriteria decision model is used to consider the collective impact of selected measures of performance on logistics support requirements. The decision model allows the logistics manager to select the desired measures of performance and establish weights which reflect the relative importance of each measure in a given system analysis. An important benefit of this approach is that it eliminates the use of the shortage penalties frequently imbedded in the objective functions of logistics models. Shortage penalties have always been difficult to define and interpret.

In evaluating overall systems effectiveness, the logistics manager must consider system operational and support factors versus total life-cycle cost. System operational and support factors include:

- **Availability** - the probability that a system will be ready or available when required for use. It is often used as a measure of system readiness. Three commonly used measures of availability are inherent availability, achieved availability, and operational availability.
- **Reliability** - the probability that a system or product will perform in a satisfactory manner for a given period of time when used under specified operating conditions.

- **Maintainability** - the ease, accuracy, safety, and economy in the performance of maintenance functions. Maintainability can be measured in terms of a combination of elapsed times, personnel labor hours rates, maintenance frequencies, maintenance cost, and related logistic support factors. Common measures include mean corrective maintenance time, maintenance downtime, maintenance manhours per maintenance action, mean time between maintenance, and cost per maintenance action.

Total life-cycle cost considerations include research and development cost, production/construction cost, operation and maintenance cost, and retirement and disposal cost.

The DSS allows the logistics manager to select from available decision models, define logistics support parameters, and determine their impact on overall system performance as measured by the multicriteria decision model. The DSS performs a simulation of the logistics system operation over a period of time specified by the logistics manager. At the end of the simulation, logistics system performance is assessed. The logistics manager can then examine alternative system designs until the desired level of performance is achieved.

3.4.1 Availability

Availability is the probability that a system will be ready or available when required for use. It is a measure of the system's ability to satisfy demands. Traditional measures of availability focus on the repairable item rather than on the population of repairable items. For example, Blanchard [1986] defines inherent availability for a single item as:

$$A_i = \frac{MTBF}{MTBF + MTTR}$$

An end item can be in one of two states: in operation or in repair. The item is in operation during the time between failures. Therefore, inherent availability is the percentage of time an item is in

operation. This research addressed a population of repairable items rather than a single repairable item. In addition, traditional measures of availability do not represent the stochastic aspects of system behavior, especially under nonstationary conditions.

Therefore, this research used a nontraditional measure of availability developed by Brammer [1985]. This measure is based upon the probability of being short (unable to meet demands) by one or more units. The calculations begin at the stream level, proceed to the operating environment level, and then address population availability. The average time spent in repair by items in stream s during period t, $\overline{MTTR_{st}}$, was determined previously in Section 3.2.5. The average time spent in repair by items in environment e during period t, $\overline{MTTR_{et}}$, is:

$$\overline{MTTR_{et}} = \frac{\sum_{s \in e} [\overline{N}_{st} \times \overline{MTTR}_{st}]}{\sum_{s \in e} \overline{N}_{st}}$$

where:

$\overline{N}_{st}$ = number of end items in stream s during period t.

The average probability of failure for an end item in environment e during period t, $P(f)_{et}$, is:

$$P(f)_{et} = \frac{\sum_{s \in e} [\overline{N}_{st} \times NODE_{1st}]}{N_{et}}$$

where:

N_{et} = number of end items in environment e during period t, and

$NODE_{1st}$ = probability of failure for an end item in stream s during period t.

$$N_{et} = \sum_{s \in e} \overline{N}_{st}$$

The average probability of failure for an end item in environment e during a period of time equal to $MTTR_{et}$ is:

$$P(f)'_{et} = [\frac{MTTR_{et}}{LTP}] \times P(f)_{et}$$

where:

LTP = length of a time period.

In order to calculate the availability of end items, the probability of being short one or more units must be calculated. The availability in environment e is:

$$A_e = 1 - \sum_{x=1}^{D_e} M_x$$

where:

D_e = the demand in environment e.

M_x = the probability of x end items out of operation.

Another way to write this would be:

$$A_e = \sum_{x=0}^{k} M_x$$

where:

$k = N_e - D_e$

It is assumed that when the number of end items in operation in a given environment becomes less than the demand for end items in that environment, then at that point in time, the availability of that system of end items is zero. The availability calculations determine the probability that, for a given time period, the number of units in operation is greater than or equal to the demand. This probability represents the availability of end items versus the demand. If the number of end items out of operation at any given time remains between zero and k for an entire time period, then the availability for that time period would be 100%.

The chance of failure of any one end item is independent from the chance of failure of any other end item within or outside of the same stream. Further, because of the homogeneity of the streams, all end items within a stream have the same probability of failure. In the above calculations, the value $P(f)_e$ establishes an average probability of failure for all end items operating in the same failure environment, so all end items operating in the same failure environment in effect have an equal probability of failure. There are two things that can happen to an end item during a given period: either the end item experiences a failure, or the end item experiences no failure.

With the characteristics of the foregoing paragraph in mind, it can be assumed that the probabilities of 0,1,2,...,N simultaneous failures of end items is binomially distributed. The binomial distribution as applied to a population of end items would be defined as follows:

$$\sum_{x=0}^{n} b(x; n,p) = 1$$

where:

n = total number of end items in the population.

p = probability of failure of an end item.

Availability can now be computed as the probability of 0 to k end items being out of operation simultaneously. The availability of end items in environment e during period t is:

$$A_{et} = \sum_{x=0}^{k} \mathrm{b(x;n,p)}$$

where:

$k = N_{et} - D_e$.

D_e =the demand for end items in environment e.

$p = P(f)'_{et}$.

The availability for the entire population of end items in period t is the availability in each environment, weighted by the number of items in each environment, summed over all environments.

$$A_t = \frac{\sum_e A_{et} \times N_{et}}{N_t}$$

where:

N_t = number of end items in operation during period t.

3.4.2 Reliability

The reliability for a single repairable item can be determined using $NODE_{1t}$, the probability of failure for the end item during period t. The reliability function, R(T), is commonly defined by Blanchard [1986] and others as:

$$R(T) = 1 - F_x(T)$$

Therefore,

$$R(T) = 1 - \sum_{t=1}^{T} NODE_{1t}$$

The cumulative failure probability after T periods of operation will be identical for every item in a given stream. However, items deployed in different operating environments exhibit different failure characteristics. Therefore, the cumulative failure probability after T periods will be different for streams fielded in different environments. Thus, the population reliability, R_t, is dependent upon the number of items deployed in each environment. The reliability for the entire population of end items after T periods of operation is:

$$R_t = \frac{\sum_{e} R_{et} \times N_{et}}{N_t}$$

where:

R_{et} = reliability for a single repairable item deployed in environment e after T periods of operation.

N_{et} = number of items in environment e during period t.

N_t = total number of end items in operation during period t.

Parallel redundant networks can be used to improve the reliability of a given component. One or more identical components are placed in parallel with the operating component so that all components must fail in order to cause system failure. The reliability function for a parallel redundant network of n components, $R'(t)$, may be expressed as:

$$\begin{aligned} R'(t) &= 1 - (1 - [R(t)])^n \\ &= 1 - (1 - [1 - F_x(t)])^n \\ &= 1 - (1 - 1 + F_x(t))^n \\ &= 1 - [F_x(t)]^n \end{aligned}$$

Therefore, the cumulative probability of failure for the parallel redundant network of n components is $F_x(t)^n$. Accordingly, parallel redundant networks can be modeled by using $F_x(t)^n$ as the cumulative probability of failure at time t. Letting n = 1 specifies a series type network. For example, the cumulative probability of component failure at time t for the uniform failure distribution was previously defined to be:

$$F_x(t) = \frac{\lambda t}{2}$$

To allow modeling of parallel redundant networks of n components, the previously defined function is replaced with:

$$F_x(t) = (\frac{\lambda t}{2})^n$$

3.4.3 Maintainability

The population maintainability can be assessed by determining the average amount of time spent in repair. Recall that:

$\overline{MTTR}_{et}$ = average repair time for end items in environment e during period t

and that:

$\overline{N}_{et}$ = average number of end items in environment e during period t

The weighted average time in repair for all end items in the population during period t, $\overline{MTTR}_t$, is given by:

$$\overline{MTTR_t} = \frac{\sum_e \overline{MTTR_{et}} \times \overline{N_{et}}}{\sum_e \overline{N_{et}}}$$

3.4.4 Life-Cycle Cost

Life cycle cost is calculated per stream per period and then totalled for the period. At the end of the simulation, the total cost for the time periods simulated is given. The period costs are converted to present worth values. All present worth values are summed and converted to a period equivalent cost. This period equivalent cost provides a measure that can be easily compared against other alternatives with unequal lives. The factors for converting to present value and to annual equivalent values are:

$$P = F(P/Fi,n) = F \times \frac{1}{(1+i)^n}$$

$$A = P(A/Pi,n) = P \times \frac{i \times (1+i)^n}{(1+i)^n - 1}$$

where:

P = present value

A = annual equivalent cost

F = future value

i = time value of money (i.e., the interest rate)

n = time periods

The costs included in the life cycle costs are:

- ***Capital Cost*** - The first cost of an end item, the salvage value received for a failed end item, and the salvage value received for a retired end item are factors considered in calculating the capital cost. Capital cost is computed on a per period basis using the following equation:

$$CC_e = (NP_e \times FC) - (NF_e \times SV_e) - (NR_e \times BV_e)$$

where:

CC_e = capital cost for end items in environment e.

NP_e = number of end items procured and placed in environment e.

FC = first cost of an end item.

NF_e = number of end items in environment e that fail fatally during a period.

SV_e = salvage value for a failed end item.

NR_e = number of end items in environment e retired at the end of a period.

BV_e = salvage value for retired end item from environment e.

- ***Repair Cost*** - The repair cost per time unit multiplied by the total amount of time that units spend out of service gives the repair cost. However, since the repair cost per unit time varies by repair level, this calculation must be performed for each bottom of the tree node and then aggregated at the item, stream, and environment levels. The repair cost per period in each environment is:

$$RC_{et} = \sum_{s \in e} \overline{N}_{st} \times [\sum_{i}^{B} NODE_{it} \times MTTR_i \times RC_i]$$

where:

RC_{et} = repair cost per period for end items in environment e.

RC_i = repair cost per unit time for $NODE_i$.

$MTTR_i$ = mean time to repair for $NODE_i$.

$NODE_{it}$ = probability of failure for $NODE_i$ in period t.

$\overline{N}_{st}$ = average number of items in stream s during period t.

B = number of bottom of the tree nodes (nodes that can incurr repairable failures).

- ***Inventory Cost*** - The sum of the TC_j described in Section 3.1.3 constitute the inventory cost. The following equation is used to calculate inventory cost:

$$IC_e = \sum_j TC_{je}$$

where:

IC_e = total inventory cost per period for end items in environment e.

TC_{je} = inventory costs for spare part j which is a stockable item for an end item in environment e.

- ***Procurement Cost*** - The administrative cost incurred when end items are procured constitutes procurement cost. This cost is a total period cost and is not allocated as an environment cost. Since only one procurement per period is allowed, if NP_t, the number of end items procured at beginning of period t, is greater than zero, then $PC_t = 1 \times PC$. Otherwise, $PC_t = 0$.

- ***Total Cost*** - The model determines total period cost for each environment. These figures are converted to present worth values and summed. These environment dependent costs are then added to the environment independent procurement cost to determine total cost. This value is then converted to a period equivalent cost.

$$TC_{te} = CC_e + RC_e + IC_e$$

$$TC_t = \sum_e TC_{te} + PC_t$$

$$TC = \sum_{t} TC_t \times (\text{P/F i,t})$$

$$AC = TC \times (\text{A/P i,n})$$

where:

TC_{te} = total cost for environment e in period t. Procurement costs of end items are not included.

TC_t = total costs during period t.

TC = total present cost of system design alternative

AC = period equivalent cost of systen design alternative.

3.4.5 System Effectiveness

The four measures of design effectiveness (life cycle cost, reliability, maintainability, and availability) are computed by the DSS. The logistics planner must specify the relative importance (weight) for each measure and the extreme points of a linear scale for measuring system performance. The logistics planner can specify any set of weights to be used by the model. These are normalized during compution of the composite measure for system evaluation.

It is suggested that these weights sum to some power of ten (e.g., 10^1, 10^2, etc.). The extreme points of the linear scale represent the minimum (score = 0) and maximum (score = 10) levels of measureable system performance. For example, consider the availability measure. Suppose the logistics planner specifies a minimum value of 90% and a maximum value of 100%. Therefore, any population availability less than or equal to 90% would score a zero and any population with an availability equal to 100% would score a ten. If the population availability was between these

two values, say availability equal to 93%, then the population score would be determined as follows:

$$\text{Score} = \frac{\text{System Performance} - \text{Minimum Value}}{\text{Maximum Value} - \text{Minimum Value}} \times 10$$
$$= \frac{93 - 90}{100 - 90} \times 10$$
$$= \frac{3}{10} \times 10 = 3$$

In the case of a fractional score (e.g., 3.7), the calculated score is rounded to the nearest integer. The following example illustrates application of this multicriteria decision model. Assume the system evaluation parameters shown in Table 1 have been input by the system designer for each of the four criteria and the CODAS model has calculated the actual system performance during the period being investigated. Note that, for life cycle cost and maintainability, a lower value for net present cost and maintenance hours per repair, respectively, is more desirable. Therefore, the lower value should receive the maximum score.

Table 1. Multicriteria System Evaluation

Criteria	Minimum Value	Maximum Value	Criteria Weight	System Performance
Life-Cycle Cost	100	60	25	75
Availability	90	100	30	97
Reliability	94	100	25	95
Maintainability	25	10	20	15

- Life Cycle Cost

$$Score = \frac{75 - 100}{60 - 100} \times 10 = 6.25 \cong 6$$

- Availability

$$Score = \frac{97 - 90}{100 - 90} \times 10 = 7$$

- Reliability

$$Score = \frac{95 - 94}{100 - 94} \times 10 = 1.67 \cong 2$$

- Maintainability

$$Score = \frac{15 - 25}{10 - 25} \times 10 = 6.67 \cong 7$$

The composite system evaluation can be determined as follows.

SE = system evaluation

ECS_i = evaluation criterion score for criterion i

ECW_i = evaluation criterion weight for criterion i

$$SE = \frac{\sum_{i=1}^{4} ECS_i \times ECW_i}{\sum_{i=1}^{4} ECW_i}$$

$$= \frac{(6 \times 25) + (7 \times 30) + (2 \times 25) + (7 \times 20)}{100}$$

$$= \frac{550}{100} = 5.5$$

This single measure of overall system effectiveness can now be used to compare alternative logistics system configurations.

3.4.6 Optimization

The measure of system effectiveness developed in Section 3.4.5 is a function of both performance and cost. Mathematically, this measure captures both the design and operation of the repairable item and its logistics support system. Fabrycky [1989, 1990] expresses this composite measure using a General Design Evaluation Function (GDEF):

$$E = f(X, Y_d, Y_i)$$

where:

X = design variables (e.g., number of repair channels, retirement age, etc.)

Y_d = design dependent parameters (e.g., cost, MTBF, MTTR, etc.), and

Y_i = design independent parameters (e.g., time value of money, end item demand, etc.).

Optimization of the GDEF requires identification of the design variables, the design dependent parameters, and the design independent parameters associated with the design decision problem. The GDEF values considered by this research are:

Decision Variables (X_i)

1. Number of parallel redundant components
2. Number of repair channels at each level
3. Level at which repair is accomplished
4. Number of end items procured
5. Operating environment deployment

6. Retirement age

Design Dependent Parameters (Y_d)

1. Maintenance concept
2. Failure characteristics (MTBF and failure distribution)
3. Repair characteristics (MTTR)
4. Maintainability characteristics (spare parts and labor hours required to repair)
5. End item costs
    - first cost
    - salvage value upon fatal failure
    - salvage value upon retirement
6. Mode of fielding (procurement profile)
7. Number of repair levels
8. Periodic inventory review policy
9. Repair costs per day per channel
10. Labor hours required to repair

Design Independent Parameters (Y_i)

1. Time value of money
2. End item demand
3. Cost of a procurement action
    - spare parts ordering cost
    - spare repairable items
4. Warehousing costs ($\$/ft^2/pd$).

This is not an exhaustive list of these values for the five decision problems considered by this research. Rather, these are the GDEF values currently addressed within the framework developed by this research. Additional GDEF values can be included as the framework is developed further.

There are frequently constraints set on both the parameter set Y_d and the variable set X (e.g., failure rate $\leq \lambda$, retirement age $\leq$ n, etc.). Optimization of the GDEF must be accomplished subject to these constraints. The approach developed by this research relies upon the expertise of the design analyst to search a set of feasible design alternatives. The best alternative in this set is identified using the multicriteria measure of system effectiveness.

A given design alternative is specified by definition of the design dependent parameters. Optimal values are then sought for the decision variables. Alternative designs will have different values for one or more of the design dependent parameters. Each design dependent parameter change may affect one or more decision variable. The design analyst must search the solution space, seeking to improve system effectiveness, based upon his/her expertise and the cost-performance tradeoffs associated with each decision problem. The interrelationships between the five design decision problems addressed in this research can be modeled using a data flow diagram. This data flow diagram, shown in Figure 9 on page 86, illustrates the impact of one decision problem upon another.

A data flow diagram is a network representation of a system [DeMarco, 1979]. It portrays the system in terms of its component pieces, with all interfaces among the components indicated. There are four component pieces to a data flow diagram in general. These are: 1) data sources and sinks; 2) data flows; 3) processes; and 4) files. A data source is a net originator or receiver of system data. In this research, the design decision problems are the data sources. These are represented in Figure 9 as rectangular boxes. The data flows (represented as named vectors in Figure 9) portray the interrelationships between components on a data flow diagram. Data flows do not represent the flow of control used by the computer or person that processes the data. The controllable GDEF values (design dependent parameters and decision variables) are represented by data flows in Figure 9. The processes (represented as circles in Figure 9) show the transformation of incom-

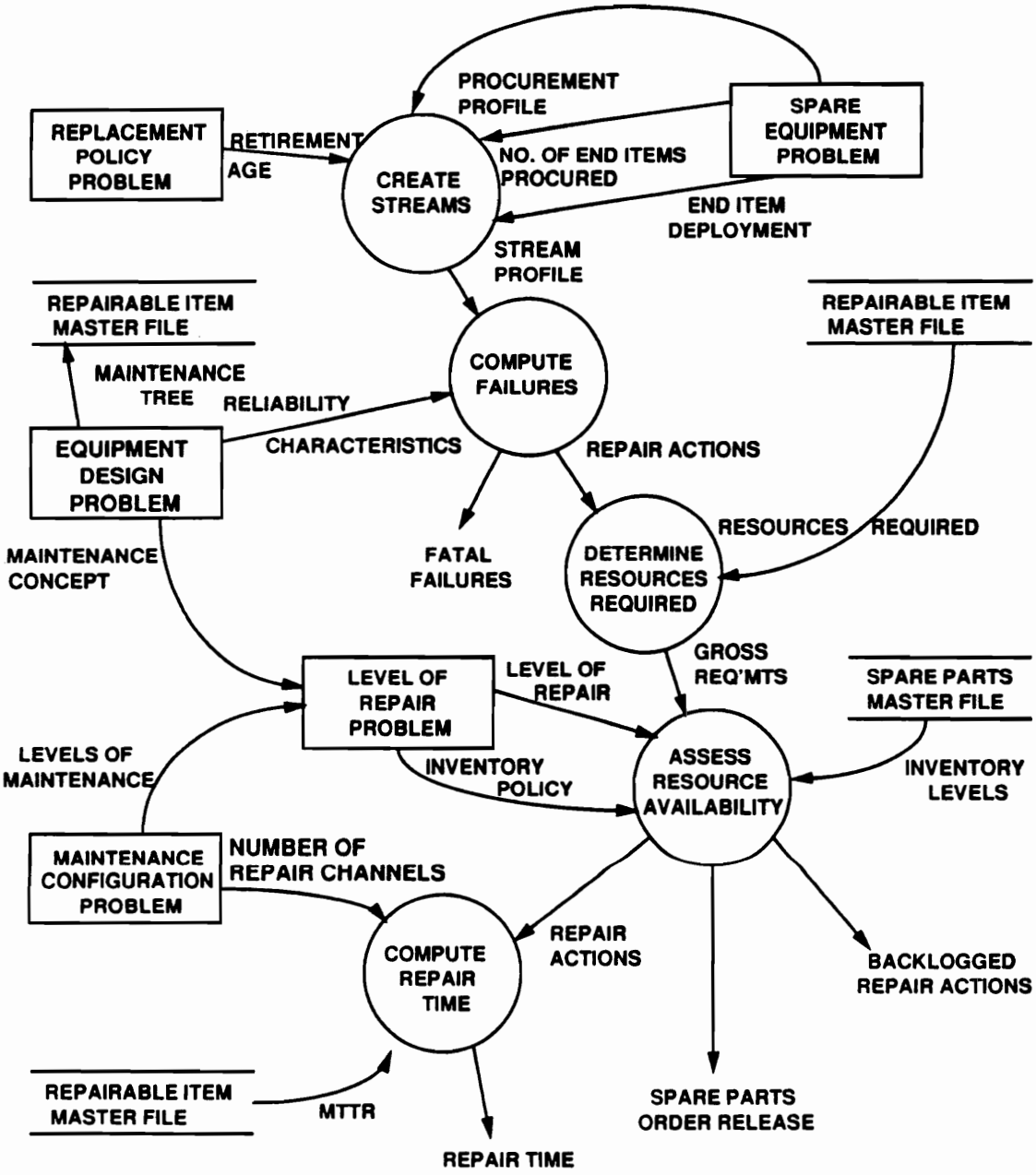


Figure 9. System Data Flow Diagram

ing data flow(s) into outgoing data flow(s). In addition to these values, these processes use data stored in files to accomplish this transformation. A file (represented in Figure 9 by its title between two parallel lines) is simply a temporary repository of data.

The data flow diagram shown in Figure 9 graphically portrays the interrelationships between the design decision problems. The data flow diagram also helps the system designer to identify the controllable values which should be modified to affect a particular aspect of system operation. For example, five controllable values affect the number of repair actions which the logistics support system must process. These can easily be identified from the data flow diagram. The number of repair actions is an output of the ***compute failures*** process. By investigating the inputs to this process, it is readily apparent that these five values are:

1. procurement profile,
2. number of end items procured,
3. end item deployment,
4. retirement age, and
5. reliability characteristics.

The first four of these values affect the fielding of the population of repairable items while the last value affects the reliability characteristics once the item is deployed. The system designer can choose to influence any or all of these values to achieve the desired impact on the number of repair actions. If the designer wanted to affect the time varying dimension of repair, alternative procurement profiles might be investigated. If the mean number of repairs needs to be reduced, the impact of early retirement or improved reliability characteristics might be evaluated.

Similarly, suppose the system designer wanted to improve the maintainability measure of performance. Maintainability is measured as a function of the time spent in repair. The data flow diagram shows that repair time is an output of the ***compute repair time*** process. The inputs to this process are:

1. number of repair channels,
2. mean time to repair (MTTR), and
3. repair actions.

The number of repair channels is an output of the maintenance configuration problem. The MTTR is retrieved from the Repairable Item Master File which is an output of the equipment design problem. Recall that the maintenance tree is the representational view of the repairable item database. The structure of this tree is dependent upon the maintenance concept established by the equipment design problem. Therefore, the MTTR is a function of the maintenance concept. The system designer could choose to influence either of these values to improve maintainability.

Similarly, the data flow diagram can be used to identify the controllable values which impact the repair actions. In this case, the number and type of repair actions is a function of many controllable values. In fact, further investigation shows that the number and type of repair actions is in some way affected by all controllable values except the number of repair channels. In this case, it becomes difficult to select which of the design decision variables to influence. This highlights a weakness of this approach. While the data flow diagram effectively portrays the interrelationships between component parts of the system, it does not provide any information on the degree or magnitude of this interaction. Here it would be helpful to have something such as a multiple regression equation where the coefficients of each controllable value represented the importance of that value to the output (e.g., repair actions) of interest.

Whatever its failings, the data flow diagram provides the system designer with a clear, easy to use portrayal of the interrelationships between the the design decision problems. It helps to structure information which previously existed only as part of an individual's experiential knowledge. It serves as an aid in the identification of those values which should be influenced during the design optimization process to achieve the desired level of system design effectiveness.

3.5 *Summary*

Chapter 3 has described the approach used to accomplish the objectives of this research. All modeling constructs employed by this research were described and equations for all associated calculations developed. This included definition of the design decision problems, explanation of the multi-indenture relationships between component parts of the repairable item, discussion of the multistream model of nonstationary demands, and establishment of measures of system effectiveness.

The five design decision problems addressed by this research were fully defined in this chapter. The scope and significance of each decision problem was explained and those aspects of each problem considered in this research were identified. In addition, the GDEF (general design evaluation function) values were established for each decision problem. These values define the design decision variables, design dependent parameters, and design independent parameters associated with each design decision problem.

The multi-indenture relationships between component parts of the repairable item were explained. The terminology used to describe the structure of a repairable item was defined and modeling assumptions about the relationships between these components were enumerated. The calculation of failure probabilities throughout the maintenance tree and the calculation of the MTTR (mean time to repair) was illustrated. In addition, the multistream model used to capture the nonstationary aspects of repair facility demands was explained.

Finally, the measures of system design effectiveness were established. Each of these four measures were defined and the equations required to compute each were developed. Incorporation of these four measures into a single measure of system design effectiveness using a weighted multicriteria evaluation technique was illustrated. Optimization of the system design using the GDEF values

was discussed. Chapter 4 describes the implementation of this approach and demonstrates the utility of the approach using a case study.

4.0 Model Implementation

The decision support system described in the previous chapter has been implemented in a case study to demonstrate the utility of the approach. The focus of this chapter is to illustrate how CODAS can be used to analyze alternative system designs. The DSS provides a framework where interrelated system (the repairable item and its logistics support system) design decisions can be analyzed in an integrated manner. The DSS supports integrated design analysis by concurrently analyzing the design of a repairable item and its logistics support system. The DSS developed by this research is named CODAS, for Concurrent Design Analysis System. The DSS was developed using PL/1 and DMS. It contains 41 interactive input/output screens. Each of these screens, their function, record layouts, a program listing, and other details of the DSS operation are described in Appendix C, CODAS User's Guide.

The analysis of design alternatives necessarily begins with specification of an initial design alternative. This specification must include definition of all decision variables, design dependent parameters, and design independent parameters. The values which must be specified for the five design decision problems were previously summarized in Section 3.4.6. Once these values have been defined, system (the repairable item and its logistics support system) operation and performance is simulated. In their discussion of simulation, Hillier and Lieberman [1980] state that "By repeating

this [simulation of system operation over time] for the various alternative design configurations and comparing their performances, the most promising configurations can be identified."

The data flows within the simulation are shown in Figure 9 on page 86. At time zero, one stream is created for each operating environment. The number of end items in each of these streams is specified by the deployment of end items from the initial procurement. Time is then advanced one period. During the first period of operation, the end items in each stream have aged one period. Using the cumulative failure probabilities specified for each bottom-of-the-tree node, the expected number of failures during the first period can be calculated. These failures generate demands upon the logistics support system for spare parts, labor, etc., based upon the specified maintenance concept repair policy. These demands for logistics support resources are then segregated by level of repair. Once the demand at each level of repair is identified, spare parts requirements, repair times, labor requirements, etc., can be determined. The procurement policy selected is tested to determine if additional end items are to be procured. If so, additional streams are created. If older streams have reached their retirement age, they are terminated.

Time is then advanced to the next period, and the whole process is repeated. This continues until the number of periods specified by the design analyst has passed. At the end of the simulation, the period spare parts demands are used to compute average inventory levels based upon the lot sizing strategy specified. Warehouse space requirements are then determined based upon unit storage requirements. In addition, the average number of items in operation in each environment can be compared to the demand in each environment to assess population availability. This deterministic simulation process can be summarized as shown in Figure 10 on page 93.

The described methodology represents an experimental approach to optimization. This approach is frequently used for problems that are too complex to solve analytically. The simulation model describes the operation of the system in terms of individual components whose behavior can be predicted, at least in terms of probability distributions, rather than directly describing the overall behavior of the system.

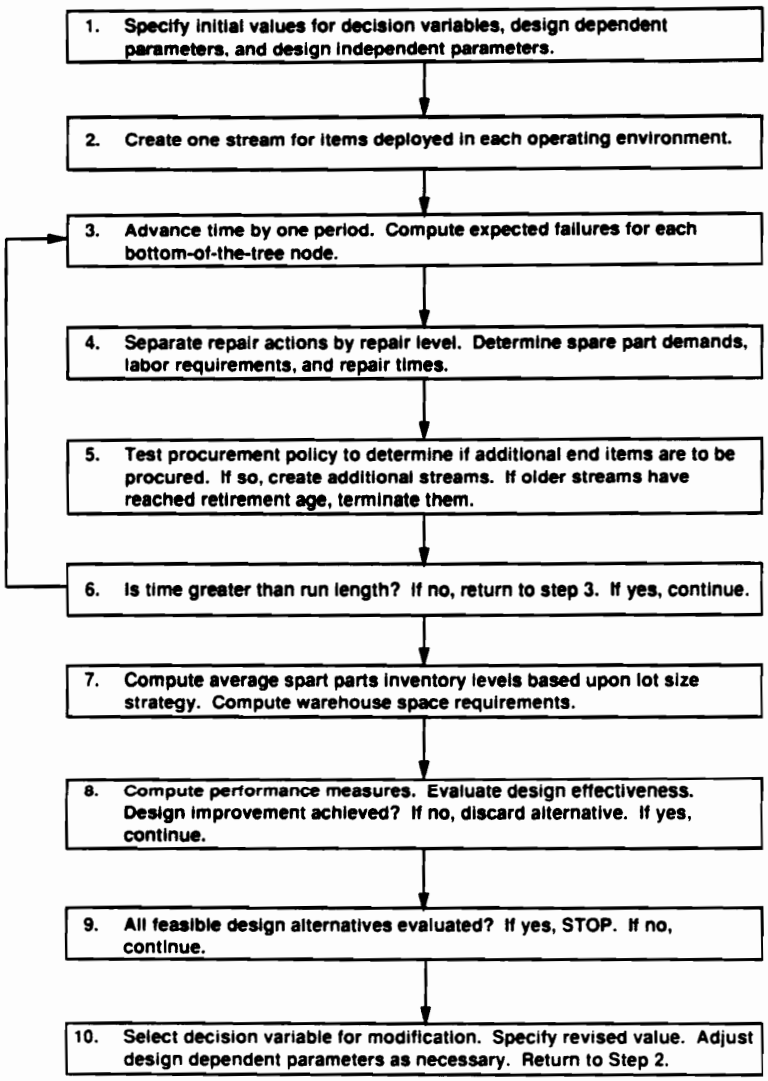


Figure 10. Simulation Flow Diagram

4.1 SLWT Case Study

A case study demonstrates the application of the developed framework. The Side Loadable Warping Tug (SLWT) is a component of the Container Offloading and Transfer System (COTS) utilized by the U.S. Navy. COTS is a deployable system designed to provide seaborne military forces of the U.S. Navy with logistics support during entry on land masses where suitable port facilities are not available. The logistics support provided is basically the unloading (from ships) and transfer to shore of the required quantities of bulk dry cargo and vehicles. The COTS is designed to operate beginning on the day after land entry by military forces through day 180 or longer. The SLWT, shown in Figure 11 on page 95, is a powered causeway section that can serve several functions as part of COTS. When equipped with a winch and A-frame, one of the major functions of the SLWT is to assist in the construction of the causeway on piers that connect ship to shore. The SLWT can also assist in the mooring of vessels, craft salvage operations, or the transport of shipping containers from ship to shore.

4.1.1 Design Specification

The SLWT is virtually immune to fatal failure when maintained properly. An exception is when the SLWT is operating in a militarily hostile environment where an SLWT may be damaged irreparably by enemy attack. In the case study, a number of SLWT's are assumed to be operating in a Support Area environment, removed from hostile military action. The remaining SLWT's are assumed to be operating in a Combat Zone. The Combat Zone environment differs from the Support Area environment in two ways. These two differences are the fatal failure probabilities and the number of hours worked daily.

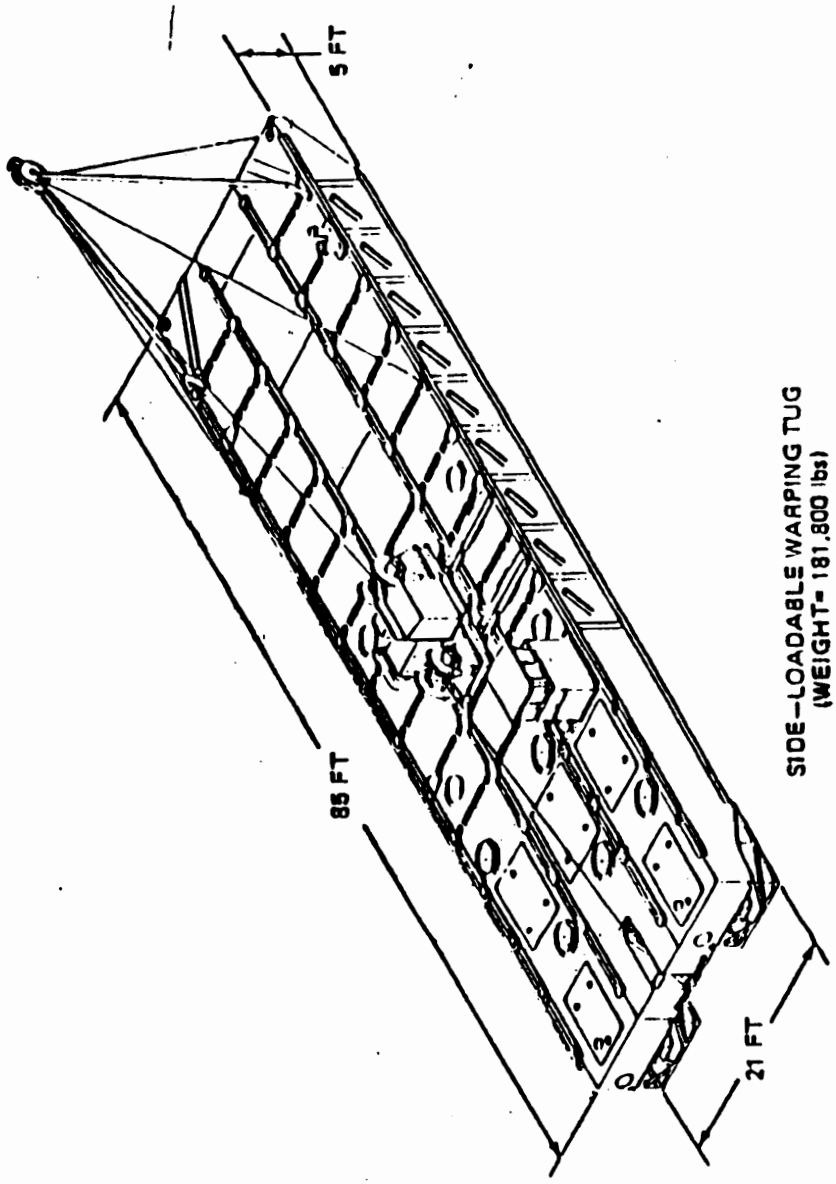


Figure 11. Side Loadable Warping Tug (SLWT)

The major difference between the two environments is the fatal failure probabilities. The SLWT's operating in the Support Area environment have a very small chance of fatal failure. The mean time between fatal failures is 100,000 operating hours per unit. The cumulative rate of fatal failure for these SLWT's follows an increasing exponential function. Therefore, early in the life of these SLWT's there will be an almost nonexistent probability of fatal failure. However, SLWT's operating in the Combat Zone environment have a mean time between failures of only 7,200 operating hours per unit. The cumulative rate of fatal failure follows a uniform distribution, meaning that there is an equal chance of fatal failure within each time period. This type of fatal failure occurs mainly due to enemy attack.

A second difference between the two environments is that SLWT's work a different number of hours per day in each environment. In the Support Area environment, a SLWT works an average of 8 hours per day. In the Combat Zone environment, an SLWT works an average of 12 hours per day. It is this difference in hours worked per day that gives the various components (bottom-of-the-tree nodes) of the SLWT different failure probabilities in each environment. For example, consider the the bottom-of-the-tree node representing a seawater pump. The seawater pump has a MTBF of 5000 hours. When the number of hours worked per day is considered, the resulting MTBF in each environment is:

$$MTBF_{support} = 5000 \text{ hours}/8 \text{ hours per day} = 625 \text{ days}$$

$$MTBF_{combat} = 5000 \text{ hours}/12 \text{ hours per day} = 417 \text{ days}$$

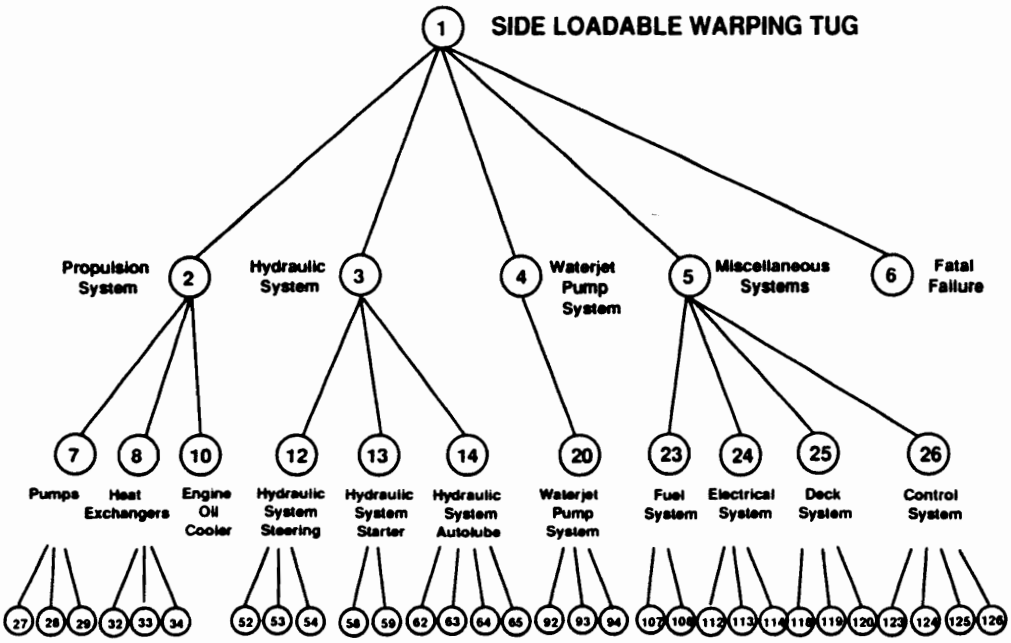
When a SLWT requires repair, three levels of repair are available. Organizational level (level 1) repairs are performed on board the SLWT. Intermediate level (level 2) repairs are performed by a repair facility set up on shore. Finally, depot level (level 3) repairs are performed at a centrally located facility. Depot level repairs are only necessary for more serious failures. The SLWT must be transported to the central depot location. Given the more serious nature of depot repairs and the transportation time required, these repairs require longer to complete. The repair level chosen

for a given failure is established by the level of repair analysis problem and remains constant regardless of the failure environment in which the end item is deployed. The spare parts needed to accomplish the assigned repairs are inventoried at each location.

The maintenance tree for the SLWT is given in Figure 12 on page 98. Failure and repair data for each of the bottom-of-the-tree nodes must be input for each of the two failure environments considered. The required data includes repair level, MTBF, MTTR, failure curve, and a stockable parts list. For each stockable part, the quantity needed to accomplish repair and various inventory parameters (unit price, order cost, holding cost, lead time, and space requirements) must be provided. The data used in this case study are summarized in Appendix A and B.

This case study considers the fielding of a proposed population of 225 SLWT's. The population is fielded using one initial block procurement with replenishment as units fail fatally (policy option 4). Of the 225 SLWT's procured, 75 will be deployed in a Combat area environment. The remaining 150 will be deployed in a Support area environment. The number of SLWT's fielded in each environment exceeds the actual demand. This is necessary because of maintenance downtime.

The specification of the decision variables and design parameters associated with this design are illustrated in Figure 13 through Figure 24. These figures are CODAS input screens for the SLWT case study. These screens allow the system designer to define simulation parameters, operating environment data, maintenance system configuration, maintenance tree data (failure and repair data), equipment cost data, level of repair data, spare parts inventory policy, the spare parts required to accomplish repair, the mode of fielding the population, and the end item retirement age. These are the data needed to completely define a design alternative.



27 Freshwater Pump
28 Seawater Pump
29 Hydraulic Pump
32 Engine Coolant Heat Exchanger
33 Freshwater Heat Exchanger
34 Hydraulic Fluid Heat Exchanger

52 Pressure Reducing Valve
53 Directional Control Valve
54 Steering Motor
58 Filter
59 Starter
62 Automatic Lubrication
63 Timer
64 Manifold
65 Solenoid

92 Waterjet Pump
93 Discharge Elbow
94 Steerable Nozzle
107 Booster Pump
108 Fuel/Water Separator
112 Inverter
113 Battery
114 Alternator

118 Hydraulic Motor
119 Hydraulic Pump
120 Cable
123 Engine On/Off
124 Throttle
125 Steering Lever
126 Clutch Lever

Figure 12. SLWT Maintenance Tree

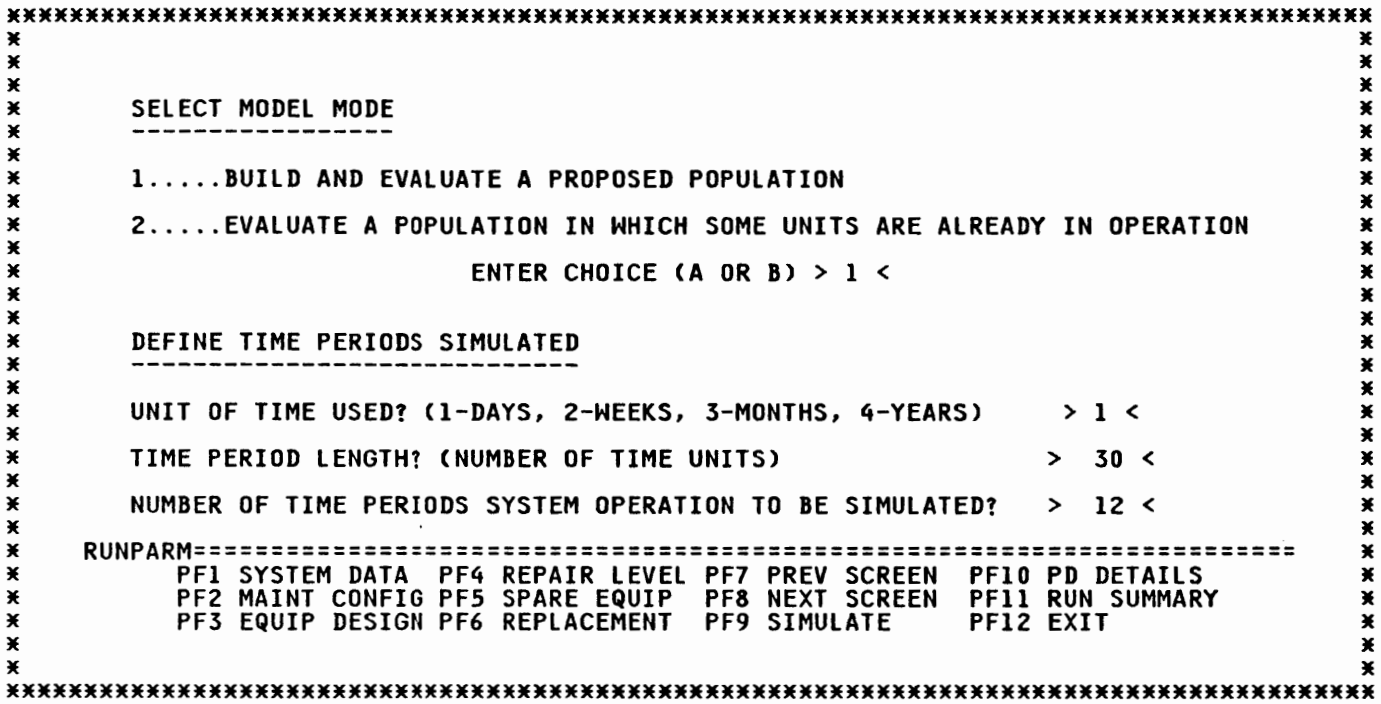


Figure 13. SLWT Simulation Parameters

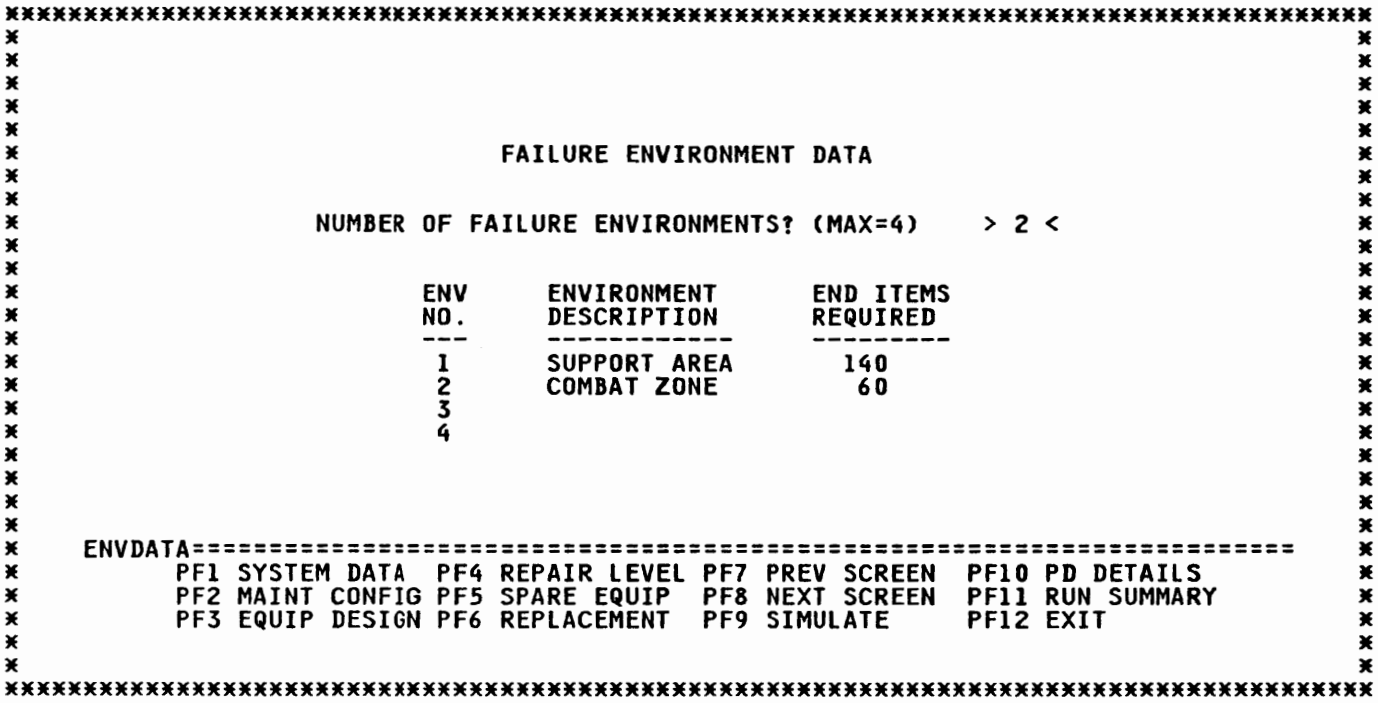


Figure 14. SLWT Operating Environments

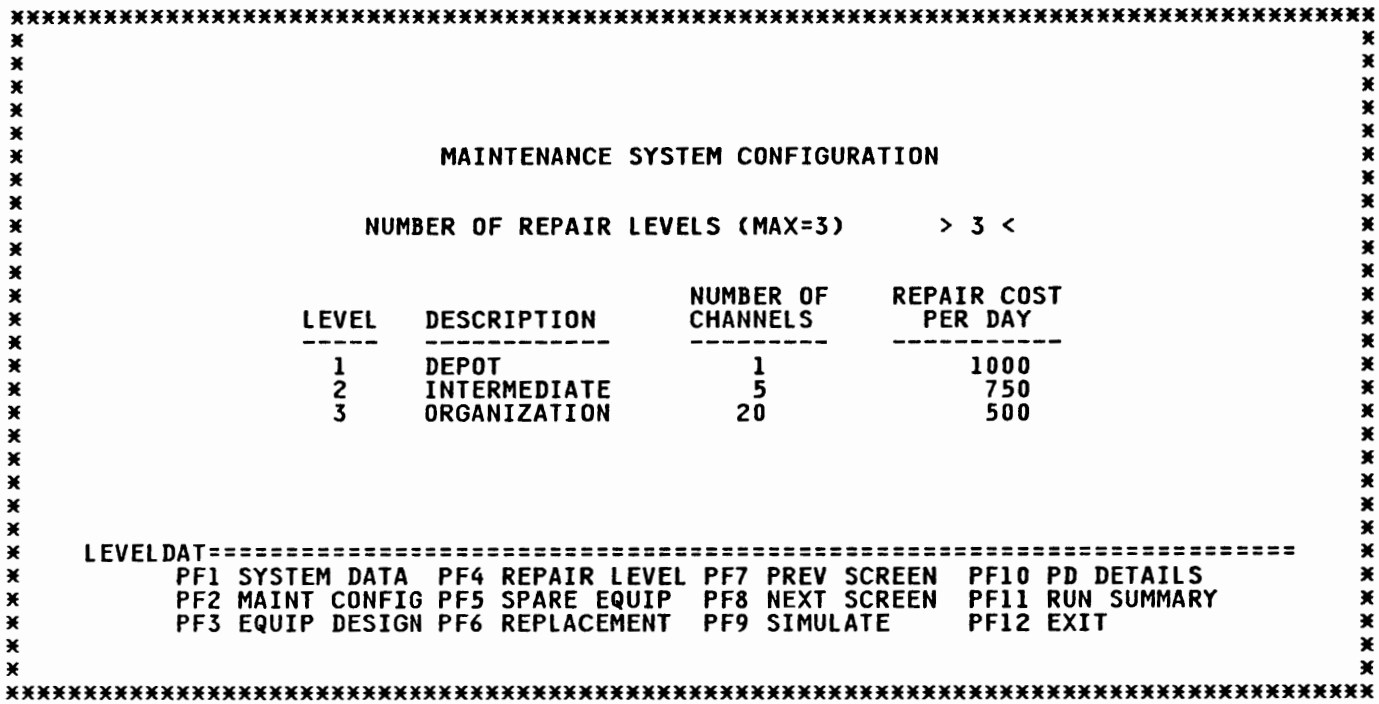
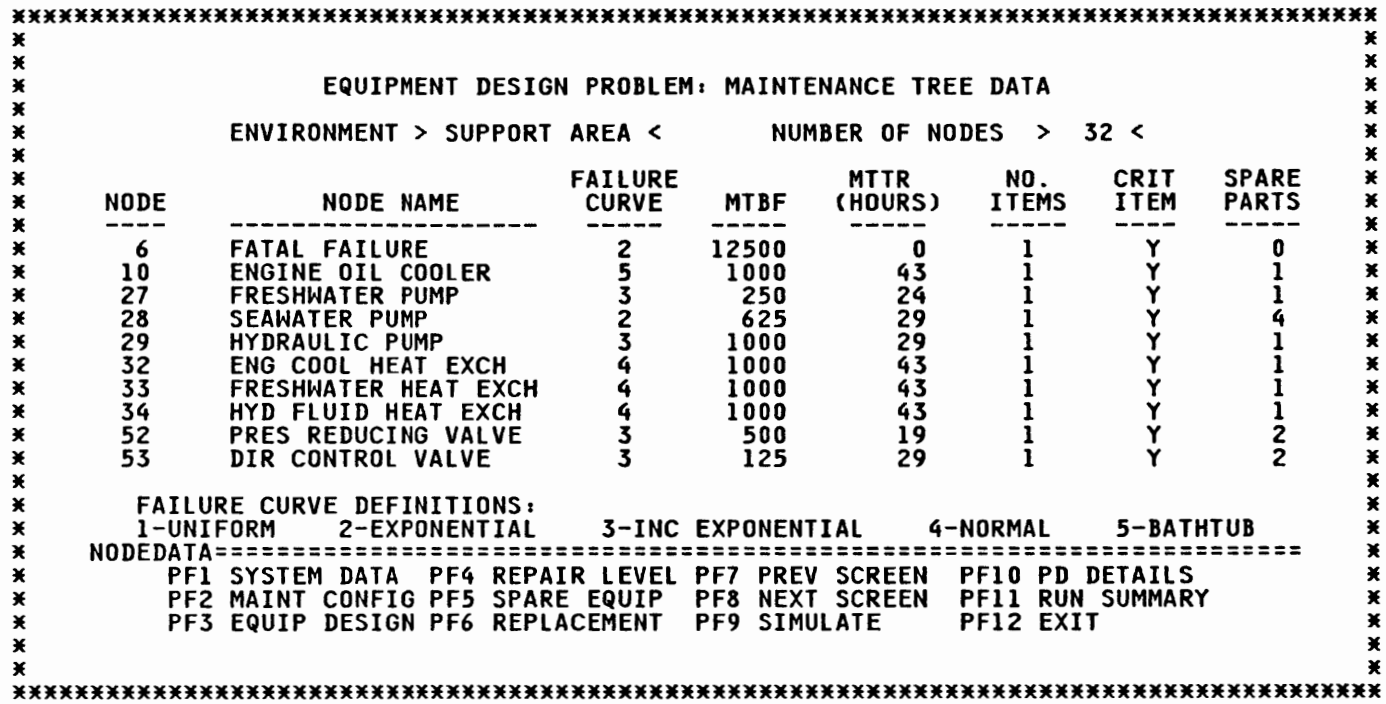


Figure 15. SLWT Maintenance System Configuration



EQUIPMENT DESIGN PROBLEM: MAINTENANCE TREE DATA

ENVIRONMENT > SUPPORT AREA < NUMBER OF NODES > 32 <

NODE	NODE NAME	FAILURE CURVE	MTBF	MTTR (HOURS)	NO. ITEMS	CRIT ITEM	SPARE PARTS
6	FATAL FAILURE	2	12500	0	1	Y	0
10	ENGINE OIL COOLER	5	1000	43	1	Y	1
27	FRESHWATER PUMP	3	250	24	1	Y	1
28	SEAWATER PUMP	2	625	29	1	Y	4
29	HYDRAULIC PUMP	3	1000	29	1	Y	1
32	ENG COOL HEAT EXCH	4	1000	43	1	Y	1
33	FRESHWATER HEAT EXCH	4	1000	43	1	Y	1
34	HYD FLUID HEAT EXCH	4	1000	43	1	Y	1
52	PRES REDUCING VALVE	3	500	19	1	Y	2
53	DIR CONTROL VALVE	3	125	29	1	Y	2

FAILURE CURVE DEFINITIONS:
1-UNIFORM 2-EXPONENTIAL 3-INC EXPONENTIAL 4-NORMAL 5-BATHTUB

NODEDATA==========

PF1 SYSTEM DATA PF4 REPAIR LEVEL PF7 PREV SCREEN PF10 PD DETAILS
PF2 MAINT CONFIG PF5 SPARE EQUIP PF8 NEXT SCREEN PF11 RUN SUMMARY
PF3 EQUIP DESIGN PF6 REPLACEMENT PF9 SIMULATE PF12 EXIT

Figure 16. SLWT Maintenance Tree Data

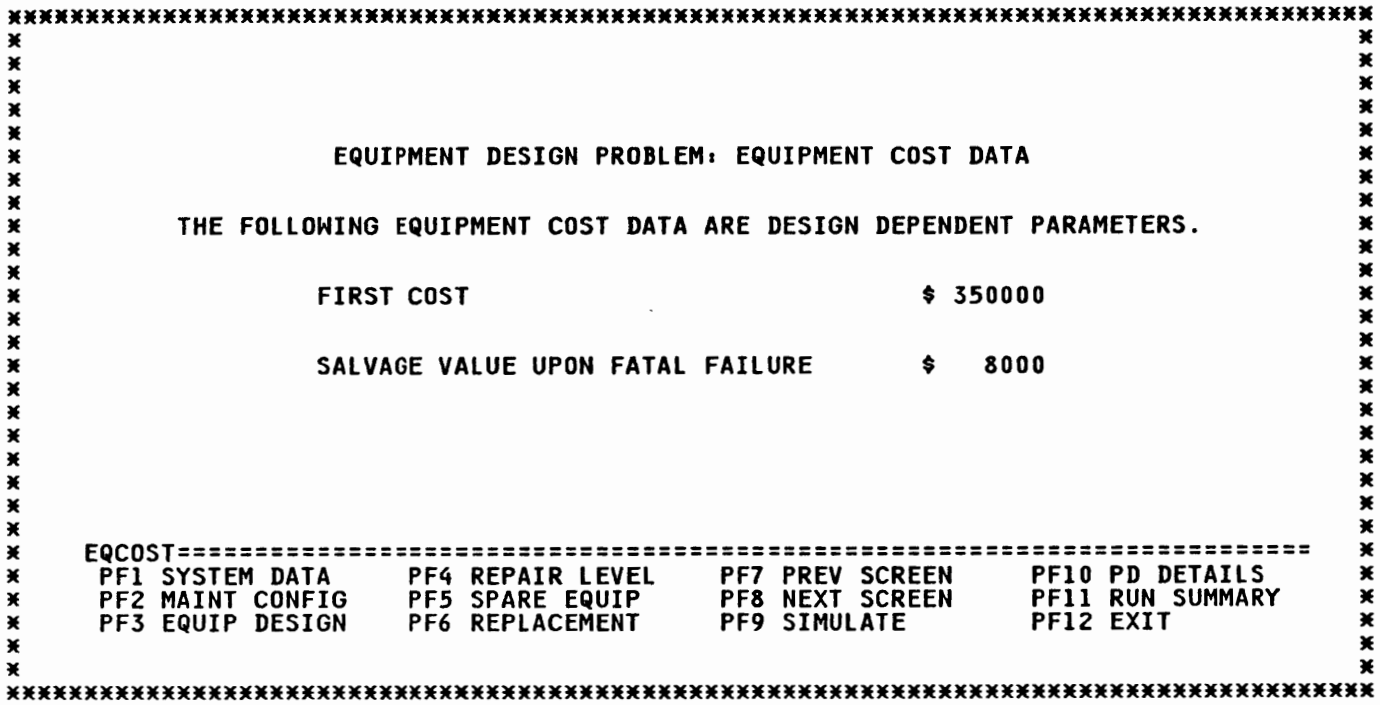


Figure 17. SLWT Cost Data

LEVEL OF REPAIR DATA

NODE	NODE NAME	REPAIR LEVEL	LABOR HOURS
6	FATAL FAILURE	1	0
10	ENGINE OIL COOLER	1	11
27	FRESHWATER PUMP	2	6
28	SEAWATER PUMP	2	7
29	HYDRAULIC PUMP	1	7
32	ENG COOL HEAT EXCH	2	11
33	FRESHWATER HEAT EXCH	1	11
34	HYD FLUID HEAT EXCH	1	11
52	PRES REDUCING VALVE	1	5
53	DIR CONTROL VALVE	1	7

LORDATA

PF1 SYSTEM DATA PF4 REPAIR LEVEL PF7 PREV SCREEN PF10 PD DETAILS
PF2 MAINT CONFIG PF5 SPARE EQUIP PF8 NEXT SCREEN PF11 RUN SUMMARY
PF3 EQUIP DESIGN PF6 REPLACEMENT PF9 SIMULATE PF12 EXIT

Figure 18. SLWT Level of Repair Data

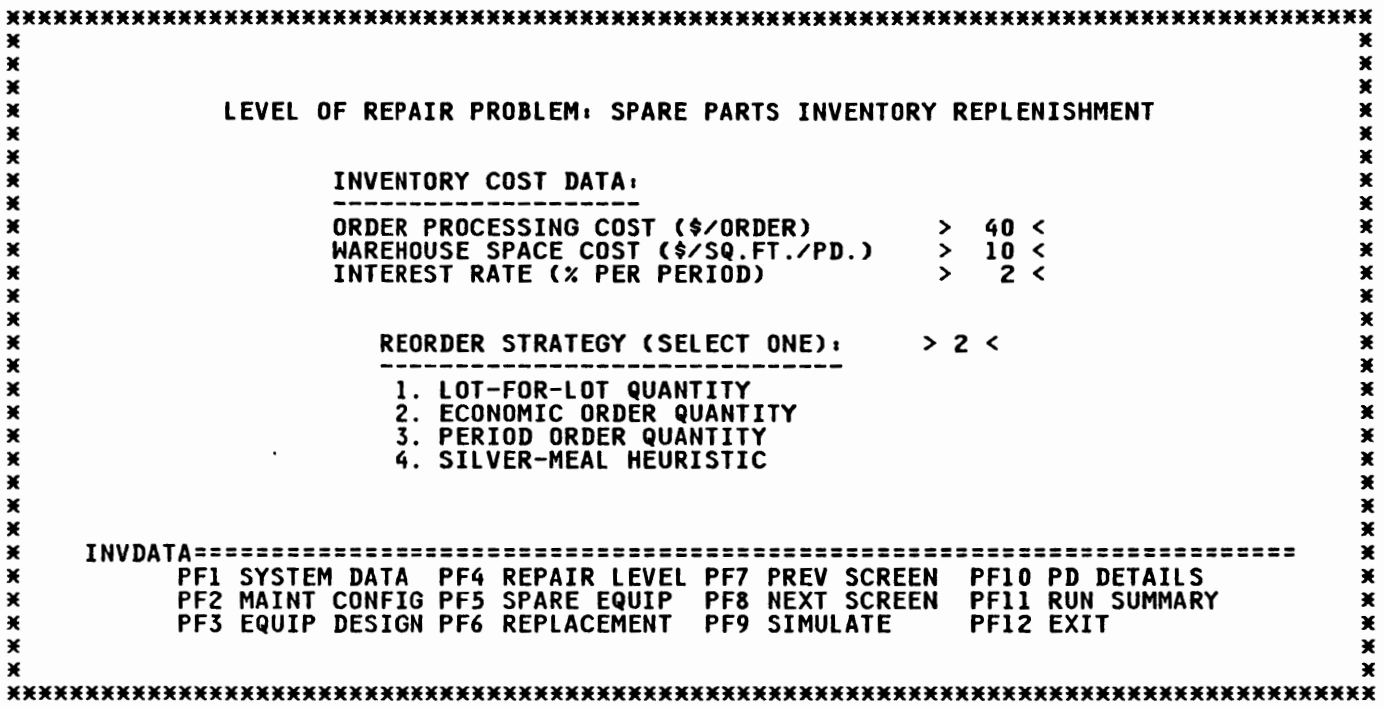
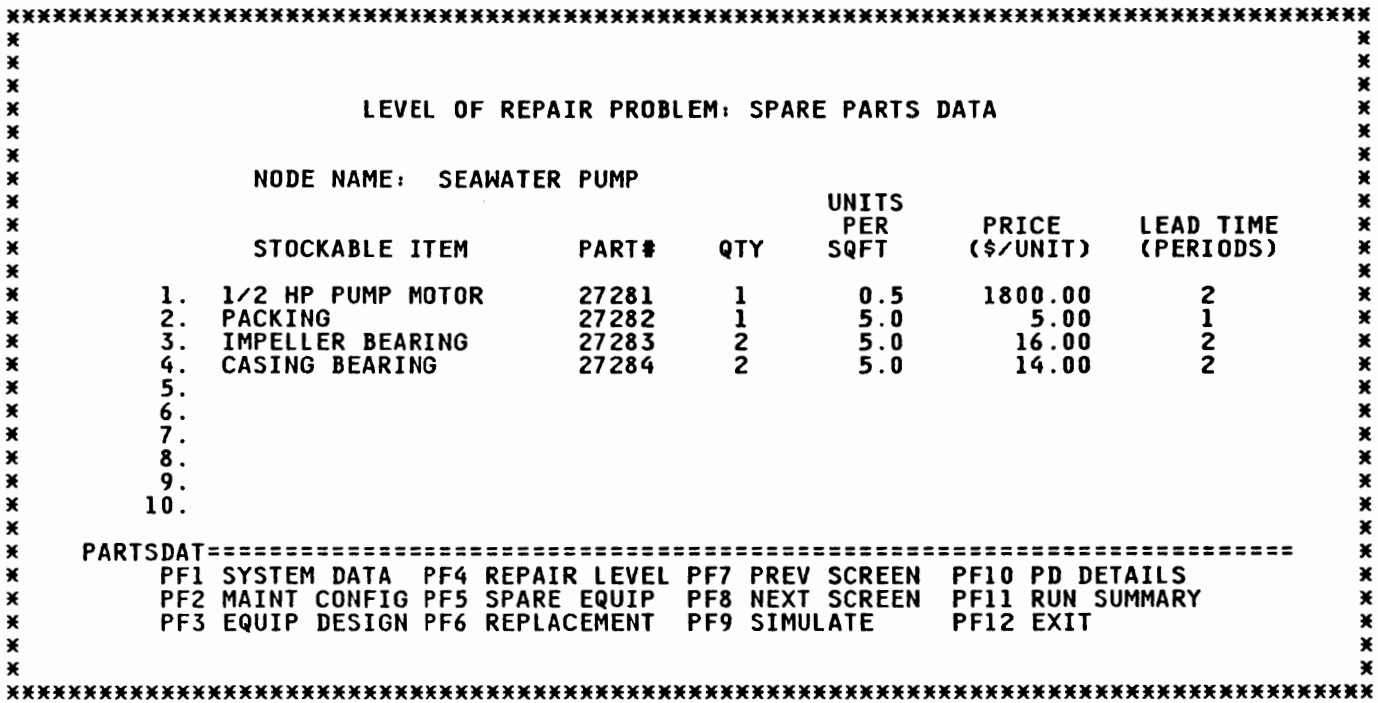


Figure 19. SLWT Spare Parts Inventory Policy



LEVEL OF REPAIR PROBLEM: SPARE PARTS DATA

NODE NAME: SEAWATER PUMP

	STOCKABLE ITEM	PART#	QTY	UNITS PER SQFT	PRICE ($/UNIT)	LEAD TIME (PERIODS)
1.	1/2 HP PUMP MOTOR	27281	1	0.5	1800.00	2
2.	PACKING	27282	1	5.0	5.00	1
3.	IMPELLER BEARING	27283	2	5.0	16.00	2
4.	CASING BEARING	27284	2	5.0	14.00	2
5.						
6.						
7.						
8.						
9.						
10.						

PARTSDAT==========

PF1 SYSTEM DATA PF4 REPAIR LEVEL PF7 PREV SCREEN PF10 PD DETAILS
PF2 MAINT CONFIG PF5 SPARE EQUIP PF8 NEXT SCREEN PF11 RUN SUMMARY
PF3 EQUIP DESIGN PF6 REPLACEMENT PF9 SIMULATE PF12 EXIT

Figure 20. SLWT Spare Parts Data

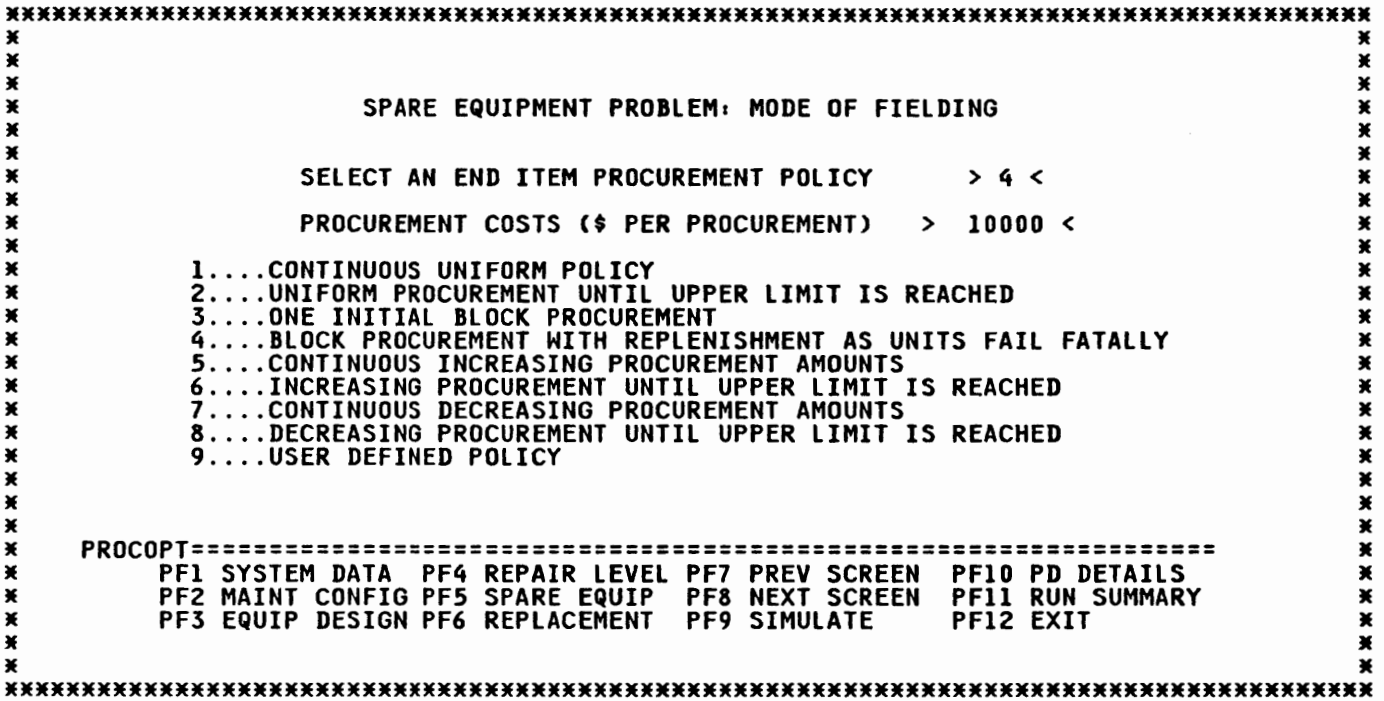


Figure 21. SLWT Procurement Policy Selection

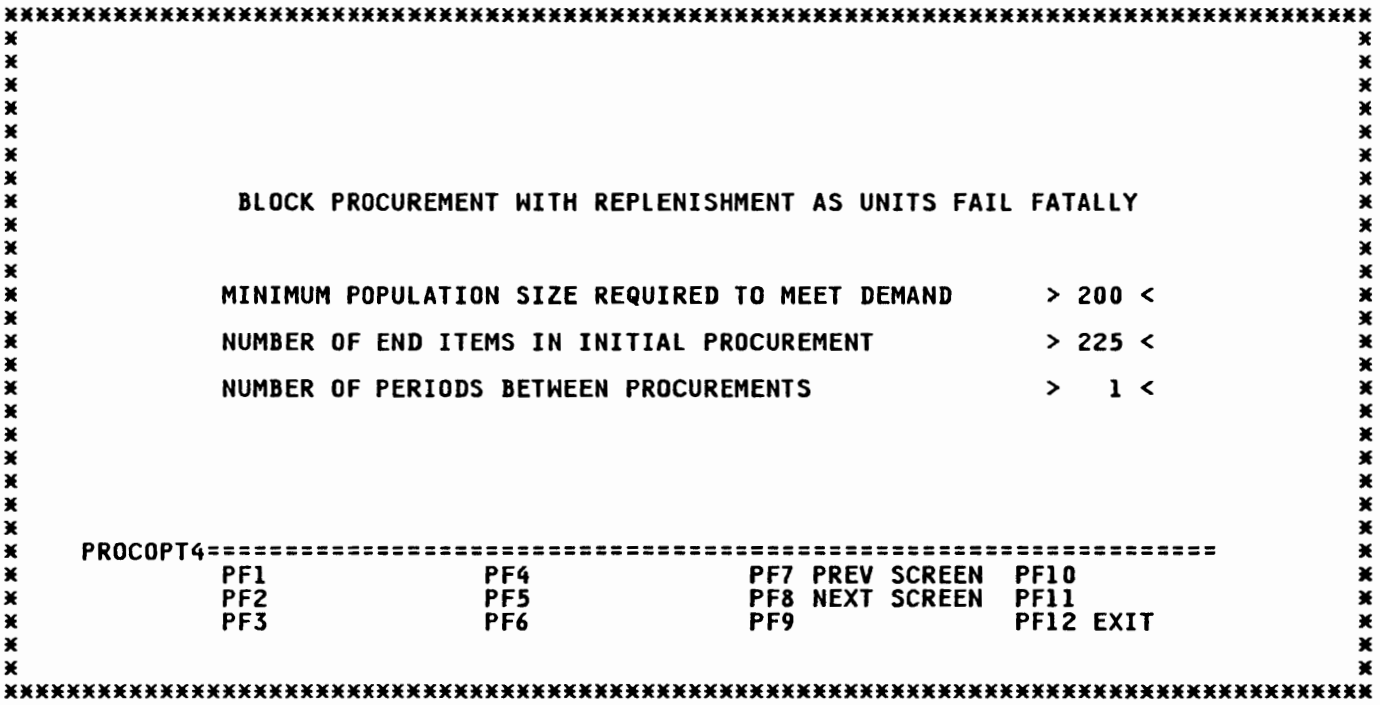


Figure 22. SLWT Procurement Policy Specification

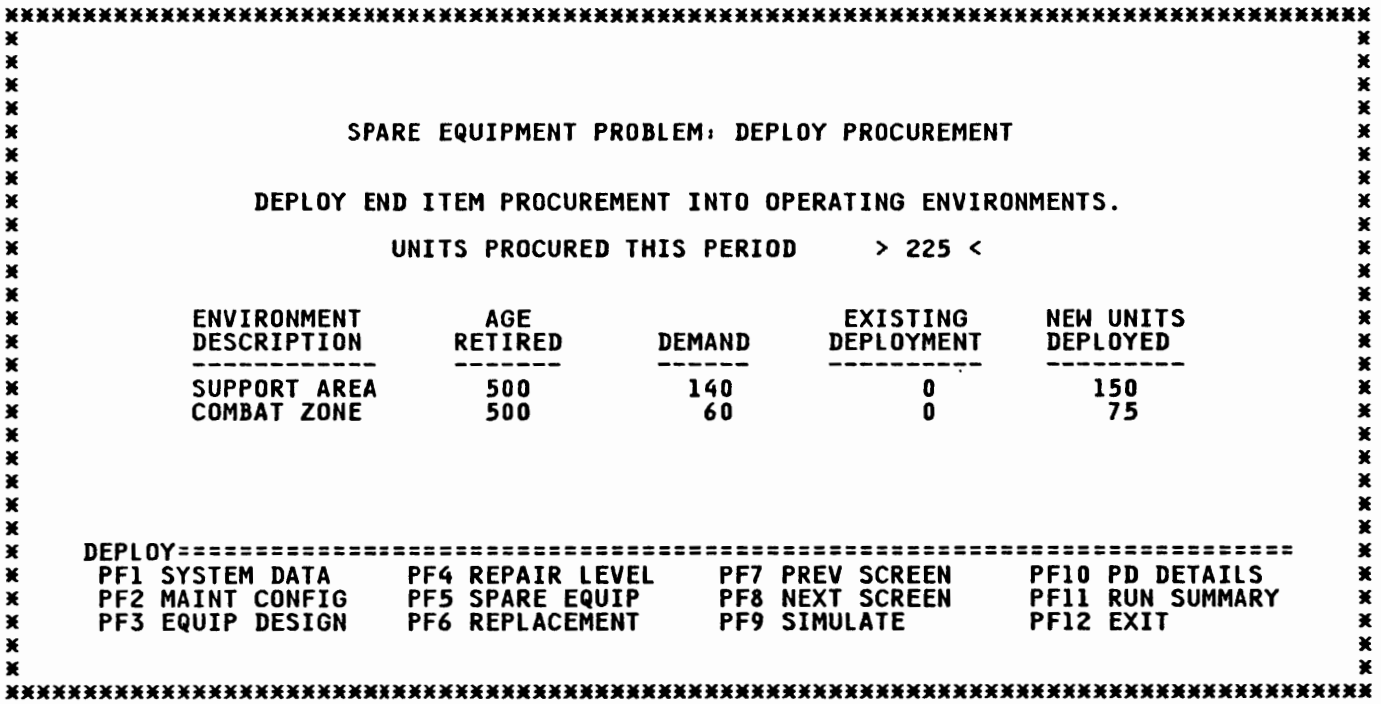
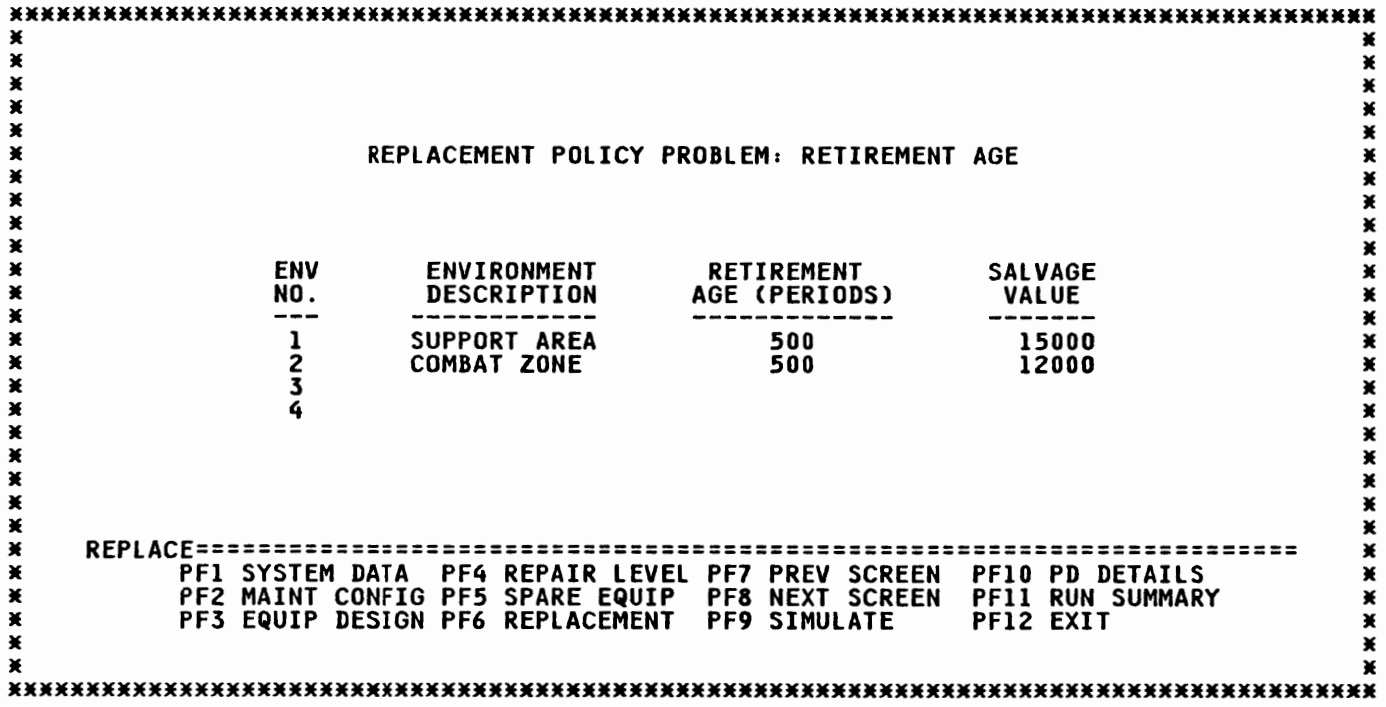


Figure 23. SLWT Deployment



REPLACEMENT POLICY PROBLEM: RETIREMENT AGE

ENV NO.	ENVIRONMENT DESCRIPTION	RETIREMENT AGE (PERIODS)	SALVAGE VALUE
1	SUPPORT AREA	500	15000
2	COMBAT ZONE	500	12000
3			
4			

REPLACE

PF1 SYSTEM DATA PF4 REPAIR LEVEL PF7 PREV SCREEN PF10 PD DETAILS
PF2 MAINT CONFIG PF5 SPARE EQUIP PF8 NEXT SCREEN PF11 RUN SUMMARY
PF3 EQUIP DESIGN PF6 REPLACEMENT PF9 SIMULATE PF12 EXIT

Figure 24. SLWT Retirement Age

4.1.2 Design Simulation

The simulation of system (the repairable item and its logistics support system) operation can be illustrated with some representative calculations. The simulation flow diagram shown in Figure 10 on page 93 illustrates the sequence of these calculations. After specification of the initial values for decision variables and design parameters, one stream is created for items deployed in each operating environment. Therefore, there are two streams in this scenario initially.

Time is advanced by one period and expected failures for each bottom-of-the-tree node are computed. For example, in the support environment, Node 28 - Seawater Pump has a MTBF = 625 days = 20.83 periods. The failure rate, $\lambda = 0.0480$, is 1/MTBF. The specified failure curve is the exponential distribution. Therefore, the cumulative failure probability after one period is:

$$F_x(t) = 1 - \frac{1}{e^{\lambda t}}$$

$$F_x(1) = 1 - \frac{1}{e^{(0.0480)(1)}}$$
$$= 0.0469$$

The expected number of Node 28 failures in the support environment is:

$$Failures = (0.0469)(150) = 7.0$$

Similiarly, the expected number of Node 28 failures in the combat environment can be determined to be 5 failures. Therefore, the total demand for logistics support resources, associated with Node 28 failures, is 12 failures. These failures are repaired at level two in the maintenance and repair system. The spare parts needed to repair Node 28 failures are given in Figure 20 on page 106. Note that two each of the impeller bearing and casing bearing are required to repair the seawater pump. Therefore, the demand for these spare parts is two times the number of failures. The MTTR = 29 hours and seven labor hours are required to complete repair on each failure. Thus,

the transportation time to the repair facility and queueing delays after arrival at the repair facility total 29-7 = 22 hours. There are five repair channels at the 2nd level of maintenance. Therefore, the expected demand at each channel is 12/5 = 2.4 items. Repaired items are returned to their respective streams and are assumed to be available at the beginning of the next period.

Node 6 represents the probability of fatal failure. Using the MTBF and failure curve specified for each environment, the probability of fatal failure in period one can be determined. In the support area environment, the probability of fatal failure is 0.0023. Therefore, $(0.0023) \times (150) = 0.36 \cong 0$ items are permanently lost from the initial support environment stream. Similarly, the probability of fatal failure in the combat area environment is 0.0487. Therefore, $(0.0488) \times (75) = 3.65 \cong 4$ items are permanently lost from the initial combat environment stream. The number of items in the combat area stream is thereby reduced to 71 items.

The procurement policy specified was block procurement with replenishment as units fail fatally. Therefore, four new SLWT's are to be procured. These will all be deployed in the combat area environment where the fatal failures occurred. Thus, only one new stream will be created. The retirement age specified is greater than one period, so no streams are due to be retired. The population of SLWT's now contains three streams. Time is advanced to the next period and the preceding calculations are repeated. A failure summary for the initial two streams over twelve simulated periods is shown in Table 2 and Table 3.

Each stream represents a subset of the total repairable item population. All repairable items in a given stream are identically aged and exhibit identical failure and repair characteristics (if fielded in the same operating environment). The individual node failures and the fatal failures for each stream are summed across all streams to determine the total number of failures for the population. Initially, there will be only one stream for each operating environment. However, as new repairable items are purchased and fielded, the number of streams grows.

Once the expected number of failures is known, the demand for spare parts and other logistics support resources can be determined. The failure summaries for two streams during the twelve periods simulated are shown in Table 2 and Table 3. The failures associated with other streams created during the simulation must be added to these to determine the total number of Node 28 failures for each time period. Assume such a tally produced the demand profile shown in Table 4 for impeller bearings. An impeller bearing is a spare part required to repair a seawater pump failure.

The replenishment lead time specified for impeller bearings is two periods. An initial inventory of one lead time period is assumed. Therefore, based upon Period 1 and Period 2 demands, the initial inventory will be 48 parts. The average demand per period for these twelve periods is 17.2 units. For an EOQ replenishment policy, the optimal order quantity is:

$$Q_{jt} = \sqrt{\frac{2 \times PC \times \overline{D}_j}{HC_j}}$$
$$= \sqrt{\frac{2 \times 40 \times 17.2}{2.32}}$$
$$= 24$$

The holding cost per period, $HC_j = 2.32$, is:

$$HC_j = i \times C_j + \frac{\$/ft^2/pd}{units/ft^2}$$
$$= 0.02 \times 16 + \frac{10}{5}$$
$$= 2.32$$

Table 2. Initial Support Area Environment Stream

Period	Fatal Failure Probability	No. of Items In Stream	No. of Fatal Failures	Node 28 Failure Probability	No. of Node 28 Failures
1	0.0023	150	0	0.0468	7.0
2	0.0023	150	0	0.0446	6.7
3	0.0023	150	0	0.0425	6.4
4	0.0023	150	0	0.0405	6.1
5	0.0023	150	0	0.0386	5.8
6	0.0023	150	0	0.0368	5.5
7	0.0023	150	0	0.0351	5.3
8	0.0023	150	0	0.0334	5.0
9	0.0023	150	0	0.0319	4.8
10	0.0023	150	0	0.0304	4.6
11	0.0023	150	0	0.0290	4.4
12	0.0023	150	0	0.0276	4.1

Table 3. Initial Combat Area Environment Stream

Period	Fatal Failure Probability	No. of Items In Stream	No. of Fatal Failures	Node 28 Failure Probability	No. of Node 28 Failures
1	0.0487	75	4	0.0694	5.2
2	0.0463	71	3	0.0645	4.6
3	0.0441	68	3	0.0601	4.1
4	0.0419	65	3	0.0559	3.6
5	0.0399	62	2	0.0520	3.2
6	0.0379	60	2	0.0484	2.9
7	0.0361	58	2	0.0450	2.6
8	0.0343	56	2	0.0419	2.3
9	0.0326	54	2	0.0390	2.1
10	0.0311	52	2	0.0363	1.9
11	0.0295	50	1	0.0338	1.7
12	0.0281	49	1	0.0314	1.5

Table 4. Impeller Bearing Demand

Period	Demand	Period	Demand	Period	Demand
1	24	5	18	9	14
2	24	6	18	10	14
3	20	7	16	11	12
4	20	8	14	12	12

Therefore the average impeller bearing inventory is $\bar{I}_{jt} = \frac{Q_{jt}}{2} = 12$ bearings. The total inventory costs, TC_j, for this part are:

$$\begin{aligned} TC_j &= (C_j \times D_j) + \frac{D_j}{Q_j} \times PC + \frac{Q_j}{2} \times HC_j \\ &= 16 \times 206 + \frac{206}{24} \times 40 + \frac{24}{2} \times 2.32 \times 12 \\ &= 3973.41 \end{aligned}$$

Once the number and type of failures per period are known, calculation of resource requirements such as labor and warehouse space are relatively straight forward. Labor requirements are simply the product of number of failures and the labor hours per repair, summed for all nonfatal failures. For example, the labor requirements for each Node 28, Seawater Pump repair, is given in Figure 18 on page 104 as seven hours. Recalling from Figure 20 on page 106 that two impeller bearings are required to repair each seawater pump failure, from Table 4 it can be concluded that there were 18/2 or nine seawater pump failures in Period 5. Therefore, the labor requirements for Node 28 Seawater Pump failures in Period 5 are:

$$7 \text{ hours/failure} \times 9 \text{ failures} = 56 \text{ labor hours}$$

This same calculation would be repeated for each nonfatal failure node in the maintenance tree and labor requirements summed to determine total period labor requirements.

Similarly, warehouse space requirements are the average inventory for a given spare part divided by the warehouse space required per part, summed for all spare parts. For impeller bearings, the warehouse space requirements are:

$$\frac{12 \text{ bearings}}{5 \text{ bearings / ft}^2} = 2.4 \text{ ft}^2$$

This same calculation would be repeated for each spare part and summed to determine total warehouse space requirements.

4.1.3 Design Evaluation

System effectiveness is a multicriteria measure which reflects the availability, reliability, maintainability, and life cycle cost of a given system design. The calculation of these measures was discussed in Section 3.4. As an example, the calculations required for the availability measure are illustrated here. Availability is the probability that a system will be ready or available when required for use. It is a measure of the system's ability to satisfy demands. At a given time, a single repairable item is either operational and available to satisfy demands or under repair and not available to satisfy demands. Thus as described in Section 3.4.1, the availability measure is based upon the binomial probability distribution.

This approach requires that the probability of failure for an end item during a typical repair cycle be estimated. The average time spent in repair depends upon the frequency and duration of each

repair. These depend upon the age of the items and the operating environments where items are deployed. Calculation of the average time spent in repair begins with calculation of the average repair times for each stream. These streams are then aggregated into operating environments. The average time spent in repair by items in stream s during period t, $\overline{MTTR_{st}}$, is:

$$\overline{MTTR_{st}} = \frac{\sum_{i=1}^{B} [MTTR_i \times NODE_{it}]}{B}$$

As shown in Figure 12 on page 98, there are 31 bottom of the tree nodes in the SLWT maintenance tree. The MTTR for each of these nodes and the probability of failure after one period of operation are shown in Table 5. Application of the equation for $\overline{MTTR_{st}}$ to the data in Table 5 yields $\overline{MTTR_{s1}} = 10.84 \cong 11$. This stream, Stream 1, is the initial support area fielding. Therefore, it is the only stream in the support area environment during period one. The average time spent in repair by items in environment e during period t, $\overline{MTTR_{et}}$, is:

$$\overline{MTTR_{et}} = \frac{\sum_{s \in e} [\overline{N}_{st} \times \overline{MTTR_{st}}]}{\sum_{s \in e} \overline{N}_{st}}$$

Since there is only one stream in the support area environment initially:

$$\overline{MTTR_{e1}} = \frac{150 \times 11}{150} = 11$$

The average probability of failure for an end item in environment e during period t, $P(f)_{et}$, is:

$$P(f)_{et} = \frac{\sum_{s \in e} [\overline{N}_{st} \times NODE_{1st}]}{N_{et}}$$

Therefore, for the support area environment, in Period 1, this probability is:

$$P(f)_{et} = \frac{150 \times 0.3104}{150} = 0.3104$$

This represents the probability of failure during the entire time period (30 days). However, the population is unable to satisfy operating environment demands only when the number of operational (not in repair) items is less than the demand. This can occur only when a significant number of items fail while others are in the repair cycle. The probability of failure for an end item in environment e during a period of time equal to $\overline{MTTR_{et}}$ (the repair cycle) is:

$$P(f)'_{et} = [\frac{MTTR_{et}}{LTP}] \times P(f)_{et}$$

Therefore,

$$P(f)'_{et} = \frac{11}{30} \times 0.3104 = 0.1138$$

The availability in environment e during period t is:

$$A_{et} = \sum_{x=0}^{k} b(x; n,p)$$

Table 5. Initial Support Area Stream Repair Time For Period 1

NODE	MTTR	Failure Probability	Expected Repair Time
10	43	0	0
27	24	0.0001	0
28	29	0.0468	1.36
29	29	0	0
32	43	0	0
33	43	0	0
34	43	0	0
52	19	0	0
53	29	0.0100	0.29
54	43	0	0
58	7	0.0532	0.37
59	48	0	0
62	24	0.0234	0.56
63	58	0.0508	2.95
64	360	0	0
65	29	0.0508	1.47
92	288	0	0
93	264	0.0032	0.84
94	288	0	0
107	19	0	0
108	24	0.0234	0.56
112	12	0.0172	0.21
113	3	0.0100	0.03
114	58	0.0172	1.00
118	19	0.0367	0.70
119	38	0.0100	0.38
120	12	0.0006	0.01
123	3	0.0032	0.01
124	24	0.0015	0.04
125	19	0.0015	0.03
126	19	0.0015	0.03

Therefore:

$$
\begin{aligned}
A_{e1} &= \sum_{x=0}^{10} b(x;\, 150,\, 0.1138) \\
&= \sum_{x=0}^{10} \binom{150}{x} 0.1138^{x} 0.8862^{150-x} \\
&= 99\%
\end{aligned}
$$

Therefore, the availability in the support area environment in Period 1 is 99%. Similar calculations must also be performed in every other operating environment. In this case there is just one additional environment, the combat area environment. These can then be aggregated to determine population availability, A_t. If the availability in the combat area environment is 97% with a fielding of 75 items, the population availability is:

$$
\begin{aligned}
A_t &= \frac{\sum_{e} A_{et} \times N_{et}}{N_t} \\
&= \frac{(99 \times 150) + (97 \times 75)}{225} \\
&= 98.3\%
\end{aligned}
$$

The reliability measure is more easily computed than the availability measure. Once again, the reliability of the items deployed in each operating environment is calculated and then aggregated to determine the population reliability. The probability of failure for an item fielded in the support area environment during the first period is, $NODE_1 = 0.3104$. Therefore, the reliability for a single repairable item fielded in this environment is:

$$
\begin{aligned}
R_{1t} &= 1 - \sum_{t=1}^{1} NODE_{1t} \\
&= 1 - 0.3104 \\
&= 69\%
\end{aligned}
$$

The reliability for the entire population of end items after T periods of operation is the weighted average of the reliability in each environment. This is given by R_t, where:

$$R_1 = \frac{\sum_e R_{et} \times N_{et}}{N_t}$$
$$= \frac{(0.6896 \times 150) + (0.3270 \times 75)}{225}$$
$$= 56.9\%$$

Maintainability is measured using the average time in repair, $\overline{MTTR_{st}}$. The average time in repair for all items in the population, $\overline{MTTR_t}$, is the weighted average of the repair times for each stream. In Period 1, there are only two streams. Therefore, the average repair time in each environment will be the same as the average repair time for each stream. Thus, $\overline{MTTR_{et}} = \overline{MTTR_{st}}$. The $\overline{MTTR_{st}}$ for the support area environment stream was previously calculated as 11 (see availability calculations in this section). If the $\overline{MTTR_{st}}$ for the combat area environment stream is 32, then the $\overline{MTTR_t}$ is:

$$\overline{MTTR_t} = \frac{\sum_e \overline{N}_{et} \times \overline{MTTR_{et}}}{\sum_e \overline{N}_{et}}$$
$$= \frac{(11 \times 150) + (32 \times 75)}{225}$$
$$= 18 \text{ hours}$$

The life cycle cost includes capital costs, repair costs, inventory costs, and procurement costs. During the first period, 150 items were procured and fielded in the support area environment. None of these failed fatally or were retired during the first period. Therefore, the capital costs incurred during Period 1 in the support area environment are:

$$CC_e = (NP_e \times FC) - (NF_e \times SV_e) - (NR_e \times BV_e)$$
$$= (150 \times \$350{,}000) - (0 \times \$8{,}000) - (0 \times \$15{,}000)$$
$$= \$52.5 \text{ million}$$

The repair cost includes both the fixed and variable costs of repair. Since this cost per unit time varies based upon the assigned level of repair, this cost must be calculated for all bottom of the tree nodes (nodes that incurr repairable failures) and then aggregated to the environment level. The repair cost in each environment is given as:

$$RC_{et} = \sum_{s \in e} \overline{N}_{st} \times [\sum_{i}^{B} NODE_{it} \times MTTR_i \times RC_i]$$

Therefore, each of the expected repair times given in Table 5 must be multiplied by the RC_i for that repair and then summed to get the expected repair cost for each stream. For example, the engine oil cooler, $NODE_{11}$, is repaired at the first level of repair. Thus, $RC_{11} = \$1{,}000$ per day. The cost for each stream is then multiplied by the number of items in that stream. The costs for all streams belonging to a given environment ($s \in e$), are then summed to get the repair cost for the environment. For the support area environment, there is only one stream in Period 1. Therefore,

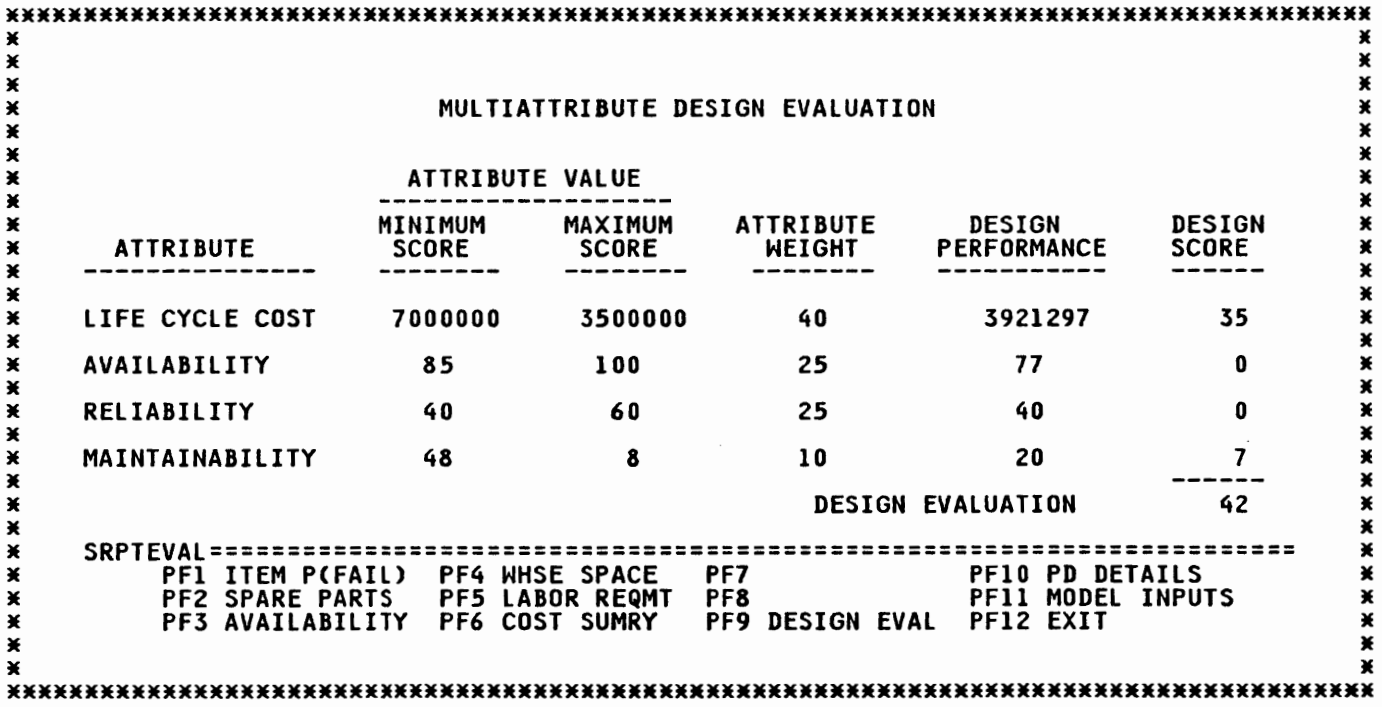
$$RC_{et} = 150 \times [\$2520.71] = \$378{,}106$$

The calculations required to determine inventory cost were previously illustrated in this section. These would need to be repeated for every spare part. The only remaining cost is the procurement cost associated with fielding new end items. This cost is incurred whenever end items are procured. In Period 1 for example, 225 end items were procured and fielded. The procurement cost is independent of the number of items procured. In Figure 21 on page 107, the procurement cost is given as \$10,000 per procurement. Since there was a procurement in Period 1, the procurement cost this period is \$10,000.

4.1.4 Design Optimization

Once all four measures of system effectiveness have been computed, they are combined into a single, composite measure of system effectiveness. The calculations required to accomplish this were illustrated in Section 3.4.5. The system designer can now search for ways to improve the initial design. To demonstrate how CODAS can be used to experimentally improve a system design, operation of the initial design specified in Section 4.1.1 was simulated for twelve periods. This simulation process was illustrated in Section 4.1.2 for a typical stream. At the end of the simulation, the four measures of system effectiveness are computed and combined into a single, composite measure of system effectiveness. The Multiattribute Design Evaluation Report shown in Figure 25 on page 125, allows the designer to identify opportunities for design improvement.

The initial design specification (Iteration 0) did not perform well with respect to the availability and reliability measures. Given the weights assigned these measures, the measure of system effectiveness may be improved significantly, if the design can be enhanced to affect performance in these two areas. An investigation of the population availability in each operating environment on a period-by-period basis (see Figure 26 on page 126) shows that the availability in the Combat Area environment (Env 2) is the problem. This can be improved by changing the initial fielding to deploy additional items in the Combat Area environment. If this fielding is changed (Iteration 1) such that 145 SLWT's are assigned to the Support Area environment and 85 SLWT's are assigned to the Combat Area environment, availability should be improved. However, Figure 27 on page 127 shows only a small improvement in the design evaluation score. While availability improved significantly, the other three measures of performance deteriorated. This deterioration is because a larger number of items are deployed in a high failure rate environment. The failures in this environment are also more severe, taking longer and costing more to repair. This illustrates the interdependency between the design decision problems and the cost versus performance tradeoffs which must be addressed during the design process.



MULTIATTRIBUTE DESIGN EVALUATION

ATTRIBUTE	ATTRIBUTE VALUE MINIMUM SCORE	ATTRIBUTE VALUE MAXIMUM SCORE	ATTRIBUTE WEIGHT	DESIGN PERFORMANCE	DESIGN SCORE
LIFE CYCLE COST	7000000	3500000	40	3921297	35
AVAILABILITY	85	100	25	77	0
RELIABILITY	40	60	25	40	0
MAINTAINABILITY	48	8	10	20	7
				DESIGN EVALUATION	42

SRPTEVAL

PF1 ITEM P(FAIL) PF4 WHSE SPACE PF7 PF10 PD DETAILS
PF2 SPARE PARTS PF5 LABOR REQMT PF8 PF11 MODEL INPUTS
PF3 AVAILABILITY PF6 COST SUMRY PF9 DESIGN EVAL PF12 EXIT

Figure 25. Initial Design: Multiattribute Design Evaluation

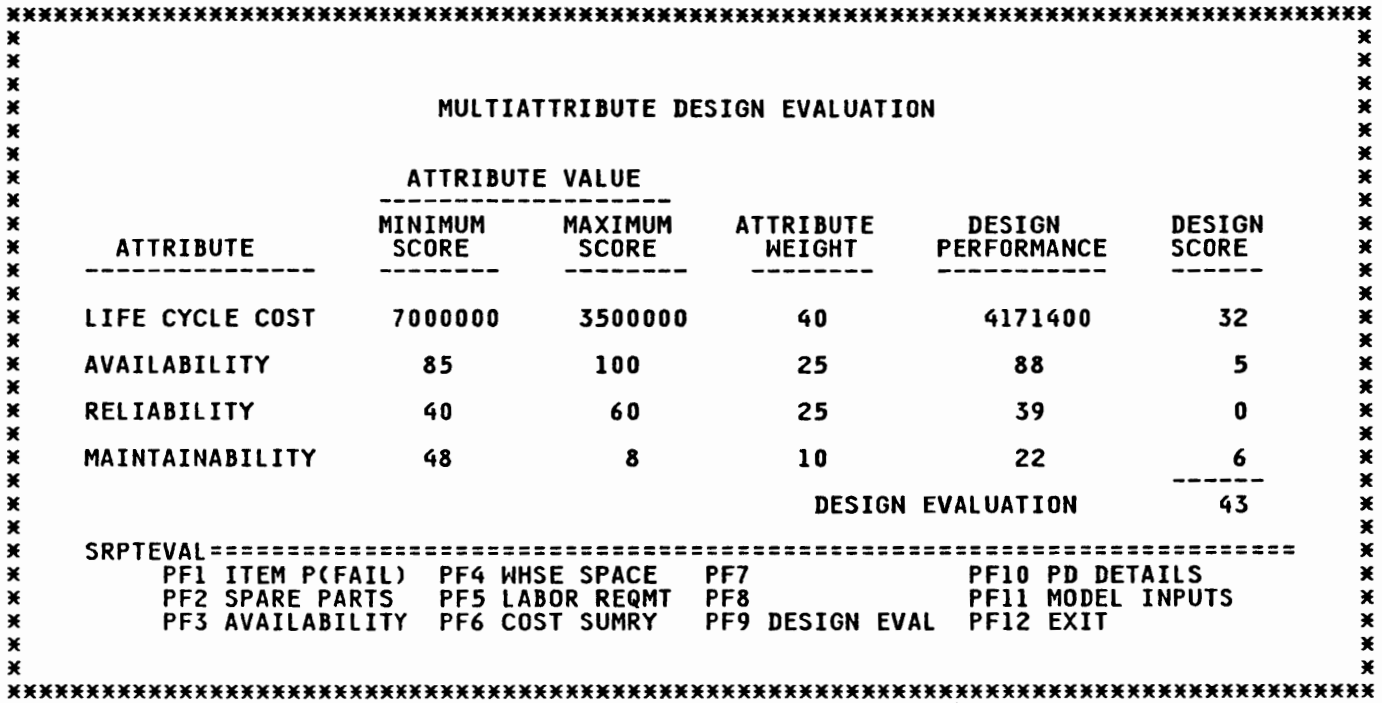
```
                    END ITEM AVAILABILITY

                  AVAILABILITY BY ENVIRONMENT

  PERIOD    ENV 1     ENV 2     ENV 3     ENV 4     SYSTEM
  ------    -----     -----     -----     -----     ------
     1        99        97                            99
     2        99        78                            93
     3        99        21                            75
     4        99        15                            74
     5        97        42                            81
     6        99        12                            75
     7        96         0                            69
     8        99         2                            72
     9        99         0                            72
    10        99         0                            73

SRPTAVAI=========================================================
  PF1 ITEM P(FAIL)  PF4 WHSE SPACE   PF7 PREV SCREEN  PF10 PD DETAIL
  PF2 SPARE PARTS   PF5 LABOR REQMT  PF8 NEXT SCREEN  PF11 MODEL INPUTS
  PF3 AVAILABILITY  PF6 COST SUMRY   PF9 DESIGN EVAL  PF12 EXIT
```

Figure 26. Initial Design: Summary Availability Report



MULTIATTRIBUTE DESIGN EVALUATION

ATTRIBUTE	ATTRIBUTE VALUE: MINIMUM SCORE	ATTRIBUTE VALUE: MAXIMUM SCORE	ATTRIBUTE WEIGHT	DESIGN PERFORMANCE	DESIGN SCORE
LIFE CYCLE COST	7000000	3500000	40	4171400	32
AVAILABILITY	85	100	25	88	5
RELIABILITY	40	60	25	39	0
MAINTAINABILITY	48	8	10	22	6
				DESIGN EVALUATION	43

SRPTEVAL

PF1 ITEM P(FAIL) PF4 WHSE SPACE PF7 PF10 PD DETAILS
PF2 SPARE PARTS PF5 LABOR REQMT PF8 PF11 MODEL INPUTS
PF3 AVAILABILITY PF6 COST SUMRY PF9 DESIGN EVAL PF12 EXIT

Figure 27. Design Iteration 1: Multiattribute Design Evaluation

An investigation of the Period Labor Requirements Report shows there is not sufficient work at the second level of repair to justify five repair channels. If on the next design iteration (Iteration 2), the number of channels are reduced from five to two, there should be a negligible increase in repair time because of the gross excess in capacity which exists. Simulation of this design change shows a 10.2% improvement in life-cycle costs with essentially no change in the other three measures, causing a five point improvement in the design evaluation score.

The reliability of the system design is dependent on the failure characteristics of the repairable item. One way to improve these failure characteristics is to use parallel redundant components. The Node Failure Probability Report shown in Figure 28 on page 129 allows the designer to identify such opportunities for improvement. For example, the probability of failure for nodes at the second level of indenture is highest at $NODE_3$, the hydraulic system. The failure probability at $NODE_3$ is dependent upon its "children" at the next lower level of indenture, Nodes 12-16. Of these nodes, the highest failure probability is at $NODE_{14}$, the hydraulic system starter. Nodes 62-66 are the "children" of $NODE_{14}$. The failure rate is highest for $Node_{63}$ (timer) and $Node_{65}$ (solenoid). On the next design iteration (Iteration 3) parallel redundant components are provided for each of these nodes. Doing so decreases the failure probability of $NODE_{14}$ to 0.0686. This decreases the failure probability of $NODE_3$ to 0.1270 and the failure probability of $NODE_1$ to 0.2699. The reliability measure for items in the support area environment is thus improved to 43%.

Alternatively, the system designer could specify a different vendor or higher quality components for the hydraulic system starter ($NODE_{14}$). This design iteration (Iteration 4), also produces the desired effect of reducing the failure probability and thus improving reliability. Many times, such decisions are situation dependent. For example, another vendor may not be available or a higher quality component does not exist. In some cases it may be possible to research and develop new components. The system designer is the best source of information on the feasibility of various alternatives. Therefore, the system designer must utilize his or her expertise to generate alternative system designs for evaluation. If the MTBF of the timer ($NODE_{63}$) and solenoid ($NODE_{65}$) can be improved by 25%, causing the cost of these components to double, reliability will be further im-

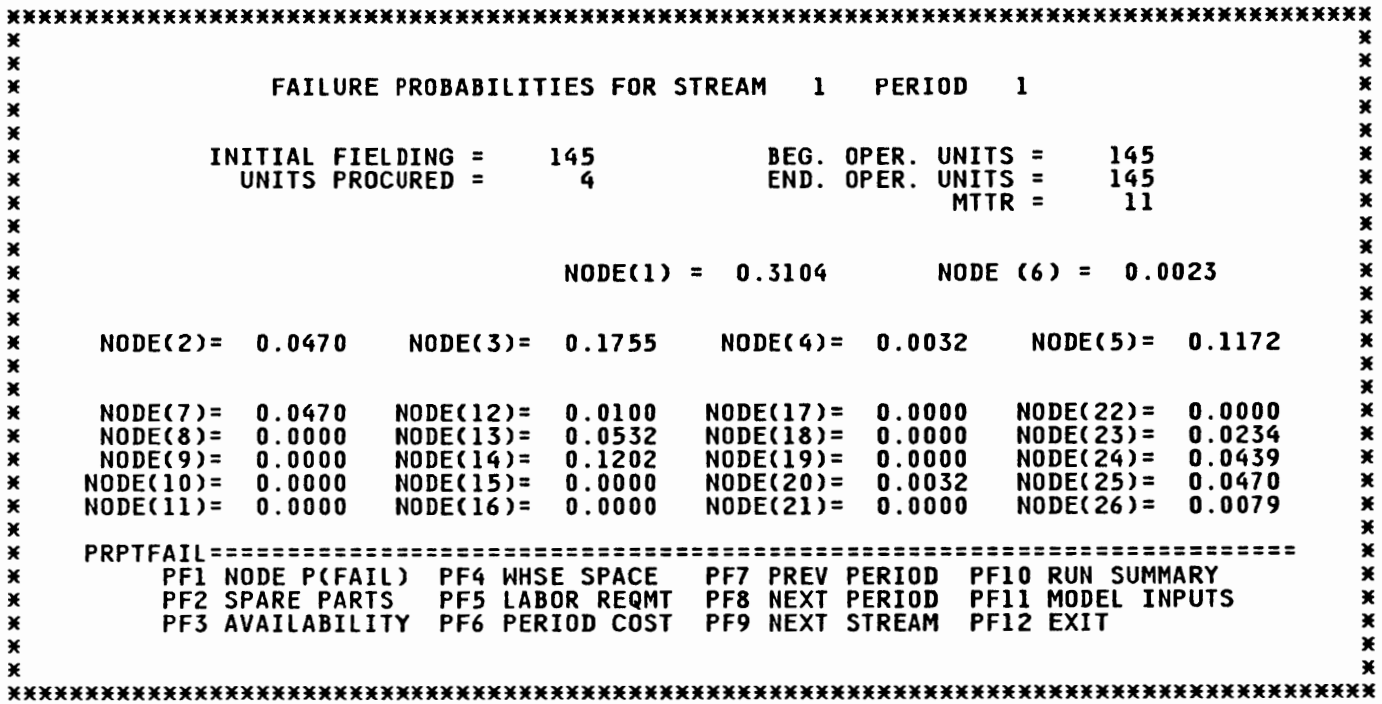


Figure 28. Design Iteration 2: Node Failure Probability Report

proved. When used in combination with parallel redundant components at $NODE_{58}$, reliability is increased to 49%. These design enhancements increase the initial cost of the SLWT. However, the improved reliability reduces life cycle costs, improves availability, and improves maintainability. This results in a thirteen point improvement in the design evaluation score.

The initial design would be repeatedly revised in the manner illustrated until the desired level of performance was achieved. The system designer would use the various reports generated by CODAS to identify additional opportunities for improvement. The following tables summarize the design iterations discussed previously in this section and the resulting change in the design evaluation measure.

Design Iteration Descriptions

- Iteration 0: Initial System Design
- Iteration 1: Modify Initial Fieldings
- Iteration 2: Reduce Number of Channels at Repair Level 2
- Iteration 3: Use Two Parallel Redundant Components at $NODE_{63}$ and $NODE_{65}$
- Iteration 4: Improve $NODE_{63}$ and $NODE_{65}$ MTBF by 25% and Use Three Parallel Redundant Components at $NODE_{58}$

Table 6. **Design Iteration Summary:** Contribution of each performance measure to the design evaluation score is given in parenthesis.

Iteration	Life-Cycle Cost	Availability	Reliability	Maintainability	Design Evaluation
0	3,921,297 (35)	77 (0)	40 (0)	20 (7)	42
1	4,171,400 (32)	88 (5)	39 (0)	22 (6)	43
2	3,745,418 (37)	88 (5)	39 (0)	22 (6)	48
3	3,635,147 (38)	94 (15)	43 (3)	17 (7)	63
4	3,584,683 (39)	96 (18)	49 (11)	16 (8)	76

The system designer must be sensitive to the impact of a design change to improve one measure of performance on the other measures of system performance. Each of the design changes suggested above will enhance performance for one or more measures of performance, sometimes at the expense of some other measure of performance. Depending upon the weighting given each of these measures, the composite measure of system performance may or may not be improved. The system designer's first task is to identify a feasible design alternative. Once found, the solution space of feasible design alternatives can then be searched to improve the composite measure of system performance. At this time, this search relies upon the expertise and creativity of the system designer. In that way, CODAS is like the early spreadsheet software for microcomputers. It provides a framework for the system designer to evaluate alternative designs (what if's). Addition of intelligence to guide the system designer to the best design alternative is an opportunity for future research.

5.0 Summary

In this chapter, the contibutions of this research are summarized, some strengths and weaknesses of the developed framework are critically analyzed, and a number of opportunities for future research are identified. This research effort represents only a first step towards effective, integrated design analysis. However, it does make a number of contributions to the integrated analysis of design decision problems for repairable items and their logistics support system. These contributions have been identified in this chapter. The developed framework has several strengths, but it also has several weaknesses. These are critically analyzed and a number of opportunities for future research identified.

5.1 Contributions

This research provides a framework in which interrelated system design decisions can be effectively analyzed in an integrated manner. This is the primary contribution of this research. As reported in the literature review, the body of research that addresses individual logistics decision problems is extensive. However, these individual decision problems are interrelated and therefore should not

be treated in an isolated manner. Several researchers have recognized the need to integrate these decision problems. However, there has been very little published work in this area. This research is one of the first to integrate these decision problems and the first to use the proposed approach.

Moore's work [1986], is the only published research found which attempts to address the degree of integration proposed in this research. Moore's research develops a concept for the mathematical modeling of multiple repairable equipment and logistic (MREAL) systems. The focus of Moore's work was on the development of new and/or improved mathematical models which yield useful and realistic predictions of the behavior of MREAL systems. This research differs in that it focuses on the development of a framework for the integration of existing mathematical models and techniques to provide the system designer with an effective decision support and design analysis tool.

The developed approach utilizes improvements in information technology which permit new and different forms of integration. In recent years, many advances in production systems design and management have resulted from integration of design decision problems and/or operational control strategies based upon improvements in information technology. These include Materials Requirements Planning, Computer Aided Design, and Computer Integrated Manufacturing. While improvements in information technology are not the only factors driving these advances, they certainly are a significant factor.

An important aspect of decision support systems, the information technology used in this research, is the ability to integrate data access and decision models. They do so by embedding the decision models in an information system that uses the database as the integration and communication mechanism between models. This research defined the interrelationships which exist between the individual design decision problems. Data flow diagrams were used to define the data flows which exist within and between decision problems. These data flow diagrams then define the data and information flows which must exist within the DSS database to effectively integrate and communicate between these decision problems.

Another contribution of this research is the level of complexity which can be included in the model of the multilevel inventory system. The model used in this research considers:

- the multi-indenture relationship between a repairable item and its component parts;
- three levels of maintenance and repair (two are more commonly considered);
- the nonstationary nature of maintenance and repair facility demands;
- multiple operating environments; and
- the stochastic nature of system operation and performance.

Most multilevel inventory models also consider a singular objective function. This research utilized a multicriteria decision model to reflect the importance of more than one design criteria on the assessment of system performance. While each of these levels of complexity taken by itself is not necessarily unique, the inclusion of all of these in a single model is unique.

Brammer's work [1985] is the most comprehensive, published model of a logistics support system. Conceptually, this model draws heavily from his work. However, the emphasis of Brammer's work was on maintenance requirements planning. This research differs in that the emphasis is on integrated design analysis. This research develops a framework in which the decision problems associated with the design of a repairable item and its logistics support system can be analyzed in an integrated manner. The integration of these decision problems was shown using a data flow diagram, the design evaluation function values for each decision problem was identified, and the varied effect of these decision problems on system design effectiveness was measured using a multicriteria measure.

A final contribution of this research is that it provides an approach which can be used throughout the life cycle of the repairable item. Most models are only useful beginning with the operational phase of the life cycle. The emphasis of this approach is on the design phases of the life cycle. However, since the model of the repairable item and its logistics support system can be easily expanded as the item progresses through its life cycle, this approach provides a tool which can be ef-

fectively used throughout the life cycle. As the item progresses from conceptual design to detail design, the maintenance tree is expanded as lower levels of detail are finalized. Since the maintenance tree is also the representational view of the database, this expansion is easily accommodated. During the operational phase of the life cycle, the design of the repairable item and its logistics support system are relatively static and the model can be used to evaluate the effect of different operating stategies. Therefore, the DSS is effectively a real-time management model of the real-world system being managed.

5.2 Critical Analysis

This research has shown that the design decision problems associated with a repairable item and its logistics support system can be effectively integrated. A framework for accomplishment of this integration was developed and its application to the design of a SLWT and its logistics support system was illustrated. The framework developed by this research provides a holistic approach to the design of the repairable item and its logistics support system. Previous research has focused on specific aspects of the ten design decision problems identified by Moore. System design has been accomplished in a piecemeal manner, assuming optimization of individual system elements would result in optimal system performance. However, because of the interrelated nature of the individual design decision problems, this was not necessarily a valid assumption.

The framework developed by this research also provides a robust modeling environment. The complex, stochastic nature of component failure and repair can be simulated. The nonstationary aspects of repair facility demands is captured using the multistream model. Multiple measures of system performance are calculated and a composite, multicriteria measure of system design effectiveness evaluated. The level of complexity which can be modeled exceeds that of any published research.

A final strength of the developed framework is that it provides an easy to use, flexible, and expandable environment for design analysis. The DSS dialog subsystem uses function keys and question-answer dialog to perform most analyses. Additional decision problems, models, and optimization routines could be added to the DSS models subsystem. The framework allows the designer to consider the entire system life cycle during design evaluation and thus to analyze the life cycle impact of design changes prior to implementation.

Given these strengths, this research represents only a first step towards effective, integrated design analysis. Much still needs to be done. A number of modeling weaknesses are identified in Section 5.3 as opportunities for future research. These include development of a more robust simulation model, inclusion of additional design decision problems, development of an enhanced multicriteria design evaluation process, and the addition of optimization procedures. The inability to optimize is a particular weakness. If a range of permissable values could be specified for each design decision variable, and the mathematical relationship between these variables and the design dependent parameters specified, a search routine or other procedure could be used to optimize the design. The more difficult aspect of such an optimization procedure would be in the definition of the mathematical relationship between the design decision variables and the design dependent parameters.

The design analysis framework developed by this research requires a significant amount of data. Currently, this data must be manually input by the system designer. Ideally, much of this data could be passed to CODAS from a CAD (Computer Aided Design) database. Specification of the repairable item design in the equipment design problem is the most data-intensive aspect of design specification. Specification of the repairable item design requires that a maintenance tree be developed and failure and repair characteristics for each bottom-of-the-tree node be specified. These data include the MTBF, failure distribution, MTTR, number of parallel redundant components, spare parts required to accomplish repair, etc.. Many of these values may already exist in the CAD database. If these could be passed to CODAS electronically, design specification would be simplified.

Establishing a link to a CAD system would also help alleviate another weaknesses of this research. The developed framework does not address the technical aspects of the repairable item design. The framework developed by this research focuses on evaluation of technically feasible equipment design alternatives. Technical aspects of the design such as size, weight, load capacity, operating range, etc. must currently be addressed outside the developed framework. The system designer is currently required to address these feasibility issues prior to design analysis. As the system designer seeks to modify the initial equipment design to improve the multicriteria measure of system design effectiveness, technical aspects of the equipment design may be compromised. For example, the addition of parallel redundant components will improve reliability. However, the additional weight of these components may have a negative effect on the operating range of an aircraft. CODAS has no mechanism for evaluating the technical feasibility of such a change. Therefore, design changes intended to improve system design effectiveness may not be technically feasible. A link to a CAD system would allow the system designer to evaluate the technical feasibility of a design change prior to design analysis.

A final weakness of this research is more a weakness of Moore's hierarchy of design decision problems than a weakness of the developed framework. Moore's hierarchy is life cycle incomplete. The design decision problems identified by Moore fail to consider the production phase of the system life cycle. The manufacture of the repairable item is only considered as it affects the initial cost of the item, not its supportability. Moore's hierarchy should be expanded to identify the decision problems assosciated with this important phase of the system life cycle. Once these decision problems have been identified, representations of these decision problems can be added to the DSS models subsystem and their impact modeled using data flow diagrams. Therefore, expansion of Moore's hierarchy could be easily accomodated in the developed framework.

5.3 *Future Research*

This research has developed a conceptual framework for integrated design analysis of a repairable item and its logistics support system. An implementation of this framework has been described and illustrated as part of this research. There are a number of dimensions along which this research can be expanded or enhanced. These include:

1. expand to address additional design decision problems;
2. expand the treatment of the five design decision problems addressed by this research effort;
3. enhance existing simulation routines;
4. enhance optimization capabilities;
5. expand and/or enhance treatment of multicriteria performance assessment; and
6. expand the nature of the design activity addressed by the framework.

The conceptual framework developed by this research was illustrated using five of the ten design decision problems identified by Moore. This research can be expanded to address the remaining five decision problems. In addition, Moore's hierarchy needs to be expanded to include additional decision problems which address producibility issues. The five decision problems, already identified, but not addressed by this initial research effort are:

1. mechanic training problem
2. preventive maintenance policy
3. inspection and training policy
4. repair facility design problem
5. operator training problem

Expansion of the existing research to include these five additional decision problems would require varying degrees of difficulty. The training problems could be incorporated relatively easily by

making the type, cost, and impact of training required inputs. These are design dependent parameters which must be updated each time a design decision variable is changed. The preventive maintenance and the inspection and testing problems are interrelated and would require revision of the way streams and their assosciated failure distributions are handled. Agee and Gallion [1986] have addressed some of the changes which would be required. The facility design problem would require an enhanced treatment of repair times and their distributions. More detailed modeling of each repair facility would also be required.

This research can also be expanded to address additional aspects of the five design decision problems included as part of this research. For example, one aspect of the replacement policy problem not currently addressed is the catastropic loss problem. One aspect of the maintenance configuration problem not currently addressed is the channel sharing problem. These and other aspects of the five decision problems addressed in this research could be added to the existing decision support system.

The simulation routine used in this research is relatively simple. Item failures are based upon expected values derived from cumulative failure distributions rather than purely random events. The simulation could be made much more robust by interfacing a more complex model, developed in FORTRAN, SIMAN, SLAM, or some other simulation language, with the decision support system. Such a model would also allow additional stochastic aspects of system operation to be captured.

The approach to optimization of the system design described in this dissertation needs to be enhanced to include search techniques and other optimization procedures to guide the system designer to the best overall design. The current approach relies too heavily upon the expertise of the system designer to search a set of feasible design alternatives. Logically, the effectiveness of this approach will vary based upon the designer's level of expertise. Alternatively, an expert system could be developed and interfaced to the decision support system to aid in this search for the best design.

The multicriteria decision model used to assess design effectiveness is a relatively simple, straightforward, multicriteria model. It requires the system designer to establish evaluation scales for each criteria. These can be difficult to establish. A more robust assessment might be obtained using the Analytic Hierarchy Process (AHP). AHP allows more extensive decomposition of the objective function into attributes, subattributes, etc. Design alternatives are then compared using a set of paired comparisons. Alternatively, goal programming could be used to identify the design which best attains stated aspiration levels for each criteria, based upon a prioritization of the attributes. The multicriteria decision model could also be expended to include additional criteria (e.g., supportability) and/or additional measures for existing criteria.

One additional avenue for future research is to expand the nature of the design activity addressed by the framework developed in this research. This research focused on the design of a repairable item and its logistics support system. There is also significant interest in the concurrent design of a product and its manufacturing and/or assembly system (design for manufacture, design for assembly, etc). The framework developed by this research could be modified to address a hierarchy of decision problems associated with the design of a product and its production system.

As can be seen from the preceding discussion, the opportunity for future research in this area is very rich. The existing effort focused on development of a descriptive approach. Much can be done to expand and/or enhance this approach to make it more robust and more complete.

5.4 Conclusions

This research effort represents only a first step towards effective, integrated design analysis. However, this research makes a number of contributions to the integrated analysis of design decision problems for repairable items and their logistics support system. While the developed framework

has several strengths, it also has several weaknesses. A number of opportunities for future research have been identified to help alleviate these weaknesses and to enhance the capabilities of the developed framework. Continued research in this area is needed and has enormous potential.

References

1. Agee, Marvin H. and Mark S. Gallion, *Simulation of Population Maintenance Requirements,* VPI&SU Research Report Submitted To The Naval Electronics Systems Command, March, 1986.

2. Alter, S.L., *Decision Support Systems - Current Practice and Continuing Challenges,* Addison-Wesley Publishing Co., Inc., 1980.

3. Benjamin, R.I. and M.S. Scott Morton, "Information Technology, Integration, and Organizational Change," *Interfaces,* Vol 18, No 3, 1988.

4. Blanchard, B.S., *Logistics Engineering and Management,* Prentice-Hall, Englewood Cliffs, NJ, 1986.

5. Blanchard, B.S. and W.J. Fabrycky, *Systems Engineering and Analysis,* Prentice-Hall, Englewood Cliffs, NJ, 1990.

6. Brammer, K.W., C.J. Malmborg, and M.H. Agee, *A Transient State Maintenance Requirements Planning Model,* Naval Electronics Systems Command Contract, No N00039-84-C-0346 (Option #2) Final Report and Master of Science Thesis, IEOR Department, VPI&SU, February, 1985.

7. Brown, G.F., Jr., L.P. Silverman, and B.L. Perlman, "Optimal Positioning of Inventory Stock in a Multi-Echelon System," Center for Naval Analyses, PP No 74, Arlington, VA, 1971.

8. Clark, A.J., "An Informal Survey of Multi-Echelon Inventory Theory," *Naval Research Logistics Quarterly,* Vol 19, No 4, 1972.

9. Cohen, M.A., I. Or, W.P. Pierskalla, and H. Yen, "A Dynamic Inventory With Recycling," *Naval Research Logistics Quarterly,* Vol 27, No 2, 1980.

10. Das, C.,"The (S-1, S) Inventory Model Under Time Limit On Backorders," *Operations Research,* Vol 25, No 5, 1977.

11. DeMarco, T., *Structured Analysis and System Specification,* Yourdon Press, 1979.

12. Demmy, W.S., "Allocation of Spares and Repair Resources to a Multi-Component System," Air Force Logistics Command Report 70-17, Wright Patterson Air Force Base, Ohio, 1970.

13. Demmy, W.S. and V.J. Presutti, "Multi-Echelon Inventory Theory in the Air Force Logistics Command," in *Multi-Level Production/Inventory Control Systems*, edited by L.B. Schwarz, TIMS Studies in the Management Sciences, North-Holland, Amsterdam, Vol 16, 1981.

14. Fabrycky, Wolter J., "RAMCAD and Concurrent Engineering Design Research at Virginia Tech," Proceedings, Fifth Annual RAMCAD Technical Interchange Meeting, April, 1989.

15. Fabrycky, Wolter J., "Concurrent Life-Cycle Engineering Research at Virginia Tech," Proceedings, Collaborative Life-Cycle Engineering Research with Virginia Industry, Research Review Workshop, March, 1990.

16. Fabrycky, W.J., C.J. Malmborg, T.P. Moore, and K.W. Brammer, "The Repairable Equipment Population System Demonstrator," User's Guide and program for the IBM-PC, VPI&SU Report for U.S. Navy Contract N00039-84-C-0346T, 1984.

17. Fawcett, W.M. and L.J. York, "Base-Depot Stockage Model," Research Report ERR-FW-621, General Dynamics, Fort Worth, TX, 1967.

18. Feeney, G.J. and C.C. Sherbrooke, "The (S-1, S) Inventory Policy Under Compound Poisson Demand," *Management Science,* Vol 12, No 5, 1966.

19. Frisch, Franz A.P., "Mortality and Spareparts: A Conceptual Analysis," Proceedings of the 1983 Federal Acquisition Research Symposium.

20. Galliher, H.P., P.M. Morse, and M. Simond, "Dymanics of Two Classes of Continuous Review Inventory Systems," *Operations Research,* Vol 7, No 3, 1959.

21. Geisler, M.A. and B.L. Murrie, "Assessment of Aircraft Logistics Planning Models," *Omega*, Vol 9, No 1, 1981.

22. Gross, D. and C.M. Harris, "On One For One Ordering Inventory Policies With State Dependent Leadtimes," *Operations Research,* Vol 19, No 3, 1971.

23. Gross, D. and Ince, J.F., "Spares Provisioning for Repairable Items: Cyclic Queues in Light Traffic," *AIIE Transactions*, Vol 10, 1978.

24. Gross, D., Kahn, H.D., and Marsh, J.D., "Queueing Models for Spares Provisioning," *Naval Research Logistics Quarterly*, Vol 24, No 4, 1977.

25. Haber, S., "Simulation of Multi-Echelon Macro-Inventory Policies," *Naval Research Logistics Quarterly,* Vol 18, 1971.

26. Haber, S.E. and R. Sitgreaves, "Inventory Model For Intermediate Echelon When Repair is Possible," *Management Science,* Vol 21, No 6, 1975.

27. Hadley, G. and T.M. Whitin, *Analysis of Inventory Systems,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1963.

28. Higa, I., A. Feyerherm, and A. Machado, "Waiting Time in an (S-1, S) Inventory System," *Operations Research,* Vol 23, No 4, 1975.

29. Hillier, F.S. and Lieberman, G.J., *Introduction to Operations Research*, 3rd Edition, Holden-Day, Inc., San Francisco, 1980.

30. Kaplan, A. and D. Orr, "An Optimum Multiechelon Repair Policy and Stockage Model," Naval Research Logistics Quarterly, Vol 32, 1985.

31. Kearney, A.T., Inc., *Measuring and Improving Productivity in Physical Distribution Management,* National Council of Physical Distribution Management, Oak Brook, IL, 1984.

32. Keen and Scott Morton, *DSS: An Organizational Perspective,* Addison-Wesley, 1978.

33. Kline, Melvin B., "A Survey of Logistics Analysis Models," Proceedings 17th Annual International Logistics Symposium, Society of Logistics Engineers, Boston, Mass., August, 1982.

34. Kruse, W.K., "Waiting Time in an (S-1, S) Inventory System With Arbitrarily Distributed Leadtimes," Technical Report, U.S. Army Inventory Research Office, 1977.

35. Lureau, F., "A Queuing Theoretic Analysis of Logistics Repair Models with Spare Units," Technical Report, No 167, Department of Operations Research, Stanford University, Stanford, CA, 1974.

36. Moore, T.P., "Optimal Design, Procurement and Support of Multiple Repairable Equipment and Logistic Systems," Unpublished Doctoral Dissertation, VPI&SU, 1986.

37. Muckstadt, John A., "A Model for a Multi-Item, Multi-Echelon, Multi-Indenture Inventory System," *Management Science*, Vol 20, No 4, Dec., 1973.

38. Muckstadt, J.A., "A Three-Echelon, Multi-Item Model For Recoverable Items," *Naval Research Logistics Quarterly,* Vol 26, No 2, 1979.

39. Nahimas, S., "Managing Repairable Item Inventory Systems: A Review," in *Multi-Level Production/Inventory Control Systems*, edited by L.B. Schwarz, TIMS Studies in the Management Sciences, North-Holland, Amsterdam, Vol 16, 1981.

40. Palm, C., "Analysis of the Erlang Traffic Formula for Busy-Signal Arrangements," *Ericsson Technics,*, No 5, 1938.

41. Phelps, E.S., "Optimal Decision Rules for the Procurement, Repair or Disposal of Spare Parts," RAND Memorandum RS-2920PR, the Rand Corp., Santa Monica, Cal., 1962.

42. Pinkus, C.E., "The Design of Multi-Echelon Inventory Systems Using a Branch-and-Bound Algorithm," The George Washington University, Institute for Management Science and Engineering, Program in Logistics, Serial T-250, Washington, D.C., 1971.

43. Porteus, E.L. and Lansdowne, A.P., "Optimal Design of a Multi-Item, Multi-Location, Multi-Repair Type Repair and Supply System," *Naval Research Logistics Quarterly*, No 21, 1974.

44. Prawda, J. and G.P. Wright, "On a Replacement Problem," *Cahiers du Centre d'Etudes de Recherche Operationnelle,* Vol 14, 1972.

45. Rose, M., "The (S-1,S) Inventory Model With Arbitrary Backordered Demand and Constant Delivery Times," *Operations Research,* Vol 20, No 5, 1972.

46. Scarf, H.E., "Stationary Operating Characteristics of an Inventory Model With Time Lag," Chapter 16 in Arrow, K.J., T. Harris, and H.E. Scarf, Editors, *Studies in the Mathematical Theory of Inventory and Production,* Stanford University Press, Stanford, CA, 1958. Problem,"

47. Schwarz, L.B.,"Introduction," in *Multi-Level Production/Inventory Control Systems,* TIMS Studies in the Management Sciences, North-Holland, Amsterdam, Vol 16, 1981.

48. Scott Morton, M.S., *Management Decision Systems: Computer Based Support For Decision Making,* Division of Research, Harvard University, Cambridge, MA, 1971.

49. Sharda, R., S.H. Barr, and J.C. McDonnell, "Decision Support System Effectiveness: A Review and An Empirical Test," *Interfaces,* Vol 18, No 1, 1988.

50. Sherbrooke, C.C., "METRIC: A Multi-Echelon Technique for Recoverable Item Control," *Operations Research*, Vol 16, No 1, 1968.

51. Sherbrooke, C.C., "An Evaluator of the Number of Operationally Ready Aircraft in a Multilevel Supply System," *Operations Research*, Vol 19, No 3, 1971.

52. Simon, R.M., "Stationary Properties of a Two Echelon Inventory Model for Low Demand Items," *Operations Research*, Vol 19, No 3, 1971.

53. Simpson, V.P., "Optimum Solution Structure For a Repairable Inventory Problem," *Operational Research*, Vol 26, No 2, 1978.

54. Sprague, R.H. and E.D. Carlson, *Building Effective Decision Support Systems*, Prentice-Hall, Inc., 1982.

55. Veinott, A.F., Jr., "The Status of Mathematical Inventory Theory," *Management Science*, Vol 12, No 11, 1966.

56. Wagner, H.M., "The Next Decade of Logistics Research," *Naval Research Logistics Quarterly*, Vol 26, 1979.

57. Williams, J.F., "The Range Model," Hq. Air Force Logistics Command, Operations Analysis Office, Operations Analysis Technical Memorandum No 8, Wright-Patterson, Ohio, 1969.

58. Yourdon, E., *Managing The System Life Cycle: A Software Development Methodology Overview*, Yourdon Press, New York, 1982.

Bibliography

1. Albright, S.C. and A. Soni, "Markovian Multiechelon Repairable Inventory System," *Naval Research Logistics,* Vol 35, 1988.

2. Allen, S.G., "Redistribution of total stock over several user locations," *Naval Research Logistics Quarterly,* Vol 5, 1958.

3. Anderson, E.E. and Y. Chen, "A Decision Support System for the Procurement of Military Equipment," *Naval Research Logistics,* Vol 35, 1988.

4. Barlow, R.E. and Hunter, L., "Optimal Preventive Maintenance Policies," *Operations Research,* Vol 8, 1960.

5. Barlow, R.E. and Proschan, F., *Mathematical Theory of Reliability,* John Wiley Sons, New York, 1967.

6. Berlucchi, E. A., "LOR... The Maintenance Engineer's Tool for Reducing Logistics Costs," *Proceedings,* 17th Annual International Logistics Symposium, Boston, MA, August, 1982.

7. Black, G. and Proschan, F., "On Optimal Redundancy," *Operations Research,* Vol 7, September/October, 1959.

8. Brown, M., "On the Reliability of Repairable Systems," *Operations Research,* Vol 32, May/June, 1984.

9. Clark, A.J., "Experiences With A Multi-Indentured, Multi-Echelon Inventory Model," in *Multi-Level Production/Inventory Control Systems,* edited by L.B. Schwarz, TIMS Studies in the Management Sciences, North-Holland, Amsterdam, Vol 16, 1981.

10. Derman, C., "On Sequential Decisions and Markov Chains," *Management Science,* Vol 9, 1962.

11. Eiger, A., J.M. Jacobs, D.B. Chung, and J.L. Selsor, "The US Army's Occupational Specialty Manpower Decision Support System," *Interfaces,* Vol 18, No 1, 1988.

12. Elam, J.J., G.P. Huber, and M.E. Hurt, "An Examination of the DSS Literature (1975-1985)," *Decision Support Systems: A Decade In Perspective,* E.R. McLean and H.G. Sol Editors, Elsevier-Science Publishers, New York, 1986.

13. Fabrycky, W.J. and Hart, J.T., "Economic Optimization of a Finite Population System Deployed to Meet a Demand," Proceedings of the Joint Conference, American Association of Cost Engineers/American Institute of Industrial Engineers, Houston, Texas, 1975.

14. Falk, H.E. and H.K. Rappoport, "An Optimization Technique for a Multi-time Period Spare Provisioning Problem," Technical Paper Serial T-408, Program in Logistics, The George Washington University, 1980.

15. Geisler, M.A., editor, "Preface" in *Logistics*, North-Holland/TIMS Studies in the Management Science Series, Volume I, North-Holland, 1975.

16. Graves, S.C., "A Multi-Echelon Inventory Model For A Repairable Item With One-For-One Replenishment," *Management Science,* Vol 31, No 10, October, 1985.

17. Graves, S.C. and J. Keilson, "A Methodology For Studying The Dynamics Of Extended Logistic Systems," *Naval Research Logistics Quarterly,* Vol 26, No 2, 1979.

18. Graves, S.C. and J. Keilson, "System Balance for Extended Logistic Systems," *Operations Research*, Mar/Apr, 1983.

19. Gross, D. and Harris, C.M., *Fundamentals of Queueing Theory*, John Wiley Sons, Inc., New York, 1974.

20. Gross, D. and D. R. Miller, "Multiechelon Repairable-Item Provisioning in a Time-Varying Environment Using the Randomization Technique," *Naval Research Logistics Quarterly,* Vol 31, 1984.

21. Gross, D., R.M. Soland, and C.E. Pinkus, "Designing A Multi-Product, Multi-Echelon Inventory System," in *Multi-Level Production/Inventory Control Systems*, edited by L.B. Schwarz, TIMS Studies in the Management Sciences, North-Holland, Amsterdam, Vol 16, 1981.

22. Hannsman, F., "Optimal inventory location and control in production and distribution networks," *Operations Research,* Vol 7, 1959.

23. Hart, J.T., "A Model for the Optimal Design of a Finite Population Queueing System Deployed to Meet a Demand," unpublished Master's Thesis, Virginia Tech, 1971.

24. Hayre, Lakbir S., "A, Note on Optimal Maintenance Policies for Deciding Whether to Repair or Replace," *European Journal of Operational Research*, Feb., 1983.

25. Hillestad, R.J., "Dyna-Metric: Dynamic Multi-Echelon Technique for Recoverable Item Control," RAND Corp., Santa Monica, Cal., July 1982.

26. Kae, E., "Optimal Replacement Rules," *Operations Research*, Vol 21, 1973.

27. Lie, C.H., Hwang, C.L., and Tillman, F.A., "Availability of Maintained Systems: A State-of-the-Art Survey," *AIIE Transactions*, Vol 9, 1977.

28. Lohmar, J.W. and Mandel, D.B., "Guided Missile Frigate TIGER Assessment," 1981 Proceedings Annual Reliability and Maintainability Symposium.

29. McCall, J.J., "Maintenance Policies for Stochastically Failing Equipment: A Survey," *Management Science*, Vol 11, 1965.

30. Mirasol, N.M., "A Queuing Approach to Logistics Systems," *Operations Research,* Vol 12, Sept/Oct, 1964.

31. Monahan, G.E. and T.L. Smurt, "A Multilevel Decision Support System for the Financial Justification of Automated Flexible Manufacturing Systems," *Interfaces,* Vol 17, No 1, 1987.

32. Morse, P.M., *Queues, Inventories and Maintenance,* John Wiley & Sons, Inc., New York, 1958.

33. Petrovic, R., Senborn, A., and Vujosevic, M., "Spares Allocation in the Presence of Uncertainty," *European Journal of Operational Research*, Sept., 1982.

34. Pierskalla, W.P. and Voelker, J.A., "A Survey of Maintenance Models: The Control and Surveillance of Deteriorating Systems," *Naval Research Logistics Quarterly*, No 23, Sep., 1976.

35. Rau, J.G., *Optimization and Probability in Systems Engineering,* Van, Nostrand Reinhold Company, New York, 1970.

36. Ross, Sheldon M., *Applied Probability Models With Optimization Applications*, Holden-Day, San Francisco, 1970.

37. Schrady, D.A., "A Deterministic Inventory Model for Repairable Items," *Naval Research Logistics Quarterly*, Vol 14, No 3, 1967.

38. Sethi, S., "Simultaneous Optimization of Preventive Maintenance and Replacement Policies for Machines: A Modern Control Theory Approach, *AIIE Transactions*, Vol 5, 1973.

39. Sherbrooke, C.C., "Vari-METRIC: Improved Approximations For Multi-Indenture, Multi-Echelon Availability Models," *Operations Research*, Vol 34, 1986.

40. Sherif, Y.S. and Smith, M.L., "Optimal Maintenance Models for Systems Subject to Failure - A Review," *Naval Research Logistics Quarterly*, March 1981.

41. Simon, H., *The New Science of Management Decision,* Harper and Row, 1960.

42. Simpson, K.F., Jr., "A Theory of Allocations of Stocks to Warehouses," *Operations Research,* Vol 7, 1960.

43. Stein, R.G., "Futuristic Logistics," *Logistics Spectrum,* Vol 19, Spring, 1985.

44. Taylor, J. and Jackson, R.R.P., "An Application of the Birth and Death Process to the Provision of Spare Machines," *Operational Research Quarterly*, 1954.

45. White, J.A., Schmidt, J.W., and Bennett, G.K., *Analysis of Queuing Systems*, New York: Academic Press, 1975.

Appendix A. Node Data

Node Data For Support Area Environment

Node	Node Description	Repair Level	MTBF (days)	MTTR (hours)	Failure Curve
6	Fatal Failure	N/A	12500	0	2
10	Engine Oil Cooler	1	1000	43	5
27	Freshwater Pump	2	250	24	3
28	Seawater Pump	2	625	29	2
29	Hydraulic Pump	1	1000	29	3
32	Eng Cool Heat Exch	2	1000	43	4
33	Freshwat Heat Exch	1	1000	43	4
34	Hyd Fluid Heat Exch	1	1000	43	4
52	Pres Reducing Valve	1	500	19	3
53	Dir Control Valve	1	125	29	3
54	Steering Motor	2	375	43	3
58	Filter	1	75	7	3
59	Starter	1	312	48	3
62	Auto Lubrication	1	188	24	5
63	Timer	1	94	58	5
64	Manifold	2	1250	360	3
65	Solenoid	1	94	29	5
92	Waterjet Pump	3	625	288	3
93	Discharge Elbow	2	1250	264	5
94	Steerable Nozzle	2	625	288	3
107	Booster Pump	2	375	19	3
108	Fuel/Water Separ	1	188	24	5
112	Inverter	1	250	12	5
113	Battery	1	125	3	3
114	Alternator	1	250	58	5
118	Hydraulic Motor	2	125	19	5
119	Hydraulic Pump	2	125	38	3
120	Cable	1	6250	12	5
123	Engine On/Off	1	1250	3	5
124	Throttle	1	2625	24	5
125	Steering Lever	1	2625	19	5
126	Clutch Lever	1	2625	19	5

Node Data For Combat Zone Environment

Node	Node Description	Repair Level	MTBF (days)	MTTR (hours)	Failure Curve
6	Fatal Failure	N/A	600	0	2
10	Engine Oil Cooler	1	667	43	2
27	Freshwater Pump	2	167	24	2
28	Seawater Pump	2	417	29	2
29	Hydraulic Pump	1	667	29	2
32	Eng Cool Heat Exch	2	667	43	2
33	Freshwat Heat Exch	1	667	43	2
34	Hyd Fluid Heat Exch	1	667	43	2
52	Pres Reducing Valve	1	333	19	3
53	Dir Control Valve	1	83	29	3
54	Steering Motor	2	250	43	3
58	Filter	1	50	7	3
59	Starter	1	208	48	3
62	Auto Lubrication	1	125	24	5
63	Timer	1	63	58	5
64	Manifold	2	833	360	3
65	Solenoid	1	63	29	5
92	Waterjet Pump	3	417	288	3
93	Discharge Elbow	2	833	264	5
94	Steerable Nozzle	2	417	288	3
107	Booster Pump	2	250	19	3
108	Fuel/Water Separ	1	125	24	5
112	Inverter	1	167	12	5
113	Battery	1	83	3	3
114	Alternator	1	167	58	5
118	Hydraulic Motor	2	83	19	5
119	Hydraulic Pump	2	83	38	3
120	Cable	1	4167	12	5
123	Engine On/Off	1	833	3	5
124	Throttle	1	1750	24	5
125	Steering Lever	1	1750	19	5
126	Clutch Lever	1	1750	19	5

Appendix B. Stockable Parts

Node	Part Description	Qty	Whse Space SqFt	Unit Price $ ea.	Lead Time (Pds)
10	Eng Oil Cooler	1	2.0	250	2
27	100 GPM Pump	1	1.0	250	2
28	.5HP Pump Motor	1	0.5	1800	2
28	Packing	1	5.0	5	1
28	Impeller Bearing	2	5.0	16	2
28	Casing Bearing	2	5.0	14	2
29	100 PSI Pump	1	1.0	600	2
32	EC Heat Exchang	1	0.3	1000	3
33	FW Heat Exchang	1	0.5	350	2
34	HF Heat Exchang	1	0.5	450	2
52	3/8″ PR Valve	1	10.0	23	2
52	3/8″ Hose Clamp	2	100.0	0.45	1
53	3/8″ DC Valve	1	10.0	30	2
53	3/8″ Hose Clamp	2	100.0	0.45	1
54	1/4 HP Motor	1	1.0	290	2
58	HSS Filter	1	5.0	11	1
59	Starter	1	1.0	190	2
62	Plunger Pump	1	2.0	80	2
62	Oil Line Filter	1	10.0	4.50	2
63	HSAL Timer	1	4.0	63	2
64	HSAL Manifold	1	1.0	22	1
64	1/2-13 Bolts	8	100.0	0.12	1
65	HSAL Solenoid	1	10.0	20	1
92	200 HP Motor	1	0.2	11000	3
93	WJ Dis. Elbow	1	1.0	50	1
93	1/2-13 Bolts	8	100.0	0.12	1
93	1/2-13 Nuts	8	200.0	0.06	1
94	3″ Steer Nozzle	1	1.0	190	2
107	.5HP Boost Pump	1	2.0	80	2
108	F/W Separator	1	2.0	60	1
112	Inverter	1	10.0	18	2
113	12V Battery	1	1.0	60	1
113	Cable	2	5.0	6	1
114	Isolation Diode	1	20.0	5	1
114	Brushes	2	20.0	2	1
114	Voltage Reg.	1	10.0	21	1
118	5HP Motor	1	1.0	800	2
119	100PSI Hyd Pump	1	1.0	600	2
120	1/8″Steel Cable	1	0.5	150	2
123	EOF Switch	1	10.0	21	2
124	Throttle	1	5.0	15	2
125	Steering Lever	1	5.0	16	2
126	Clutch Lever	1	5.0	17	2

Appendix C. CODAS User's Guide

C.1 Introduction

CODAS (Concurrent Design Analysis System) is a decision support system for concurrently analyzing the design of a repairable item and its logistics support system. The repairable item population may be operating under conditions where steady state or nonsteady state maintenance requirements are being generated. CODAS simulates population operation over a defined planning horizon and evaluates design performance using a multiattribute evaluation model. Period-by-period logistics support requirements are determined and summarized for the planning horizon specified.

CODAS is programmed in PL/I and uses the IBM Display Management System (DMS) to support interactive user sessions in a VM/CMS environment. The system currently resides on a VPI&SU mainframe computer (IBM 3090). This user's guide illustrates all introductory, input, and output screens for the system. All model operations are function key (PF key) driven. Model inputs must be typed by the user initially. However, these can be saved in a data file and retrieved for subsequent analysis. Output screens provide period detail and run summary reports to the user. The following steps are required to access the system from your "A" disk. These may vary slightly for different installations.

1. Logon your VM/CMS user id.
2. Type CODAS on the command line and press return.

Codas is now running. The first screen displayed is the system banner screen shown in Figure 29 on page 156. The PF keys available from this screen are described below.

PF1 Concepts - review basic concepts used by CODAS
PF8 Next Screen - display next screen
PF12 Exit - exit CODAS

CODAS supports active access to ten different scenarios. Each scenario is a set of input values representing a complete repairable equipment population system. The systems modelled may be distinctly different (e.g., automobiles versus airplanes) or slightly different versions of the same system (e.g., SLWT with 2 levels of repair versus SLWT with 3 levels of repair). Scenario management functions (retrieve data and save data) are performed from the Scenario Management Screen shown in Figure 30 on page 157. Scenarios are numbered (0-9) and are given a seven character (alphanumeric) name. Space is also provided for a short, 40 character scenario description. The user must enter the number of the desired scenario and press the appropriate PF key to execute a scenario management function. PF key operations for this screen are:

PF1 Concepts - review basic concepts used by CODAS
PF4 Get Data - retrieve data for selected scenario
PF5 Save Data - save and/or replace data for specified scenario
PF7 Prev Screen - display previous system screen
PF8 Next Screen - display next system screen
PF12 Exit - exit to CODAS banner screen

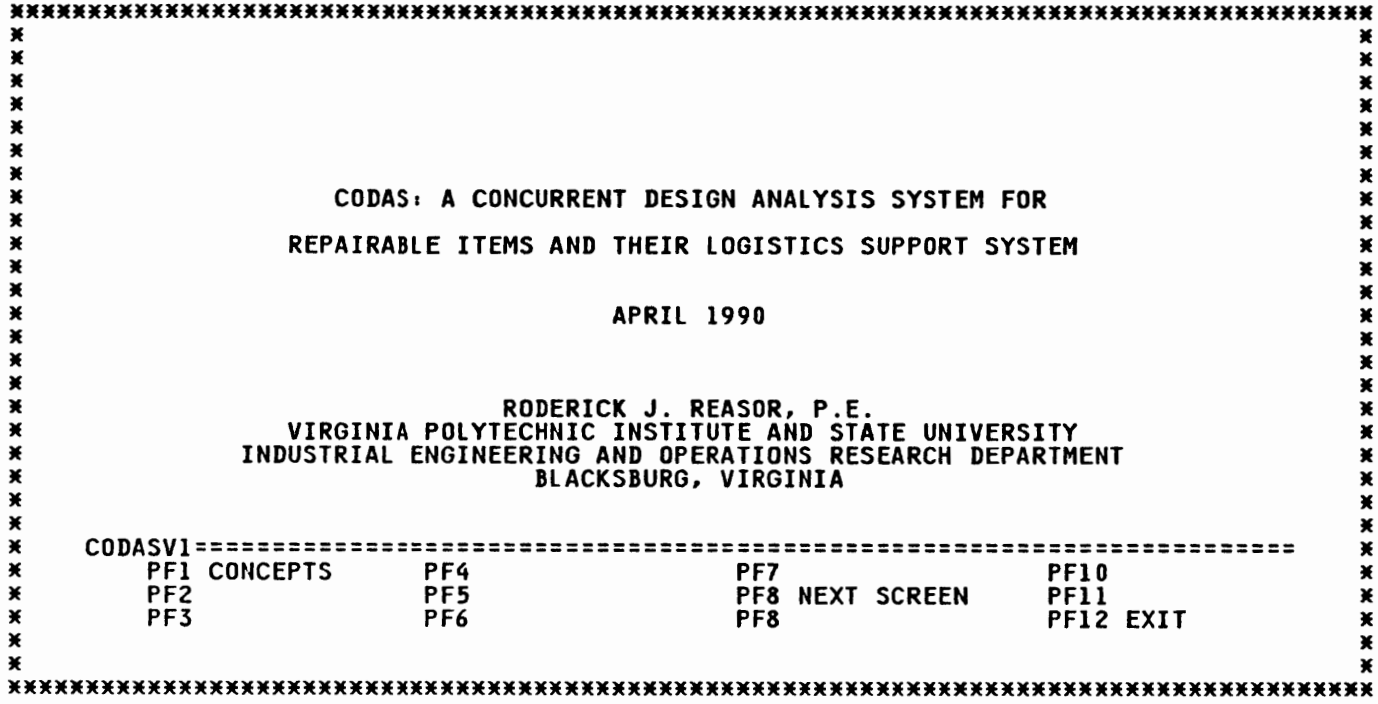


Figure 29. System Banner Screen

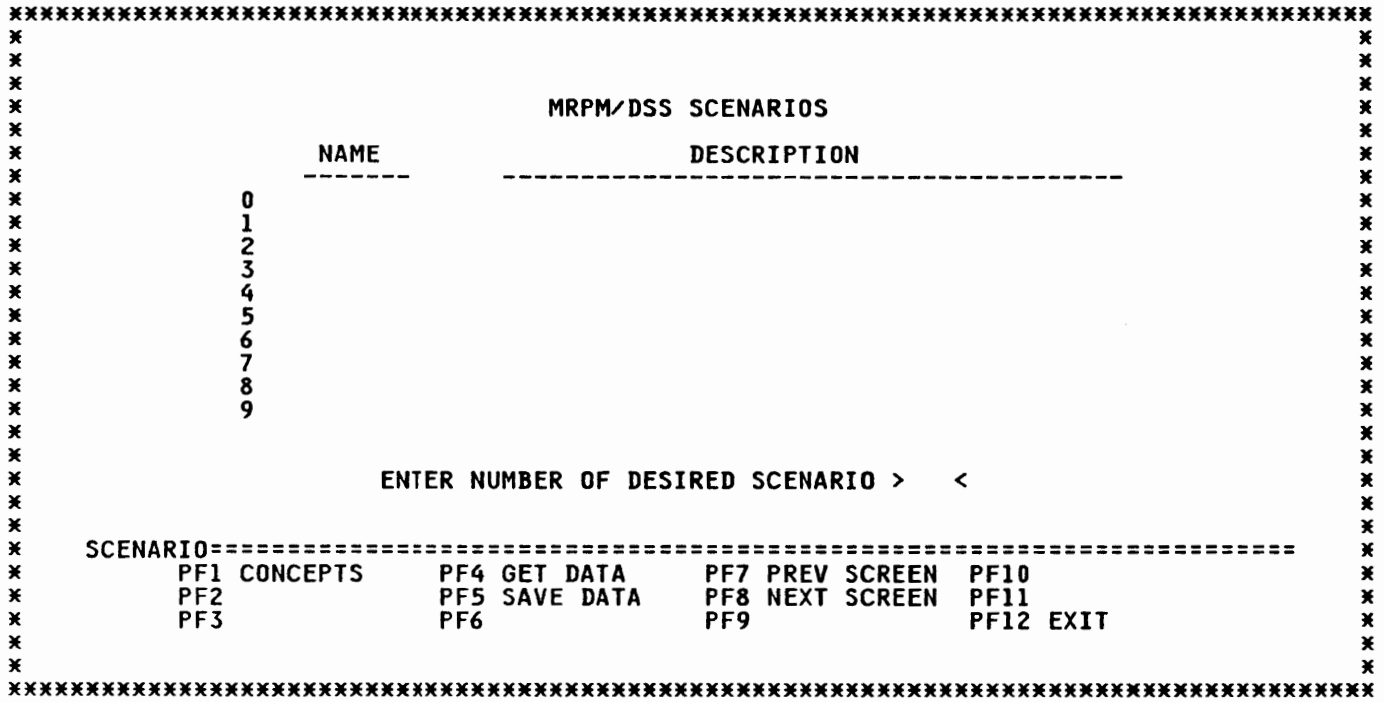


Figure 30. Scenario Management Screen

C.2 Basic Concepts

There are five "Concept" screens included in CODAS, shown in Figure 31 on page 159 through Figure 35 on page 163. These screens are intended to give the model user an overview of the basic concepts and terminology used by CODAS. These screens illustrate the multistream concept and the maintenance tree concept. The five screens included are:

- Multistream concept
- Maintenance tree concept
- Maximum maintenance tree
- Example maintenance tree
- Maintenance tree definitions

These screens may be accessed from any input screen using the PF1 key. However, these screens will be most helpful when specifying the repairable item design required by the equipment design problem. PF key operations for these five screens, once accessed, are:

PF7 Prev Screen - display previous concept screen
PF8 Next Screen - display next concept screen
PF12 Exit - exit concept screens and return to point of access

```
                                MULTISTREAM CONCEPT

                          T1        T2        T3        T4        T5

                AGE3                          .1A.1B    .2A.2B    .3A.3B

                AGE2                .1A.1B    .2A.2B    .3A.3B    .4A.4B

                AGE1      1A 1B     2A 2B     3A 3B     4A 4B     5A 5B


              T1,T2,.....   = TIME PERIOD
              AGE1,AGE2,...  = AGE OF END ITEMS (AGE1<AGE2<AGE...)
              1A 1B.....     = END ITEMS OF THE SAME AGE, OPERATING
                               IN FAILURE ENVIRONMENTS A AND B
              1A.1A.1A.....  = A STREAM OF SIMILARLY AGED ITEMS (A STREAM
                               ENDS WHEN ALL ITEMS FAIL OR ARE RETIRED.)

    INTROMSC==========================================================
             PF1               PF4               PF7 PREV SCREEN   PF10
             PF2               PF5               PF8 NEXT SCREEN   PF11
             PF3               PF6               PF9               PF12 EXIT
```

Figure 31. Multistream Concept

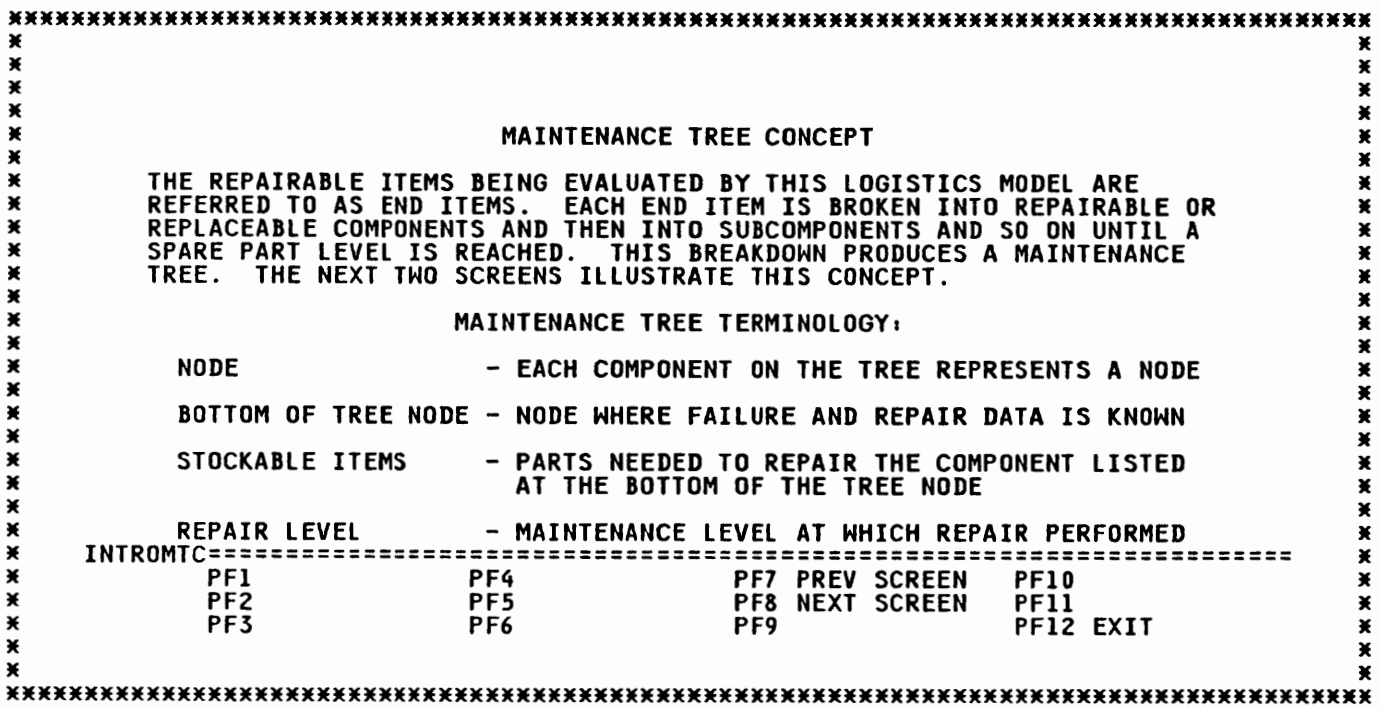


Figure 32. Maintenance Tree Concept

Figure 33. Maximum Maintenance Tree

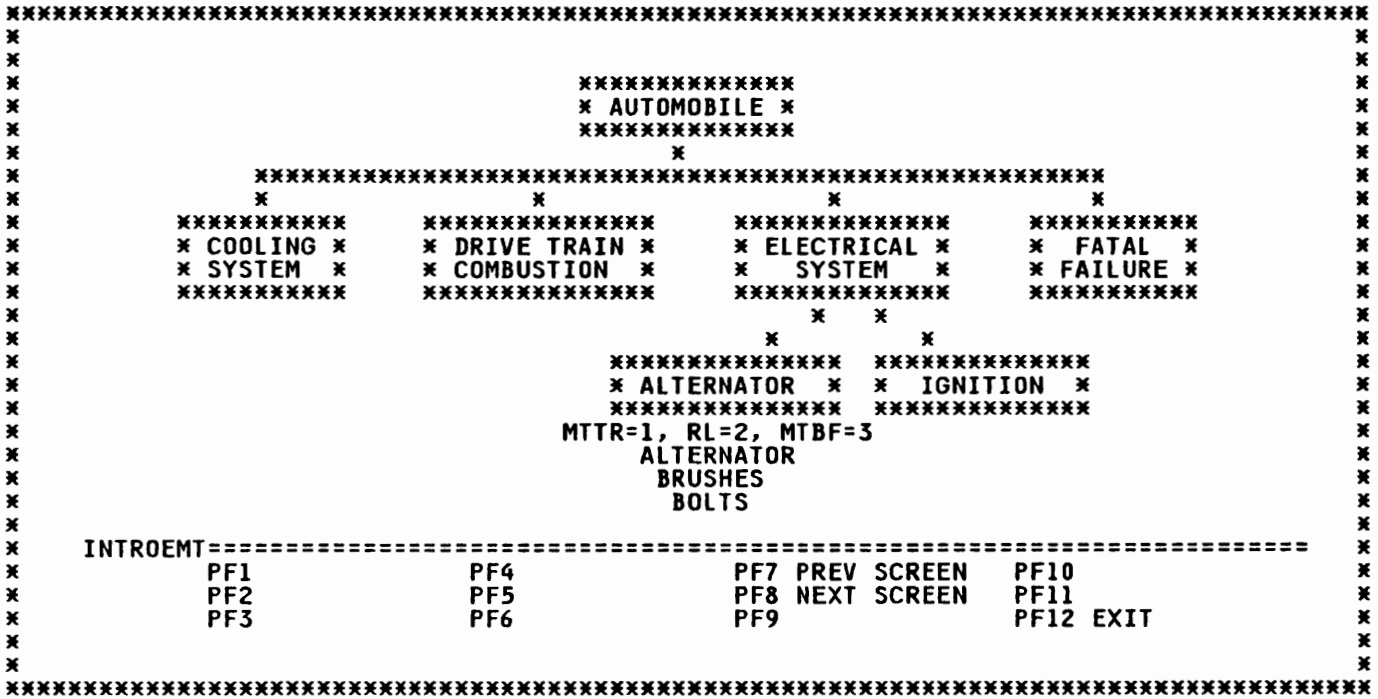


Figure 34. Example Maintenance Tree

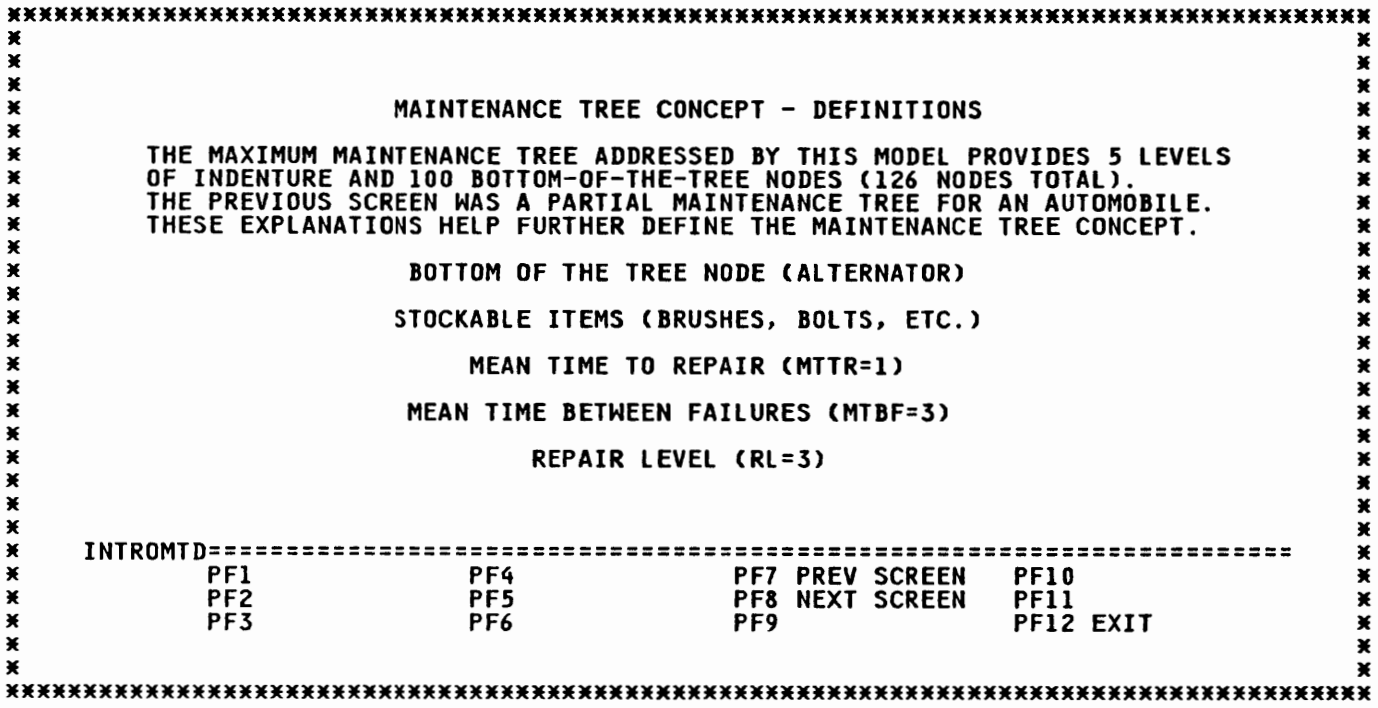


Figure 35. Maintenance Tree Definitions

C.3 *Input Screens*

The input screens are the CODAS screens used to interactively define a design alternative. The user must specify the decision variables, the design dependent parameters, and the design independent parameters associated with the design of the repairable item and its logistics support system. The input of these values is structured around the five design problems addressed by this research. Once these values have been input, CODAS will evaluate the impact of the specified decisions on design performance. The limitations imposed by the current version of the software upon specification of any design alternative are:

Table 7. Programmed CODAS Limitations

Specification	Maximum
Levels of Repair	3
Operating environments	4
Levels of indenture	5
Spare parts for each bottom-of-the-tree node	10
Number of active streams	50

Input data should be entered in the areas designated on each screen. Each required data entry is described on the screen. The TAB key will move the cursor sequentially from one input field to another. After entering the required data, press PF8 to move to the next input screen. When first creating the input data for a scenario, the user should move sequentially through all input screens using PF8. When experimenting with different sets of conditions or constraints, the user may skip directly to the desired input screen (using PF keys), change the desired value(s) only, and execute a new run without having to review all input screens. Previous data entries may be changed by simply typing over the old data. PF key operation for these screens are:

PF1 System Data - system and/or population values
PF2 Maintenance Configuration Problem
PF3 Equipment Design Problem
PF4 Level of Repair Analysis Problem
PF5 Spare Equipment Problem
PF6 Replacement Policy Problem
PF7 Prev Screen - display previous screen
PF8 Next Screen - display next screen
PF9 Simulate - simulate system operation
PF10 Pd Details - review detailed period reports
PF11 Run Summary - review run summary reports
PF12 Exit - exit to Scenario Management Screen

C.3.1 System Data Screens

The user must specify the general system and population parameters to be used by the model. There are three input screens used to specify these parameters (Figure 36 on page 167 through Figure 38 on page 169). These screens support user specification of:

1. model mode and simulation parameters,
2. failure environment parameters, and
3. existing end item parameters (displayed only if Model Mode B specified).

C.3.2 Maintenance Configuration Problem

The number of maintenance levels in the logistics support system must be specified. In addition, the number of repair channels at each level must be established. Each of these are design decision variables. The repair cost per day at each of these levels must also be defined. This cost would typically include the amortized cost of the repair facilities plus the operating cost.

C.3.3 Equipment Design Problem

The equipment design problem screens define end item maintenance tree values. There are three input screens used to specify these values. The primary input screen requires definition of bottom-of-the-tree node data (see Figure 40 on page 171). This includes node number, node name, failure curve, MTBF, MTTR, repair level, number of parallel components, critical item (Y or N) and number of spare parts required to repair. These values must be defined for every operating environment.

The maximum maintenance tree was previously illustrated in Figure 33 on page 161. The bounds of this structure cannot be exceeded due to self-imposed limitations of the existing software. These are not constraints of the programming language or the modeling methodology and can be removed with additional software development. Even though the maximum structure is fixed, the structure is otherwise relatively flexible. For example, at the second level of indenture, the end item can be decomposed into a maximum of four subassemblies. However, the end item does not have to be decomposed into four subassemblies. It could be decomposed into only three subassemblies or less. In addition, each of these subassemblies can individually be decomposed to varying degrees. One subassembly may be decomposed to the fifth level of indenture while another may go down to only the third level of indenture. One subassembly may be decomposed into only two nodes at the third level of indenture while another may be fully decomposed into five nodes. There are only two restrictions. These are:

1. the maximum structure cannot be exceeded, and
2. the node numbers illustrated in Figure 33 on page 161 are fixed and must be used in the decomposition.

The second restriction is imposed because the logical relationship between individual nodes in the maintenance tree is imbedded in the software. Otherwise, additional software development would be required to allow these relationships to be defined by the user upon input. User definition of these relationships would complicate the input process. The existing software developemnt represents a trade-off between flexibility, simplicity of use, and ease of software development.

End item cost and salvage values must be specified as shown in Figure 41 on page 172.

C.3.4 Level of Repair Analysis Problem

The level at which each repair will be accomplished is a design decision variable. Spare parts inventories are maintained at each repair facility. The lot sizing method used to determine replenishment shipments is another design decision variable. The model determines spare parts inventory and warehouse space requirements using a periodic review policy. Inventory cost data is also input here. The last screen in this section is for input of the stockable spare parts needed for repair of bottom-of-the-tree node failures.

C.3.5 Spare Equipment Problem

The model supports analysis of alternative modes of fielding the population of end items. The model user may select from eight predefined procurement policies or may alternatively chose to input a customized policy. The user will then be shown a screen for input of the parameters needed to implement the selected policy. After selecting a procurement policy, the initial procurement must be deployed into one or more operating environments. The last screen in this section requires the user to specify how items should be deployed. A default deployment is suggested initially, based upon the relative level of demand in each operating environment.

C.3.6 Replacement Policy Problem

The replacement policy screen supports specification of the end item retirement age and salvage value. These may differ for each operating environment.

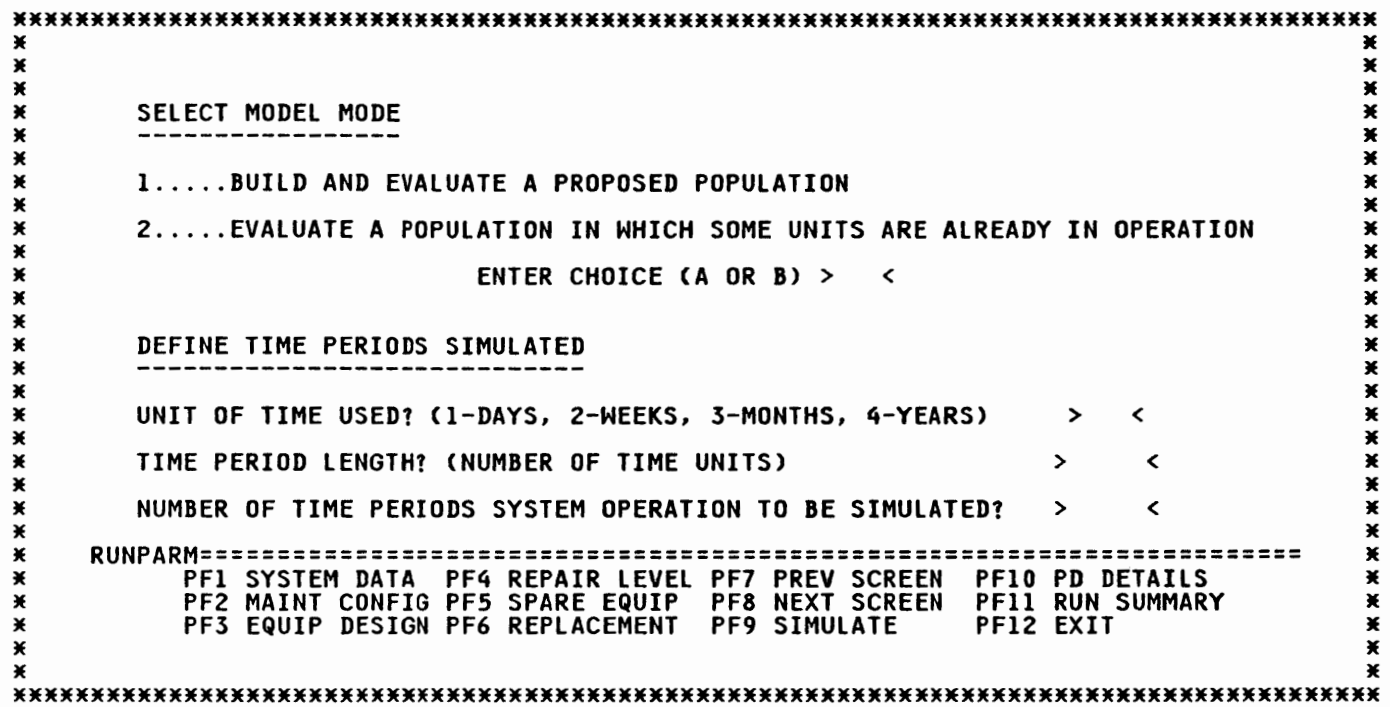


Figure 36. Simulation Parameters

```
                    FAILURE ENVIRONMENT DATA

NUMBER OF FAILURE ENVIRONMENTS? (MAX=4)    >  <

          ENV      ENVIRONMENT          END ITEMS
          NO.      DESCRIPTION          REQUIRED
          ---      -----------          --------
            1
            2
            3
            4

ENVDATA=========================================================
   PF1 SYSTEM DATA  PF4 REPAIR LEVEL  PF7 PREV SCREEN  PF10 PD DETAILS
   PF2 MAINT CONFIG PF5 SPARE EQUIP   PF8 NEXT SCREEN  PF11 RUN SUMMARY
   PF3 EQUIP DESIGN PF6 REPLACEMENT   PF9 SIMULATE     PF12 EXIT
```

Figure 37. Operating Environments

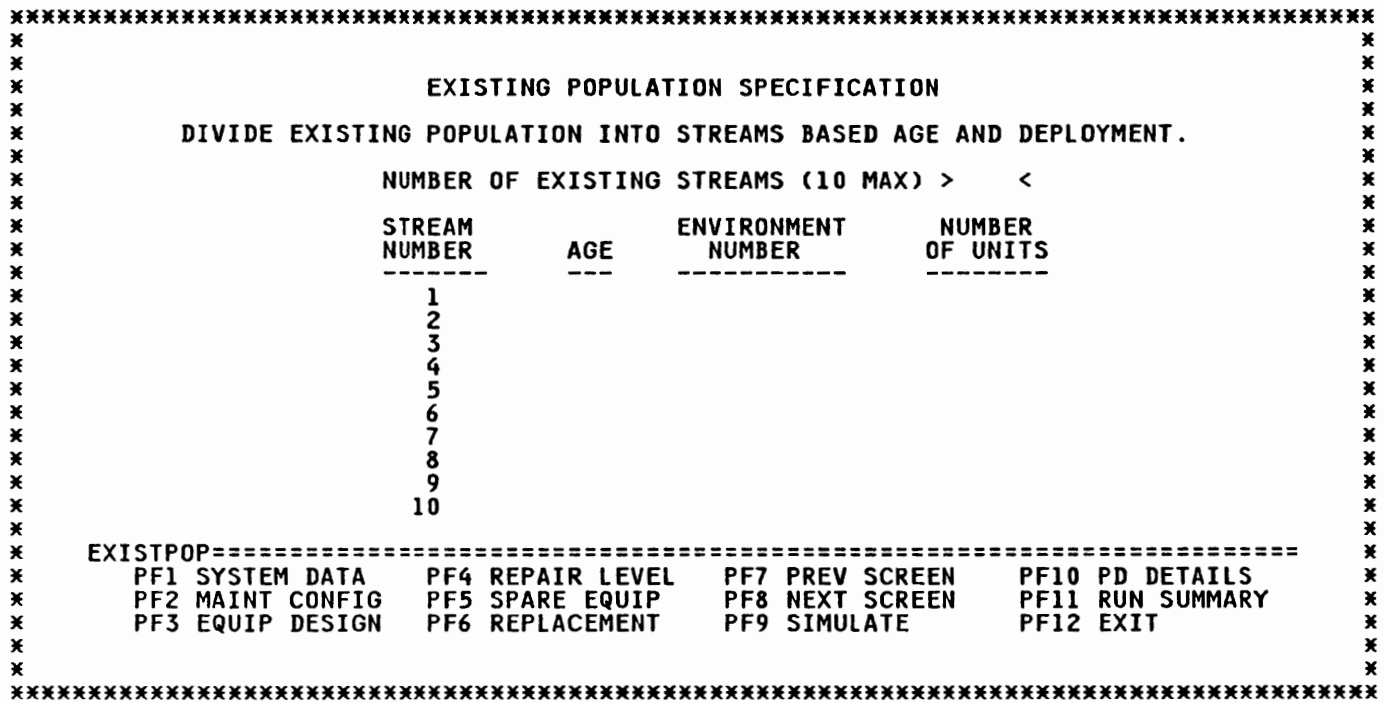


Figure 38. Existing End Items

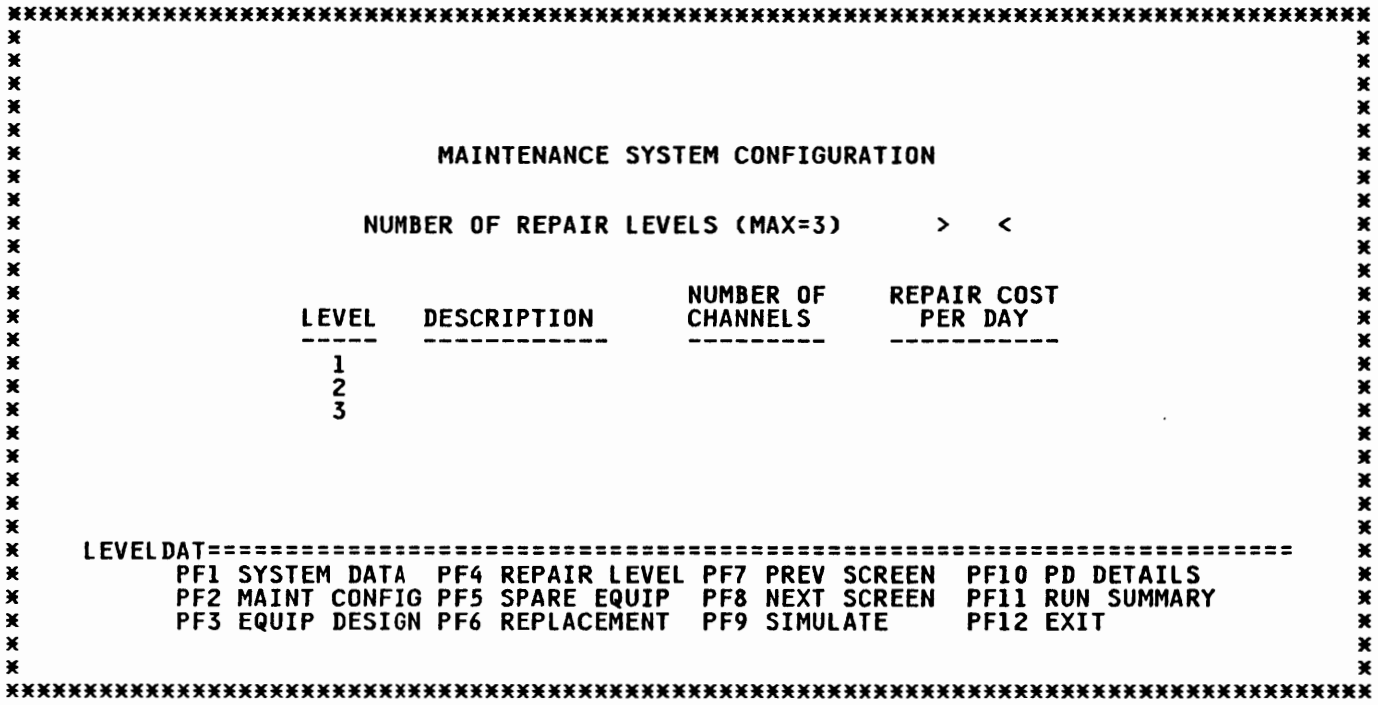


Figure 39. Levels and Channels

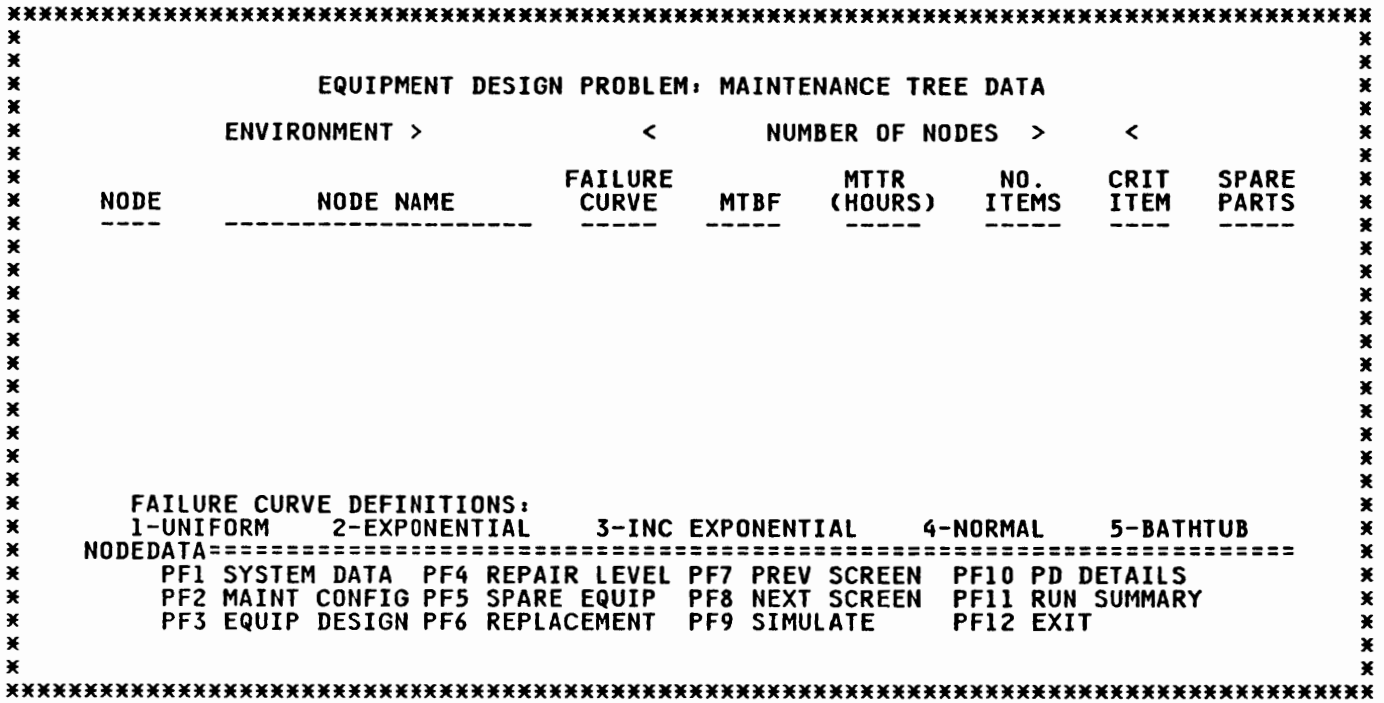


Figure 40. Maintenance Tree Data

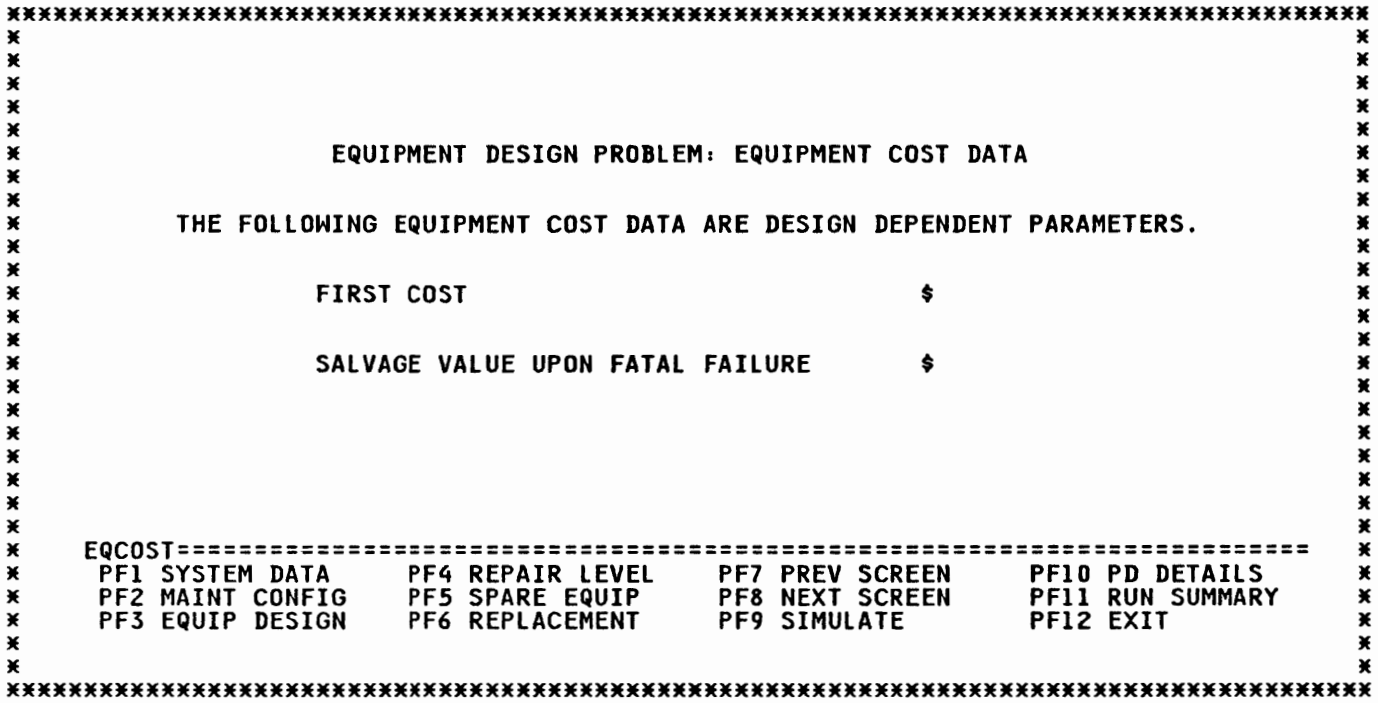


Figure 41. Repairable Item Cost

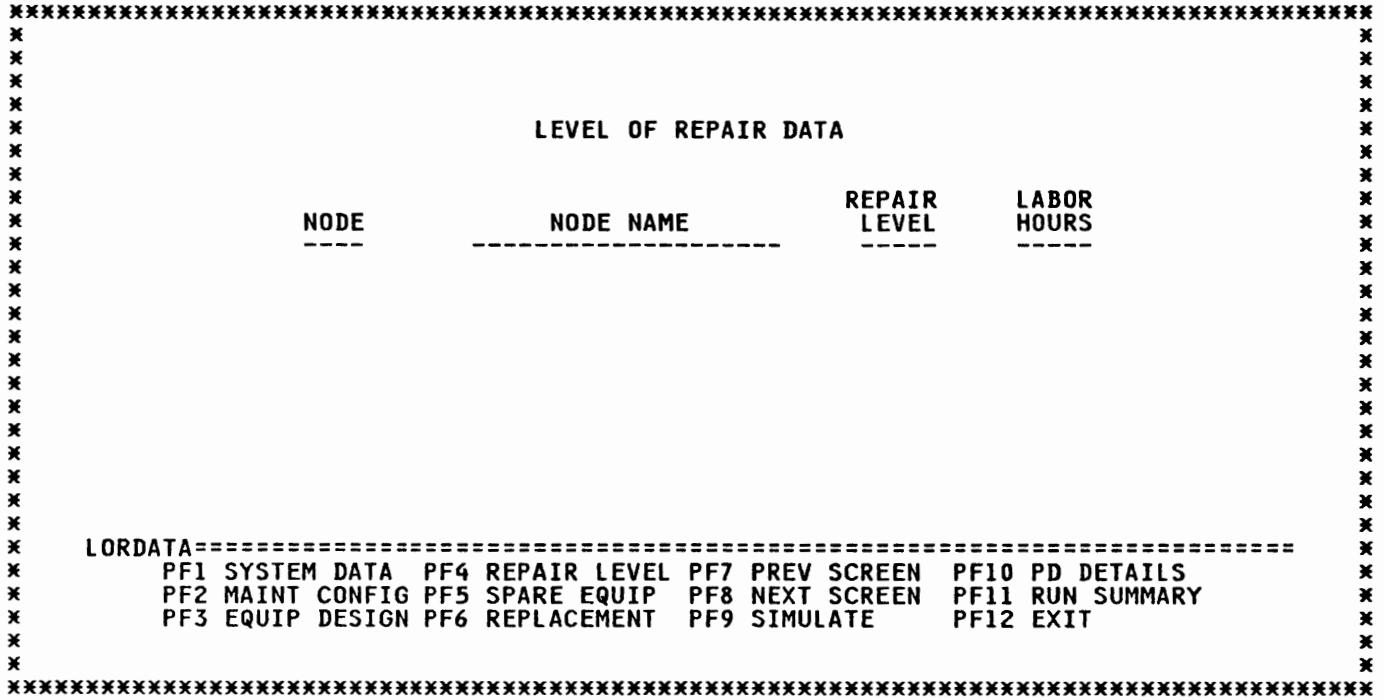


Figure 42. Level of Repair Data

```
LEVEL OF REPAIR PROBLEM: SPARE PARTS INVENTORY REPLENISHMENT

   INVENTORY COST DATA:
   --------------------
   ORDER PROCESSING COST ($/ORDER)      >     <
   WAREHOUSE SPACE COST ($/SQ.FT./PD.)  >     <
   INTEREST RATE (% PER PERIOD)         >     <

      REORDER STRATEGY (SELECT ONE):    >   <
      ------------------------------
       1. LOT-FOR-LOT QUANTITY
       2. ECONOMIC ORDER QUANTITY
       3. PERIOD ORDER QUANTITY
       4. SILVER-MEAL HEURISTIC

INVDATA==========================================================
   PF1 SYSTEM DATA   PF4 REPAIR LEVEL  PF7 PREV SCREEN  PF10 PD DETAILS
   PF2 MAINT CONFIG  PF5 SPARE EQUIP   PF8 NEXT SCREEN  PF11 RUN SUMMARY
   PF3 EQUIP DESIGN  PF6 REPLACEMENT   PF9 SIMULATE     PF12 EXIT
```

Figure 43. Spare Parts Inventory Policies

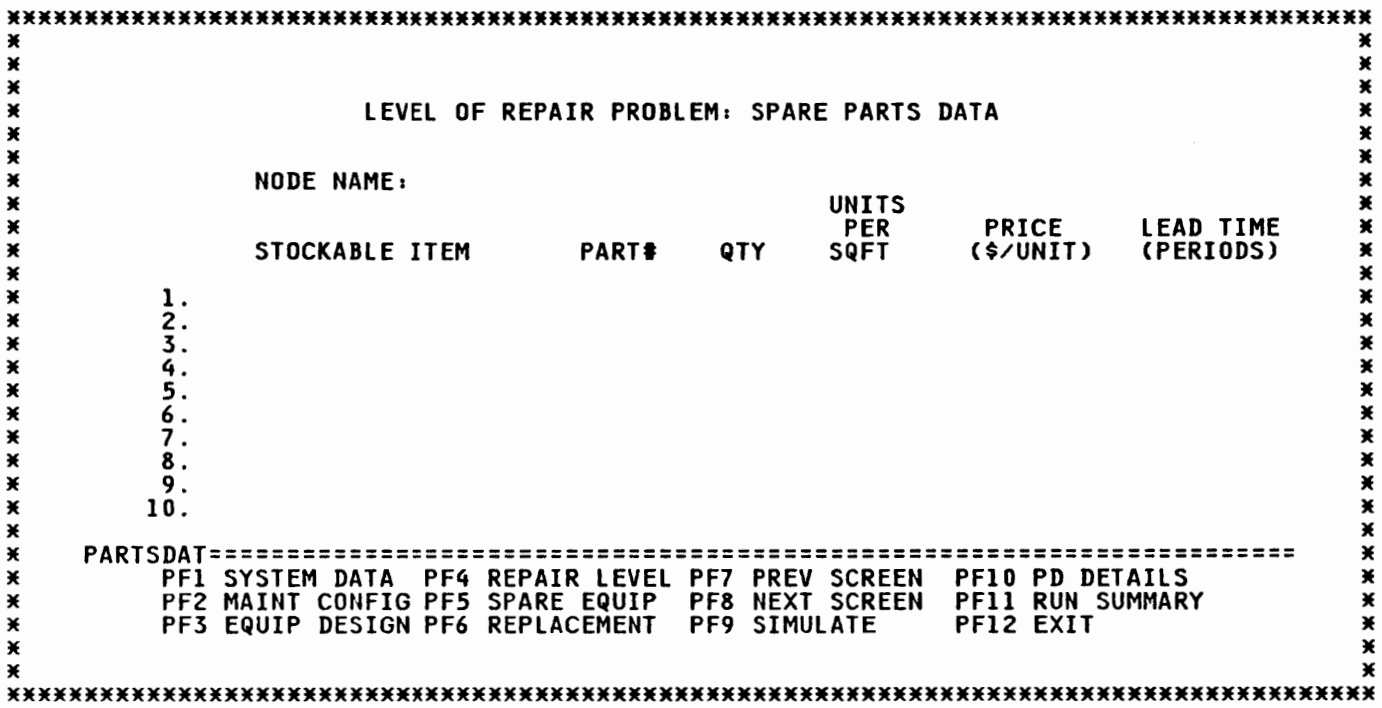


Figure 44. Spare Parts Data

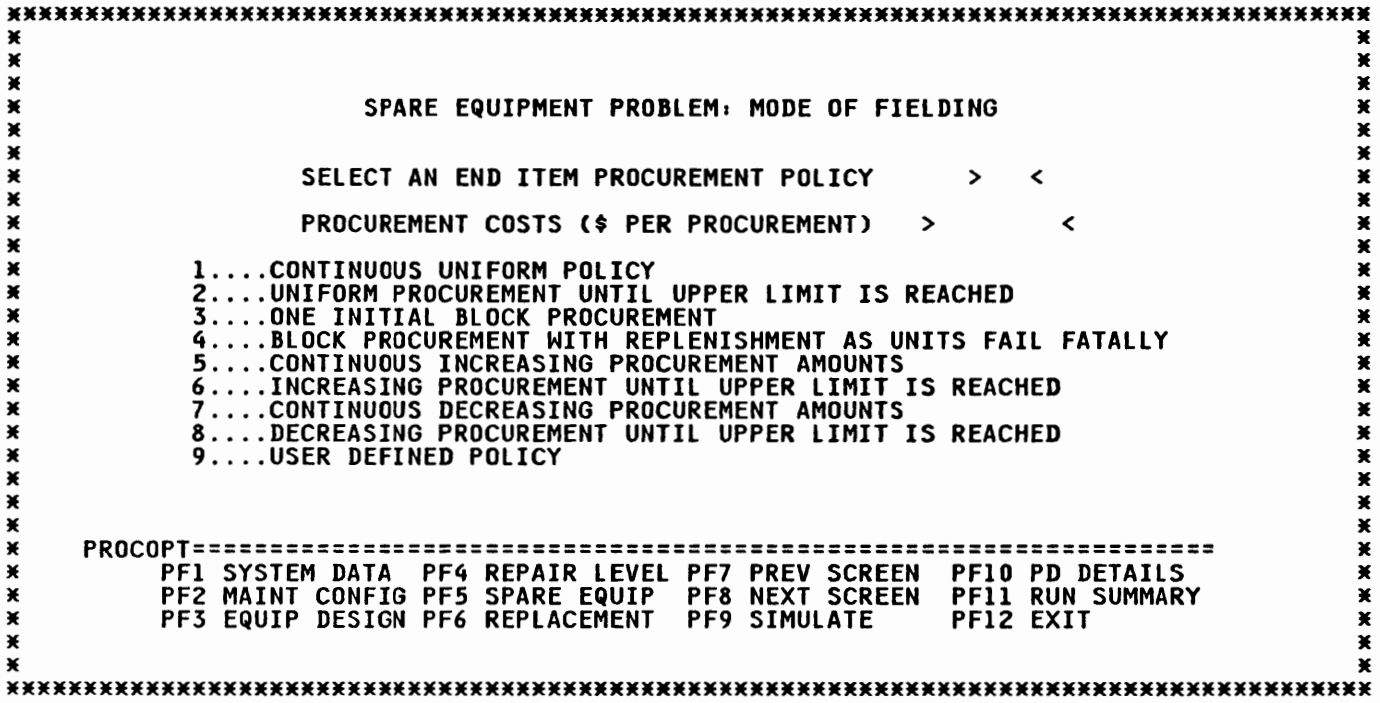


Figure 45. Procurement Policy Options

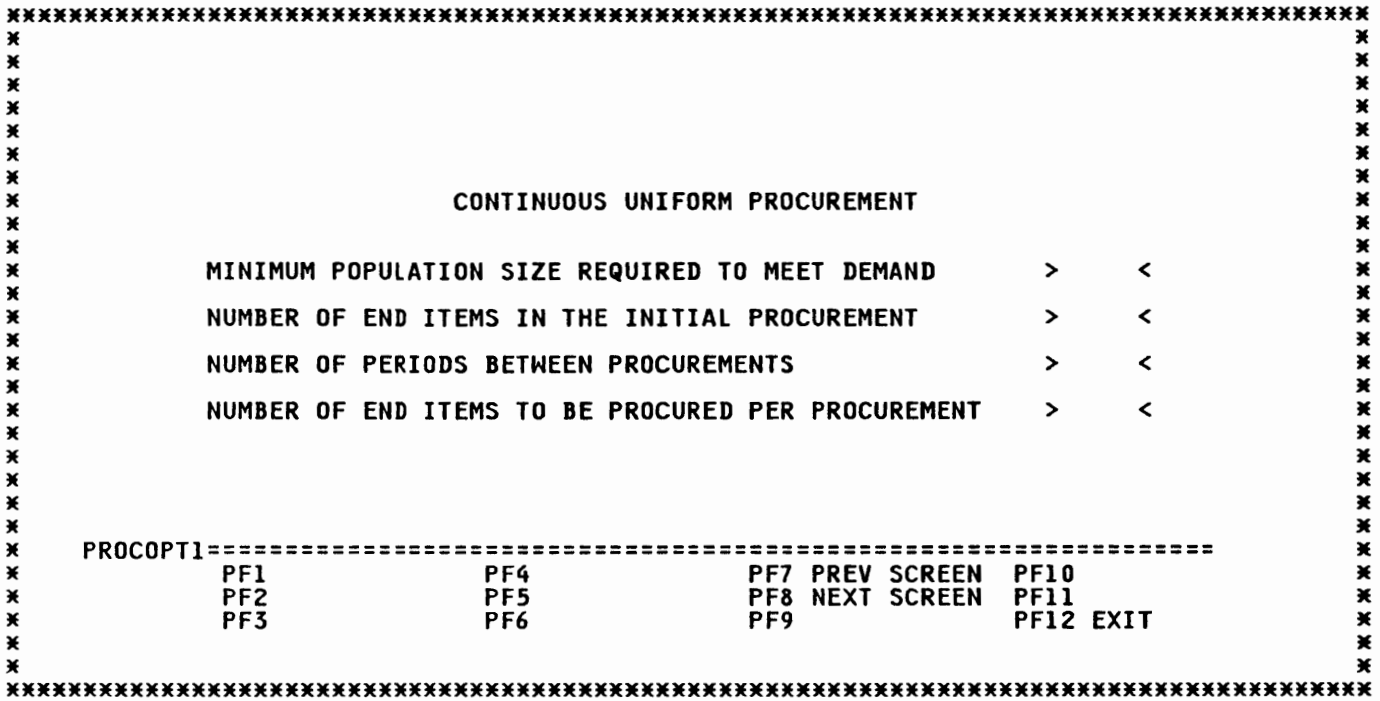


Figure 46. Procurement Policy Option #1

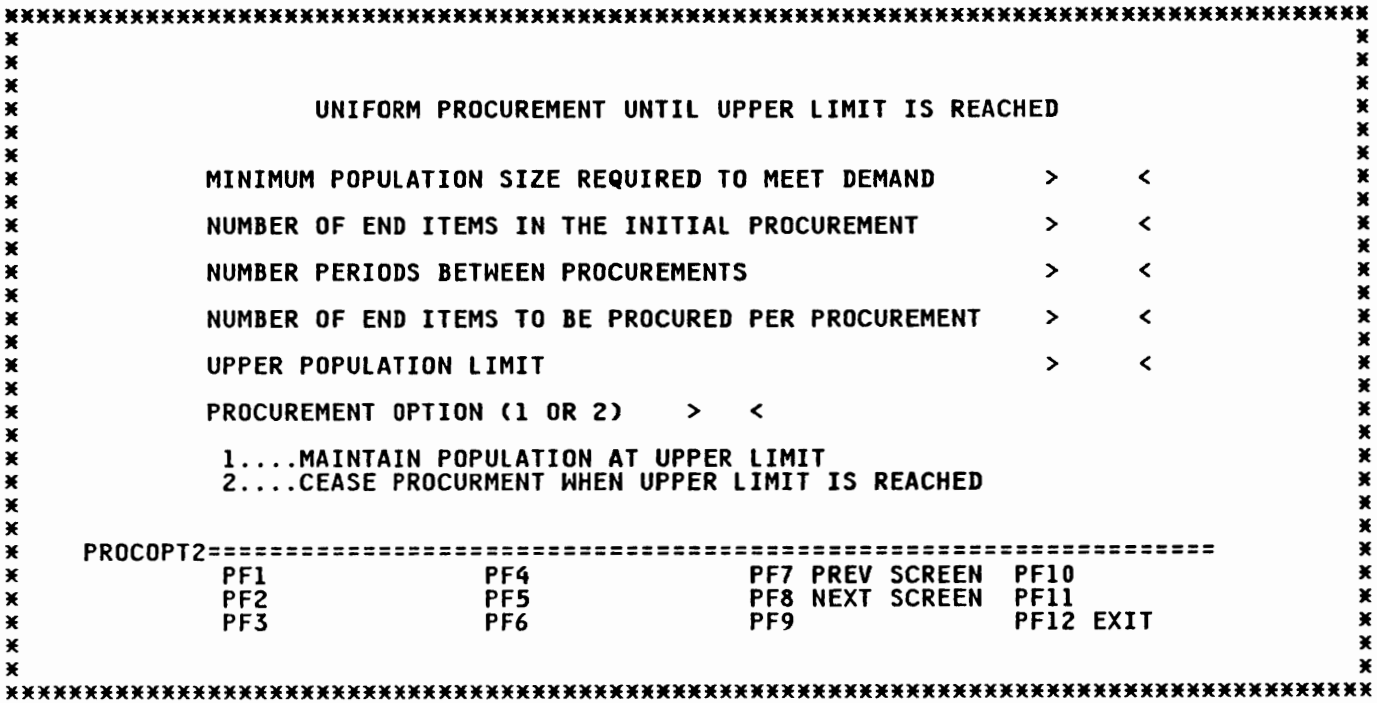


Figure 47. Procurement Policy option #2

```
                    ONE INITIAL BLOCK PROCUREMENT

  MINIMUM POPULATION SIZE REQUIRED TO MEET DEMAND      >    <

  NUMBER OF END ITEMS IN THE INITIAL BLOCK PROCUREMENT  >    <

PROCOPT3=============================================================
     PF1                PF4                PF7 PREV SCREEN  PF10
     PF2                PF5                PF8 NEXT SCREEN  PF11
     PF3                PF6                PF9              PF12 EXIT
```

Figure 48. Procurement Policy option #3

```
                    BLOCK PROCUREMENT WITH REPLENISHMENT AS UNITS FAIL FATALLY

    MINIMUM POPULATION SIZE REQUIRED TO MEET DEMAND      >     <
    NUMBER OF END ITEMS IN INITIAL PROCUREMENT            >     <
    NUMBER OF PERIODS BETWEEN PROCUREMENTS                >     <

PROCOPT4=============================================================
      PF1                PF7 PREV SCREEN  PF10
      PF2                PF8 NEXT SCREEN  PF11
      PF3                PF9              PF12 EXIT
      PF4
      PF5
      PF6
```

Figure 49. Procurement Policy option #4

```
                    INCREASING PROCUREMENT AMOUNTS

   MINIMUM POPULATION SIZE REQUIRED TO MEET DEMAND    >    <
   NUMBER OF END ITEMS IN INITIAL PROCUREMENT          >    <
   NUMBER OF PERIODS BETWEEN PROCUREMENTS              >    <
   PERCENT INCREASE IN PROCUREMENT AMOUNTS             >    <

PROCOPT5=========================================================
     PF1          PF4          PF7 PREV SCREEN  PF10
     PF2          PF5          PF8 NEXT SCREEN  PF11
     PF3          PF6          PF9              PF12 EXIT
```

Figure 50. Procurement Policy option #5

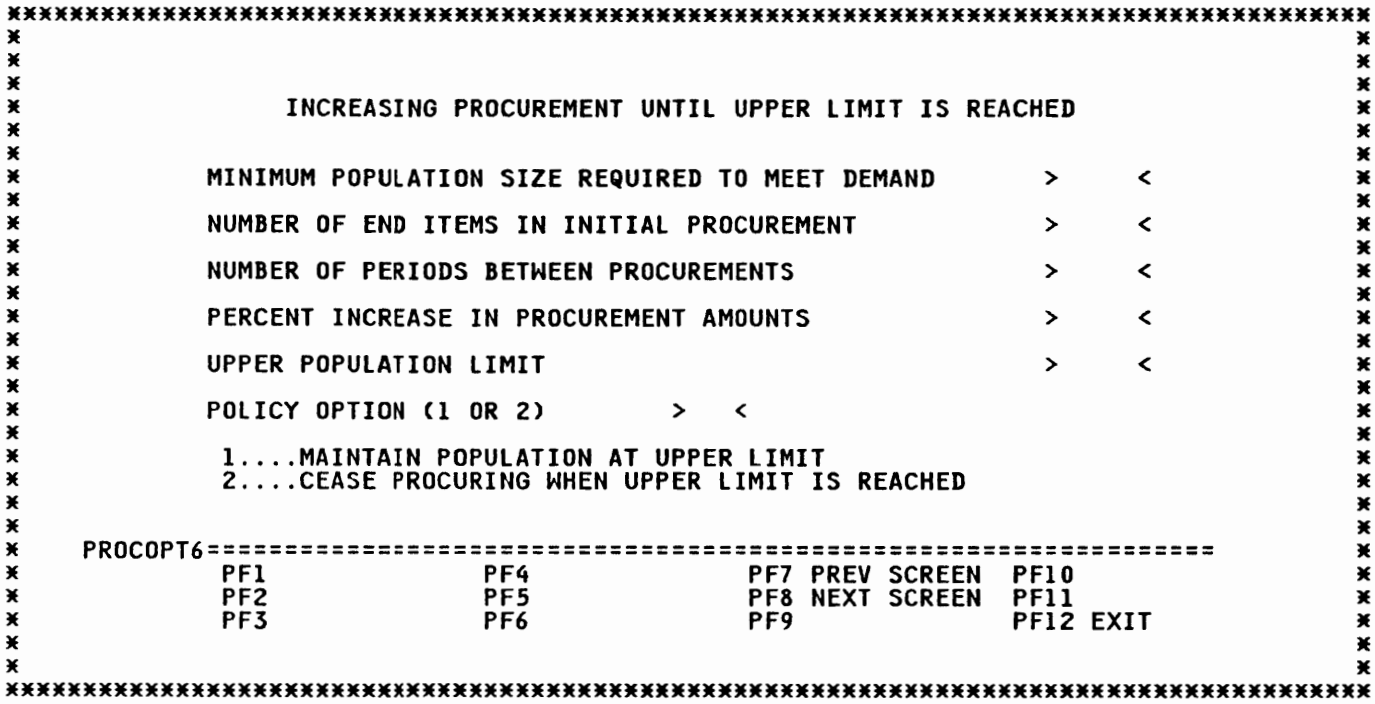


Figure 51. Procurement Policy option #6

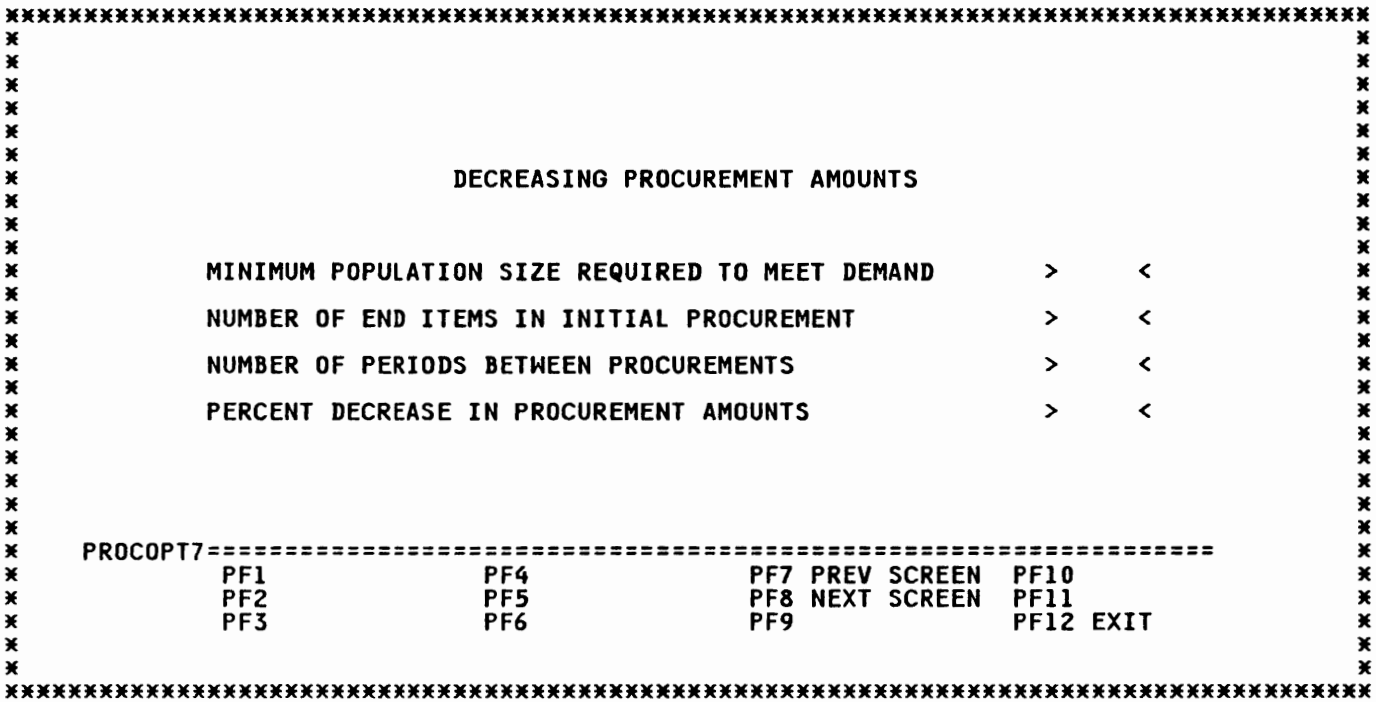


Figure 52. Procurement Policy option #7

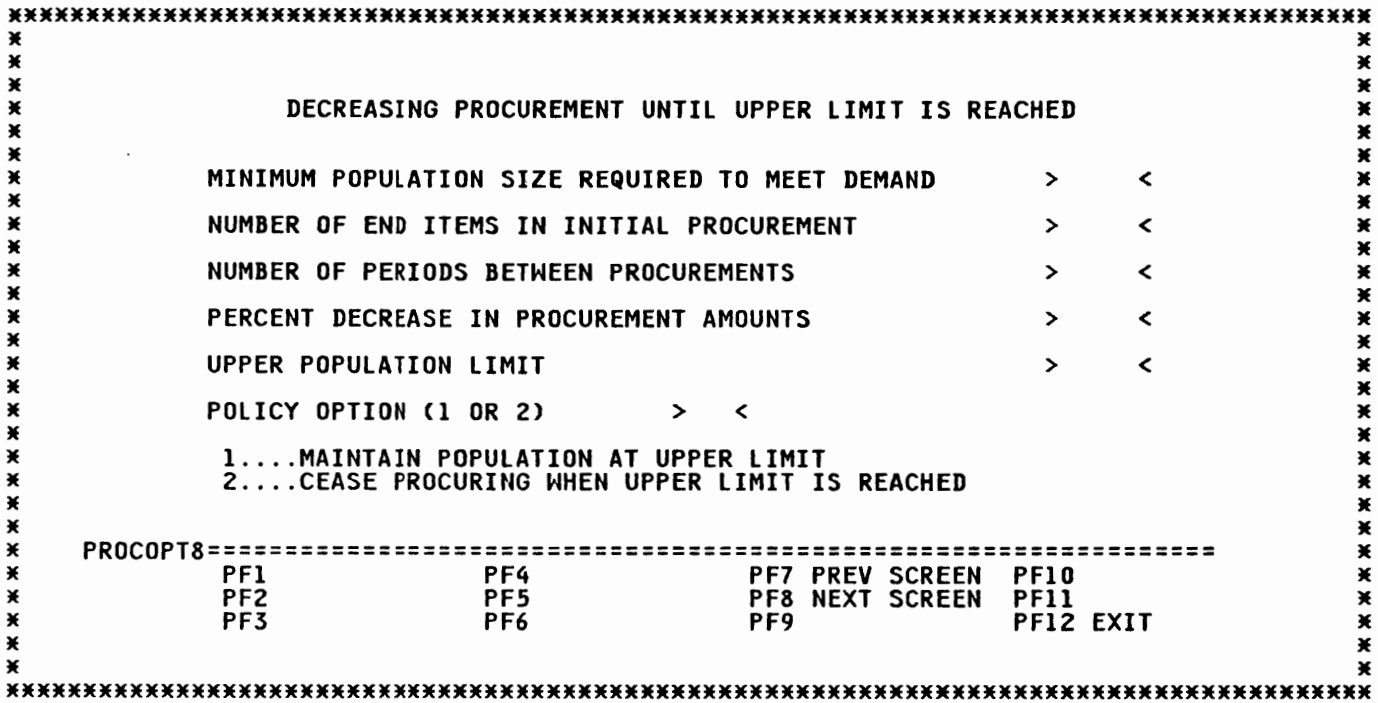


Figure 53. Procurement Policy option #8

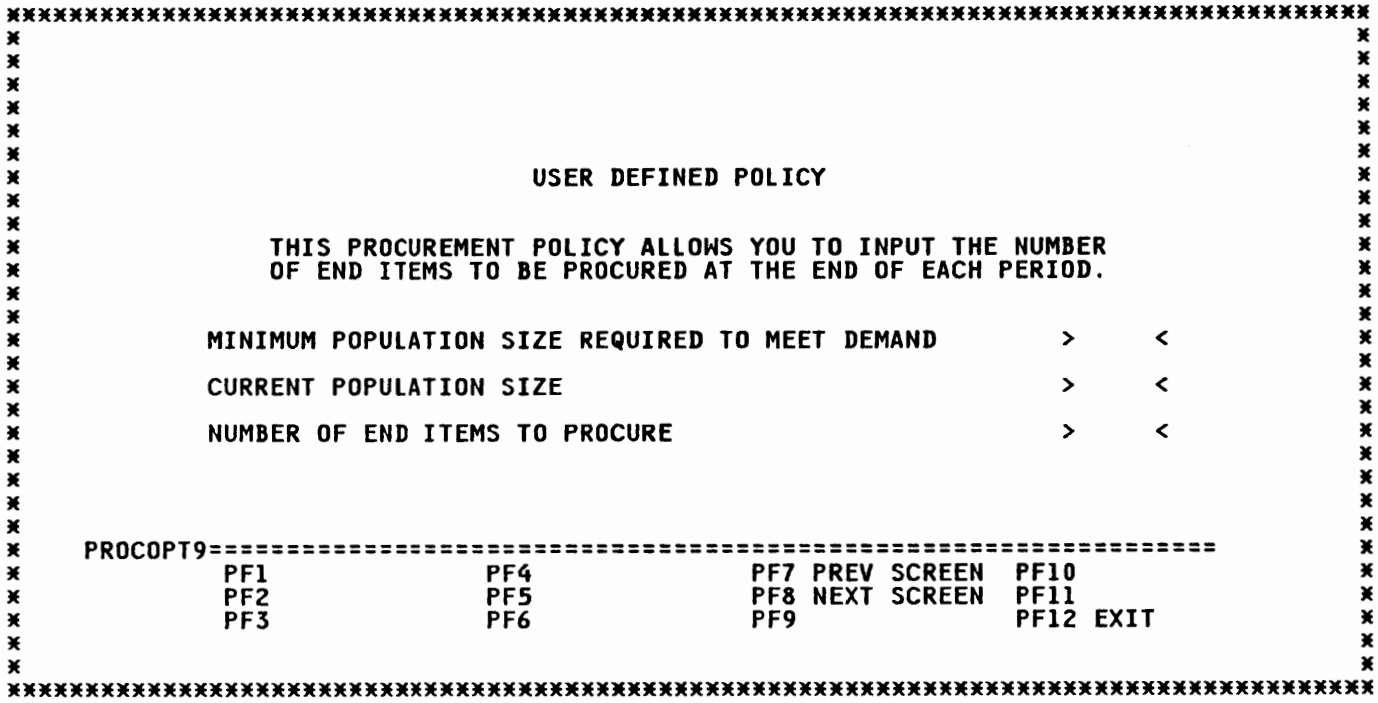


Figure 54. Procurement Policy option #9

Figure 55. Deploy Procured Items

```
SPARE EQUIPMENT PROBLEM: DEPLOY PROCUREMENT

DEPLOY END ITEM PROCUREMENT INTO OPERATING ENVIRONMENTS.

UNITS PROCURED THIS PERIOD     >     <

ENVIRONMENT       AGE                          EXISTING     NEW UNITS
DESCRIPTION       RETIRED      DEMAND          DEPLOYMENT   DEPLOYED
------------      -------      ------          ----------   ---------

DEPLOY==========================================================================
 PF1 SYSTEM DATA     PF4 REPAIR LEVEL    PF7 PREV SCREEN     PF10 PD DETAILS
 PF2 MAINT CONFIG    PF5 SPARE EQUIP     PF8 NEXT SCREEN     PF11 RUN SUMMARY
 PF3 EQUIP DESIGN    PF6 REPLACEMENT     PF9 SIMULATE        PF12 EXIT
```

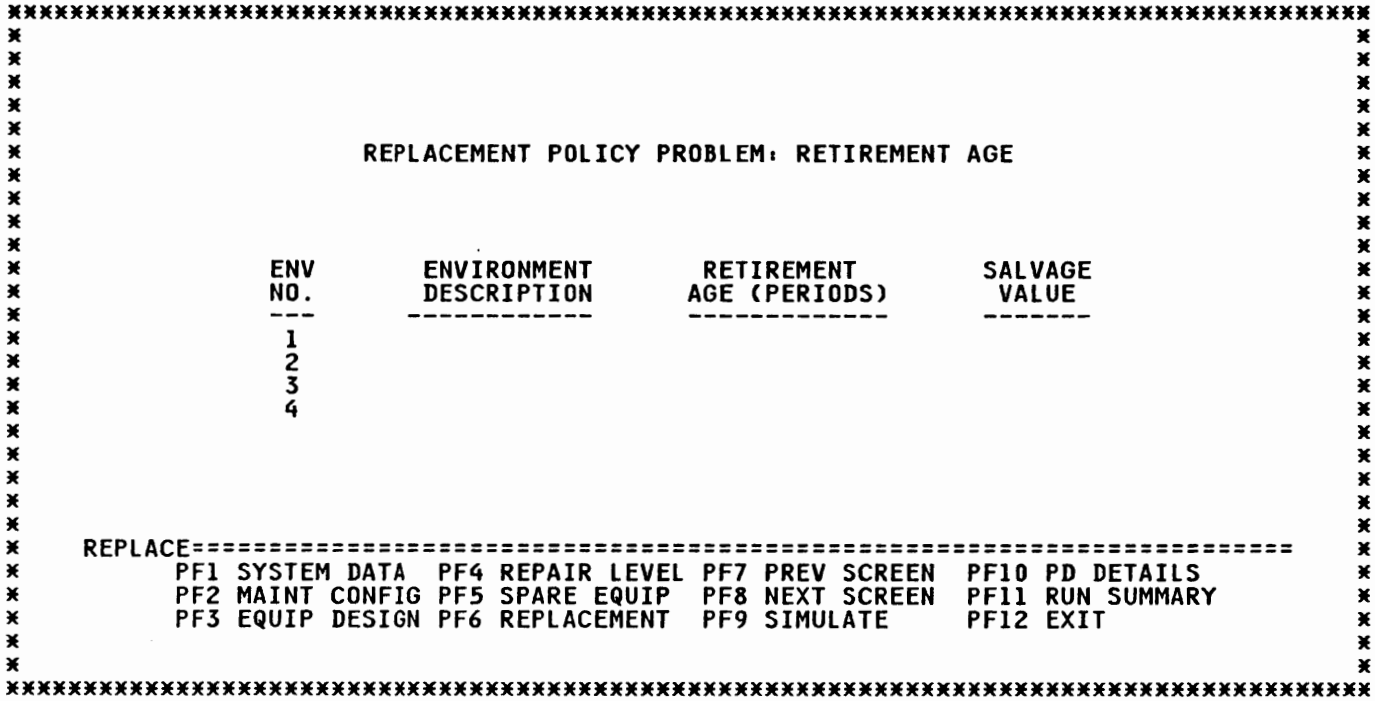


Figure 56. Retirement Age

C.4 Period Detail Reports

The Period Detail Reports are a series of six output screens designed to allow the model user to examine specific, detailed aspects of system performance on a period-by-period basis. The time period used to produce these reports is specified by the user and is the time period used in the population simulation. Each report is produced for each time period simulated. The six reports available in CODAS provide the model user information on node failure probabilities, spare parts requirements, end item availability, warehouse space requirements, labor requirements for item repair, and period costs.

Random access to each report is provided through PF keys. PF keys are also used to view these reports for each period simulated and to provide access to other system functions. PF key operations for these reports are summarized below.

PF1 Node P(Fail) - display node failure probabilities
PF2 Spare Parts - display spare parts requirements
PF3 Availability - display end item availability
PF4 WHSE Space - display warehouse space requirements
PF5 Labor Reqmt - display labor requirements
PF6 Period Cost - display period cost report
PF7 Prev Period - display previous period report
PF8 Next Period - display next period report
PF10 Run Summary - access run summary reports
PF11 Model Inputs - access model input data screens
PF12 Exit - exit to Scenario Management Screen

C.4.1 Node Failure Probabilities Report

The Node Failure Probabilities Report shows the probability of failure for maintenance tree nodes 1 to 26. These nodes represent the first three levels of indenture in the end item maintenance tree. Display screen limitations prohibit output of all 126 nodes in the maintenance tree. Node failure probabilities are used to determine the expected number of end item failures and the resulting spare parts, labor, and warehouse requirements; end item availability; and period costs. The screen also provides current and historical information on the number of end items in each stream. In addition to the standard PF keys, PF9 Next Stream, is used to step through the streams.

C.4.2 Spare Parts Report

Spare parts inventory requirements are determined using a periodic review policy.

C.4.3 Availability Report

Availability is the probability of there being enough units in operation to meet the given demand. Demand is known for each operating environment and is not meaningful at the stream level. Therefore, all streams for a given environment are aggregated to determine availability.

C.4.4 Warehouse Space Report

This report gives warehouse space requirements at each level of repair. These requirements are driven by the spare parts inventory requirements identified earlier.

C.4.5 Labor Requirements Report

This report summarizes the period labor hours needed to accomplish critical and noncritical repair of failed items within each environment at each repair facility.

C.4.6 Period Costs Report

This report provides information on repair costs, inventory costs, and capital costs (new end items purchase cost) during each period for each operating environment.

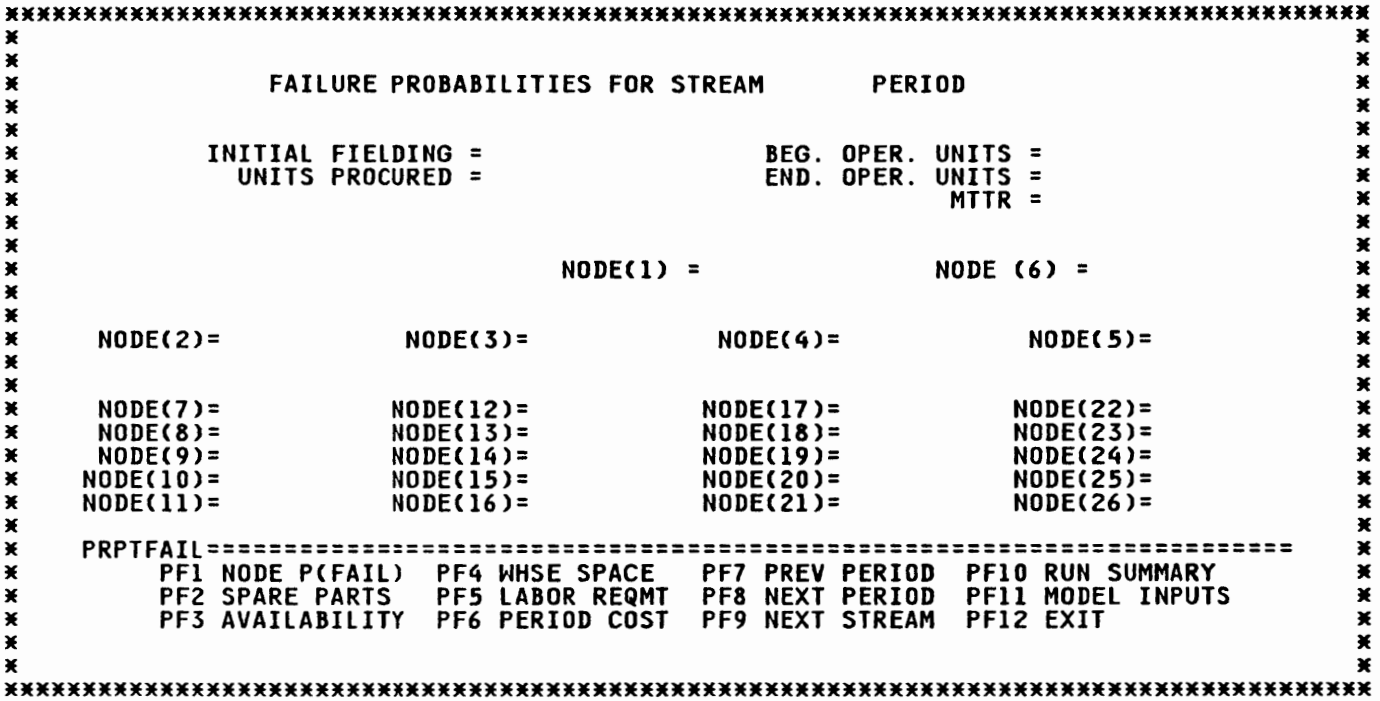


Figure 57. Node Failure Probabilities

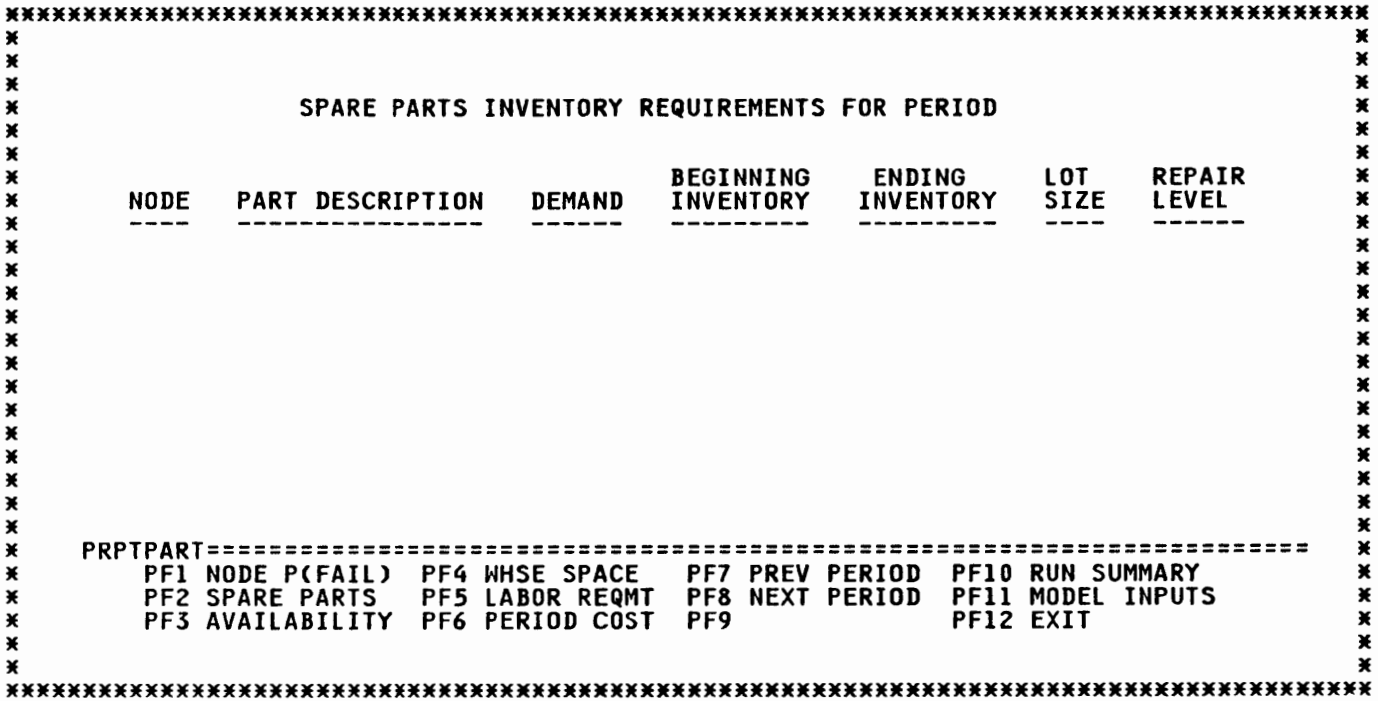


Figure 58. Spare Parts Inventory Requirements

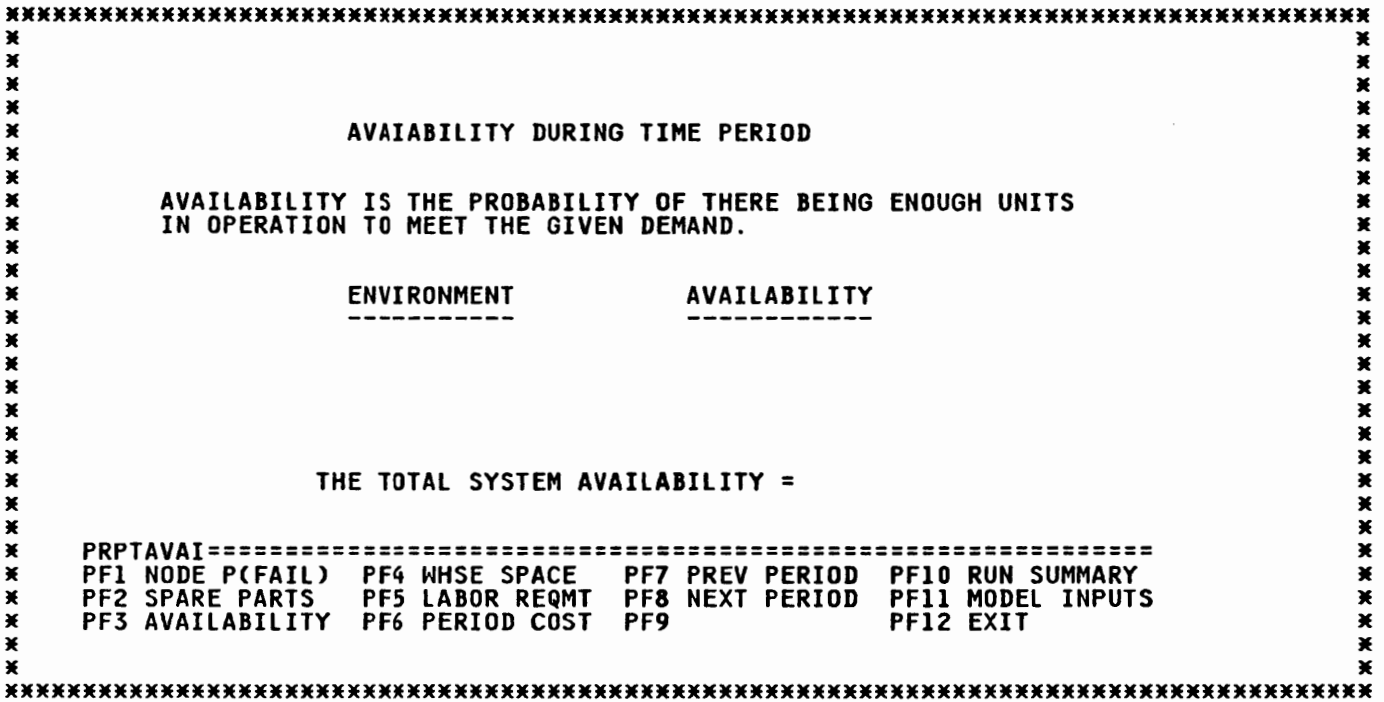


Figure 59. End Item Availability

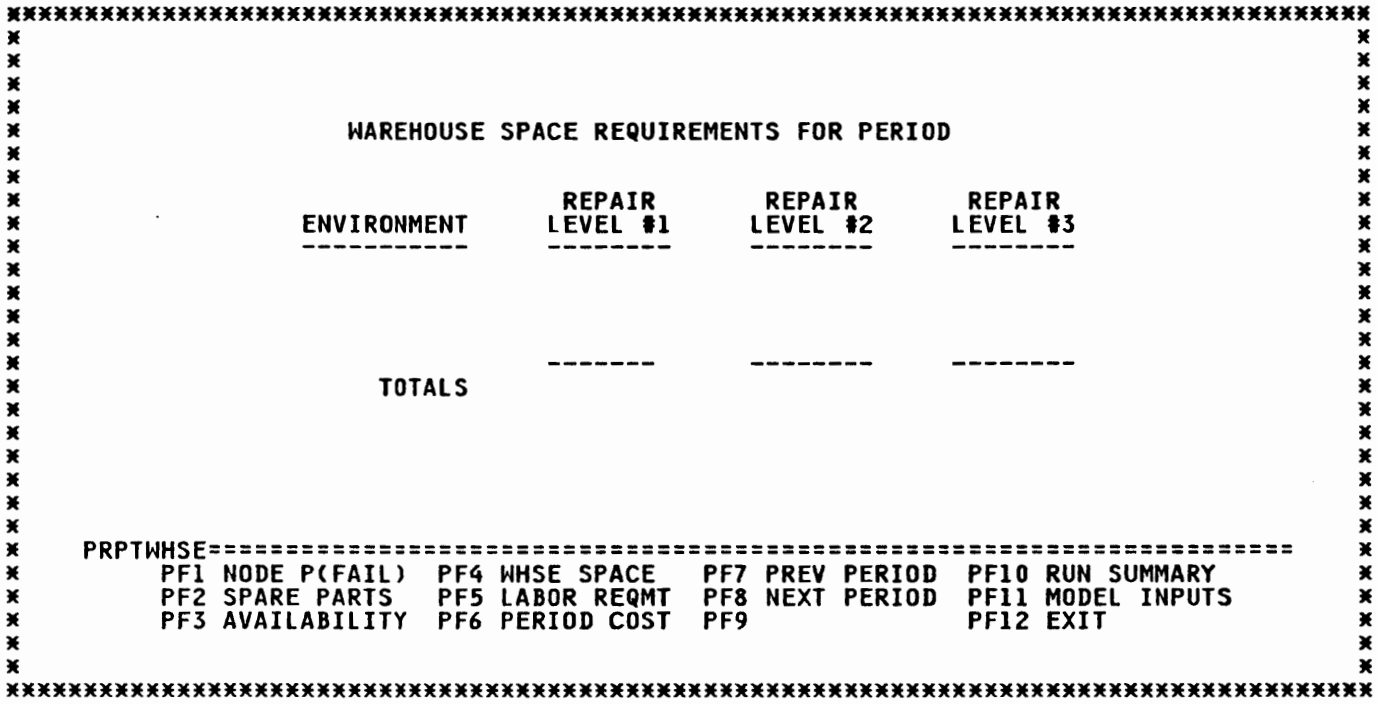


Figure 60. Warehouse Space Requirements

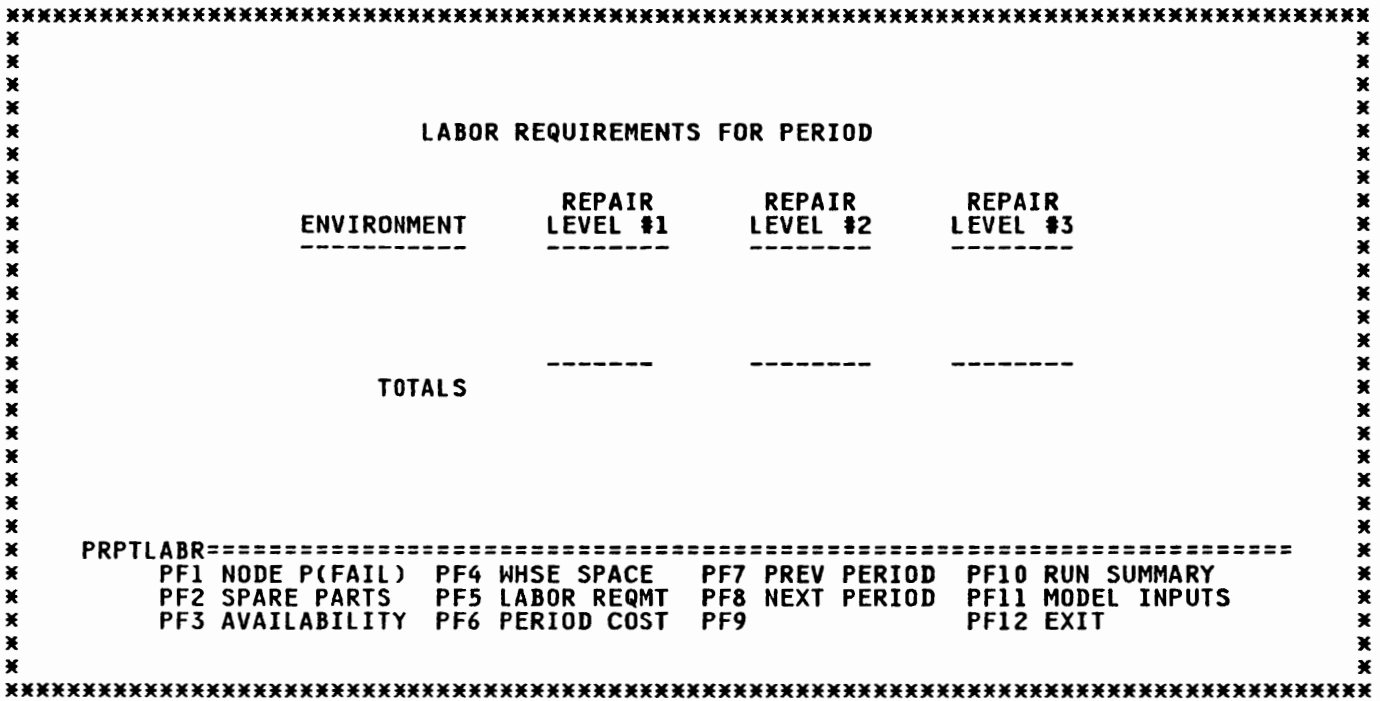


Figure 61. Labor Requirements

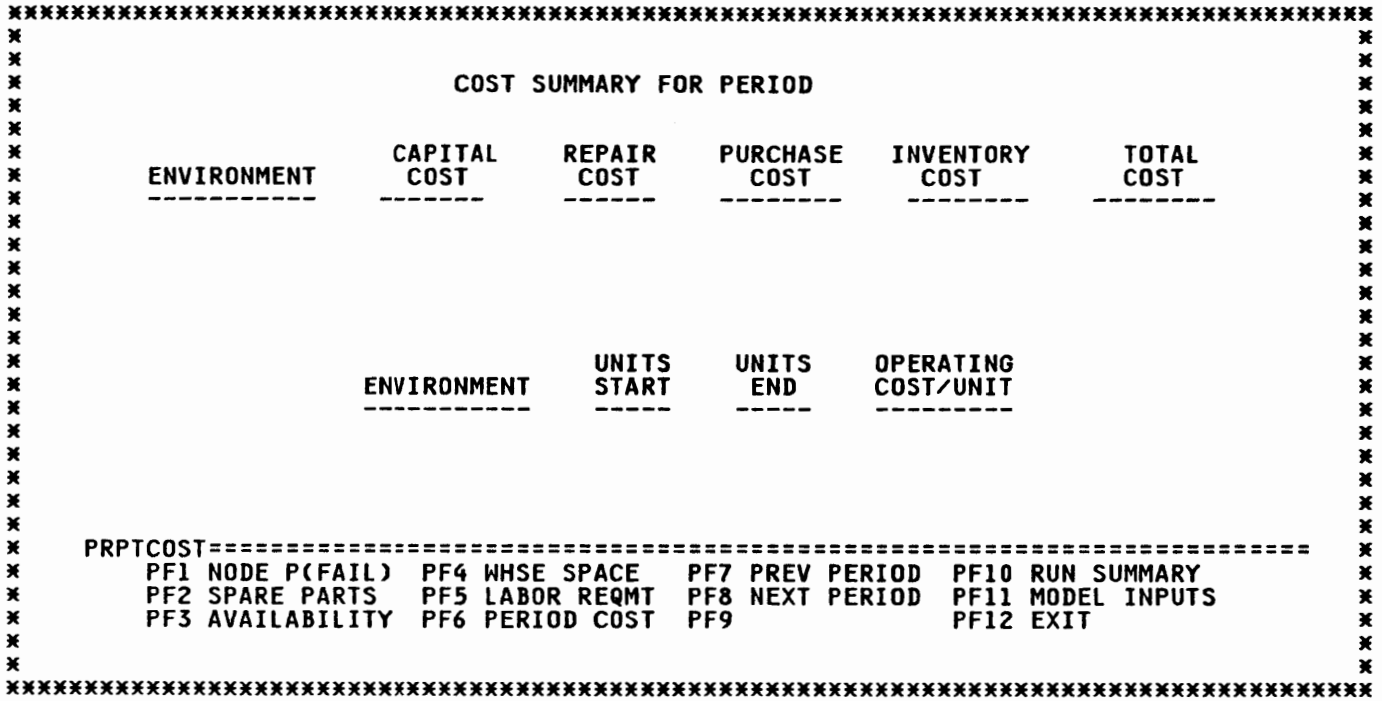


Figure 62. Period Costs

C.5 Run Summary Reports

The Run Summary Reports are a series of seven output screens for evaluation of overall population and logistics system performance over the time periods simulated. These seven reports provide the user information on end item failure probabilities, spare parts requirements, end item availability, warehouse requirements, labor requirements, life cycle costs, and a multicriteria measure of overall system performance. Random and sequential access to these reports is provided through PF keys. The PF key definitions for these reports are:

PF1 Item P(Fail) - end item failure probabilities
PF2 Spare Parts - spare parts requirements
PF3 Availability - end item availability
PF4 WHSE Space - warehouse space requirements
PF5 Labor Reqmt - labor requirements
PF6 Cost Sumry - life cycle cost summary
PF7 Prev Screen - previous screen
PF8 Next Screen - next screen
PF9 Design Eval - design evaluation model
PF10 PD Detail - access period detail reports
PF11 Model Inputs - access model input data screens
PF12 Exit - exit to Scenario Management Screen

C.5.1 End-Item Failure Probabilities

This report summarizes the cumulative end item (Node 1) failure probabilities for end items operating in the different failure environments simulated.

C.5.2 Spare Parts Requirements

This report summarizes the spare parts requirements for each node in the maintenance tree.

C.5.3 End-Item Availability

This report summarizes end item availability by environment for each period simulated.

C.5.4 Warehouse Space Requirements

This report summarizes warehouse space required to store the spare parts inventories needed at each level of repair.

C.5.5 Labor Requirements

This report summarizes the labor hours needed to accomplish end item repairs at each repair facility over the time periods simulated.

C.5.6 Life-Cycle Costs

This report summarizes period repair costs, inventory costs, purchase costs, and capital costs for each operating environment. The PF7 and PF8 function keys are used to view the cost performance for the different environments.

C.5.7 Multicriteria Decision Model

This report allows the model user to evaluate the overall performance of the logistics system based upon a weighted scoring of four criteria. The user must define the minimum (score = 0) and maximum (score = 10) values and the weight for each criteria. The model determines system performance for each of these criteria and computes a system evaluation measure based upon the scales and weights defined by the user. The user may examine the effect of different scales and/or weights on the system evaluation measure by changing these and pressing PF9 to recompute the measure.

C.5.8 Optimization

Once a design alternative has been completely specified, system operation can be simulated and design performance evaluated. The design evaluation is the first run summary report displayed (the multicriteria decision model). The user would primarily use this screen and the input screens to investigate alternative system designs. The remaining report screens would be primarily used to provide insight to design improvement opportunities. For example, if the reliability performance needed to be improved, the Node Failure Probabilities Report could be investigated to identify those nodes with high failure probabilities. Parallel redundant components could be used to improve this failure probability or an alternative supplier's component could be used to improve reliability.

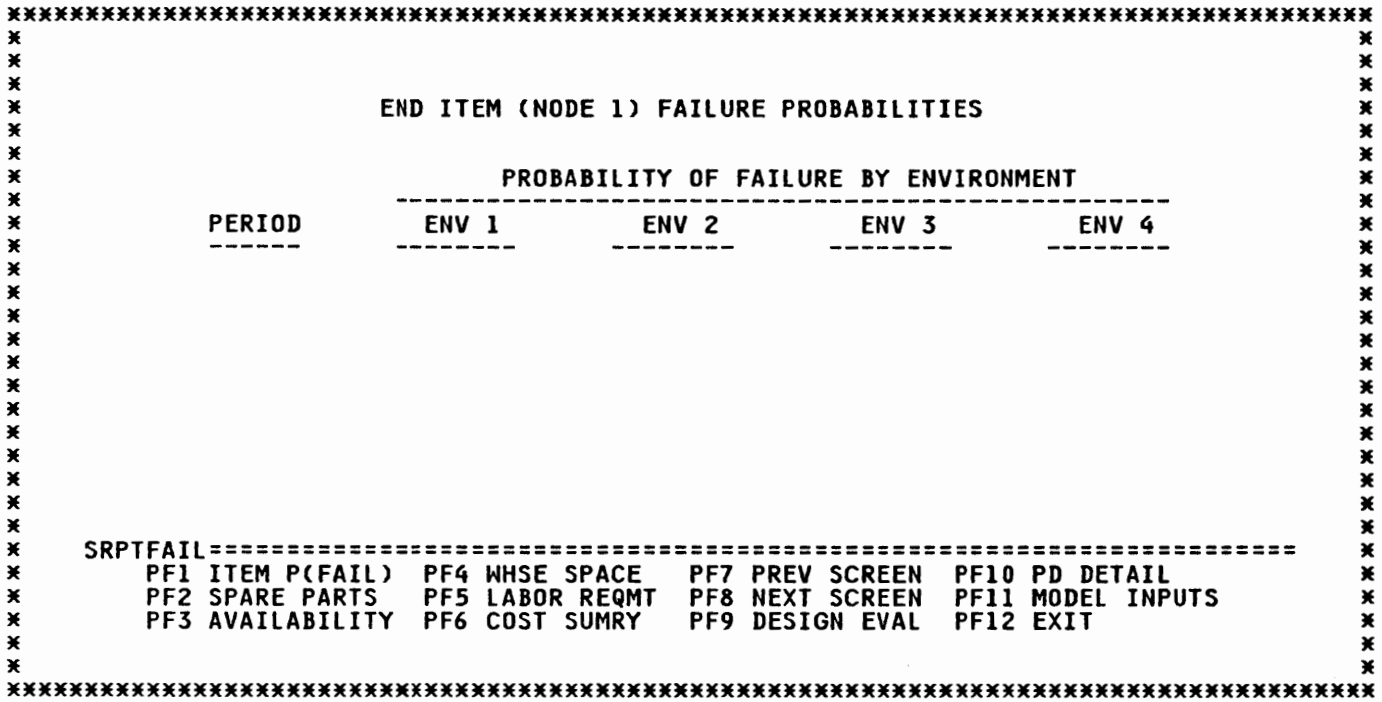


Figure 63. End Item Failure Probabilities

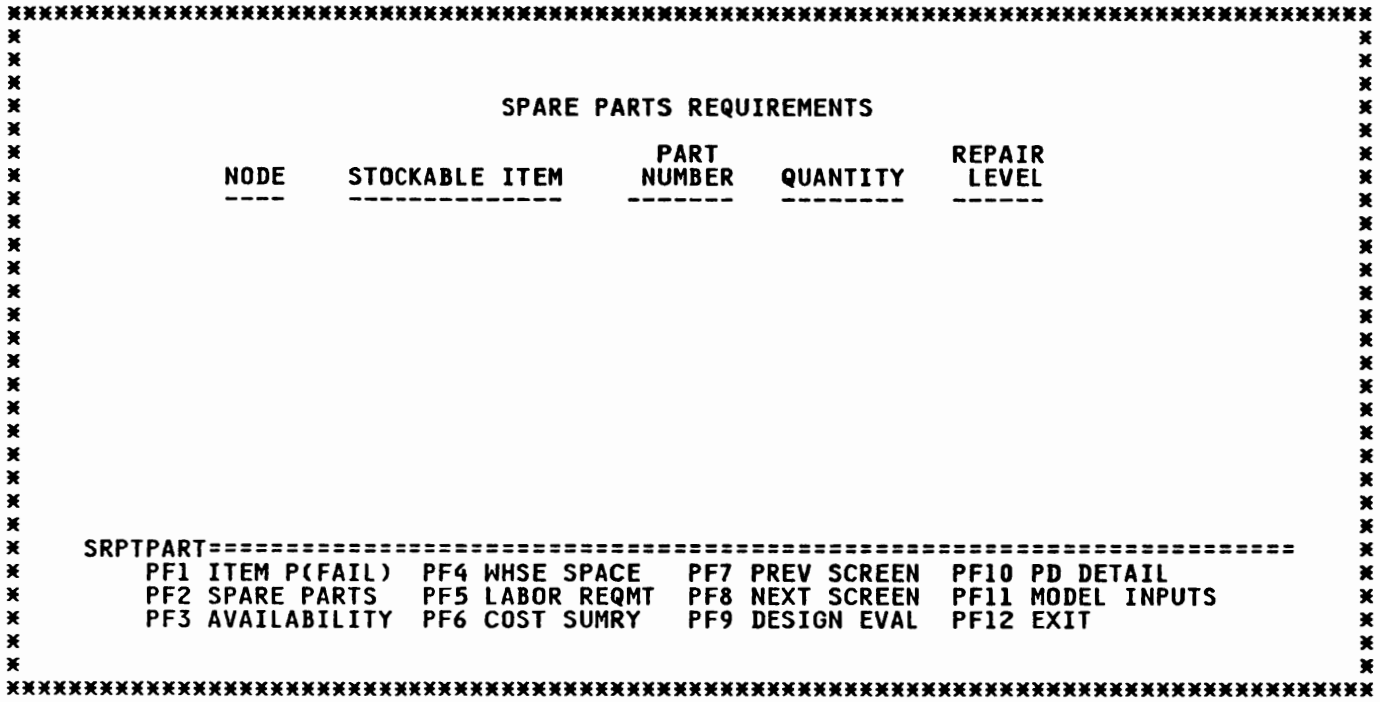


Figure 64. Spare Parts Requirements

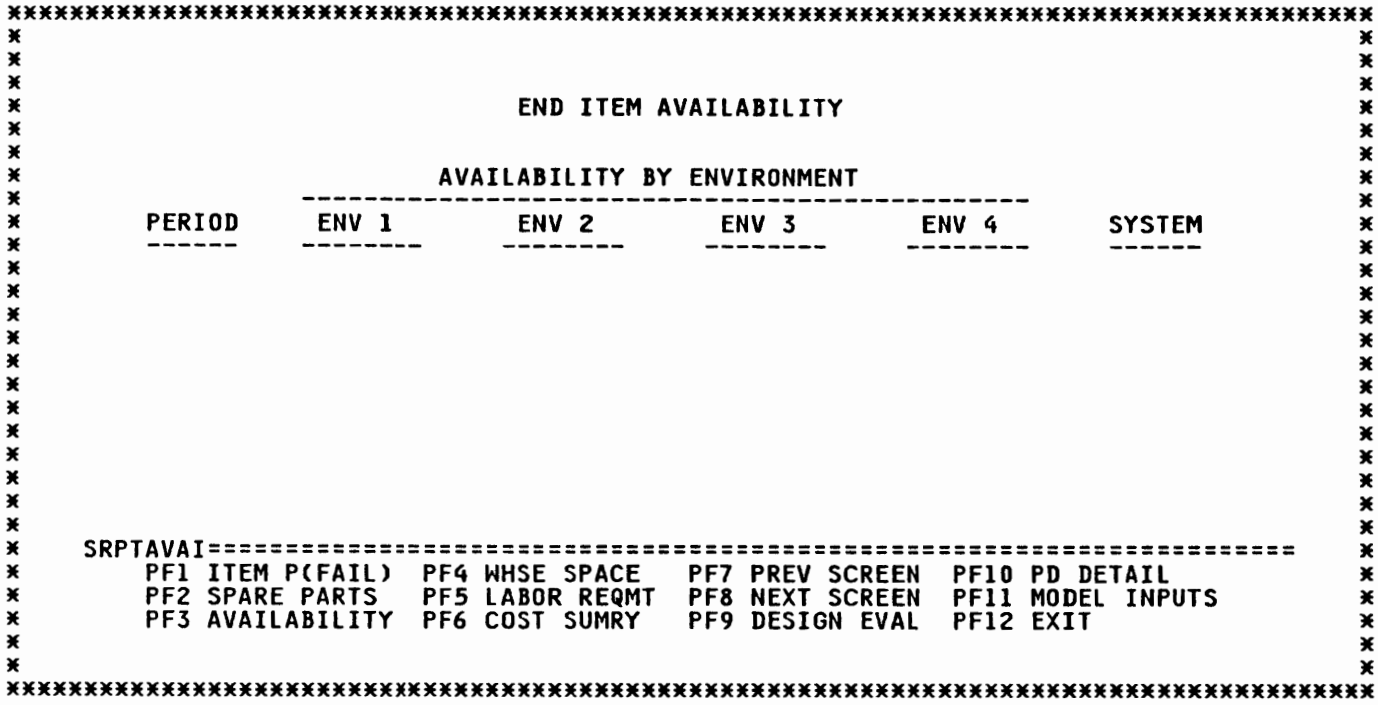


Figure 65. End Item Availability

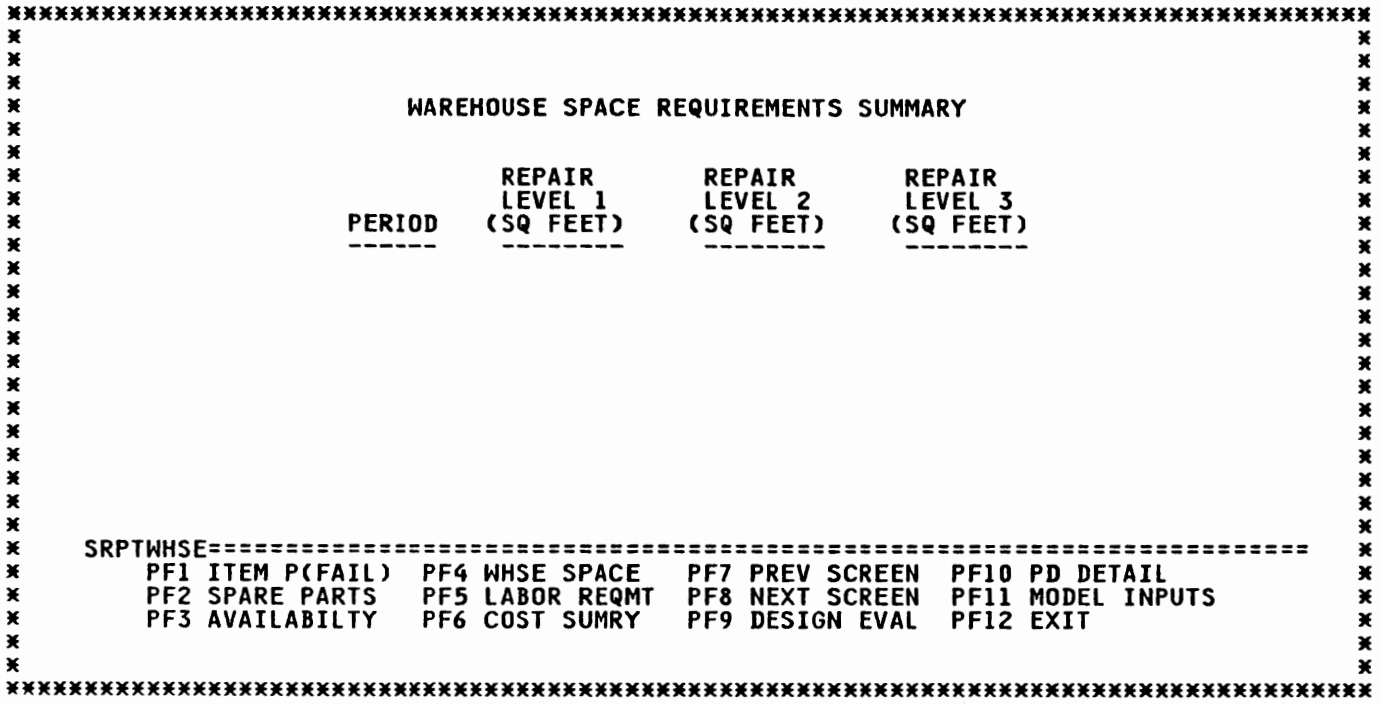


Figure 66. Warehouse Space Requirements

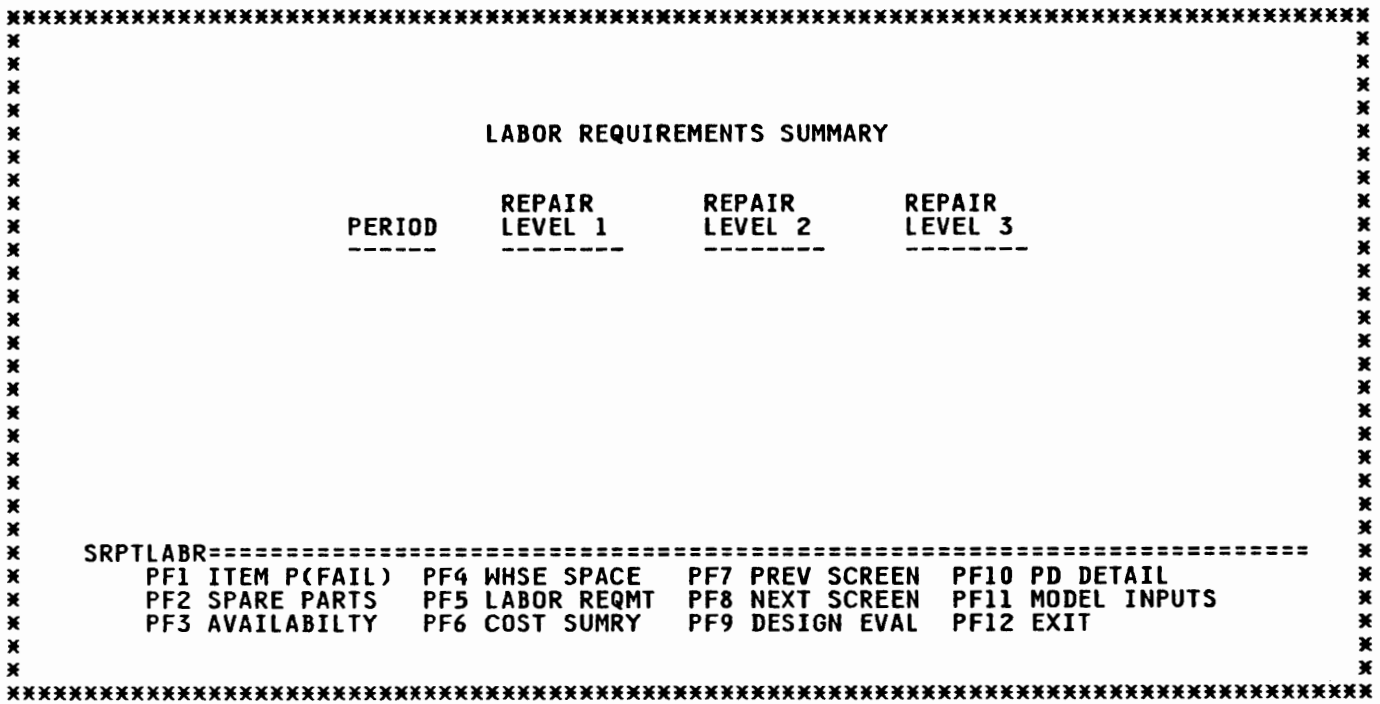


Figure 67. Labor Requirements

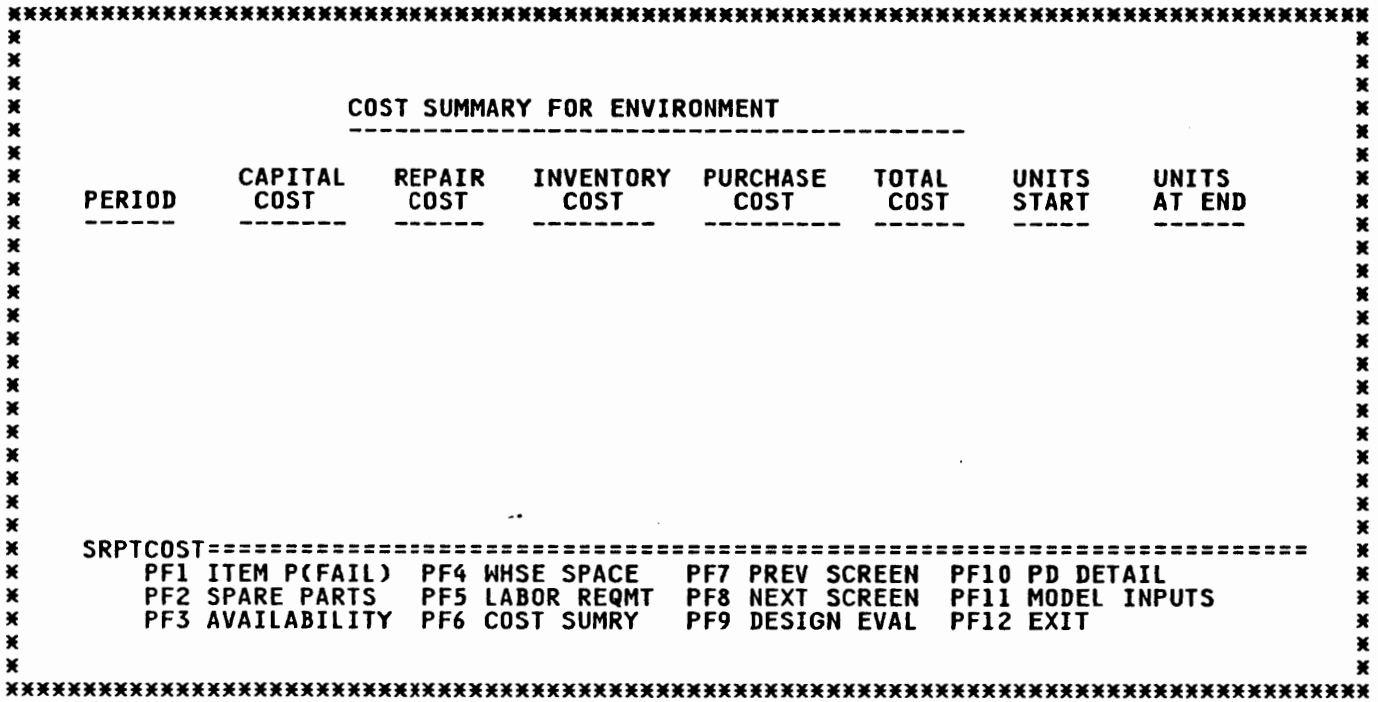
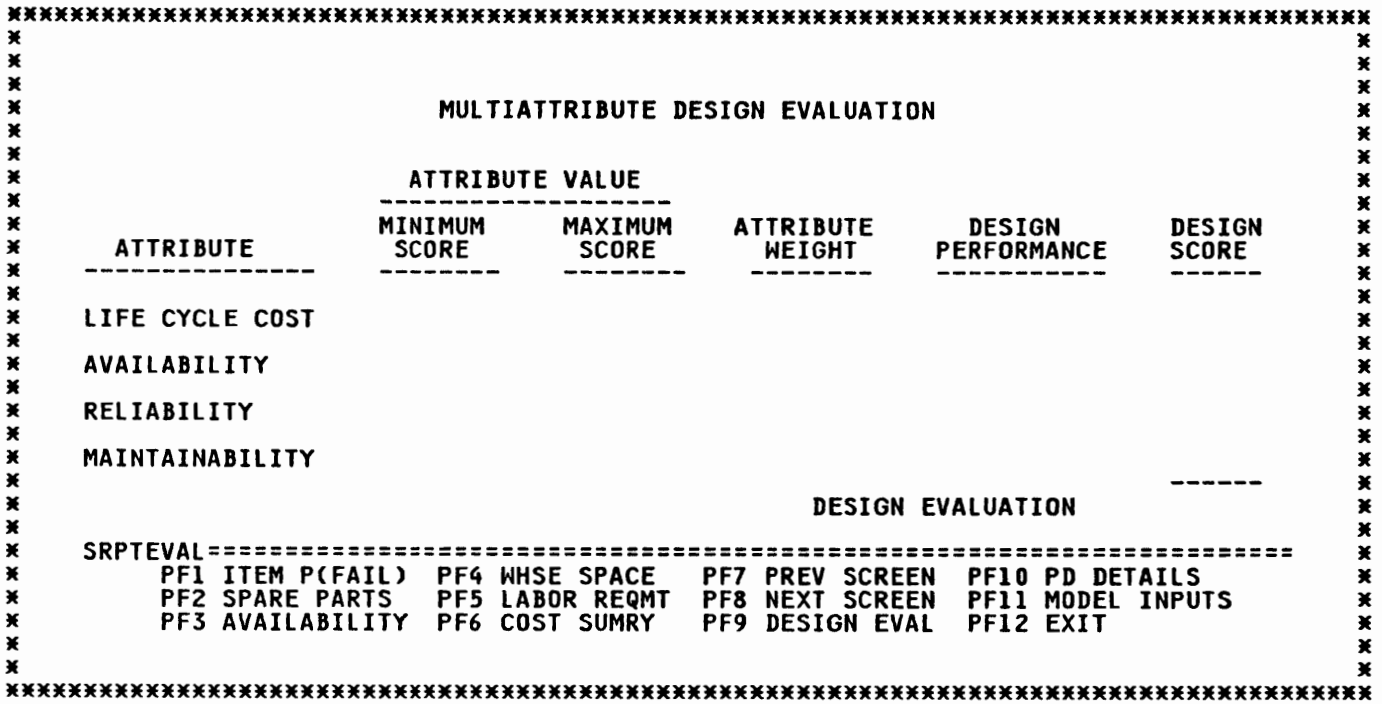


Figure 68. Life-Cycle Costs

Figure 69. Design Evaluation Model



C.6 Record Layouts

Eleven data files are used by CODAS. One of these files contains the names of the scenario data files and a short 40 character description of each scenario. The record layout for this file is :

Filename: FILE SCENARIO A

Column	Field Description
4-10	Scenario Name
16-55	Scenario Description

The remaining ten files contain scenario data. Each of these files contains the data needed to completely describe one scenario (design alternative). Each of these files contain seven different types of records which describe different aspects of the design alternative. These seven record types are:

1. System parameters (PARM_RC)
2. Levels data (LEVELS_RC)
3. Cost data (COST_RC)
4. Operating environment data (ENV_RC)
5. Maintenance tree data (NODE_DATA_REC)
6. Spare part data (PART_RC)
7. Procurement policy data (PROC_RC)

The record structure in each of these files is as follows:

PARM_RC (1 record)

LEVELS_RC (1 record for each level; 3 max.)

COST_RC (1 record)

PROC_RC (1 record)

MCDM_RC (1 record)

ENV_RC (1st environment; used as header for NODE_DATA_REC's; 1 record for each operating environment)

NODE_DATA_REC (1 record for each bottom-of-the-tree node; 100 max.)

ENV_RC (2nd environment)

NODE_DATA_REC (1 record for each bottom-of-the-tree node; must have same number of records as 1st environment)

Repeat ENV_RC and NODE_DATA_REC's for each environment.

PART_RC (1 record for each spare part required to accomplish repair of the bottom-of-the-tree nodes; 100 max.)

The layout of each of these records is shown below.

Record: PARM_RC

Column	Field Description
10	Time period
13-15	Period length

18-20	Run length
25	Number of environments
30	Model mode
33-35	Number of nodes
40	Repair levels
43-45	Number spare parts
50	Lot sizing strategy
54-55	Number of existing streams

Record: LEVELS_RC

Column	Field Description
5-16	Level description
21-22	Number of channels
26-31	Repair cost per day

Record: COSTS_RC

Column	Field Description
5-10	First cost (end item)
15-20	Salvage value (fatal failure)
25-30	Procurement cost
38-40	Order processing cost
48-50	Warehouse space cost
54-55	Time value of money

Record: PROC_RC

Column	Field Description
5	Policy number
8-10	Order size
13-15	Periods between procurement
18-20	Initial procurement
25	Policy option
28-30	Upper population limit
33-35	Percent increase
38-40	Percent decrease

Record: MCDM_RC

Column	Field Description
3-10	Life-Cycle Cost (Minimum Score)
13-20	Life-Cycle Cost (Maximum Score)
23-25	Availability (Minimum Score)
28-30	Availability (Maximum Score)
33-35	Reliability (Minimum Score)
38-40	Reliability (Maximum Score)
43-45	Maintainability (Minimum Score)
48-50	Maintainability (Maximum Score)
53-55	Life-Cycle Cost (Weight)
58-60	Availability (Weight)
63-65	Reliability (Weight)
68-70	Maintainability (Weight)

Record: ENV_RC

Column	Field Description

5	Environment number
9-20	Environment description)
23-25	Retirement age
28-30	Environment demand
33-35	Number of items deployed
45-50	Salvage value (retirement)

Record: NODE_DATA_REC

Column	Field Description
2-4	Node number
7-26	Node description
29	Environment number
32	Failure curve
35-39	MTBF
42-46	MTTR
49	Level of repair
52	Number of parallel redundant components
55	Critical failure (Y or N)
58-62	Labor hours required to repair
65-66	Number of spare parts required to repair

Record: PART_RC

Column	Field Description
2-4	Node number
7-24	Node description
27-32	Part number
35	Quantity
38-42	Warehouse space
45-51	Unit price
56-57	Lead time

Appendix D. CODASV1 Program Listing

```
*PROCESS ATTRIBUTES MACRO;
 CODA: PROCEDURE OPTIONS(MAIN);

 /*************************************************************************/
 /*      CODAS Version 1: A DSS For Integrated Design Analysis            */
 /*      Of A Repairable Item and Its Logistics Support System            */
 /*        Written By Roderick J. Reasor Using PL/1 and DMS               */
 /*************************************************************************/

  %INCLUDE SYSLIB(EUDPLI);

 /*************************************************************************/
 /************ VARIABLES DECLARATIONS AND INITIALIZATIONS *****************/
 /*************************************************************************/

   DECLARE CODALST INPUT STREAM ENV (F BLKSIZE(80));

   DECLARE DATAREC INPUT STREAM ENV (F BLKSIZE(80));

   DECLARE
    1 NODE_DATA_REC,
     3 NODE_NUM          PIC'ZZ9',
     3 NODE_DESC         CHAR(20),
     3 ENVIRON           PIC'9',
     3 FAIL_CURVE        PIC'9',
     3 MTBF              PIC'ZZZZ9',
     3 MTTR              PIC'ZZZ9',
     3 LOR               PIC'9',
     3 COMPONENTS        PIC'9',
     3 CRIT_FAIL         PIC'A',
     3 MAN_HRS           PIC'ZZZZ9',
     3 SPARE_PARTS       PIC'Z9',

    1 NODE_DATA(4,50)    LIKE NODE_DATA_REC;

  DECLARE
    1 ENV_RC,
     3 ENV_NUM           PIC'9',
     3 ENV_DESC          CHAR(12),
     3 RETIRE_AGE        PIC'ZZ9',
     3 DEMAND            PIC'ZZ9',
     3 DEPLOYED          PIC'ZZ9',
     3 NEW_UNITS         PIC'ZZ9',
     3 SV_RETIRE         PIC'ZZZZZ9',

    1 ENV_DATA(4)        LIKE ENV_RC;

  DECLARE
    1 PART_RC,
     3 NODE_NUM          PIC'ZZ9',
```

```
    3 PART_DESC          CHAR(18),
    3 PART_NUM           CHAR(6),
    3 QUANTITY           PIC'9',
    3 WH_SPACE           PIC'ZZ9V.9',
    3 UNIT_PRICE         PIC'ZZZZ9V.99',
    3 LEAD_TIME          PIC'Z9',

   1 PART_DATA(50)       LIKE PART_RC;

DECLARE
   1 PARM_RC,
    3 TIME_PD            PIC'9',
    3 PD_LENGTH          PIC'ZZ9',
    3 RUN_LENGTH         PIC'ZZ9',
    3 NUM_ENV            PIC'9',
    3 MODEL_MODE         PIC'9',
    3 NUM_NODES          PIC'ZZ9',
    3 REPAIR_LEVELS      PIC'9',
    3 NUM_STK_RC         PIC'ZZ9',
    3 LOT_SIZE           PIC'9',
    3 EXIST_STR          PIC'Z9';

 DECLARE
   1 COST_RC,
    3 FIRST_COST         PIC'ZZZZZ9',
    3 SV_FFAIL           PIC'ZZZZZ9',
    3 PROC_COST          PIC'ZZZZZ9',
    3 ORDER_COST         PIC'ZZ9',
    3 WHSE_COST          PIC'ZZ9',
    3 INTEREST           PIC'ZZ9';

 DECLARE
   1 LEVELS_RC,
    3 DESC               CHAR(12),
    3 CHANNELS           PIC'Z9',
    3 COST               PIC'ZZZZZ9',

   1 LEVELS_DATA(3)      LIKE LEVELS_RC;

DECLARE
   1 PROC_RC,
    3 POLICY_NUM         PIC'9',
    3 ORDER_SIZE         PIC'ZZ9',
    3 PD_BETW_PROC       PIC'ZZ9',
    3 INIT_PROC          PIC'ZZ9',
    3 POLICY_OPT         PIC'9',
    3 UP_POP_LIMIT       PIC'ZZ9',
    3 PCT_INC            PIC'ZZ9',
    3 PCT_DEC            PIC'ZZ9';
```

```
DECLARE
  1 SCEN_RC,
   3 FILENAME          CHAR(7),
   3 FILEDESC          CHAR(40),

  1 SCENES(10)         LIKE SCEN_RC;

DECLARE
  1 MCDM_RC,
   3 LCC_MIN           PIC'ZZZZZZZ9',
   3 LCC_MAX           PIC'ZZZZZZZ9',
   3 AVA_MIN           PIC'ZZ9',
   3 AVA_MAX           PIC'ZZ9',
   3 REL_MIN           PIC'ZZ9',
   3 REL_MAX           PIC'ZZ9',
   3 MAI_MIN           PIC'ZZZ9',
   3 MAI_MAX           PIC'ZZZ9',
   3 LCC_WT            PIC'ZZ9',
   3 AVA_WT            PIC'ZZ9',
   3 REL_WT            PIC'ZZ9',
   3 MAI_WT            PIC'ZZ9';

DECLARE
  SCNUM               FIXED BIN(31) INIT(1),
  SCENDAT             CHAR(3)   INIT(' NO'),
  FNAME               CHAR(7)   INIT('EXAMPLE'),
  SCEN_NUM            PIC'9'    INIT(1),
  GSCEN               FIXED BINARY (15)   INIT(1);

DECLARE
  MORE_SCREENS        BIT(1),
  YES                 BIT(1)    INIT('1'B),
  NO                  BIT(1)    INIT('0'B);

DECLARE
  MORE_INTSC          BIT(1),
  INTSC               FIXED BIN(15),
  PART_COUNT          FIXED BIN(15) INIT(0),
  IFILL               FIXED BIN(15),
  FILL_DATA(75)       CHAR(5) INIT((75)'    '),
  TEST_NODE_NUM       PIC'ZZ9',
  NUM_PARTS(50)       FIXED BIN(15) INIT(0);

DECLARE
  1 CURSOR,
   2 FIELD             FIXED BIN(15)  INIT(1),
   2 OFFSET            FIXED BIN(15)  INIT(0),
   2 TYPE              CHAR(1)        INIT('D');

DECLARE
```

```
  1 SC2_CURSOR,
   2 FIELD              FIXED BIN(15)  INIT(21),
   2 OFFSET             FIXED BIN(15)  INIT(0),
   2 TYPE               CHAR(1)        INIT('D');

DECLARE
  1 RSTATUS             BIT(8),
  1 ADDR                BUILTIN,
  1 CEIL                BUILTIN,
  1 FLOOR               BUILTIN,
  1 EXP                 BUILTIN,
  1 LOG                 BUILTIN,
  1 TRUNC               BUILTIN,
  1 ROUND               BUILTIN,
  1 MOD                 BUILTIN,
  1 SQRT                BUILTIN,
  1 PROTECT             CHAR(121)      INIT((121)'N');

DECLARE
  (ISN, IA2, IA, IA1, IE, IN, IP, IPN)     FIXED BIN(15) INIT(1),
  (J, IL, I, JS, NX1, JN, JE, IS, I1, JP)  FIXED BIN(15) INIT(1),
  (IL1, IL2, ILT, NUDEPLOY, OQTY)          FIXED BIN(15) INIT(1),
  (TDEMAND, ENVUNITS, RDEPLOY, SDEPLOY1)   FIXED BIN(15) INIT(0),
  (SDEPLOY, TPDMD, LTREQMTS, AGE1, INVPOS) FIXED BIN(15) INIT(0),
  (EOQ, PQTY, POQ, XENV, INVCOST)          FIXED BIN(15) INIT(0),
  (TVM, HCOST1, HCOST2, AVDMD, HCOST)      FLOAT DECIMAL (15),
  (IHCOST, ACPTU, MCPTU, FAC3, SALVAGE)    FLOAT DECIMAL (15),
  (ADJAGE, ULSWITCH, PROCAMT, ORD_ARRIVAL) FIXED BIN(15) INIT(0),
  (STRFAIL, NXTPROC, PROCNUM, PROCSIZE)    FIXED BIN(15) INIT(0),
  (CTIME, IENV, IENV1, FC, AGE, XNG, NX)   FIXED BIN(15) INIT(1),
  (NUM_STREAMS, IRL, IENV2, SUMSC, J1)     FIXED BIN(15) INIT(1),
  (K, I2, I3, I4, I5, IV, NG, JNUM, JMAX)  FIXED BIN(15) INIT(1);

DECLARE
 MORE_SUMSC                                BIT(1),
 (PROB, PFAIL, PROB5, PROB4, ET, ET1)      FLOAT DEC (15);

DECLARE
  SC2_DMASK(21)    BIT(8)   INIT((21)'01000000'B),
  SCEN_MSG         CHAR(25) INIT((25)' '),
  MORE_SCENES      BIT(1);

DECLARE
 MORE_OUTSC           BIT(1),
 OUTSC                FIXED BIN(31),
 PARTNO               FIXED BIN(15) INIT(1),
 INP                  FIXED BIN(15) INIT(1),
 LNUM                 FIXED BIN(15) INIT(1),
 PERNO                PIC'Z9',
 STRNO                PIC'Z9';
```

```
DECLARE
  POPSIZE              PIC'ZZ9'  INIT(0),
  PROCQTY(30)          PIC'ZZ9'  INIT((30)0),
  MIN_POPSIZE          PIC'ZZ9'  INIT(0);

DECLARE
 1 STREAM(50),
  2 CDATE              PIC'-ZZ9',
  2 FAIL_ENV           PIC'9',
  2 UNITS              PIC'ZZZ9',
  2 INIT_FIELD         PIC'ZZZ9';

DECLARE
 1 EXSTR(10),
  2 AGE                PIC'ZZ9',
  2 ENV                PIC'9',
  2 UNITS              PIC'ZZ9';

DECLARE
 1 STRQTY(30,50),
  2 BEG_PD             PIC'ZZZ9',
  2 END_PD             PIC'ZZZ9';

DECLARE
 LAMDA                 FLOAT DECIMAL (15),
 NFPROB(50,50)         FLOAT DECIMAL (15),
 EFAIL(30,50)          FIXED BINARY (15) INIT((1500)0),
 ENFAIL                FIXED BINARY (15),
 SMTTR(30,50)          FIXED BINARY (15) INIT((1500)0),
 SNMTTR                FLOAT DECIMAL (15),
 ENVMTTR               FLOAT DECIMAL (15),
 FFAILS(50)            FIXED BINARY (15) INIT((50)0),
 MTPROB(50,126)        FLOAT DECIMAL (15) INIT((6300)0);

DECLARE
 (LTP, N, X, POPUNITS)           FIXED BINARY (15),
 ENVQTY(30,4)                    FIXED BINARY (15),
 EMTTR(30,4)                     FIXED BINARY (15),
 EPFAIL(30,4)                    FLOAT DECIMAL (15),
 EPFAIL1(30,4)                   FLOAT DECIMAL (15),
 ENVAVAIL(30,4)                  FLOAT DECIMAL (15),
 PAVAIL(30)                      FLOAT DECIMAL (15),
 DF(4)                           FLOAT DECIMAL (15),
 (ENVPFAIL, P, Q, FAC1, FAC2)    FLOAT DECIMAL (15),
 (PX, QNLX, FX, SUMFX, POPAVAIL) FLOAT DECIMAL (15);

DECLARE
 1 PART_DMD(30,50),
  2 PD_DMD             PIC'ZZZZ9',
```

```
   2 BEG_INV            PIC'ZZZZZ9',
   2 END_INV            PIC'ZZZZZ9',
   2 LOT_SIZE           PIC'ZZZ9';

DECLARE
 PLOR                   PIC'9';

DECLARE
 NFAILRPT(50,30,26)    FLOAT DECIMAL (15);
/** 50 STREAMS, 30 PERIODS, 26 NODES **/

DECLARE
 1 PFAIL_RPT,
   2 STRNO              PIC'Z9'       INIT(1),
   2 PERNO              PIC'Z9'       INIT(1),
   2 INIT_FIELD         PIC'ZZZZ9'    INIT(0),
   2 UNITS_PROC         PIC'ZZZZ9'    INIT(0),
   2 BEG_UNITS          PIC'ZZZZ9'    INIT(0),
   2 END_UNITS          PIC'ZZZZ9'    INIT(0),
   2 MTTR               PIC'ZZZZ9'    INIT(0),
   2 FPROB(26)          PIC'9V.9999'  INIT((26)0);

DECLARE
 1 EFAIL_RPT,
   2 FAIL(10,4)         PIC'9V.9999',
   2 PDNO(10)           PIC'Z9';

DECLARE
 1 EIAVAIL_RPT,
   2 AVAIL(10,4)         PIC'ZZ9',
   2 SYS_AVAIL(10)       PIC'ZZ9',
   2 PDNO(10)            PIC'Z9';

DECLARE
 1 SPACE_RPT,
   2 SQFT(10,3)         PIC'ZZZZZ9',
   2 PDNO(10)            PIC'Z9';

DECLARE
 1 LABOR_RPT,
   2 HOUR(10,3)         PIC'ZZZ9',
   2 PDNO(10)            PIC'Z9';

DECLARE
 1 PARTS_RPT(12),
   2 NODE_NUM           PIC'ZZ9',
   2 PART_DESC          CHAR(15),
   2 DEMAND             PIC'ZZZ9',
   2 BEG_INV            PIC'ZZZZZ9',
   2 END_INV            PIC'ZZZZZ9',
```

```
  2 LOT_SIZE          PIC'ZZZ9',
  2 PART_LOR          PIC'9';

DECLARE
 1 SPART_RPT(12),
  2 NODE_NUM          PIC'ZZ9',
  2 PART_DESC         CHAR(15),
  2 PART_NUM          CHAR(6),
  2 DEMAND            PIC'ZZZ9',
  2 LOR               PIC'9';

DECLARE
 1 AVAIL_RPT(4),
  2 ENV_DESC       CHAR(12),
  2 ENV_AVAIL      PIC'ZZ9';

DECLARE
   SYS_AVAIL       PIC'ZZ9';

DECLARE
 1 WHSE_RPT,
  2 ENV_DESC(4)       CHAR(12),
  2 REP_LEV(4,3)      PIC'ZZZZZ9',
  2 TOTALS(3)         PIC'ZZZZZ9';

DECLARE
 SPACE(30,3)          FIXED BINARY (15),
 LABOR(30,3)          FIXED BINARY (15),
 AVEINV               FIXED BINARY (15),
 LABOR1               FIXED BINARY (15),
 SPACE1               FLOAT DECIMAL (15),
 EQTY                 FIXED BINARY (15);

DECLARE
 1 MANPR_RPT,
  2 ENV_DESC(4)       CHAR(12),
  2 ENV_REQ(4,3)      PIC'ZZZZZ9',
  2 TOTALS(3)         PIC'ZZZZZ9';

DECLARE
1 COST_RPT(30,4),
 2 ENV_DESC           CHAR(12),
 2 CAP_COST           PIC'-ZZZZZZZZ9',
 2 REP_COST           PIC'ZZZZZ9',
 2 PUR_COST           PIC'ZZZZZ9',
 2 INV_COST           PIC'ZZZZZ9',
 2 TOT_COST           PIC'-ZZZZZZZZ9',
 2 UNITS_START        PIC'ZZZ9',
 2 UNITS_END          PIC'ZZZ9',
 2 PDNO               PIC'Z9',
```

```
   2 UNIT_COST          PIC'ZZZZZ9';

   DECLARE
    1 MCDM_RPT,
     2 LIFE_CYCLE_COST(3)   PIC'ZZZZZZZ9',
     2 AVAILABILITY(3)      PIC'ZZ9',
     2 RELIABILITY(3)       PIC'ZZ9',
     2 MAINTAINABILITY(3)   PIC'ZZZ9',
     2 WEIGHT(4)            PIC'ZZ9',
     2 SCORE(4)             PIC'ZZ9',
     2 SYSTEM_EVAL          PIC'ZZ9';

  DECLARE
   PMTTR(30)              FIXED BINARY (15),
   MABILITY               FIXED BINARY (15),
   SAVAIL                 FIXED BINARY (15),
   PWCOST                 FLOAT DECIMAL (15),
   PDCOST                 FLOAT DECIMAL (15),
   LCYCLE_COST            FLOAT DECIMAL (15),
   RELABILITY             FLOAT DECIMAL (15),
   ENVREL(4)              FLOAT DECIMAL (15),
   AGPIN                  FLOAT DECIMAL (15);

/*************************************************************************/
/******************** PANEL DISPLAY AND I/O CONTROL **********************/
/*************************************************************************/

/*************** INITIAL BANNER SCREEN DISPLAY ***************************/

   PDISPLAY PNAME('CODASV1 ') RSTATUS(RSTATUS);
   SELECT (RSTATUS);
     WHEN(PF1 ) SCNUM = SCNUM + 100;
     WHEN(PF8 ) SCNUM = SCNUM + 1;
     WHEN(PF12) STOP;
     OTHERWISE SCNUM = SCNUM + 0;
     END;

/*************** SELECT DISPLAY SCREEN ***********************************/

   MORE_SCREENS = YES;

   DO WHILE (MORE_SCREENS);
    SELECT;
     WHEN (SCNUM = 1) CALL BANNER_SCREEN;
     WHEN (SCNUM = 2) CALL SCENARIO_SCREEN;
     WHEN (SCNUM = 3) CALL RUN_PARAMETERS_SCREEN;
     WHEN (SCNUM = 4) CALL ENV_DATA_SCREEN;
     WHEN (SCNUM = 5) CALL LOG_SUPT_SYS_SCREEN;
     WHEN (SCNUM = 6) CALL NODE_DATA_SCREEN;
     WHEN (SCNUM = 7) CALL COST_DATA_SCREEN;
```

```
     WHEN (SCNUM = 8) CALL REPAIR_LEVEL_SCREEN;
     WHEN (SCNUM = 9) CALL INV_REORDER_SCREEN;
     WHEN (SCNUM = 10) CALL SPARE_PARTS_SCREEN;
     WHEN (SCNUM = 11) CALL PROC_POLICY_SCREEN;
     WHEN (SCNUM = 12) CALL DEPLOY_PROC_SCREEN;
     WHEN (SCNUM = 13) CALL RETIRE_AGE_SCREEN;
     WHEN (SCNUM = 14) CALL PD_REPORTS_SCREEN;
     WHEN (SCNUM = 15) CALL RUN_SUMRY_REPORTS;
     WHEN (SCNUM = 16) CALL EXIST_POP_SCREEN;
     WHEN (SCNUM  100) CALL CONCEPTS_SCREEN;
     OTHERWISE STOP;
     END;
    END;
    RETURN;

/*****************************************************************************/
/************************ DISPLAY BANNER SCREEN ************************/
/*****************************************************************************/

BANNER_SCREEN: PROCEDURE;

  PDISPLAY PNAME('CODASV1 ') RSTATUS(RSTATUS);
  SELECT (RSTATUS);
    WHEN(PF1 ) SCNUM = SCNUM + 100;
    WHEN(PF8 ) SCNUM = SCNUM + 1;
    WHEN(PF12) STOP;
    OTHERWISE SCNUM = SCNUM + 0;
    END;
  END BANNER_SCREEN;

/*****************************************************************************/
/******** DISPLAY SCENARIO SELECTION AND DATA OPTIONS SCREEN ********/
/*****************************************************************************/

SCENARIO_SCREEN: PROCEDURE;

/*********************************************/
/** READ IN NAMES OF AVAILABLE SCENARIOS **/
/*********************************************/

    ISN = 1;
    MORE_SCENES = YES;
    OPEN FILE(CODALST);
    ON ENDFILE (CODALST) MORE_SCENES = NO;
    GET FILE(CODALST) EDIT
      (SCEN_RC) (COLUMN(4),A(7),X(5),A(40));
    DO WHILE (MORE_SCENES);
      SCENES(ISN) = SCEN_RC;
      ISN = ISN +1;
      GET FILE(CODALST) EDIT
```

```
        (SCEN_RC) (COLUMN(4),A(7),X(5),A(40));
      END;

/************************************/
/***** DISPLAY SCENARIO SCREEN *****/
/************************************/

  DECLARE
   1 SC2_LOADLIST,
     2 LOAD(22)      PTR,
     2 FENCE         BIT(32)   INIT((32)'1'B);

  DO IA = 1 TO 10;
    IA1 = (IA-1)*2 + 1;
    IA2 = IA*2;
    SC2_LOADLIST.LOAD(IA1) = ADDR(SCENES.FILENAME(IA));
    SC2_LOADLIST.LOAD(IA2) = ADDR(SCENES.FILEDESC(IA));
    END;

  SC2_LOADLIST.LOAD(21) = ADDR(SCEN_NUM);
  SC2_LOADLIST.LOAD(22) = ADDR(SCEN_MSG);

  PDISPLAY PNAME('SCENARIO') RSTATUS(RSTATUS) LDLIST(SC2_LOADLIST)
       ULDLIST(SC2_LOADLIST) DCURSOR(SC2_CURSOR) DMASK(SC2_DMASK);

  SELECT (RSTATUS);
   WHEN(PF1 ) SCNUM = SCNUM + 100;

   /****** WHEN (PF4) GET DATA FOR REQUESTED SCENARIO ******/

   WHEN(PF4 ) DO;
    SCEN_MSG = ' SCENARIO DATA RETRIEVED ';
    GSCEN = SCEN_NUM + 1;
    FNAME = SCENES.FILENAME(GSCEN);
    OPEN FILE(DATAREC) TITLE (FNAME);

    /*** INPUT SYSTEM PARAMETERS ***/

    GET FILE (DATAREC) EDIT
      (PARM_RC) (COLUMN(10),F(1),2(X(2),F(3)),X(4),F(1),X(4),F(1),
                 X(2),F(3),X(4),F(1),X(2),F(3),X(4),F(1),X(3),F(2));

    /*** INPUT REPAIR LEVEL DATA ***/

    DO IL = 1 TO REPAIR_LEVELS;
    GET FILE (DATAREC) EDIT
      (LEVELS_RC) (COLUMN(5),A(12),X(4),F(2),X(3),F(6));
    LEVELS_DATA(IL) = LEVELS_RC;
    END;
```

```
/*** INPUT ITEM COST DATA ***/

GET FILE (DATAREC) EDIT
  (COST_RC) (COLUMN(5),F(6),2(X(4),F(6)),2(X(7),F(3)),X(3),F(3));

 /*** INPUT PROCUREMENT POLICY RECORD ***/

GET FILE (DATAREC) EDIT
  (PROC_RC) (COLUMN(5),F(1),3(X(2),F(3)),X(4),F(1),3(X(2),F(3)));

 /*** INPUT DESIGN EVALUATION SCORING PARAMETERS ***/

GET FILE (DATAREC) EDIT
  (MCDM_RC) (COLUMN(3),F(8),X(2),F(8),4(X(2),F(3)),2(X(1),F(4)),
             4(X(2),F(3)));

  MCDM_RPT.LIFE_CYCLE_COST(1) = MCDM_RC.LCC_MIN;
  MCDM_RPT.LIFE_CYCLE_COST(2) = MCDM_RC.LCC_MAX;
  MCDM_RPT.AVAILABILITY(1) = MCDM_RC.AVA_MIN;
  MCDM_RPT.AVAILABILITY(2) = MCDM_RC.AVA_MAX;
  MCDM_RPT.RELIABILITY(1) = MCDM_RC.REL_MIN;
  MCDM_RPT.RELIABILITY(2) = MCDM_RC.REL_MAX;
  MCDM_RPT.MAINTAINABILITY(1) = MCDM_RC.MAI_MIN;
  MCDM_RPT.MAINTAINABILITY(2) = MCDM_RC.MAI_MAX;
  MCDM_RPT.WEIGHT(1) = MCDM_RC.LCC_WT;
  MCDM_RPT.WEIGHT(2) = MCDM_RC.AVA_WT;
  MCDM_RPT.WEIGHT(3) = MCDM_RC.REL_WT;
  MCDM_RPT.WEIGHT(4) = MCDM_RC.MAI_WT;

/*** INPUT ENVIRONMENT DATA ***/

DO IE = 1 TO NUM_ENV;
 GET FILE (DATAREC) EDIT
  (ENV_RC) (COLUMN(5),F(1),X(3),A(12),4(X(2),F(3)),X(4),F(6));
 ENV_DATA(IE) = ENV_RC;

 DO IN = 1 TO PARM_RC.NUM_NODES;
  GET FILE (DATAREC) EDIT
    (NODE_DATA_REC) (R(NODE_DATA_FMT));
  NODE_DATA(IE,IN) = NODE_DATA_REC;
  END;
 END;

 NODE_DATA_FMT: FORMAT (COLUMN(2),F(3), X(2),A(20), 2(X(2),F(1)),
      2(X(2),F(5)),2(X(2),F(1)),X(2),A(1),X(2),F(5),X(2),F(2));

 /*** INPUT STOCKABLE PARTS DATA ***/

 DO IP = 1 TO NUM_STK_RC;
  GET FILE (DATAREC) EDIT
```

```
        (PART_RC) (COLUMN(2),F(3),X(2),A(18),X(2),A(6),X(2),F(1),
                  X(2),F(5),X(2),F(7),X(4),F(2));
       PART_DATA(IP) = PART_RC;
      END;

     /*** DETERMINE NUMBER OF STOCKABLE PARTS AT EACH NODE ***/

      DO IN = 1 TO PARM_RC.NUM_NODES;
       PART_COUNT = 0;
       TEST_NODE_NUM = NODE_DATA.NODE_NUM(1,IN);
       DO IP = 1 TO NUM_STK_RC;
        IF PART_DATA.NODE_NUM(IP) = TEST_NODE_NUM THEN
                  PART_COUNT = PART_COUNT + 1;
        END;
       NUM_PARTS(IN) = PART_COUNT;
       END;
      END;

    /***** WHEN(PF5 ) SAVE DATA FOR CURRENT SCENARIO *****/

    WHEN(PF5 ) DO;
     SCEN_MSG = ' FUNCTION NOT AVAILABLE  ';
     END;

    WHEN(PF7 ) SCNUM = SCNUM - 1;
    WHEN(PF8 ) SCNUM = SCNUM + 1;
    WHEN(PF12) SCNUM = 1;
    OTHERWISE SCNUM = SCNUM + 0;
    END;

END SCENARIO_SCREEN;

/*************************************************************************/
/************ INPUT MODEL MODE AND SIMULATION PARAMETERS ************/
/*************************************************************************/

RUN_PARAMETERS_SCREEN: PROCEDURE;

  DECLARE
   1 IS3_LOADLIST,
     2 LOAD(4)     PTR,
     2 FENCE       BIT(32)  INIT((32)'1'B);

  IS3_LOADLIST.LOAD(1) = ADDR(PARM_RC.MODEL_MODE);
  IS3_LOADLIST.LOAD(2) = ADDR(PARM_RC.TIME_PD);
  IS3_LOADLIST.LOAD(3) = ADDR(PARM_RC.PD_LENGTH);
  IS3_LOADLIST.LOAD(4) = ADDR(PARM_RC.RUN_LENGTH);

  PDISPLAY PNAME('RUNPARM ') RSTATUS(RSTATUS) LDLIST(IS3_LOADLIST)
           ULDLIST(IS3_LOADLIST) DCURSOR(CURSOR);
```

```
   /*** CALCULATE LENGTH OF TIME PERIOD (LTP) IN HOURS ***/
   /*** FOR SUBSEQUENT USE IN AVAILABILITY CALCULATIONS ***/
   SELECT (PARM_RC.TIME_PD);
    WHEN (1) LTP = 8 * PARM_RC.PD_LENGTH;
    WHEN (2) LTP = 40 * PARM_RC.PD_LENGTH;
    WHEN (3) LTP = 173.33 * PARM_RC.PD_LENGTH;
    WHEN (4) LTP = 2080 * PARM_RC.PD_LENGTH;
    END;

   SELECT(RSTATUS);
    WHEN(PF1 ) SCNUM = 3;
    WHEN(PF2 ) SCNUM = 5;
    WHEN(PF3 ) SCNUM = 6;
    WHEN(PF4 ) SCNUM = 8;
    WHEN(PF5 ) SCNUM = 11;
    WHEN(PF6 ) SCNUM = 13;
    WHEN(PF7 ) SCNUM = SCNUM - 1;
    WHEN(PF8 ) DO;
     IF PARM_RC.MODEL_MODE = 2 THEN SCNUM = 16;
     ELSE SCNUM = SCNUM + 1;
     END;
    WHEN(PF9 ) CALL SIMULATE;
    WHEN(PF10) SCNUM = 14;
    WHEN(PF11) SCNUM = 15;
    WHEN(PF12) SCNUM = 1;
    OTHERWISE SCNUM = SCNUM + 0;
    END;
   PRELEASE;
   END RUN_PARAMETERS_SCREEN;

/*************************************************************************/
/******************* INPUT FAILURE ENVIRONMENT DATA ******************/
/*************************************************************************/

ENV_DATA_SCREEN: PROCEDURE;

   DECLARE
    1 IS73_LOADLIST,
      2 LOAD(9)      PTR,
      2 FENCE        BIT(32)  INIT((32)'1'B);

   IS73_LOADLIST.LOAD(1) = ADDR(PARM_RC.NUM_ENV);
   DO IE = 1 TO 4;
   IS73_LOADLIST.LOAD((IE-1)*2+2) = ADDR(ENV_DATA.ENV_DESC(IE));
   IS73_LOADLIST.LOAD((IE-1)*2+3) = ADDR(ENV_DATA.DEMAND(IE));
   END;
   PDISPLAY PNAME('ENVDATA ') RSTATUS(RSTATUS) LDLIST(IS73_LOADLIST)
            ULDLIST(IS73_LOADLIST) DCURSOR(CURSOR);
```

```
   MIN_POPSIZE = 0;
   DO IE = 1 TO 4;
    MIN_POPSIZE = MIN_POPSIZE + ENV_DATA.DEMAND(IE);
    END;

   SELECT(RSTATUS);
    WHEN(PF1 ) SCNUM = 3;
    WHEN(PF2 ) SCNUM = 5;
    WHEN(PF3 ) SCNUM = 6;
    WHEN(PF4 ) SCNUM = 8;
    WHEN(PF5 ) SCNUM = 11;
    WHEN(PF6 ) SCNUM = 13;
    WHEN(PF7 ) SCNUM = SCNUM - 1;
    WHEN(PF8 ) SCNUM = SCNUM + 1;
    WHEN(PF9 ) CALL SIMULATE;
    WHEN(PF10) SCNUM = 14;
    WHEN(PF11) SCNUM = 15;
    WHEN(PF12) SCNUM = 1;
    OTHERWISE SCNUM = SCNUM + 0;
    END;
   PRELEASE;
END ENV_DATA_SCREEN;

/*****************************************************************************/
/***************** INPUT LOGISTICS SUPPORT PARAMETERS ****************/
/*****************************************************************************/

LOG_SUPT_SYS_SCREEN: PROCEDURE;

   DECLARE
    1 IS5_LOADLIST,
      2 LOAD(10)    PTR,
      2 FENCE       BIT(32)  INIT((32)'1'B);

   IS5_LOADLIST.LOAD(1) = ADDR(PARM_RC.REPAIR_LEVELS);

   DO IL = 1 TO 3;
     IS5_LOADLIST.LOAD((IL-1)*3+2) = ADDR(LEVELS_DATA.DESC(IL));
     IS5_LOADLIST.LOAD((IL-1)*3+3) = ADDR(LEVELS_DATA.CHANNELS(IL));
     IS5_LOADLIST.LOAD((IL-1)*3+4) = ADDR(LEVELS_DATA.COST(IL));
     END;

   PDISPLAY PNAME('LEVELDAT') RSTATUS(RSTATUS) LDLIST(IS5_LOADLIST)
            ULDLIST(IS5_LOADLIST) DCURSOR(CURSOR);

   SELECT(RSTATUS);
    WHEN(PF1 ) SCNUM = 3;
    WHEN(PF2 ) SCNUM = 5;
    WHEN(PF3 ) SCNUM = 6;
    WHEN(PF4 ) SCNUM = 8;
```

```
   WHEN(PF5 ) SCNUM = 11;
   WHEN(PF6 ) SCNUM = 13;
   WHEN(PF7 ) SCNUM = SCNUM - 1;
   WHEN(PF8 ) SCNUM = SCNUM + 1;
   WHEN(PF9 ) CALL SIMULATE;
   WHEN(PF10) SCNUM = 14;
   WHEN(PF11) SCNUM = 15;
   WHEN(PF12) SCNUM = 1;
   OTHERWISE SCNUM = SCNUM + 0;
   END;
  PRELEASE;
  END LOG_SUPT_SYS_SCREEN;

/*************************************************************************/
/***************** SPECIFY EXISTING POPULATION ***************************/
/*************************************************************************/

EXIST_POP_SCREEN: PROCEDURE;

  DECLARE
   1 EPOP_LOADLIST,
     2 LOAD(31)    PTR,
     2 FENCE       BIT(32)  INIT((32)'1'B);

  EPOP_LOADLIST.LOAD(1) = ADDR(PARM_RC.EXIST_STR);

  DO I = 1 TO 10;
   EPOP_LOADLIST.LOAD((I-1)*3+2) = ADDR(EXSTR.AGE(I));
   EPOP_LOADLIST.LOAD((I-1)*3+3) = ADDR(EXSTR.ENV(I));
   EPOP_LOADLIST.LOAD((I-1)*3+4) = ADDR(EXSTR.UNITS(I));
   END;

  PDISPLAY PNAME('EXISTPOP') RSTATUS(RSTATUS) LDLIST(EPOP_LOADLIST)
           ULDLIST(EPOP_LOADLIST) DCURSOR(CURSOR);

  IF PARM_RC.EXIST_STRN DO;
   DO I = 1 TO PARM_RC.NUM_ENV;
    ENV_DATA.DEPLOYED(I) = 0;
    END;
   DO I = 1 TO PARM_RC.EXIST_STR;
    J = EXSTR.ENV(I);
    ENV_DATA.DEPLOYED(J) = ENV_DATA.DEPLOYED(J) + EXSTR.UNITS(I);
    END;
   END;

  SELECT(RSTATUS);
   WHEN(PF1 ) SCNUM = 3;
   WHEN(PF2 ) SCNUM = 5;
   WHEN(PF3 ) SCNUM = 6;
   WHEN(PF4 ) SCNUM = 8;
```

```
   WHEN(PF5 ) SCNUM = 11;
   WHEN(PF6 ) SCNUM = 13;
   WHEN(PF7 ) SCNUM = 3;
   WHEN(PF8 ) SCNUM = 4;
   WHEN(PF9 ) CALL SIMULATE;
   WHEN(PF10) SCNUM = 14;
   WHEN(PF11) SCNUM = 15;
   WHEN(PF12) SCNUM = 1;
   OTHERWISE SCNUM = 16;
   END;
  PRELEASE;
  END EXIST_POP_SCREEN;

/*************************************************************************/
/********** INPUT MAINTENANCE TREE FAILURE AND REPAIR DATA **********/
/*************************************************************************/

NODE_DATA_SCREEN: PROCEDURE;

  DECLARE
   1 IS6A_LOADLIST,
     2 LOAD(82)        PTR,
     2 FENCE           BIT(32)  INIT((32)'1'B);

 IE = 1;
 DO WHILE (IE <= NUM_ENV);

  IS = 1;
  XNG = PARM_RC.NUM_NODES/10 + 1;
  NG = CEIL(XNG);
  DO WHILE (IS <= NG);
  IS6A_LOADLIST.LOAD(1) = ADDR(ENV_DATA.ENV_DESC(IE));
  IS6A_LOADLIST.LOAD(2) = ADDR(PARM_RC.NUM_NODES);
  DO I = (IS-1)*10+1 TO IS*10;
  J = I - (IS-1)*10;
  IS6A_LOADLIST.LOAD((J-1)*8+3) = ADDR(NODE_DATA.NODE_NUM(IE,I));
  IS6A_LOADLIST.LOAD((J-1)*8+4) = ADDR(NODE_DATA.NODE_DESC(IE,I));
  IS6A_LOADLIST.LOAD((J-1)*8+5) = ADDR(NODE_DATA.FAIL_CURVE(IE,I));
  IS6A_LOADLIST.LOAD((J-1)*8+6) = ADDR(NODE_DATA.MTBF(IE,I));
  IS6A_LOADLIST.LOAD((J-1)*8+7) = ADDR(NODE_DATA.MTTR(IE,I));
  IS6A_LOADLIST.LOAD((J-1)*8+8) = ADDR(NODE_DATA.COMPONENTS(IE,I));
  IS6A_LOADLIST.LOAD((J-1)*8+9) = ADDR(NODE_DATA.CRIT_FAIL(IE,I));
  IS6A_LOADLIST.LOAD((J-1)*8+10) = ADDR(NODE_DATA.SPARE_PARTS(IE,I));
   END;

  PDISPLAY PNAME('NODEDATA') RSTATUS(RSTATUS) LDLIST(IS6A_LOADLIST)
           ULDLIST(IS6A_LOADLIST) DCURSOR(CURSOR);

  SELECT(RSTATUS);
    WHEN(PF7 ) DO;
```

```
      IF IS <= 1 THEN DO;
        IF IE <= 1 THEN DO;
          SCNUM = SCNUM - 1;
          IE = NUM_ENV + 1;
          IS = NG + 1;
          END;
        ELSE DO;
         IE = IE - 1;
         IS = NG;
         END;
        END;
      ELSE IS = IS - 1;
      END;

    WHEN(PF8 ) DO;
      IS = IS + 1;
      IF IS      NG THEN DO;
        IE = IE + 1;
        IF IE    NUM_ENV THEN DO;
          SCNUM = SCNUM + 1;
          END;
        END;
      END;

    OTHERWISE DO;
      SELECT(RSTATUS);
        WHEN(PF1 ) SCNUM = 3;
        WHEN(PF2 ) SCNUM = 5;
        WHEN(PF3 ) SCNUM = 6;
        WHEN(PF4 ) SCNUM = 8;
        WHEN(PF5 ) SCNUM = 11;
        WHEN(PF6 ) SCNUM = 13;
        WHEN(PF9 ) CALL SIMULATE;
        WHEN(PF10) SCNUM = 14;
        WHEN(PF11) SCNUM = 15;
        WHEN(PF12) SCNUM = 1;
        OTHERWISE SCNUM = SCNUM + 0;
        END;
      IE = NUM_ENV + 1;
      IS = NG + 1;
      END;
    END;
   END;
  END;
  PRELEASE;
  END NODE_DATA_SCREEN;

/*************************************************************************/
/********************** INPUT END ITEM COST DATA **********************/
/*************************************************************************/
```

```
COST_DATA_SCREEN: PROCEDURE;

  DECLARE
   1 IS4_LOADLIST,
     2 LOAD(2)      PTR,
     2 FENCE        BIT(32)  INIT((32)'1'B);

  IS4_LOADLIST.LOAD(1) = ADDR(COST_RC.FIRST_COST);
  IS4_LOADLIST.LOAD(2) = ADDR(COST_RC.SV_FFAIL);

  PDISPLAY PNAME('EQCOST  ') RSTATUS(RSTATUS) LDLIST(IS4_LOADLIST)
           ULDLIST(IS4_LOADLIST) DCURSOR(CURSOR);

  SELECT(RSTATUS);
   WHEN(PF1 ) SCNUM = 3;
   WHEN(PF2 ) SCNUM = 5;
   WHEN(PF3 ) SCNUM = 6;
   WHEN(PF4 ) SCNUM = 8;
   WHEN(PF5 ) SCNUM = 11;
   WHEN(PF6 ) SCNUM = 13;
   WHEN(PF7 ) SCNUM = SCNUM - 1;
   WHEN(PF8 ) SCNUM = SCNUM + 1;
   WHEN(PF9 ) CALL SIMULATE;
   WHEN(PF10) SCNUM = 14;
   WHEN(PF11) SCNUM = 15;
   WHEN(PF12) SCNUM = 1;
   OTHERWISE SCNUM = SCNUM + 0;
   END;
  PRELEASE;
  END COST_DATA_SCREEN;

/*************************************************************************/
/********************* LEVEL OF REPAIR DATA ******************************/
/*************************************************************************/

 REPAIR_LEVEL_SCREEN: PROCEDURE;

  DECLARE
   1 LOR_LOADLIST,
    2 LOAD(40)      PTR,
    2 FENCE         BIT(32) INIT((32)'1'B);

  IS = 1;
  XNG = PARM_RC.NUM_NODES/10 + 1;
  NG = CEIL(XNG);
  DO WHILE (IS <= NG);
  DO I = (IS-1)*10+1 TO IS*10;
   J = I - (IS-1)*10;
   LOR_LOADLIST.LOAD((J-1)*4+1) = ADDR(NODE_DATA.NODE_NUM(1,I));
```

```
   LOR_LOADLIST.LOAD((J-1)*4+2) = ADDR(NODE_DATA.NODE_DESC(1,I));
   LOR_LOADLIST.LOAD((J-1)*4+3) = ADDR(NODE_DATA.LOR(1,I));
   LOR_LOADLIST.LOAD((J-1)*4+4) = ADDR(NODE_DATA.MAN_HRS(1,I));
  END;

 PDISPLAY PNAME('LORDATA ') RSTATUS(RSTATUS) LDLIST(LOR_LOADLIST)
          ULDLIST(LOR_LOADLIST) DCURSOR(CURSOR);

  SELECT (RSTATUS);
   WHEN(PF7 ) DO;
    IF IS <= 1 THEN DO;
     SCNUM = SCNUM - 1;
     IS = NG + 1;
     END;
    ELSE IS = IS - 1;
    END;

   WHEN(PF8 ) DO;
    IS = IS + 1;
    IF IS        NG THEN DO;
     SCNUM = SCNUM + 1;
     END;
    END;

   OTHERWISE DO;
    SELECT (RSTATUS);
     WHEN(PF1 ) SCNUM = 3;
     WHEN(PF2 ) SCNUM = 5;
     WHEN(PF3 ) SCNUM = 6;
     WHEN(PF4 ) SCNUM = 8;
     WHEN(PF5 ) SCNUM = 11;
     WHEN(PF6 ) SCNUM = 13;
     WHEN(PF9 ) CALL SIMULATE;
     WHEN(PF10) SCNUM = 14;
     WHEN(PF11) SCNUM = 15;
     WHEN(PF12) SCNUM = 1;
     OTHERWISE SCNUM = SCNUM + 0;
     END;
    IS = NG + 1;
    END;
   END;
  END;
  PRELEASE;
  END REPAIR_LEVEL_SCREEN;

/*************************************************************************/
/********************* SPARE PARTS INVENTORY POLICY **********************/
/*************************************************************************/

 INV_REORDER_SCREEN: PROCEDURE;
```

```
 DECLARE
  1 INV_LOADLIST,
   2 LOAD(4)       PTR,
   2 FENCE         BIT(32) INIT((32)'1'B);

  INV_LOADLIST.LOAD(1) = ADDR(COST_RC.ORDER_COST);
  INV_LOADLIST.LOAD(2) = ADDR(COST_RC.WHSE_COST);
  INV_LOADLIST.LOAD(3) = ADDR(COST_RC.INTEREST);
  INV_LOADLIST.LOAD(4) = ADDR(PARM_RC.LOT_SIZE);

 PDISPLAY PNAME('INVDATA ') RSTATUS(RSTATUS) LDLIST(INV_LOADLIST)
          ULDLIST(INV_LOADLIST) DCURSOR(CURSOR);

 SELECT (RSTATUS);
   WHEN(PF1 ) SCNUM = 3;
   WHEN(PF2 ) SCNUM = 5;
   WHEN(PF3 ) SCNUM = 6;
   WHEN(PF4 ) SCNUM = 8;
   WHEN(PF5 ) SCNUM = 11;
   WHEN(PF6 ) SCNUM = 13;
   WHEN(PF7 ) SCNUM = SCNUM - 1;
   WHEN(PF8 ) SCNUM = SCNUM + 1;
   WHEN(PF9 ) CALL SIMULATE;
   WHEN(PF10) SCNUM = 14;
   WHEN(PF11) SCNUM = 15;
   WHEN(PF12) SCNUM = 1;
   OTHERWISE SCNUM = SCNUM + 0;
   END;
  PRELEASE;
  END INV_REORDER_SCREEN;

/*************************************************************************************/
/************************* STOCKABLE PARTS SCREENS ***************************/
/*************************************************************************************/

SPARE_PARTS_SCREEN: PROCEDURE;

 DECLARE
   1 I102_LOADLIST,
     2 LOAD(61)          PTR,
     2 FENCE             BIT(32) INIT((32)'1'B);

   IP = 1;
   IN = 1;

 NUM_STK_RC = 0;
 DO I = 1 TO PARM_RC.NUM_NODES;
  NUM_STK_RC = NUM_STK_RC + NODE_DATA.SPARE_PARTS(1,I);
  END;
```

```
 DO WHILE (IN <= PARM_RC.NUM_NODES);
  NUM_PARTS(IN) = NODE_DATA.SPARE_PARTS(1,IN);
  I102_LOADLIST.LOAD(1) = ADDR(NODE_DATA.NODE_DESC(1,IN));

  DO I = 1 TO NUM_PARTS(IN);
   I102_LOADLIST.LOAD((I-1)*6+2) = ADDR(PART_DATA.PART_DESC(IP));
   I102_LOADLIST.LOAD((I-1)*6+3) = ADDR(PART_DATA.PART_NUM(IP));
   I102_LOADLIST.LOAD((I-1)*6+4) = ADDR(PART_DATA.QUANTITY(IP));
   I102_LOADLIST.LOAD((I-1)*6+5) = ADDR(PART_DATA.WH_SPACE(IP));
   I102_LOADLIST.LOAD((I-1)*6+6) = ADDR(PART_DATA.UNIT_PRICE(IP));
   I102_LOADLIST.LOAD((I-1)*6+7) = ADDR(PART_DATA.LEAD_TIME(IP));
   IP = IP + 1;
   END;

/**** LOAD REMAINDER OF SPARE PARTS DATA SCREEN ****/

IFILL = 6*NUM_PARTS(IN)+2;
DO I = IFILL TO 61;
  I102_LOADLIST.LOAD(I) = ADDR(FILL_DATA(I));
  END;

 BB: PDISPLAY PNAME('PARTSDAT') RSTATUS(RSTATUS)
              LDLIST(I102_LOADLIST) ULDLIST(I102_LOADLIST)
              DCURSOR(CURSOR);

  SELECT(RSTATUS);
    WHEN(PF7 ) DO;
     IF IN = 1 THEN DO;
      SCNUM = SCNUM - 1;
      IN = PARM_RC.NUM_NODES + 1;
      END;
     ELSE DO;
      IP = IP - NUM_PARTS(IN) - NUM_PARTS(IN-1);
      IN = IN - 1;
      END;
     END;
    WHEN(PF8 ) DO;
     IF IN = PARM_RC.NUM_NODES THEN DO;
      SCNUM = SCNUM + 1;
      IN = PARM_RC.NUM_NODES + 1;
      END;
     ELSE IN = IN + 1;
     END;
    WHEN(PF9 ) IN = IN + 0;
    OTHERWISE DO;
      SELECT(RSTATUS);
        WHEN(PF1 ) SCNUM = 3;
        WHEN(PF2 ) SCNUM = 5;
        WHEN(PF3 ) SCNUM = 6;
```

```
            WHEN(PF4 ) SCNUM = 8;
            WHEN(PF5 ) SCNUM = 11;
            WHEN(PF6 ) SCNUM = 13;
            WHEN(PF9 ) CALL SIMULATE;
            WHEN(PF10) SCNUM = 14;
            WHEN(PF11) SCNUM = 15;
            WHEN(PF12) SCNUM = 1;
            OTHERWISE SCNUM = SCNUM + 0;
            END;
          IN = PARM_RC.NUM_NODES + 1;
          END;
        END;
      END;

END SPARE_PARTS_SCREEN;

/*************************************************************************/
/********************** PROCUREMENT POLICY SCREENS ***********************/
/*************************************************************************/

PROC_POLICY_SCREEN: PROCEDURE;

/*******************************************************/
/****** PROCUREMENT POLICY SELECTION SCREEN ******/
/*******************************************************/

  DECLARE
   1 IS8_LOADLIST,
     2 LOAD(2)                    PTR,
     2 FENCE                      BIT(32)          INIT((32)'1'B);

  IS8_LOADLIST.LOAD(1) = ADDR(PROC_RC.POLICY_NUM);
  IS8_LOADLIST.LOAD(2) = ADDR(COST_RC.PROC_COST);

  PDISPLAY PNAME('PROCOPT ') RSTATUS(RSTATUS)  LDLIST(IS8_LOADLIST)
          ULDLIST(IS8_LOADLIST) DCURSOR(CURSOR);

     SELECT (RSTATUS);
      WHEN(PF1 ) SCNUM = 3;
      WHEN(PF2 ) SCNUM = 5;
      WHEN(PF3 ) SCNUM = 6;
      WHEN(PF4 ) SCNUM = 8;
      WHEN(PF5 ) SCNUM = 11;
      WHEN(PF6 ) SCNUM = 13;
      WHEN(PF7 ) SCNUM = SCNUM - 1;
      WHEN(PF8 ) DO;
         SELECT(PROC_RC.POLICY_NUM);
           WHEN(1) CALL PROC_POLICY_SC1;
           WHEN(2) CALL PROC_POLICY_SC2;
           WHEN(3) CALL PROC_POLICY_SC3;
```

```
            WHEN(4) CALL PROC_POLICY_SC4;
            WHEN(5) CALL PROC_POLICY_SC5;
            WHEN(6) CALL PROC_POLICY_SC6;
            WHEN(7) CALL PROC_POLICY_SC7;
            WHEN(8) CALL PROC_POLICY_SC8;
            WHEN(9) CALL PROC_POLICY_SC9;
            OTHERWISE SCNUM = SCNUM + 0;
          END;
        END;
       WHEN(PF9) CALL SIMULATE;
       WHEN(PF10) SCNUM = 14;
       WHEN(PF11) SCNUM = 15;
       WHEN(PF12) SCNUM = 1;
       OTHERWISE  SCNUM = SCNUM + 0;
       END;

/*********************************************/
/***** CONTINUOUS UNIFORM PROCUREMENT *****/
/*********************************************/

PROC_POLICY_SC1: PROCEDURE;

 DECLARE
   1 IS91_LOADLIST,
     2 LOAD(4)                  PTR,
     2 FENCE                    BIT(32)         INIT((32)'1'B);

  IS91_LOADLIST.LOAD(1) = ADDR(MIN_POPSIZE);
  IS91_LOADLIST.LOAD(2) = ADDR(PROC_RC.INIT_PROC);
  IS91_LOADLIST.LOAD(3) = ADDR(PROC_RC.PD_BETW_PROC);
  IS91_LOADLIST.LOAD(4) = ADDR(PROC_RC.ORDER_SIZE);

  PP_SC1: PDISPLAY PNAME('PROCOPT1') RSTATUS(RSTATUS)
      LDLIST(IS91_LOADLIST) ULDLIST(IS91_LOADLIST)  DCURSOR(CURSOR);

    SELECT (RSTATUS);
      WHEN(PF1 ) SCNUM = SCNUM + 100;
      WHEN(PF7 ) DO;
        PRELEASE PNAME('PROCOPT1');
        END;
      WHEN(PF8 ) SCNUM = SCNUM + 1;
      WHEN(PF12) SCNUM = 1;
      OTHERWISE GO TO PP_SC1;
      END;
   END PROC_POLICY_SC1;

/*****************************************************************/
/***** UNIFORM PROCUREMENT UNTIL UPPER LIMIT IS REACHED *****/
/*****************************************************************/
```

```
PROC_POLICY_SC2: PROCEDURE;

 DECLARE
   1 IS92_LOADLIST,
     2 LOAD(6)                  PTR,
     2 FENCE                    BIT(32)         INIT((32)'1'B);

  IS92_LOADLIST.LOAD(1) = ADDR(MIN_POPSIZE);
  IS92_LOADLIST.LOAD(2) = ADDR(PROC_RC.INIT_PROC);
  IS92_LOADLIST.LOAD(3) = ADDR(PROC_RC.PD_BETW_PROC);
  IS92_LOADLIST.LOAD(4) = ADDR(PROC_RC.ORDER_SIZE);
  IS92_LOADLIST.LOAD(5) = ADDR(PROC_RC.UP_POP_LIMIT);
  IS92_LOADLIST.LOAD(6) = ADDR(PROC_RC.POLICY_OPT);

  PP_SC2: PDISPLAY PNAME('PROCOPT2') RSTATUS(RSTATUS)
      LDLIST(IS92_LOADLIST) ULDLIST(IS92_LOADLIST)  DCURSOR(CURSOR);

    SELECT (RSTATUS);
      WHEN(PF1 ) SCNUM = SCNUM + 100;
      WHEN(PF7 ) DO;
        PRELEASE PNAME('PROCOPT2');
        END;
      WHEN(PF8 ) SCNUM = SCNUM + 1;
      WHEN(PF12) SCNUM = 1;
      OTHERWISE GO TO PP_SC2;
      END;
   END PROC_POLICY_SC2;

/*********************************************/
/***** ONE INITIAL BLOCK PROCUREMENT *****/
/*********************************************/

PROC_POLICY_SC3: PROCEDURE;

 DECLARE
   1 IS93_LOADLIST,
     2 LOAD(2)                  PTR,
     2 FENCE                    BIT(32)         INIT((32)'1'B);

  IS93_LOADLIST.LOAD(1) = ADDR(MIN_POPSIZE);
  IS93_LOADLIST.LOAD(2) = ADDR(PROC_RC.INIT_PROC);

  PROC_RC.PD_BETW_PROC = PARM_RC.RUN_LENGTH + 1;

  PP_SC3: PDISPLAY PNAME('PROCOPT3') RSTATUS(RSTATUS)
      LDLIST(IS93_LOADLIST) ULDLIST(IS93_LOADLIST)  DCURSOR(CURSOR);

    SELECT (RSTATUS);
      WHEN(PF1 ) SCNUM = SCNUM + 100;
      WHEN(PF7 ) DO;
```

```
         PRELEASE PNAME('PROCOPT3');
         END;
       WHEN(PF8 ) SCNUM = SCNUM + 1;
       WHEN(PF12) SCNUM = 1;
       OTHERWISE GO TO PP_SC3;
       END;
   END PROC_POLICY_SC3;

/*************************************************************************/
/**** BLOCK PROCUREMENT WITH REPLENISHMENT AS UNITS FAIL FATALLY ***/
/*************************************************************************/

PROC_POLICY_SC4: PROCEDURE;

 DECLARE
   1 IS94_LOADLIST,
     2 LOAD(3)                  PTR,
     2 FENCE                    BIT(32)         INIT((32)'1'B);

  IS94_LOADLIST.LOAD(1) = ADDR(MIN_POPSIZE);
  IS94_LOADLIST.LOAD(2) = ADDR(PROC_RC.INIT_PROC);
  IS94_LOADLIST.LOAD(3) = ADDR(PROC_RC.PD_BETW_PROC);

  PP_SC4: PDISPLAY PNAME('PROCOPT4') RSTATUS(RSTATUS)
      LDLIST(IS94_LOADLIST) ULDLIST(IS94_LOADLIST)  DCURSOR(CURSOR);

    SELECT (RSTATUS);
      WHEN(PF1 ) SCNUM = SCNUM + 100;
      WHEN(PF7 ) DO;
         PRELEASE PNAME('PROCOPT4');
         END;
      WHEN(PF8 ) SCNUM = SCNUM + 1;
      WHEN(PF12) SCNUM = 1;
      OTHERWISE GO TO PP_SC4;
      END;
   END PROC_POLICY_SC4;

/*************************************************/
/****** INCREASING PROCUREMENT AMOUNTS ******/
/*************************************************/

PROC_POLICY_SC5: PROCEDURE;

 DECLARE
   1 IS95_LOADLIST,
     2 LOAD(4)                  PTR,
     2 FENCE                    BIT(32)         INIT((32)'1'B);

  IS95_LOADLIST.LOAD(1) = ADDR(MIN_POPSIZE);
  IS95_LOADLIST.LOAD(2) = ADDR(PROC_RC.INIT_PROC);
```

```
   IS95_LOADLIST.LOAD(3) = ADDR(PROC_RC.PD_BETW_PROC);
   IS95_LOADLIST.LOAD(4) = ADDR(PROC_RC.PCT_INC);

   PP_SC5: PDISPLAY PNAME('PROCOPT5') RSTATUS(RSTATUS)
       LDLIST(IS95_LOADLIST) ULDLIST(IS95_LOADLIST)  DCURSOR(CURSOR);

     SELECT (RSTATUS);
       WHEN(PF1 ) SCNUM = SCNUM + 100;
       WHEN(PF7 ) DO;
          PRELEASE PNAME('PROCOPT5');
          END;
       WHEN(PF8 ) SCNUM = SCNUM + 1;
       WHEN(PF12) SCNUM = 1;
       OTHERWISE GO TO PP_SC5;
       END;
    END PROC_POLICY_SC5;

/*************************************************************************/
/****** INCREASING PROCUREMENT UNTIL UPPER LIMIT IS REACHED ******/
/*************************************************************************/

PROC_POLICY_SC6: PROCEDURE;

 DECLARE
   1 IS96_LOADLIST,
     2 LOAD(6)                  PTR,
     2 FENCE                    BIT(32)         INIT((32)'1'B);

  IS96_LOADLIST.LOAD(1) = ADDR(MIN_POPSIZE);
  IS96_LOADLIST.LOAD(2) = ADDR(PROC_RC.INIT_PROC);
  IS96_LOADLIST.LOAD(3) = ADDR(PROC_RC.PD_BETW_PROC);
  IS96_LOADLIST.LOAD(4) = ADDR(PROC_RC.PCT_INC);
  IS96_LOADLIST.LOAD(5) = ADDR(PROC_RC.UP_POP_LIMIT);
  IS96_LOADLIST.LOAD(6) = ADDR(PROC_RC.POLICY_OPT);

  PP_SC6: PDISPLAY PNAME('PROCOPT6') RSTATUS(RSTATUS)
      LDLIST(IS96_LOADLIST) ULDLIST(IS96_LOADLIST)  DCURSOR(CURSOR);

    SELECT (RSTATUS);
      WHEN(PF1 ) SCNUM = SCNUM + 100;
      WHEN(PF7 ) DO;
         PRELEASE PNAME('PROCOPT6');
         END;
      WHEN(PF8 ) SCNUM = SCNUM + 1;
      WHEN(PF12) SCNUM = 1;
      OTHERWISE GO TO PP_SC6;
      END;
   END PROC_POLICY_SC6;

/*************************************************/
```

```
/***** DECREASING PROCUREMENT AMOUNTS *****/
/*******************************************/

PROC_POLICY_SC7: PROCEDURE;

 DECLARE
   1 IS97_LOADLIST,
     2 LOAD(4)                  PTR,
     2 FENCE                    BIT(32)         INIT((32)'1'B);

  IS97_LOADLIST.LOAD(1) = ADDR(MIN_POPSIZE);
  IS97_LOADLIST.LOAD(2) = ADDR(PROC_RC.INIT_PROC);
  IS97_LOADLIST.LOAD(3) = ADDR(PROC_RC.PD_BETW_PROC);
  IS97_LOADLIST.LOAD(4) = ADDR(PROC_RC.PCT_DEC);

  PP_SC7: PDISPLAY PNAME('PROCOPT7') RSTATUS(RSTATUS)
      LDLIST(IS97_LOADLIST) ULDLIST(IS97_LOADLIST)  DCURSOR(CURSOR);

    SELECT (RSTATUS);
      WHEN(PF1 ) SCNUM = SCNUM + 100;
      WHEN(PF7 ) DO;
         PRELEASE PNAME('PROCOPT7');
         END;
      WHEN(PF8 ) SCNUM = SCNUM + 1;
      WHEN(PF12) SCNUM = 1;
      OTHERWISE GO TO PP_SC7;
      END;
   END PROC_POLICY_SC7;

/*************************************************************************/
/***** DECREASING PROCUREMENT UNTIL UPPER LIMIT IS REACHED *****/
/*************************************************************************/

PROC_POLICY_SC8: PROCEDURE;

 DECLARE
   1 IS98_LOADLIST,
     2 LOAD(6)                  PTR,
     2 FENCE                    BIT(32)         INIT((32)'1'B);

  IS98_LOADLIST.LOAD(1) = ADDR(MIN_POPSIZE);
  IS98_LOADLIST.LOAD(2) = ADDR(PROC_RC.INIT_PROC);
  IS98_LOADLIST.LOAD(3) = ADDR(PROC_RC.PD_BETW_PROC);
  IS98_LOADLIST.LOAD(4) = ADDR(PROC_RC.PCT_DEC);
  IS98_LOADLIST.LOAD(5) = ADDR(PROC_RC.UP_POP_LIMIT);
  IS98_LOADLIST.LOAD(6) = ADDR(PROC_RC.POLICY_OPT);

  PP_SC8: PDISPLAY PNAME('PROCOPT8') RSTATUS(RSTATUS)
      LDLIST(IS98_LOADLIST) ULDLIST(IS98_LOADLIST)  DCURSOR(CURSOR);
```

```
    SELECT (RSTATUS);
      WHEN(PF1 ) SCNUM = SCNUM + 100;
      WHEN(PF7 ) DO;
         PRELEASE PNAME('PROCOPT8');
         END;
      WHEN(PF8 ) SCNUM = SCNUM + 1;
      WHEN(PF12) SCNUM = 1;
      OTHERWISE GO TO PP_SC8;
      END;
   END PROC_POLICY_SC8;

/*************************************/
/***** OTHER PROCUREMENT POLICY *****/
/*************************************/

PROC_POLICY_SC9: PROCEDURE;

 DECLARE
   1 IS99_LOADLIST,
     2 LOAD(3)                  PTR,
     2 FENCE                    BIT(32)         INIT((32)'1'B);

  IS99_LOADLIST.LOAD(1) = ADDR(MIN_POPSIZE);
  IS99_LOADLIST.LOAD(2) = ADDR(POPSIZE);
  IS99_LOADLIST.LOAD(3) = ADDR(PROC_RC.INIT_PROC);

  PROC_RC.PD_BETW_PROC = 1;

  PP_SC9: PDISPLAY PNAME('PROCOPT9') RSTATUS(RSTATUS)
      LDLIST(IS99_LOADLIST) ULDLIST(IS99_LOADLIST)  DCURSOR(CURSOR);

    SELECT (RSTATUS);
      WHEN(PF1 ) SCNUM = SCNUM + 100;
      WHEN(PF7 ) DO;
         PRELEASE PNAME('PROCOPT9');
         END;
      WHEN(PF8 ) SCNUM = SCNUM + 1;
      WHEN(PF12) SCNUM = 1;
      OTHERWISE GO TO PP_SC9;
      END;
   END PROC_POLICY_SC9;

END PROC_POLICY_SCREEN;

/*****************************************************************************/
/********************* DEPLOY PROCUREMENT ************************************/
/*****************************************************************************/

 DEPLOY_PROC_SCREEN: PROCEDURE;
```

```
DECLARE
 1 DEP_LOADLIST,
  2 LOAD(21)      PTR,
  2 FENCE         BIT(32) INIT((32)'1'B);

 DEP_LOADLIST.LOAD(1) = ADDR(PROC_RC.INIT_PROC);

 TDEMAND = 0;
 DO I = 1 TO PARM_RC.NUM_ENV;
  TDEMAND = TDEMAND + ENV_DATA.DEMAND(I);
  END;

 NUDEPLOY = 0;
 DO I = 1 TO PARM_RC.NUM_ENV;
  NUDEPLOY = NUDEPLOY + ENV_DATA.NEW_UNITS(I);
  END;

 IF NUDEPLOY   0 THEN DO;
 /** CALCULATE DEPLOYMENT BASED UPON INITIAL DEPLOYMENT **/
  ENV_DATA.NEW_UNITS(PARM_RC.NUM_ENV) = PROC_RC.INIT_PROC;
 DO I = 1 TO PARM_RC.NUM_ENV-1;
  ENVUNITS = ENV_DATA.NEW_UNITS(I)/NUDEPLOY*PROC_RC.INIT_PROC;
  ENV_DATA.NEW_UNITS(I) = ROUND(ENVUNITS,0) + 1;
  ENV_DATA.NEW_UNITS(PARM_RC.NUM_ENV) =
  ENV_DATA.NEW_UNITS(PARM_RC.NUM_ENV) - ENV_DATA.NEW_UNITS(I);
  END;
 END;

 ELSE DO;
 /** CALCULATE DEPLOYMENT BASED UPON DEMAND **/
  ENV_DATA.NEW_UNITS(PARM_RC.NUM_ENV) = PROC_RC.INIT_PROC;
 DO I = 1 TO PARM_RC.NUM_ENV-1;
  ENVUNITS = ENV_DATA.DEMAND(I)/TDEMAND*PROC_RC.INIT_PROC;
  ENV_DATA.NEW_UNITS(I) = ROUND(ENVUNITS,0) + 1;
  ENV_DATA.NEW_UNITS(PARM_RC.NUM_ENV) =
  ENV_DATA.NEW_UNITS(PARM_RC.NUM_ENV) - ENV_DATA.NEW_UNITS(I);
  END;
 END;

DO I = 1 TO 4;
 DEP_LOADLIST.LOAD((I-1)*5+2) = ADDR(ENV_DATA.ENV_DESC(I));
 DEP_LOADLIST.LOAD((I-1)*5+3) = ADDR(ENV_DATA.RETIRE_AGE(I));
 DEP_LOADLIST.LOAD((I-1)*5+4) = ADDR(ENV_DATA.DEMAND(I));
 DEP_LOADLIST.LOAD((I-1)*5+5) = ADDR(ENV_DATA.DEPLOYED(I));
 DEP_LOADLIST.LOAD((I-1)*5+6) = ADDR(ENV_DATA.NEW_UNITS(I));
END;

PDISPLAY PNAME('DEPLOY  ') RSTATUS(RSTATUS) LDLIST(DEP_LOADLIST)
         ULDLIST(DEP_LOADLIST) DCURSOR(CURSOR);
```

```
 SELECT (RSTATUS);
   WHEN(PF1 ) SCNUM = 3;
   WHEN(PF2 ) SCNUM = 5;
   WHEN(PF3 ) SCNUM = 6;
   WHEN(PF4 ) SCNUM = 8;
   WHEN(PF5 ) SCNUM = 11;
   WHEN(PF6 ) SCNUM = 13;
   WHEN(PF7 ) SCNUM = SCNUM - 1;
   WHEN(PF8 ) SCNUM = SCNUM + 1;
   WHEN(PF9 ) CALL SIMULATE;
   WHEN(PF10) SCNUM = 14;
   WHEN(PF11) SCNUM = 15;
   WHEN(PF12) SCNUM = 1;
   OTHERWISE SCNUM = SCNUM + 0;
   END;
  PRELEASE;
  END DEPLOY_PROC_SCREEN;

/*************************************************************************/
/***************** RETIREMENT AGE AND SALVAGE VALUE ******************/
/*************************************************************************/

 RETIRE_AGE_SCREEN: PROCEDURE;

  DECLARE
   1 RET_LOADLIST,
    2 LOAD(12)     PTR,
    2 FENCE        BIT(32) INIT((32)'1'B);

  DO I = 1 TO 4;
   RET_LOADLIST.LOAD((I-1)*3+1) = ADDR(ENV_DATA.ENV_DESC(I));
   RET_LOADLIST.LOAD((I-1)*3+2) = ADDR(ENV_DATA.RETIRE_AGE(I));
   RET_LOADLIST.LOAD((I-1)*3+3) = ADDR(ENV_DATA.SV_RETIRE(I));
  END;

 PDISPLAY PNAME('REPLACE ') RSTATUS(RSTATUS) LDLIST(RET_LOADLIST)
          ULDLIST(RET_LOADLIST) DCURSOR(CURSOR);

 SELECT (RSTATUS);
   WHEN(PF1 ) SCNUM = 3;
   WHEN(PF2 ) SCNUM = 5;
   WHEN(PF3 ) SCNUM = 6;
   WHEN(PF4 ) SCNUM = 8;
   WHEN(PF5 ) SCNUM = 11;
   WHEN(PF6 ) SCNUM = 13;
   WHEN(PF7 ) SCNUM = SCNUM - 1;
   WHEN(PF8 ) SCNUM = SCNUM + 1;
   WHEN(PF9 ) CALL SIMULATE;
   WHEN(PF10) SCNUM = 14;
   WHEN(PF11) SCNUM = 15;
```

```
   WHEN(PF12) SCNUM = 1;
   OTHERWISE SCNUM = SCNUM + 0;
   END;
  PRELEASE;
  END RETIRE_AGE_SCREEN;

/*************************************************************************/
/******************* SIMULATE OPERATIONS *********************************/
/*************************************************************************/

 SIMULATE: PROCEDURE;

 /*** INITIALIZE VARIABLES ***/

 DO I = 1 TO 50;
  STREAM.CDATE(I) = 0;
  STREAM.FAIL_ENV(I) = 0;
  STREAM.UNITS(I) = 0;
  STREAM.INIT_FIELD(I) = 0;

  FFAILS(I) = 0;

  DO J = 1 TO 50;
   NFPROB(I,J) = 0;
   END;

  DO J = 1 TO 30;
   EFAIL(J,I) = 0;
   SMTTR(J,I) = 0;
   STRQTY.BEG_PD(J,I) = 0;
   STRQTY.END_PD(J,I) = 0;

   DO K = 1 TO 26;
    NFAILRPT(I,J,K) = 0;
    END;

   END;
  END;

  DO I = 1 TO NUM_STK_RC;
   DO J = 1 TO PARM_RC.RUN_LENGTH;
    PART_DMD.PD_DMD(J,I) = 0;
    PART_DMD.BEG_INV(J,I) = 0;
    PART_DMD.END_INV(J,I) = 0;
    PART_DMD.LOT_SIZE(J,I) = 0;
    END;
   END;

 DO I = 1 TO 30;
```

```
 PMTTR(I) = 0;
 PROCQTY(I) = 0;
 PAVAIL(I) = 0;

 DO J = 1 TO 4;
  ENVQTY(I,J) = 0;
  EMTTR(I,J) = 0;
  EPFAIL(I,J) = 0;
  EPFAIL1(I,J) = 0;
  ENVAVAIL(I,J) = 0;
  END;

 DO J = 1 TO PARM_RC.NUM_ENV;
  COST_RPT.CAP_COST(I,J) = 0;
  COST_RPT.REP_COST(I,J) = 0;
  COST_RPT.PUR_COST(I,J) = 0;
  COST_RPT.INV_COST(I,J) = 0;
  COST_RPT.TOT_COST(I,J) = 0;
  COST_RPT.UNITS_START(I,J) = 0;
  COST_RPT.UNITS_END(I,J) = 0;
  COST_RPT.UNIT_COST(I,J) = 0;
  END;

 DO J = 1 TO 3;
  SPACE(I,J) = 0;
  LABOR(I,J) = 0;
  END;

 END;

DO I = 1 TO 50;
 DO J = 1 TO 126;
  MTPROB(I,J) = 0;
  END;
 END;

PROCNUM = 0;
ULSWITCH = 0;
CTIME = 0;
POPSIZE = 0;

 /*** CREATE STREAMS FOR EXISTING POPULATION ***/

IF PARM_RC.MODEL_MODE = 2 THEN DO;
 DO JS = 1 TO PARM_RC.EXIST_STR;
  STREAM.CDATE(JS) = CTIME - EXSTR.AGE(JS);
  STREAM.FAIL_ENV(JS) = EXSTR.ENV(JS);
  STREAM.UNITS(JS) = EXSTR.UNITS(JS);
  STREAM.INIT_FIELD(JS) = EXSTR.UNITS(JS);
  POPSIZE = POPSIZE + STREAM.UNITS(JS);
```

```
 END;
END;

ELSE PARM_RC.EXIST_STR = 0;

NUM_STREAMS = PARM_RC.NUM_ENV + PARM_RC.EXIST_STR;
NXTPROC = PROC_RC.PD_BETW_PROC;

/*** CREATE ONE NEW STREAM FOR EACH OPERATING ENVIRONMENT ***/

DO JS = 1 + PARM_RC.EXIST_STR TO NUM_STREAMS;
 IE = JS - PARM_RC.EXIST_STR;
 STREAM.CDATE(JS) = CTIME;
 STREAM.FAIL_ENV(JS) = IE;
 STREAM.UNITS(JS) = ENV_DATA.NEW_UNITS(IE);
 STREAM.INIT_FIELD(JS) = ENV_DATA.NEW_UNITS(IE);
 COST_RPT.CAP_COST(CTIME+1,IE) = COST_RPT.CAP_COST(CTIME+1,IE)
                        + COST_RC.FIRST_COST * STREAM.UNITS(JS);
 COST_RPT.PUR_COST(CTIME+1,IE) = COST_RC.PROC_COST/PARM_RC.NUM_ENV;
 POPSIZE = POPSIZE + STREAM.UNITS(JS);
 END;

/*** SET CLOCK TIME ***/
CTIME = 1;

DO WHILE (CTIME <= PARM_RC.RUN_LENGTH);

/*** COMPUTE NODE FAILURE PROBABILITIES FOR EACH STREAM ***/

DO JS = 1 TO NUM_STREAMS;
  DO JN = 1 TO PARM_RC.NUM_NODES;
    JE = STREAM.FAIL_ENV(JS);
    FC = NODE_DATA.FAIL_CURVE(JE,JN);
    LAMDA = 1/(NODE_DATA.MTBF(JE,JN) *
            NODE_DATA.COMPONENTS(JE,JN) / PARM_RC.PD_LENGTH);
    ADJAGE = TRUNC(2/LAMDA);
    AGE1 = CTIME - STREAM.CDATE(JS);
    AGE = MOD(AGE1,ADJAGE);

SELECT (FC);

/*** UNIFORM DISTRIBUTION ***/
WHEN ( 1) PFAIL = LAMDA/2;

/*** EXPONENTIAL DISTRIBUTION ***/
WHEN (2) PFAIL = 1/(EXP(LAMDA*(AGE-1)))-1/(EXP(LAMDA*AGE));

/*** INCREASING EXPONENTIAL DISTRIBUTION ***/
WHEN (3) DO;
 PFAIL = 1/EXP(1/LAMDA)*(EXP(AGE/2)-EXP((AGE-1)/2));
```

```
    END;

   /*** NORMAL DIST. (WEIBULL WITH SHAPE PARAMETER = 3 ***/
   WHEN (4) DO;
    ET = (LAMDA*AGE)**3;
    ET1 = (LAMDA*(AGE-1))**3;
    PFAIL = 1/EXP(ET1) - 1/EXP(ET);
    END;

   /*** BATHTUB DISTRIBUTION ***/
   WHEN (5) DO;
    PFAIL = 1/EXP(2)*(EXP(LAMDA*AGE)-EXP(LAMDA*(AGE-1)));
    END;

   OTHERWISE PFAIL = 0;
   END;

   IF NODE_DATA.CRIT_FAIL(JE,JN) = 'N' THEN NFPROB(JS,JN) = 0;
   ELSE NFPROB(JS,JN) = PFAIL;
   END;
  END;

   /*** DETERMINE EXPECTED NUMBER OF FAILURES AT EACH NODE ***/
   DO JN = 1 TO PARM_RC.NUM_NODES;
     ENFAIL = 0;
     DO JS = 1 TO NUM_STREAMS;
       ENFAIL = ENFAIL + NFPROB(JS,JN)*STREAM.UNITS(JS) + 0.5;
       END;
     EFAIL(CTIME,JN) = ROUND(ENFAIL,0);
     END;

   /*** COMPUTE REPAIR COSTS ***/
   DO JN = 1 TO PARM_RC.NUM_NODES;
    DO JS = 1 TO NUM_STREAMS;
     XENV = STREAM.FAIL_ENV(JS);
     IRL = NODE_DATA.LOR(1,JN);
     COST_RPT.REP_COST(CTIME,XENV) = COST_RPT.REP_COST(CTIME,XENV)
     + NFPROB(JS,JN) * NODE_DATA.MTTR(XENV,JN) * STREAM.UNITS(JS)/8
     * LEVELS_DATA.COST(IRL) * LEVELS_DATA.CHANNELS(IRL);
     END;
    END;

  /*** DETERMINE MTTR FOR EACH STREAM ***/
  DO JS = 1 TO NUM_STREAMS;
   IE = STREAM.FAIL_ENV(JS);
   SNMTTR = 0;
   DO JN = 1 TO PARM_RC.NUM_NODES;
    SNMTTR = SNMTTR + NFPROB(JS,JN)*NODE_DATA.MTTR(IE,JN);
    END;
   SNMTTR = SNMTTR + 0.5;
```

```
 SMTTR(CTIME,JS) = ROUND(SNMTTR,0);
 END;

/*** DETERMINE SPARE PART DEMAND ***/
JP = 0;
J1 = 1;
DO JN = 1 TO PARM_RC.NUM_NODES;
  JP = JP + NUM_PARTS(JN);
  DO K = J1 TO JP;

  PART_DMD.PD_DMD(CTIME,K) = PART_DATA.QUANTITY(K)*EFAIL(CTIME,JN);
  END;
  J1 = JP + 1;
  END;

/*** COMPUTE FAILURE PROBABILITIES FOR NODES 1-26 ***/
DO JS = 1 TO NUM_STREAMS;
  DO JN = 1 TO PARM_RC.NUM_NODES;
    JNUM = NODE_DATA.NODE_NUM(1,JN);
    MTPROB(JS,JNUM) = NFPROB(JS,JN);
    END;

  DO NX = 26 TO 1 BY -1;
    SELECT;
     WHEN(NX   6) NX1 = (NX-2)*5+2;
     WHEN(NX<6) NX1 = (NX-1)*5+2;
     OTHERWISE GO TO NXTNX;
     END;

    PROB = 0;

    /*** SUM PROBABILITIES TAKEN ONE-AT-A-TIME ***/
    DO I1 = NX1 TO NX1+4;
      PROB = PROB + MTPROB(JS,I1);
      END;

    /*** SUM PROBABILITIES TAKEN TWO-AT-A-TIME ***/
    DO I1 = NX1 TO NX1+3;
     DO I2 = I1+1 TO NX1+4;
      PROB = PROB - MTPROB(JS,I1)*MTPROB(JS,I2);
     END;
    END;

    /*** SUM PROBABILITIES TAKEN THREE-AT-A-TIME ***/
    DO I1 = NX1 TO NX1+2;
     DO I2 = I1+1 TO NX1+3;
      DO I3 = I2+1 TO NX1+4;
       PROB = PROB + MTPROB(JS,I1)*MTPROB(JS,I2)*MTPROB(JS,I3);
      END;
     END;
```

```
    END;

    /*** SUM PROBABILITIES TAKEN FOUR-AT-A-TIME ***/
    DO I1 = NX1 TO NX1+1;
     DO I2 = I1+1 TO NX1+2;
      DO I3 = I2+1 TO NX1+3;
       DO I4 = I3+1 TO NX1+4;
        PROB4 = MTPROB(JS,I1)*MTPROB(JS,I2)*MTPROB(JS,I3);
        PROB = PROB - PROB4*MTPROB(JS,I4);
       END;
      END;
     END;
    END;

    /*** SUM PROBABILITIES TAKEN FIVE-AT-A-TIME ***/
    PROB5 = 1.0;
    DO I5 = NX1 TO NX1+4;
      PROB5 = PROB5*MTPROB(JS,I5);
      END;
    PROB = PROB + PROB5;

   /*** SUMMARIZE NODE NX FAILURE PROBABILITY ***/
   MTPROB(JS,NX) = PROB;

   NXTNX: END;

   DO JN = 1 TO 26;
    NFAILRPT(JS,CTIME,JN) = MTPROB(JS,JN);
    END;

/** COMPUTE FATAL FAILURES **/

STRFAIL = MTPROB(JS,6)*STREAM.UNITS(JS) + 0.5;
FFAILS(JS) = ROUND(STRFAIL,0);
STRQTY.BEG_PD(CTIME,JS) = STREAM.UNITS(JS);
STRQTY.END_PD(CTIME,JS) = STREAM.UNITS(JS) - FFAILS(JS);
STREAM.UNITS(JS) = STREAM.UNITS(JS) - FFAILS(JS);
POPSIZE = POPSIZE - FFAILS(JS);
XENV = STREAM.FAIL_ENV(JS);
COST_RPT.CAP_COST(CTIME,XENV) = COST_RPT.CAP_COST(CTIME,XENV)
                              - FFAILS(JS)*COST_RC.SV_FFAIL;

 NXTJS: END;

/*** PERIOD AVAILABILITY CALCULATIONS ***/

DO IE = 1 TO PARM_RC.NUM_ENV;
 ENVUNITS = 0;
 ENVMTTR = 0;
 ENVPFAIL = 0;
```

```
  DO JS = 1 TO NUM_STREAMS;
   IF STREAM.FAIL_ENV(JS) = IE THEN DO;
    ENVUNITS = ENVUNITS + STREAM.UNITS(JS);
    ENVMTTR = ENVMTTR + STREAM.UNITS(JS) * SMTTR(CTIME,JS);
    ENVPFAIL = ENVPFAIL + STREAM.UNITS(JS) * MTPROB(JS,1);
    END;
   END;
   ENVQTY(CTIME,IE) = ENVUNITS;
   EMTTR(CTIME,IE) = ENVMTTR/ENVUNITS;
   EPFAIL(CTIME,IE) = ENVPFAIL/ENVUNITS;
   EPFAIL1(CTIME,IE) = EMTTR(CTIME,IE)/LTP * EPFAIL(CTIME,IE);

   /*** BINOMIAL PROBABILITY CALCULATIONS ***/

   N = ENVQTY(CTIME,IE);
   K = ENVQTY(CTIME,IE) - ENV_DATA.DEMAND(IE);
   P = EPFAIL1(CTIME,IE);
   Q = 1 - P;

   SUMFX = 0;
   DO X = 0 TO K;
    IF X       = 1 THEN DO;
    FAC1 = 1;
    DO J = 1 TO X;
     FAC1 = FAC1 * (N-X+J) / J;
     ON OVERFLOW FAC1 = 0;
     END;
    PX = P**X;
     ON UNDERFLOW PX = 0;
    QNLX = Q**(N-X);
    FX = FAC1*PX*QNLX;
    END;
    ELSE FX = Q**N;
    SUMFX = SUMFX + FX;
    END;
  ENVAVAIL(CTIME,IE) = SUMFX;
  END;

/** DETERMINE PERIOD PROCUREMENT SIZE **/
IF CTIME = NXTPROC THEN DO;
 PROCNUM = PROCNUM + 1;
 NXTPROC = CTIME + PROC_RC.PD_BETW_PROC;
 SELECT (PROC_RC.POLICY_NUM);

  WHEN (1) PROCSIZE = PROC_RC.ORDER_SIZE;

  WHEN (2) DO;
   IF POPSIZE = PROC_RC.UP_POP_LIMIT THEN ULSWITCH = 1;
   IF PROC_RC.LP_POP_LIMIT THEN ULSWITCH = 1;
   IF POPSIZE = PROC_RC.UP_POP_LIMIT THEN PROCSIZE = 0;
```

```
   IF POPSIZE < PROC_RC.UP_POP_LIMIT THEN DO;
    IF ULSWITCH < 1 THEN PROCSIZE = PROC_RC.ORDER_SIZE;
    ELSE PROCSIZE = 0;
    END;
   IF PROCSIZE = 0 THEN PROCNUM = PROCNUM - 1;
   END;

  WHEN (3) DO;
   PROCSIZE = 0;
   PROCNUM = PROCNUM - 1;
   END;

  WHEN (4) PROCSIZE = PROC_RC.INIT_PROC - POPSIZE;

  WHEN (5) DO;
   PROCAMT = PROC_RC.INIT_PROC * (1+PROC_RC.PCT_INC/100)**PROCNUM;
   PROCSIZE = ROUND(PROCAMT,0);
   END;

  WHEN (6) DO;
   IF POPSIZE = PROC_RC.UP_POP_LIMIT THEN ULSWITCH = 1;
   IF PROC_RC.INIT_PROC * (1+PROC_RC.PCT_INC/100)**PROCNUM > PROC_RC.UP_POP_LIMIT THEN ULSWITCH = 1;
   IF POPSIZE = PROC_RC.UP_POP_LIMIT THEN PROCSIZE = 0;
   IF POPSIZE < PROC_RC.UP_POP_LIMIT THEN DO;
    PROCAMT = PROC_RC.INIT_PROC * (1+PROC_RC.PCT_INC/100)**PROCNUM;
    PROCSIZE = ROUND(PROCAMT,0);
    IF ULSWITCH = 1 THEN PROCSIZE = 0;
    END;
   IF PROCSIZE = 0 THEN PROCNUM = PROCNUM - 1;
   END;

  WHEN (7) DO;
   PROCAMT = PROC_RC.INIT_PROC * (1-PROC_RC.PCT_DEC/100)**PROCNUM;
   PROCSIZE = ROUND(PROCAMT,0);
   END;

  WHEN (8) DO;
   IF POPSIZE = PROC_RC.UP_POP_LIMIT THEN ULSWITCH = 1;
   IF PROC_RC.INIT_PROC * (1-PROC_RC.PCT_DEC/100)**PROCNUM > PROC_RC.UP_POP_LIMIT THEN ULSWITCH = 1;
   IF POPSIZE = PROC_RC.UP_POP_LIMIT THEN PROCSIZE = 0;
   IF POPSIZE < PROC_RC.UP_POP_LIMIT THEN DO;
    PROCAMT = PROC_RC.INIT_PROC * (1-PROC_RC.PCT_DEC/100)**PROCNUM;
    PROCSIZE = ROUND(PROCAMT,0);
    IF ULSWITCH = 1 THEN PROCSIZE = 0;
    END;
   IF PROCSIZE = 0 THEN PROCNUM = PROCNUM - 1;
   END;

  WHEN (9) PROCSIZE = 0;
 END;
```

```
 IF PROCSIZE    0 THEN DO;
  RDEPLOY = PROCSIZE;
  PROCQTY(CTIME) = PROCSIZE;
  DO I = 1 TO PARM_RC.NUM_ENV-1;
   SDEPLOY = ENV_DATA.NEW_UNITS(I) * PROCSIZE / PROC_RC.INIT_PROC;
   SDEPLOY1 = ROUND(SDEPLOY,0);
   IF SDEPLOY1  0 THEN DO;
    NUM_STREAMS = NUM_STREAMS + 1;
    STREAM.CDATE(NUM_STREAMS) = CTIME;
    STREAM.FAIL_ENV(NUM_STREAMS) = I;
    STREAM.UNITS(NUM_STREAMS) = SDEPLOY1;
    STREAM.INIT_FIELD(NUM_STREAMS) = SDEPLOY1;
    POPSIZE = POPSIZE + SDEPLOY1;
    RDEPLOY = RDEPLOY - SDEPLOY1;
    IF CTIME+1 <= PARM_RC.RUN_LENGTH THEN DO;
     COST_RPT.CAP_COST(CTIME+1,I) = COST_RPT.CAP_COST(CTIME+1,I)
     + SDEPLOY1*COST_RC.FIRST_COST;
     COST_RPT.PUR_COST(CTIME+1,I) = COST_RPT.PUR_COST(CTIME+1,I)
     + COST_RC.PROC_COST/PROCSIZE * SDEPLOY1;
     END;
    END;
   END;
   IF RDEPLOY   0 THEN DO;
    NUM_STREAMS = NUM_STREAMS + 1;
    STREAM.CDATE(NUM_STREAMS) = CTIME;
    STREAM.FAIL_ENV(NUM_STREAMS) = PARM_RC.NUM_ENV;
    STREAM.UNITS(NUM_STREAMS) = RDEPLOY;
    STREAM.INIT_FIELD(NUM_STREAMS) = RDEPLOY;
    POPSIZE = POPSIZE + RDEPLOY;
    I = PARM_RC.NUM_ENV;
    IF CTIME+1 <= PARM_RC.RUN_LENGTH THEN DO;
     COST_RPT.CAP_COST(CTIME+1,I) = COST_RPT.CAP_COST(CTIME+1,I)
     + RDEPLOY*COST_RC.FIRST_COST;
     COST_RPT.PUR_COST(CTIME+1,I) = COST_RPT.PUR_COST(CTIME+1,I)
     + COST_RC.PROC_COST/PROCSIZE * RDEPLOY;
     END;
    END;
   END;
 END;

/*** RETIRE ITEMS AS NECESSARY ***/
 DO JS = 1 TO NUM_STREAMS;
  AGE = CTIME - STREAM.CDATE(JS);
  IE = STREAM.FAIL_ENV(JS);
  IF AGE       = ENV_DATA.RETIRE_AGE(IE) THEN DO;
    COST_RPT.CAP_COST(CTIME,IE) = COST_RPT.CAP_COST(CTIME,IE) -
      ENV_DATA.SV_RETIRE(IE) * STREAM.UNITS(JS);
    POPSIZE = POPSIZE - STREAM.UNITS(JS);
    STREAM.UNITS(JS) = 0;
```

```
    END;
   END;

  /** ADVANCE SIMULATION CLOCK **/
  CTIME = CTIME + 1;
  NXTPD: END;

  /*** RETIRE END ITEMS AT END OF THE SIMULATION ***/

  DO IE = 1 TO PARM_RC.NUM_ENV;
   FAC2 = ENV_DATA.SV_RETIRE(IE)/COST_RC.FIRST_COST;
   FAC3 = LOG(FAC2)/ENV_DATA.RETIRE_AGE(IE);
   DF(IE) = EXP(FAC3);
   END;

  DO JS = 1 TO NUM_STREAMS;
  IF STREAM.CDATE(JS) < PARM_RC.RUN_LENGTH THEN DO;
  AGE = PARM_RC.RUN_LENGTH - STREAM.CDATE(JS);
  IE = STREAM.FAIL_ENV(JS);
  SALVAGE = COST_RC.FIRST_COST * DF(IE)**AGE;
  COST_RPT.CAP_COST(PARM_RC.RUN_LENGTH,IE) =
  COST_RPT.CAP_COST(PARM_RC.RUN_LENGTH,IE)-SALVAGE*STREAM.UNITS(JS);
  END;
  END;

  /*** CALCULATE POPULATION AVAILABILITY ***/

  DO IP = 1 TO PARM_RC.RUN_LENGTH;
   POPAVAIL = 0;
   POPUNITS = 0;
   DO IE = 1 TO PARM_RC.NUM_ENV;
    POPAVAIL = POPAVAIL + ENVAVAIL(IP,IE) * ENVQTY(IP,IE);
    POPUNITS = POPUNITS + ENVQTY(IP,IE);
    END;
   PAVAIL(IP) = POPAVAIL/POPUNITS;
  END;

 /****** LOT SIZE CALCULATIONS ******/

 SELECT (PARM_RC.LOT_SIZE);

  WHEN (1) DO;
   /*** LOT FOR LOT ***/

    DO IP = 1 TO PARM_RC.NUM_STK_RC;
    PART_DMD.BEG_INV(1,IP) = 0;
    DO I = 1 TO PART_DATA.LEAD_TIME(IP);
     PART_DMD.BEG_INV(1,IP) = PART_DMD.BEG_INV(1,IP)
                            + PART_DMD.PD_DMD(I,IP);
     END;
```

```
    END;

   DO I = 1 TO PARM_RC.RUN_LENGTH;
    INVCOST = 0;
    DO IP = 1 TO PARM_RC.NUM_STK_RC;
     PART_DMD.END_INV(I,IP) = PART_DMD.BEG_INV(I,IP)
                            - PART_DMD.PD_DMD(I,IP);
     ILT = I + PART_DATA.LEAD_TIME(IP);
     IF ILT <= PARM_RC.RUN_LENGTH THEN
       PART_DMD.LOT_SIZE(I,IP) = PART_DMD.PD_DMD(ILT,IP);
      ELSE PART_DMD.LOT_SIZE(I,IP) = 0;
     PART_DMD.BEG_INV(ILT,IP) = PART_DMD.LOT_SIZE(I,IP);
     PART_DMD.BEG_INV(I+1,IP) = PART_DMD.BEG_INV(I+1,IP)
                              + PART_DMD.END_INV(I,IP);
     INVCOST=INVCOST+PART_DMD.PD_DMD(I,IP)*PART_DATA.UNIT_PRICE(IP);
    TVM = COST_RC.INTEREST/100;
    HCOST1 = PART_DATA.UNIT_PRICE(IS) * TVM;
    HCOST2 = COST_RC.WHSE_COST/PART_DATA.WH_SPACE(IP);
    HCOST = HCOST1 + HCOST2;
    INVCOST = INVCOST + PART_DMD.PD_DMD(I,IP)/2*HCOST;
     END;

    INVCOST = INVCOST + COST_RC.ORDER_COST;
    EQTY = 0;
    DO IE = 1 TO PARM_RC.NUM_ENV;
     EQTY = EQTY + ENVQTY(I,IE);
     END;
    DO IE = 1 TO PARM_RC.NUM_ENV;
     COST_RPT.INV_COST(I,IE) = COST_RPT.INV_COST(I,IE)
                             + INVCOST*ENVQTY(I,IE)/EQTY;
     END;
    END;
   END;  /** END LOT FOR LOT **/

  WHEN (2) DO;
   /*** ECONOMIC ORDER QUANTITY ***/

   DO IP = 1 TO NUM_STK_RC;
    TPDMD = 0;
    DO I = 1 TO PARM_RC.RUN_LENGTH;
     TPDMD = TPDMD + PART_DMD.PD_DMD(I,IP);
     END;
    AVDMD = TPDMD/PARM_RC.RUN_LENGTH;
    TVM = COST_RC.INTEREST/100;
    HCOST1 = PART_DATA.UNIT_PRICE(IP) * TVM;
    HCOST2 = COST_RC.WHSE_COST/PART_DATA.WH_SPACE(IP);
    HCOST = HCOST1 + HCOST2;
    EOQ = SQRT(2*AVDMD*COST_RC.ORDER_COST/HCOST) + 0.5;

    PART_DMD.BEG_INV(1,IP) = 0;
```

```
DO I = 1 TO PART_DATA.LEAD_TIME(IP);
 PART_DMD.BEG_INV(1,IP) = PART_DMD.BEG_INV(1,IP)
                        + PART_DMD.PD_DMD(I,IP);
 END;

INVPOS = PART_DMD.BEG_INV(1,IP);
DO I = 1 TO PARM_RC.RUN_LENGTH-PART_DATA.LEAD_TIME(IP);
 PART_DMD.END_INV(I,IP) = PART_DMD.BEG_INV(I,IP)
                        - PART_DMD.PD_DMD(I,IP);
 INVPOS = INVPOS - PART_DMD.PD_DMD(I,IP);

 LTREQMTS = 0;
 DO I1 = I+1 TO I+PART_DATA.LEAD_TIME(IP);
  LTREQMTS = LTREQMTS + PART_DMD.PD_DMD(I1,IP);
  END;
 IF LTREQMTS INVPOS THEN DO;
  PART_DMD.LOT_SIZE(I,IP) = ROUND(EOQ,0);
  IF PART_DMD.LOT_SIZE(I,IP) < LTREQMTS-INVPOS THEN
     PART_DMD.LOT_SIZE(I,IP) = LTREQMTS-INVPOS;
  INVPOS = INVPOS + PART_DMD.LOT_SIZE(I,IP);
  ORD_ARRIVAL = I + PART_DATA.LEAD_TIME(IP);
  PART_DMD.BEG_INV(ORD_ARRIVAL,IP) = PART_DMD.LOT_SIZE(I,IP);
  END;

 PART_DMD.BEG_INV(I+1,IP) = PART_DMD.BEG_INV(I+1,IP)
                          + PART_DMD.END_INV(I,IP);
 END;
END;

DO JP = 1 TO PARM_RC.RUN_LENGTH;
 INVCOST = 0;
 DO J = I TO PARM_RC.NUM_STK_RC;
  TVM = COST_RC.INTEREST/100;
  HCOST1 = PART_DATA.UNIT_PRICE(J) * TVM;
  HCOST2 = COST_RC.WHSE_COST/PART_DATA.WH_SPACE(J);
  HCOST = HCOST1 + HCOST2;
  INVCOST = PART_DMD.PD_DMD(JP,J) * PART_DATA.UNIT_PRICE(J)
   + (PART_DMD.BEG_INV(JP,J)+PART_DMD.END_INV(JP,J))/2*HCOST
   +INVCOST;
  IF PART_DMDOLUMENIZE(JP,J)
    INVCOST = INVCOST + COST_RC.ORDER_COST;
  END;
 EQTY = 0;
 DO IE = 1 TO PARM_RC.NUM_ENV;
  EQTY = EQTY + ENVQTY(JP,IE);
  END;
 DO IE = 1 TO PARM_RC.NUM_ENV;
  COST_RPT.INV_COST(JP,IE)=INVCOST*ENVQTY(JP,IE)/EQTY;
  END;
 END;
```

```
END;  /** END ECONOMIC ORDER QUANTITY **/

WHEN (3) DO;
 /*** PERIOD ORDER QUANTITY ***/
 DO IP = 1 TO NUM_STK_RC;
  TPDMD = 0;
  DO I = 1 TO PARM_RC.RUN_LENGTH;
   TPDMD = TPDMD + PART_DMD.PD_DMD(I,IP);
   END;
  AVDMD = TPDMD/PARM_RC.RUN_LENGTH;
  TVM = COST_RC.INTEREST/100;
  HCOST1 = PART_DATA.UNIT_PRICE(IP) * TVM;
  HCOST2 = COST_RC.WHSE_COST/PART_DATA.WH_SPACE(IP);
  HCOST = HCOST1 + HCOST2;
  EOQ = SQRT(2*AVDMD*COST_RC.ORDER_COST/HCOST) + 0.5;
  IF AVDMD < 1 THEN AVDMD = 1;
  PQTY = EOQ/AVDMD + 0.5;
  POQ = ROUND(PQTY,0);
  IF POQ < 1 THEN POQ = 1;

  PART_DMD.BEG_INV(1,IP) = 0;
  DO I = 1 TO PART_DATA.LEAD_TIME(IP);
   PART_DMD.BEG_INV(1,IP) = PART_DMD.BEG_INV(1,IP)
                          + PART_DMD.PD_DMD(I,IP);
   END;

  INVPOS = PART_DMD.BEG_INV(1,IP);
  DO I = 1 TO PARM_RC.RUN_LENGTH-PART_DATA.LEAD_TIME(IP);
   PART_DMD.END_INV(I,IP) = PART_DMD.BEG_INV(I,IP)
                          - PART_DMD.PD_DMD(I,IP);
   INVPOS = INVPOS - PART_DMD.PD_DMD(I,IP);

   LTREQMTS = 0;
   DO I1 = I+1 TO I+PART_DATA.LEAD_TIME(IP);
    LTREQMTS = LTREQMTS + PART_DMD.PD_DMD(I1,IP);
    END;

   IF LTREQMTS INVPOS THEN DO;
    IL1 = I+PART_DATA.LEAD_TIME(IP);
    IL2 = IL1 + POQ - 1;
    IF IL2     PARM_RC.RUN_LENGTH THEN IL2 = PARM_RC.RUN_LENGTH;
    PART_DMD.LOT_SIZE(I,IP) = 0;
    DO IL = IL1 TO IL2;
     PART_DMD.LOT_SIZE(I,IP) = PART_DMD.LOT_SIZE(I,IP)
                             + PART_DMD.PD_DMD(IL,IP);
     END;
    INVPOS = INVPOS + PART_DMD.LOT_SIZE(I,IP);
    ORD_ARRIVAL = I + PART_DATA.LEAD_TIME(IP);
    PART_DMD.BEG_INV(ORD_ARRIVAL,IP) = PART_DMD.LOT_SIZE(I,IP);
    END;
```

```
    PART_DMD.BEG_INV(I+1,IP) = PART_DMD.BEG_INV(I+1,IP)
                             + PART_DMD.END_INV(I,IP);
    END;
   END;

   DO JP = 1 TO PARM_RC.RUN_LENGTH;
    INVCOST = 0;
    DO J = I TO PARM_RC.NUM_STK_RC;
     TVM = COST_RC.INTEREST/100;
     HCOST1 = PART_DATA.UNIT_PRICE(J) * TVM;
     HCOST2 = COST_RC.WHSE_COST/PART_DATA.WH_SPACE(J);
     HCOST = HCOST1 + HCOST2;
     INVCOST = PART_DMD.PD_DMD(JP,J) * PART_DATA.UNIT_PRICE(J)
      + (PART_DMD.BEG_INV(JP,J)+PART_DMD.END_INV(JP,J))/2*HCOST
      +INVCOST;
     IF PART_DMDOLUMENIZE(JP,J)
       INVCOST = INVCOST + COST_RC.ORDER_COST;
     END;
    EQTY = 0;
    DO IE = 1 TO PARM_RC.NUM_ENV;
     EQTY = EQTY + ENVQTY(JP,IE);
     END;
    DO IE = 1 TO PARM_RC.NUM_ENV;
     COST_RPT.INV_COST(JP,IE)=INVCOST*ENVQTY(JP,IE)/EQTY;
     END;
    END;
  END; /** END PERIOD ORDER QUANTITY **/

 WHEN (4) DO;
 /*** SILVER MEAL HEURISTIC ***/

  DO IP = 1 TO NUM_STK_RC;

   TVM = COST_RC.INTEREST/100;
   HCOST1 = PART_DATA.UNIT_PRICE(IP) * TVM;
   HCOST2 = COST_RC.WHSE_COST/PART_DATA.WH_SPACE(IP);
   HCOST = HCOST1 + HCOST2;

   PART_DMD.BEG_INV(1,IP) = 0;
   DO I = 1 TO PART_DATA.LEAD_TIME(IP);
    PART_DMD.BEG_INV(1,IP) = PART_DMD.BEG_INV(1,IP)
                           + PART_DMD.PD_DMD(I,IP);
    END;

   INVPOS = PART_DMD.BEG_INV(1,IP);
   DO I = 1 TO PARM_RC.RUN_LENGTH-PART_DATA.LEAD_TIME(IP);
    PART_DMD.END_INV(I,IP) = PART_DMD.BEG_INV(I,IP)
                           - PART_DMD.PD_DMD(I,IP);
    INVPOS = INVPOS - PART_DMD.PD_DMD(I,IP);
```

```
 LTREQMTS = 0;
 DO I1 = I+1 TO I+PART_DATA.LEAD_TIME(IP);
  LTREQMTS = LTREQMTS + PART_DMD.PD_DMD(I1,IP);
  END;

 IF LTREQMTS INVPOS THEN DO;
  IL1 = I+PART_DATA.LEAD_TIME(IP);
  IHCOST = 0;
  OQTY = 1;
  DO IL = IL1 TO PARM_RC.RUN_LENGTH;
   IHCOST = IHCOST + PART_DMD.PD_DMD(IL,IP)*(IL-IL1)*HCOST;
   ACPTU = (COST_RC.ORDER_COST + IHCOST)/(IL-IL1+1);
   IF IL = IL1 THEN MCPTU = ACPTU;
   IF ACPTU < MCPTU THEN DO;
    MCPTU = ACPTU;
    OQTY = OQTY + 1;
    END;
   END;

  IL2 = I + PART_DATA.LEAD_TIME(IP) + OQTY;
  IF IL2     PARM_RC.RUN_LENGTH THEN IL2 = PARM_RC.RUN_LENGTH;
  PART_DMD.LOT_SIZE(I,IP) = 0;
  DO IL = IL1 TO IL2;
   PART_DMD.LOT_SIZE(I,IP) = PART_DMD.LOT_SIZE(I,IP)
                           + PART_DMD.PD_DMD(IL,IP);
   END;
  INVPOS = INVPOS + PART_DMD.LOT_SIZE(I,IP);
  ORD_ARRIVAL = I + PART_DATA.LEAD_TIME(IP);
  PART_DMD.BEG_INV(ORD_ARRIVAL,IP) = PART_DMD.LOT_SIZE(I,IP);
  END;

 PART_DMD.BEG_INV(I+1,IP) = PART_DMD.BEG_INV(I+1,IP)
                          + PART_DMD.END_INV(I,IP);
 END;
END;

DO JP = 1 TO PARM_RC.RUN_LENGTH;
 INVCOST = 0;
 DO J = I TO PARM_RC.NUM_STK_RC;
  TVM = COST_RC.INTEREST/100;
  HCOST1 = PART_DATA.UNIT_PRICE(J) * TVM;
  HCOST2 = COST_RC.WHSE_COST/PART_DATA.WH_SPACE(J);
  HCOST = HCOST1 + HCOST2;
  INVCOST = PART_DMD.PD_DMD(JP,J) * PART_DATA.UNIT_PRICE(J)
   + (PART_DMD.BEG_INV(JP,J)+PART_DMD.END_INV(JP,J))/2*HCOST
   +INVCOST;
  IF PART_DMD.LOT_SIZE(JP,J)
    INVCOST = INVCOST + COST_RC.ORDER_COST;
  END;
```

```
      EQTY = 0;
      DO IE = 1 TO PARM_RC.NUM_ENV;
       EQTY = EQTY + ENVQTY(JP,IE);
       END;
      DO IE = 1 TO PARM_RC.NUM_ENV;
       COST_RPT.INV_COST(JP,IE)=INVCOST*ENVQTY(JP,IE)/EQTY;
       END;
      END;
     END; /** END SILVER MEAL HEURISTIC **/

   END; /** END INVENTORY REVIEW OPTIONS **/

   /*** CALCULATE PERIOD SPACE REQUIREMENTS ***/
   DO IP = 1 TO PARM_RC.RUN_LENGTH;
    DO IS = 1 TO NUM_STK_RC;
     PLOR = 3;
     DO J1 = 1 TO PARM_RC.NUM_NODES;
      IF PART_DATA.NODE_NUM(IS) = NODE_DATA.NODE_NUM(1,J1)
         THEN PLOR = NODE_DATA.LOR(1,J1);
      END;
     AVEINV = (PART_DMD.BEG_INV(IP,IS)+PART_DMD.END_INV(IP,IS))/2;
     SPACE1 = AVEINV/PART_DATA.WH_SPACE(IS);
     SPACE(IP,PLOR) = SPACE(IP,PLOR) + CEIL(SPACE1);
    END;
   END;

   /*** CALCULATE PERIOD LABOR REQUIREMENTS ***/
   DO IP = 1 TO PARM_RC.RUN_LENGTH;
    DO IN = 1 TO PARM_RC.NUM_NODES;
     LABOR1 = EFAIL(IP,IN)*NODE_DATA.MAN_HRS(1,IN);
     PLOR = NODE_DATA.LOR(1,IN);
     LABOR(IP,PLOR) = LABOR(IP,PLOR) + CEIL(LABOR1);
    END;
   END;

   END SIMULATE;

/*************************************************************************/
/******************** PERIOD DETAIL REPORT SCREENS ********************/
/*************************************************************************/

PD_REPORTS_SCREEN: PROCEDURE;

 OUTSC = 1;
 MORE_OUTSC = YES;

 DO WHILE (MORE_OUTSC = YES);
   SELECT;
    WHEN(OUTSC = 1) CALL OUTSC_1;
    WHEN(OUTSC = 2) CALL OUTSC_2;
```

```
    WHEN(OUTSC = 3) CALL OUTSC_3;
    WHEN(OUTSC = 4) CALL OUTSC_4;
    WHEN(OUTSC = 5) CALL OUTSC_5;
    WHEN(OUTSC = 6) CALL OUTSC_6;
    OTHERWISE CALL OUTSC_1;
    END;
  END;
  RETURN;

/*****************************************/
/* PERIOD DETAIL REPORT - NODE P(FAIL) */
/*****************************************/

OUTSC_1: PROCEDURE;

 DECLARE
 1 OS1A_LOADLIST,
  2 LOAD(33)          PTR,
  2 FENCE             BIT(32) INIT((32)'1'B);

 PERNO = 1;
 DO WHILE (PERNO <= PARM_RC.RUN_LENGTH);

  STRNO = 1;
  DO WHILE (STRNO <= NUM_STREAMS);

  DO I = 1 TO 26;
   PFAIL_RPT.FPROB(I) = NFAILRPT(STRNO,PERNO,I);
   END;

   PFAIL_RPT.INIT_FIELD = STREAM.INIT_FIELD(STRNO);
   PFAIL_RPT.BEG_UNITS = STRQTY.BEG_PD(PERNO,STRNO);
   PFAIL_RPT.END_UNITS = STRQTY.END_PD(PERNO,STRNO);
   PFAIL_RPT.UNITS_PROC = PROCQTY(PERNO);
   PFAIL_RPT.MTTR = SMTTR(PERNO,STRNO);

   OS1A_LOADLIST.LOAD(1) = ADDR(STRNO);
   OS1A_LOADLIST.LOAD(2) = ADDR(PERNO);
   OS1A_LOADLIST.LOAD(3) = ADDR(PFAIL_RPT.INIT_FIELD);
   OS1A_LOADLIST.LOAD(4) = ADDR(PFAIL_RPT.BEG_UNITS);
   OS1A_LOADLIST.LOAD(5) = ADDR(PFAIL_RPT.UNITS_PROC);
   OS1A_LOADLIST.LOAD(6) = ADDR(PFAIL_RPT.END_UNITS);
   OS1A_LOADLIST.LOAD(7) = ADDR(PFAIL_RPT.MTTR);
   OS1A_LOADLIST.LOAD(8) = ADDR(PFAIL_RPT.FPROB(1));
   OS1A_LOADLIST.LOAD(9) = ADDR(PFAIL_RPT.FPROB(6));

   DO INP = 1 TO 4;
    OS1A_LOADLIST.LOAD(10+(INP-1)) = ADDR(PFAIL_RPT.FPROB(INP+1));
    END;
```

```
DO INP = 1 TO 5;
OS1A_LOADLIST.LOAD(14+4*(INP-1)) = ADDR(PFAIL_RPT.FPROB(7+(INP-1)));
OS1A_LOADLIST.LOAD(15+4*(INP-1)) = ADDR(PFAIL_RPT.FPROB(12+(INP-1)));
OS1A_LOADLIST.LOAD(16+4*(INP-1)) = ADDR(PFAIL_RPT.FPROB(17+(INP-1)));
OS1A_LOADLIST.LOAD(17+4*(INP-1)) = ADDR(PFAIL_RPT.FPROB(22+(INP-1)));
END;

DOSC1:
 PDISPLAY PNAME('PRPTFAIL') RSTATUS(RSTATUS) LDLIST(OS1A_LOADLIST)
          ULDLIST(OS1A_LOADLIST) DCURSOR(CURSOR);

 SELECT (RSTATUS);
  WHEN(PF7 ) DO;
   IF PERNO     1 THEN PERNO = PERNO - 1;
    ELSE PERNO = PARM_RC.RUN_LENGTH;
   END;
  WHEN(PF8 ) DO;
   IF PERNO < PARM_RC.RUN_LENGTH THEN PERNO = PERNO + 1;
    ELSE PERNO = 1;
   END;
  WHEN(PF9 ) DO;
   IF STRNO < NUM_STREAMS THEN STRNO = STRNO + 1;
    ELSE STRNO = 1;
   END;
  OTHERWISE DO;
    SELECT(RSTATUS);
      WHEN(PF1 ) OUTSC = 1;
      WHEN(PF2 ) OUTSC = 2;
      WHEN(PF3 ) OUTSC = 3;
      WHEN(PF4 ) OUTSC = 4;
      WHEN(PF5 ) OUTSC = 5;
      WHEN(PF6 ) OUTSC = 6;
      WHEN(PF10) DO;
        SCNUM = SCNUM + 1;
        MORE_OUTSC = NO;
        END;
      WHEN(PF11) DO;
        SCNUM = SCNUM - 1;
        MORE_OUTSC = NO;
        END;
      WHEN(PF12) DO;
        SCNUM = 1;
        MORE_OUTSC = NO;
        END;
      OTHERWISE OUTSC = 1;
      END;
   PERNO = PARM_RC.RUN_LENGTH + 1;
   STRNO = NUM_STREAMS + 1;
   END;
 END;
```

```
  NEXTSTR: END;
 NEXTPD1: END;
END OUTSC_1;

/*************************************************************/
/* PERIOD DETAIL REPORT - SPARE PARTS REQUIREMENTS */
/*************************************************************/

 OUTSC_2: PROCEDURE;

 DECLARE
 1 OS2_LOADLIST,
  2 LOAD(85)          PTR,
  2 FENCE             BIT(32) INIT((32)'1'B);

 IS = 1;
 XNG = PARM_RC.NUM_STK_RC/12;
 NG = CEIL(XNG);

 PERNO = 1;
 JMAX = 12;
 DO WHILE (PERNO <= PARM_RC.RUN_LENGTH);
  DO WHILE (IS <= NG);
   OS2_LOADLIST.LOAD(1) = ADDR(PERNO);
   DO I = (IS-1)*12+1 TO IS*12;
   J = I - (IS-1)*12;
   OS2_LOADLIST.LOAD((J-1)*7+2) = ADDR(PART_DATA.NODE_NUM(I));
   OS2_LOADLIST.LOAD((J-1)*7+3) = ADDR(PART_DATA.PART_DESC(I));
   OS2_LOADLIST.LOAD((J-1)*7+4) = ADDR(PART_DMD.PD_DMD(PERNO,I));
   OS2_LOADLIST.LOAD((J-1)*7+5) = ADDR(PART_DMD.BEG_INV(PERNO,I));
   OS2_LOADLIST.LOAD((J-1)*7+6) = ADDR(PART_DMD.END_INV(PERNO,I));
   OS2_LOADLIST.LOAD((J-1)*7+7) = ADDR(PART_DMD.LOT_SIZE(PERNO,I));
   PLOR = 3;
   DO J1 = 1 TO PARM_RC.NUM_NODES;
    IF PART_DATA.NODE_NUM(I) = NODE_DATA.NODE_NUM(1,J1) THEN
       PLOR = NODE_DATA.LOR(1,J1);
    END;
   OS2_LOADLIST.LOAD((J-1)*7+8) = ADDR(PLOR);
   END;

  DOSC2:
  PDISPLAY PNAME('PRPTPART') RSTATUS(RSTATUS) LDLIST(OS2_LOADLIST)
           ULDLIST(OS2_LOADLIST) DCURSOR(CURSOR);

  SELECT(RSTATUS);
   WHEN(PF7 ) DO;
    IF IS        1 THEN IS = IS - 1;
    ELSE DO;
     IS = 1;
```

```
      IF PERNO     1 THEN PERNO = PERNO - 1;
      END;
     END;

    WHEN(PF8 ) DO;
     IF IS < NG THEN IS = IS + 1;
     ELSE DO;
      IS = 1;
      IF PERNO < PARM_RC.RUN_LENGTH THEN PERNO = PERNO + 1;
      END;
     END;

    OTHERWISE DO;
      SELECT(RSTATUS);
        WHEN(PF1 ) OUTSC = 1;
        WHEN(PF2 ) OUTSC = 2;
        WHEN(PF3 ) OUTSC = 3;
        WHEN(PF4 ) OUTSC = 4;
        WHEN(PF5 ) OUTSC = 5;
        WHEN(PF6 ) OUTSC = 6;
        WHEN(PF10) DO;
          SCNUM = SCNUM + 1;
          MORE_OUTSC = NO;
          END;
        WHEN(PF11) DO;
          SCNUM = SCNUM - 1;
          MORE_OUTSC = NO;
          END;
        WHEN(PF12) DO;
          SCNUM = 1;
          MORE_OUTSC = NO;
          END;
        OTHERWISE OUTSC = 2;
        END;
      IS = NG + 1;
      PERNO = PARM_RC.RUN_LENGTH + 1;
      END;
     END;
    END;
   END;
   PRELEASE;

   END OUTSC_2;

  /*********************************************/
  /* PERIOD DETAIL REPORT - AVAILABILITY */
  /*********************************************/

  OUTSC_3: PROCEDURE;
```

```
DECLARE
1 OS4_LOADLIST,
 2 LOAD(10)          PTR,
 2 FENCE             BIT(32) INIT((32)'1'B);

 PERNO = 1;
 DO WHILE (PERNO <= PARM_RC.RUN_LENGTH);
 OS4_LOADLIST.LOAD(1) = ADDR(PERNO);
 DO IE = 1 TO PARM_RC.NUM_ENV;
  AVAIL_RPT.ENV_DESC(IE) = ENV_DATA.ENV_DESC(IE);
  AVAIL_RPT.ENV_AVAIL(IE) = ENVAVAIL(PERNO,IE) * 100;
  END;

 DO IENV = 1 TO 4;
  IV = (IENV-1)*2;
  OS4_LOADLIST.LOAD(IV+2) = ADDR(AVAIL_RPT.ENV_DESC(IENV));
  OS4_LOADLIST.LOAD(IV+3) = ADDR(AVAIL_RPT.ENV_AVAIL(IENV));
  END;

 SYS_AVAIL = PAVAIL(PERNO) * 100;
 OS4_LOADLIST.LOAD(10) = ADDR(SYS_AVAIL);

 DOSC3:
 PDISPLAY PNAME('PRPTAVAI') RSTATUS(RSTATUS) LDLIST(OS4_LOADLIST)
          ULDLIST(OS4_LOADLIST) DCURSOR(CURSOR);

 SELECT(RSTATUS);

  WHEN(PF7 ) DO;
   IF PERNO     1 THEN PERNO = PERNO - 1;
   ELSE PERNO = 1;
   END;
  WHEN(PF8 ) DO;
  IF PERNO < PARM_RC.RUN_LENGTH THEN PERNO = PERNO + 1;
   ELSE PERNO = 1;
   END;
  WHEN(PF9 ) GO TO DOSC3;
  OTHERWISE DO;
    SELECT(RSTATUS);
      WHEN(PF1 ) OUTSC = 1;
      WHEN(PF2 ) OUTSC = 2;
      WHEN(PF3 ) OUTSC = 3;
      WHEN(PF4 ) OUTSC = 4;
      WHEN(PF5 ) OUTSC = 5;
      WHEN(PF6 ) OUTSC = 6;
      WHEN(PF10) DO;
        SCNUM = SCNUM + 1;
        MORE_OUTSC = NO;
        END;
      WHEN(PF11) DO;
```

```
          SCNUM = SCNUM - 1;
          MORE_OUTSC = NO;
          END;
        WHEN(PF12) DO;
          SCNUM = 1;
          MORE_OUTSC = NO;
          END;
        OTHERWISE OUTSC = 3;
        END;
      PERNO = PARM_RC.RUN_LENGTH + 1;
      END;
    END;
   NEXTPD3: END;
   PRELEASE;
   END OUTSC_3;

/*********************************************************/
/* PERIOD DETAIL REPORT - WAREHOUSE REQUIREMENTS */
/*********************************************************/

 OUTSC_4: PROCEDURE;

 DECLARE
 1 OS41_LOADLIST,
  2 LOAD(20)            PTR,
  2 FENCE               BIT(32)       INIT((32)'1'B);

 DO IE = 1 TO PARM_RC.NUM_ENV;
  WHSE_RPT.ENV_DESC(IE) = ENV_DATA.ENV_DESC(IE);
  END;

 PERNO = 1;
 DO WHILE (PERNO <= PARM_RC.RUN_LENGTH);

   OS41_LOADLIST.LOAD(1) = ADDR(PERNO);

   EQTY = 0;
   DO IE = 1 TO PARM_RC.NUM_ENV;
    EQTY = EQTY + ENVQTY(PERNO,IE);
    END;
   DO IE = 1 TO PARM_RC.NUM_ENV;
   DO IRL = 1 TO PARM_RC.REPAIR_LEVELS;
   WHSE_RPT.REP_LEV(IE,IRL) = ENVQTY(PERNO,IE)/EQTY*SPACE(PERNO,IRL);
   END;
   END;

   DO IRL = 1 TO PARM_RC.REPAIR_LEVELS;
    WHSE_RPT.TOTALS(IRL) = SPACE(PERNO,IRL);
    END;
```

```
 DO IENV = 1 TO 4;
 IENV1 = 2 + (IENV -1)*4;
 OS41_LOADLIST.LOAD(IENV1) = ADDR(WHSE_RPT.ENV_DESC(IENV));

  DO IRL = 1 TO 3;
  IENV2 = IENV1 + IRL;
  OS41_LOADLIST.LOAD(IENV2) = ADDR(WHSE_RPT.REP_LEV(IENV,IRL));
  END;
 END;

 DO IRL = 1 TO 3;
  IENV2 = 17 + IRL;
  OS41_LOADLIST.LOAD(IENV2) = ADDR(WHSE_RPT.TOTALS(IRL));
  END;


DOSC4:
 PDISPLAY PNAME('PRPTWHSE') RSTATUS(RSTATUS) LDLIST(OS41_LOADLIST)
          ULDLIST(OS41_LOADLIST) DCURSOR(CURSOR);

 SELECT (RSTATUS);
  WHEN(PF7 ) DO;
   IF PERNO     1 THEN PERNO = PERNO - 1;
    ELSE PERNO = PARM_RC.RUN_LENGTH;
   END;
  WHEN(PF8 ) DO;
   IF PERNO < PARM_RC.RUN_LENGTH THEN PERNO = PERNO + 1;
    ELSE PERNO = 1;
    END;
  WHEN(PF9 ) GO TO DOSC4;
  OTHERWISE DO;
    SELECT(RSTATUS);
      WHEN(PF1 ) OUTSC = 1;
      WHEN(PF2 ) OUTSC = 2;
      WHEN(PF3 ) OUTSC = 3;
      WHEN(PF4 ) OUTSC = 4;
      WHEN(PF5 ) OUTSC = 5;
      WHEN(PF6 ) OUTSC = 6;
      WHEN(PF10) DO;
        SCNUM = SCNUM + 1;
        MORE_OUTSC = NO;
        END;
      WHEN(PF11) DO;
        SCNUM = SCNUM - 1;
        MORE_OUTSC = NO;
        END;
      WHEN(PF12) DO;
        SCNUM = 1;
        MORE_OUTSC = NO;
        END;
```

```
        OTHERWISE OUTSC = 4;
        END;
      PERNO = PARM_RC.RUN_LENGTH + 1;
      END;
    END;
 NEXTPD4: END;
 END OUTSC_4;

/*************************************************/
/* PERIOD DETAIL REPORT - LABOR REQUIREMENTS */
/*************************************************/

 OUTSC_5: PROCEDURE;

 DECLARE
 1 OS43_LOADLIST,
  2 LOAD(20)            PTR,
  2 FENCE               BIT(32)       INIT((32)'1'B);

 DO IE = 1 TO PARM_RC.NUM_ENV;
  MANPR_RPT.ENV_DESC(IE) = ENV_DATA.ENV_DESC(IE);
  END;

 PERNO = 1;
 DO WHILE (PERNO <= PARM_RC.RUN_LENGTH);

   OS43_LOADLIST.LOAD(1) = ADDR(PERNO);

   EQTY = 0;
   DO IE = 1 TO PARM_RC.NUM_ENV;
    EQTY = EQTY + ENVQTY(PERNO,IE);
    END;
   DO IE = 1 TO PARM_RC.NUM_ENV;
   DO IRL = 1 TO PARM_RC.REPAIR_LEVELS;
  MANPR_RPT.ENV_REQ(IE,IRL) = ENVQTY(PERNO,IE)/EQTY*LABOR(PERNO,IRL);
   END;
   END;

   DO IRL = 1 TO PARM_RC.REPAIR_LEVELS;
    MANPR_RPT.TOTALS(IRL) = LABOR(PERNO,IRL);
    END;

   DO IENV = 1 TO 4;
   IENV1 = 2 + (IENV -1)*4;
   OS43_LOADLIST.LOAD(IENV1) = ADDR(MANPR_RPT.ENV_DESC(IENV));

    DO IRL = 1 TO 3;
    IENV2 = IENV1 + IRL;
    OS43_LOADLIST.LOAD(IENV2) = ADDR(MANPR_RPT.ENV_REQ(IENV,IRL));
    END;
```

```
   END;

   DO IRL = 1 TO 3;
    IENV2 = 17 + IRL;
    OS43_LOADLIST.LOAD(IENV2) = ADDR(MANPR_RPT.TOTALS(IRL));
    END;


 DOSC5:
  PDISPLAY PNAME('PRPTLABR') RSTATUS(RSTATUS) LDLIST(OS43_LOADLIST)
           ULDLIST(OS43_LOADLIST) DCURSOR(CURSOR);

  SELECT (RSTATUS);
   WHEN(PF7 ) DO;
    PERNO = PERNO - 1;
    IF PERNO < 1 THEN DO;
     PERNO = 1;
     END;
    END;
   WHEN(PF8 ) PERNO = PERNO + 1;
   WHEN(PF9 ) GO TO DOSC5;
   OTHERWISE DO;
     SELECT(RSTATUS);
       WHEN(PF1 ) OUTSC = 1;
       WHEN(PF2 ) OUTSC = 2;
       WHEN(PF3 ) OUTSC = 3;
       WHEN(PF4 ) OUTSC = 4;
       WHEN(PF5 ) OUTSC = 5;
       WHEN(PF6 ) OUTSC = 6;
       WHEN(PF10) DO;
         SCNUM = SCNUM + 1;
         MORE_OUTSC = NO;
         END;
       WHEN(PF11) DO;
         SCNUM = SCNUM - 1;
         MORE_OUTSC = NO;
         END;
       WHEN(PF12) DO;
         SCNUM = 1;
         MORE_OUTSC = NO;
         END;
       OTHERWISE OUTSC = 5;
       END;
     PERNO = PARM_RC.RUN_LENGTH + 1;
     END;
   END;
 NEXTPD5: END;
 END OUTSC_5;

/*********************************************/
```

```
/* PERIOD DETAIL REPORT - PERIOD COSTS */
/*****************************************/

 OUTSC_6: PROCEDURE;

 DECLARE
 1 OS5_LOADLIST,
  2 LOAD(41)            PTR,
  2 FENCE               BIT(32)       INIT((32)'1'B);

  DO PERNO = 1 TO PARM_RC.RUN_LENGTH;
   DO IE = 1 TO PARM_RC.NUM_ENV;
    COST_RPT.ENV_DESC(PERNO,IE) = ENV_DATA.ENV_DESC(IE);
    COST_RPT.TOT_COST(PERNO,IE) = COST_RPT.CAP_COST(PERNO,IE)
    + COST_RPT.REP_COST(PERNO,IE) + COST_RPT.PUR_COST(PERNO,IE)
                                  + COST_RPT.INV_COST(PERNO,IE);
    COST_RPT.UNITS_START(PERNO,IE) = 0;
    COST_RPT.UNITS_END(PERNO,IE) = 0;
    DO JS = 1 TO NUM_STREAMS;
     IF STREAM.FAIL_ENV(JS) = IE THEN DO;
     COST_RPT.UNITS_START(PERNO,IE) = COST_RPT.UNITS_START(PERNO,IE)
                                    + STRQTY.BEG_PD(PERNO,JS);
     COST_RPT.UNITS_END(PERNO,IE) = COST_RPT.UNITS_END(PERNO,IE)
                                    + STRQTY.END_PD(PERNO,JS);
     END;
    END;
    COST_RPT.UNIT_COST(PERNO,IE) = (COST_RPT.REP_COST(PERNO,IE)
     + COST_RPT.INV_COST(PERNO,IE))/COST_RPT.UNITS_END(PERNO,IE);
   END;
  END;

 PERNO = 1;
 DO WHILE (PERNO <= PARM_RC.RUN_LENGTH);

   OS5_LOADLIST.LOAD(1) = ADDR(PERNO);

   DO IENV = 1 TO 4;
   IENV1 = 2 + (IENV -1)*6;
   OS5_LOADLIST.LOAD(IENV1) = ADDR(COST_RPT.ENV_DESC(PERNO,IENV));
   IENV1 = IENV1 + 1;
   OS5_LOADLIST.LOAD(IENV1) = ADDR(COST_RPT.CAP_COST(PERNO,IENV));
   IENV1 = IENV1 + 1;
   OS5_LOADLIST.LOAD(IENV1) = ADDR(COST_RPT.REP_COST(PERNO,IENV));
   IENV1 = IENV1 + 1;
   OS5_LOADLIST.LOAD(IENV1) = ADDR(COST_RPT.PUR_COST(PERNO,IENV));
   IENV1 = IENV1 + 1;
   OS5_LOADLIST.LOAD(IENV1) = ADDR(COST_RPT.INV_COST(PERNO,IENV));
   IENV1 = IENV1 + 1;
   OS5_LOADLIST.LOAD(IENV1) = ADDR(COST_RPT.TOT_COST(PERNO,IENV));
   END;
```

```
   DO IENV = 1 TO 4;
   IENV1 = 26 + (IENV -1)*4;
   OS5_LOADLIST.LOAD(IENV1) = ADDR(COST_RPT.ENV_DESC(PERNO,IENV));
   IENV1 = IENV1 + 1;
   OS5_LOADLIST.LOAD(IENV1) = ADDR(COST_RPT.UNITS_START(PERNO,IENV));
   IENV1 = IENV1 + 1;
   OS5_LOADLIST.LOAD(IENV1) = ADDR(COST_RPT.UNITS_END(PERNO,IENV));
   IENV1 = IENV1 + 1;
   OS5_LOADLIST.LOAD(IENV1) = ADDR(COST_RPT.UNIT_COST(PERNO,IENV));
   END;

DOSC6:
 PDISPLAY PNAME('PRPTCOST') RSTATUS(RSTATUS) LDLIST(OS5_LOADLIST)
          ULDLIST(OS5_LOADLIST) DCURSOR(CURSOR);

 SELECT (RSTATUS);
  WHEN(PF7 ) DO;
   PERNO = PERNO - 1;
   IF PERNO < 1 THEN PERNO = PARM_RC.RUN_LENGTH;
   END;
  WHEN(PF8 ) DO;
   PERNO = PERNO + 1;
   IF PERNO     PARM_RC.RUN_LENGTH THEN PERNO = 1;
   END;

  WHEN(PF9 ) GO TO DOSC6;
  OTHERWISE DO;
    SELECT(RSTATUS);
      WHEN(PF1 ) OUTSC = 1;
      WHEN(PF2 ) OUTSC = 2;
      WHEN(PF3 ) OUTSC = 3;
      WHEN(PF4 ) OUTSC = 4;
      WHEN(PF5 ) OUTSC = 5;
      WHEN(PF6 ) OUTSC = 6;
      WHEN(PF10) DO;
        SCNUM = SCNUM + 1;
        MORE_OUTSC = NO;
        END;
      WHEN(PF11) DO;
        SCNUM = SCNUM - 1;
        MORE_OUTSC = NO;
        END;
      WHEN(PF12) DO;
        SCNUM = 1;
        MORE_OUTSC = NO;
        END;
      OTHERWISE OUTSC = 6;
      END;
    PERNO = PARM_RC.RUN_LENGTH + 1;
```

```
     END;
   END;
 NEXTPD6: END;
 END OUTSC_6;

 PRELEASE;
 END PD_REPORTS_SCREEN;

/*****************************************************************************/
/*************** DISPLAY INTRODUCTORY CONCEPTS SCREENS ***************/
/*****************************************************************************/

CONCEPTS_SCREEN: PROCEDURE;

  INTSC = 1;
  MORE_INTSC = YES;

  DO WHILE (MORE_INTSC = YES);
   SELECT;
    WHEN (INTSC = 1) CALL INTSC_1;
    WHEN (INTSC = 2) CALL INTSC_2;
    WHEN (INTSC = 3) CALL INTSC_3;
    WHEN (INTSC = 4) CALL INTSC_4;
    WHEN (INTSC = 5) CALL INTSC_5;
    OTHERWISE DO;
      SCNUM = SCNUM - 100;
      MORE_INTSC = NO;
      END;
   END;
  END;
  RETURN;

  INTSC_1: PROCEDURE;
      PDISPLAY PNAME ('INTROMSC') RSTATUS(RSTATUS);
      SELECT (RSTATUS);
        WHEN (PF8 ) INTSC = INTSC + 1;
        WHEN (PF12) INTSC = INTSC + 100;
        OTHERWISE  INTSC = INTSC + 0;
        END;
       PRELEASE;
       END INTSC_1;

  INTSC_2: PROCEDURE;
      PDISPLAY PNAME ('INTROMTC') RSTATUS(RSTATUS);
      SELECT (RSTATUS);
        WHEN (PF7 ) INTSC = INTSC - 1;
        WHEN (PF8 ) INTSC = INTSC + 1;
        WHEN (PF12) INTSC = INTSC + 100;
        OTHERWISE  INTSC = INTSC + 0;
        END;
```

```
      PRELEASE;
      END INTSC_2;

  INTSC_3: PROCEDURE;
      PDISPLAY PNAME ('INTROMMT') RSTATUS(RSTATUS);
      SELECT (RSTATUS);
        WHEN (PF7 ) INTSC = INTSC - 1;
        WHEN (PF8 ) INTSC = INTSC + 1;
        WHEN (PF12) INTSC = INTSC + 100;
        OTHERWISE  INTSC = INTSC + 0;
        END;
       PRELEASE;
       END INTSC_3;

  INTSC_4: PROCEDURE;
      PDISPLAY PNAME ('INTROEMT') RSTATUS(RSTATUS);
      SELECT (RSTATUS);
        WHEN (PF7 ) INTSC = INTSC - 1;
        WHEN (PF12) INTSC = INTSC + 100;
        OTHERWISE  INTSC = INTSC + 0;
        END;
       PRELEASE;
       END INTSC_4;

  INTSC_5: PROCEDURE;
      PDISPLAY PNAME ('INTROMTD') RSTATUS(RSTATUS);
      SELECT (RSTATUS);
        WHEN (PF7 ) INTSC = INTSC - 1;
        WHEN (PF8 ) INTSC = INTSC + 1;
        WHEN (PF12) INTSC = INTSC + 100;
        OTHERWISE  INTSC = INTSC + 0;
        END;
       PRELEASE;
       END INTSC_5;

END CONCEPTS_SCREEN;

/*************************************************************************/
/******************* RUN SUMMARY REPORT SCREENS **********************/
/*************************************************************************/

 RUN_SUMRY_REPORTS: PROCEDURE;

  SUMSC = 1;
  MORE_SUMSC = YES;
  DO WHILE (MORE_SUMSC = YES);
   SELECT;
    WHEN (SUMSC = 1) CALL SUMSC_1;
    WHEN (SUMSC = 2) CALL SUMSC_2;
    WHEN (SUMSC = 3) CALL SUMSC_3;
```

```
    WHEN (SUMSC = 4) CALL SUMSC_4;
    WHEN (SUMSC = 5) CALL SUMSC_5;
    WHEN (SUMSC = 6) CALL SUMSC_6;
    WHEN (SUMSC = 7) CALL SUMSC_7;
    OTHERWISE CALL SUMSC_1;
    END;
   END;
  RETURN;

/*****************************************************************/
/********** MULTIATTRIBUTE DECISION MODEL *********************/
/*****************************************************************/

SUMSC_1: PROCEDURE;

DECLARE
 1 SS1_LOADLIST,
  2 LOAD(21)       PTR,
  2 FENCE          BIT(32) INIT((32)'1'B);

/*** COMPUTE AVAILABILITY ***/
SAVAIL = 0;
DO IP = 1 TO PARM_RC.RUN_LENGTH;
 SAVAIL = SAVAIL + PAVAIL(IP) * 100;
 END;
SAVAIL = SAVAIL/PARM_RC.RUN_LENGTH;
MCDM_RPT.AVAILABILITY(3) = SAVAIL;

/*** COMPUTE RELIABILITY ***/
EQTY = 0;
DO IE = 1 TO PARM_RC.NUM_ENV;
 PROB4 = 0;
 DO IP = 1 TO PARM_RC.RUN_LENGTH;
  PROB4 = PROB4 + EPFAIL(IP,IE);
  END;
 PROB4 = PROB4/PARM_RC.RUN_LENGTH;
 ENVREL(IE) = 1 - PROB4;
 EQTY = EQTY + ENVQTY(PARM_RC.RUN_LENGTH,IE);
 END;
RELABILITY = 0;
DO IE = 1 TO PARM_RC.NUM_ENV;
 RELABILITY = RELABILITY +
   ENVREL(IE) * ENVQTY(PARM_RC.RUN_LENGTH,IE)/EQTY;
 END;
MCDM_RPT.RELIABILITY(3) = RELABILITY * 100;

/*** COMPUTE MAINTAINABILITY ***/
MABILITY = 0;
DO IP = 1 TO PARM_RC.RUN_LENGTH;
 EQTY = 0;
```

```
 DO IE = 1 TO NUM_ENV;
  EQTY = EQTY + ENVQTY(IP,IE);
  END;
 PMTTR(IP) = 0;
 DO IE = 1 TO PARM_RC.NUM_ENV;
  PMTTR(IP) = PMTTR(IP) + EMTTR(IP,IE)*ENVQTY(IP,IE)/EQTY;
  END;
 MABILITY = PMTTR(IP)/PARM_RC.RUN_LENGTH + MABILITY;
END;
MCDM_RPT.MAINTAINABILITY(3) = MABILITY;

/*** COMPUTE LIFE CYCLE COSTS ***/
PWCOST = 0;
TVM = COST_RC.INTEREST/100;
DO IP = 1 TO PARM_RC.RUN_LENGTH;
 PDCOST = 0;
 DO IE = 1 TO NUM_ENV;
    COST_RPT.TOT_COST(IP,IE) = COST_RPT.CAP_COST(IP,IE)
     + COST_RPT.REP_COST(IP,IE) + COST_RPT.PUR_COST(IP,IE)
                                + COST_RPT.INV_COST(IP,IE);
  PDCOST = PDCOST + COST_RPT.TOT_COST(IP,IE);
  END;
 PWCOST = PWCOST + PDCOST/((1+TVM)**IP);
 END;
IRL = PARM_RC.RUN_LENGTH;
AGPIN = (TVM*((1+TVM)**IRL))/(((1+TVM)**IRL)-1);
LCYCLE_COST = PWCOST * AGPIN;
MCDM_RPT.LIFE_CYCLE_COST(3) = LCYCLE_COST;

/*** COMPUTE DESIGN EVALUATION SCORE ***/
  MCDM_RPT.SCORE(1) =
 (MCDM_RPT.LIFE_CYCLE_COST(1) - MCDM_RPT.LIFE_CYCLE_COST(3)) /
 (MCDM_RPT.LIFE_CYCLE_COST(1) - MCDM_RPT.LIFE_CYCLE_COST(2)) *
  MCDM_RPT.WEIGHT(1);
IF MCDM_RPT.LIFE_CYCLE_COST(1) < MCDM_RPT.LIFE_CYCLE_COST(1)
  THEN MCDM_RPT.SCORE(1) = 0;
IF MCDM_RPT.LIFE_CYCLE_COST(3) < MCDM_RPT.LIFE_CYCLE_COST(2)
  THEN MCDM_RPT.SCORE(1) = MCDM_RPT.WEIGHT(1);

  MCDM_RPT.SCORE(2) =
 (MCDM_RPT.AVAILABILITY(3) - MCDM_RPT.AVAILABILITY(1)) /
 (MCDM_RPT.AVAILABILITY(2) - MCDM_RPT.AVAILABILITY(1)) *
  MCDM_RPT.WEIGHT(2);
IF MCDM_RPT.AVAILABILITY(3) < MCDM_RPT.AVAILABILITY(1)
  THEN MCDM_RPT.SCORE(2) = 0;
IF MCDM_RPT.AVAILABILITY(3) > MCDM_RPT.AVAILABILITY(2)
  THEN MCDM_RPT.SCORE(2) = MCDM_RPT.WEIGHT(2);

 MCDM_RPT.SCORE(3) =
 (MCDM_RPT.RELIABILITY(3) - MCDM_RPT.RELIABILITY(1)) /
```

```
 (MCDM_RPT.RELIABILITY(2) - MCDM_RPT.RELIABILITY(1)) *
  MCDM_RPT.WEIGHT(3);
IF MCDM_RPT.RELIABILITY(3) < MCDM_RPT.RELIABILITY(1)
  THEN MCDM_RPT.SCORE(3) = 0;
IF MCDM_RPT.RELIABILITY(3) > MCDM_RPT.RELIABILITY(2)
  THEN MCDM_RPT.SCORE(3) = MCDM_RPT.WEIGHT(3);

  MCDM_RPT.SCORE(4) =
 (MCDM_RPT.MAINTAINABILITY(1) - MCDM_RPT.MAINTAINABILITY(3)) /
 (MCDM_RPT.MAINTAINABILITY(1) - MCDM_RPT.MAINTAINABILITY(2)) *
  MCDM_RPT.WEIGHT(4);
IF MCDM_RPT.MAINTAINABILITY(3) > MCDM_RPT.MAINTAINABILITY(1)
  THEN MCDM_RPT.SCORE(4) = 0;
IF MCDM_RPT.MAINTAINABILITY(3) < MCDM_RPT.MAINTAINABILITY(2)
  THEN MCDM_RPT.SCORE(4) = MCDM_RPT.WEIGHT(4);

MCDM_RPT.SYSTEM_EVAL = 0;
DO I = 1 TO 4;
 MCDM_RPT.SYSTEM_EVAL = MCDM_RPT.SYSTEM_EVAL + MCDM_RPT.SCORE(I);
 END;

 SS1_LOADLIST.LOAD(1) = ADDR(MCDM_RPT.LIFE_CYCLE_COST(1));
 SS1_LOADLIST.LOAD(2) = ADDR(MCDM_RPT.LIFE_CYCLE_COST(2));
 SS1_LOADLIST.LOAD(4) = ADDR(MCDM_RPT.LIFE_CYCLE_COST(3));

 SS1_LOADLIST.LOAD(6) = ADDR(MCDM_RPT.AVAILABILITY(1));
 SS1_LOADLIST.LOAD(7) = ADDR(MCDM_RPT.AVAILABILITY(2));
 SS1_LOADLIST.LOAD(9) = ADDR(MCDM_RPT.AVAILABILITY(3));

 SS1_LOADLIST.LOAD(11) = ADDR(MCDM_RPT.RELIABILITY(1));
 SS1_LOADLIST.LOAD(12) = ADDR(MCDM_RPT.RELIABILITY(2));
 SS1_LOADLIST.LOAD(14) = ADDR(MCDM_RPT.RELIABILITY(3));

 SS1_LOADLIST.LOAD(16) = ADDR(MCDM_RPT.MAINTAINABILITY(1));
 SS1_LOADLIST.LOAD(17) = ADDR(MCDM_RPT.MAINTAINABILITY(2));
 SS1_LOADLIST.LOAD(19) = ADDR(MCDM_RPT.MAINTAINABILITY(3));

 SS1_LOADLIST.LOAD(3) = ADDR(MCDM_RPT.WEIGHT(1));
 SS1_LOADLIST.LOAD(8) = ADDR(MCDM_RPT.WEIGHT(2));
 SS1_LOADLIST.LOAD(13) = ADDR(MCDM_RPT.WEIGHT(3));
 SS1_LOADLIST.LOAD(18) = ADDR(MCDM_RPT.WEIGHT(4));

 SS1_LOADLIST.LOAD(5) = ADDR(MCDM_RPT.SCORE(1));
 SS1_LOADLIST.LOAD(10) = ADDR(MCDM_RPT.SCORE(2));
 SS1_LOADLIST.LOAD(15) = ADDR(MCDM_RPT.SCORE(3));
 SS1_LOADLIST.LOAD(20) = ADDR(MCDM_RPT.SCORE(4));

 SS1_LOADLIST.LOAD(21) = ADDR(MCDM_RPT.SYSTEM_EVAL);

 PDISPLAY PNAME('SRPTEVAL') RSTATUS(RSTATUS) LDLIST(SS1_LOADLIST)
```

```
          ULDLIST(SS1_LOADLIST) DCURSOR(CURSOR);

 SELECT (RSTATUS);
     WHEN (PF1 ) SUMSC = 2;
     WHEN (PF2 ) SUMSC = 3;
     WHEN (PF3 ) SUMSC = 4;
     WHEN (PF4 ) SUMSC = 5;
     WHEN (PF5 ) SUMSC = 6;
     WHEN (PF6 ) SUMSC = 7;
     WHEN (PF9 ) SUMSC = 1;
  WHEN(PF10) DO;
   SCNUM = 14;
   MORE_SUMSC = NO;
   END;
  WHEN(PF11) DO;
   SCNUM = 13;
   MORE_SUMSC = NO;
   END;
  WHEN(PF12) DO;
   SCNUM = 1;
   MORE_SUMSC = NO;
   END;
  OTHERWISE SUMSC = 1;
  END;
  PRELEASE;
 END SUMSC_1;

/*********************************************/
/**** END ITEM FAILURE PROBABILITIES ****/
/*********************************************/

SUMSC_2: PROCEDURE;

DECLARE
 1 SS2_LOADLIST,
  2 LOAD(50)          PTR,
  2 FENCE             BIT(32) INIT((32)'1'B);

IS = 1;
XNG = PARM_RC.RUN_LENGTH/10 + 1;
NG = CEIL(XNG);
DO WHILE (IS <= NG);
  DO IP = (IS-1)*10+1 TO IS*10;
    J = IP - (IS-1)*10;
    DO IE = 1 TO PARM_RC.NUM_ENV;
     EFAIL_RPT.FAIL(J,IE) = EPFAIL(IP,IE);
     END;
    EFAIL_RPT.PDNO(J) = IP;
    SS2_LOADLIST.LOAD((J-1)*5+1)=ADDR(EFAIL_RPT.PDNO(J));
    SS2_LOADLIST.LOAD((J-1)*5+2)=ADDR(EFAIL_RPT.FAIL(J,1));
```

```
      SS2_LOADLIST.LOAD((J-1)*5+3)=ADDR(EFAIL_RPT.FAIL(J,2));
      SS2_LOADLIST.LOAD((J-1)*5+4)=ADDR(EFAIL_RPT.FAIL(J,3));
      SS2_LOADLIST.LOAD((J-1)*5+5)=ADDR(EFAIL_RPT.FAIL(J,4));
      END;

    PDISPLAY PNAME('SRPTFAIL') RSTATUS(RSTATUS) LDLIST(SS2_LOADLIST)
             ULDLIST(SS2_LOADLIST) DCURSOR(CURSOR);
    SELECT (RSTATUS);
     WHEN (PF7 ) DO;
      IF IS       1 THEN IS = IS - 1;
      ELSE IS = 1;
      END;
     WHEN (PF8 ) DO;
      IF IS < NG THEN IS = IS + 1;
      ELSE IS = NG;
      END;
     OTHERWISE DO;
      SELECT (RSTATUS);
       WHEN (PF1 ) SUMSC = 2;
       WHEN (PF2 ) SUMSC = 3;
       WHEN (PF3 ) SUMSC = 4;
       WHEN (PF4 ) SUMSC = 5;
       WHEN (PF5 ) SUMSC = 6;
       WHEN (PF6 ) SUMSC = 7;
       WHEN (PF9 ) SUMSC = 1;
       WHEN (PF10) DO;
        SCNUM = 14;
        MORE_SUMSC = NO;
        END;
       WHEN (PF11) DO;
        SCNUM = 13;
        MORE_SUMSC = NO;
        END;
       WHEN (PF12) DO;
        SCNUM = 1;
        MORE_SUMSC = NO;
        END;
       OTHERWISE SUMSC = 2;
       END;
     IS = NG + 5;
     END;
    END;
   END;
   PRELEASE;
   END SUMSC_2;

  /*************************************/
  /***** SPARE PARTS REQUIREMENTS *****/
  /*************************************/
```

```
SUMSC_3: PROCEDURE;
DECLARE
 1 SS3_LOADLIST,
  2 LOAD(60)          PTR,
  2 FENCE             BIT(32) INIT((32)'1'B);

IS = 1;
XNG = PARM_RC.NUM_STK_RC/12 + 1;
NG = CEIL(XNG);
DO WHILE (IS <= NG);
  DO I = (IS-1)*12+1 TO IS*12;
    J = I - (IS-1)*12;
    SPART_RPT.NODE_NUM(J) = PART_DATA.NODE_NUM(I);
    SPART_RPT.PART_DESC(J) = PART_DATA.PART_DESC(I);
    SPART_RPT.PART_NUM(J) = PART_DATA.PART_NUM(I);
    SPART_RPT.DEMAND(J)=0;
    DO J1 = 1 TO PARM_RC.RUN_LENGTH;
    SPART_RPT.DEMAND(J)=SPART_RPT.DEMAND(J)+PART_DMD.PD_DMD(J1,I);
    END;
    PLOR=3;
    DO J1 = 1 TO PARM_RC.NUM_NODES;
     IF PART_DATA.NODE_NUM(I) = NODE_DATA.NODE_NUM(1,J1) THEN
        PLOR = NODE_DATA.LOR(1,J1);
     END;
    SPART_RPT.LOR(J) = PLOR;

    SS3_LOADLIST.LOAD((J-1)*5+1)=ADDR(SPART_RPT.NODE_NUM(J));
    SS3_LOADLIST.LOAD((J-1)*5+2)=ADDR(SPART_RPT.PART_DESC(J));
    SS3_LOADLIST.LOAD((J-1)*5+3)=ADDR(SPART_RPT.PART_NUM(J));
    SS3_LOADLIST.LOAD((J-1)*5+4)=ADDR(SPART_RPT.DEMAND(J));
    SS3_LOADLIST.LOAD((J-1)*5+5)=ADDR(SPART_RPT.LOR(J));
    END;

  PDISPLAY PNAME('SRPTPART') RSTATUS(RSTATUS) LDLIST(SS3_LOADLIST)
           ULDLIST(SS3_LOADLIST) DCURSOR(CURSOR);
  SELECT (RSTATUS);
   WHEN (PF7 ) DO;
    IF IS       1 THEN IS = IS - 1;
    ELSE IS = 1;
    END;
   WHEN (PF8 ) DO;
    IF IS < NG THEN IS = IS + 1;
    ELSE IS = NG;
    END;
   OTHERWISE DO;
    SELECT (RSTATUS);
     WHEN (PF1 ) SUMSC = 2;
     WHEN (PF2 ) SUMSC = 3;
     WHEN (PF3 ) SUMSC = 4;
     WHEN (PF4 ) SUMSC = 5;
```

```
      WHEN (PF5 ) SUMSC = 6;
      WHEN (PF6 ) SUMSC = 7;
      WHEN (PF9 ) SUMSC = 1;
      WHEN (PF10) DO;
       SCNUM = 14;
       MORE_SUMSC = NO;
       END;
      WHEN (PF11) DO;
       SCNUM = 13;
       MORE_SUMSC = NO;
       END;
      WHEN (PF12) DO;
       SCNUM = 1;
       MORE_SUMSC = NO;
       END;
      OTHERWISE SUMSC = 3;
      END;
    IS = NG + 5;
    END;
   END;
  END;
  PRELEASE;
  END SUMSC_3;

/*********************************/
/**** END ITEM AVAILABILITY ****/
/*********************************/

SUMSC_4: PROCEDURE;
DECLARE
 1 SS4_LOADLIST,
  2 LOAD(60)          PTR,
  2 FENCE             BIT(32) INIT((32)'1'B);

IS = 1;
XNG = PARM_RC.RUN_LENGTH/10 + 1;
NG = CEIL(XNG);
DO WHILE (IS <= NG);
  DO IP = (IS-1)*10+1 TO IS*10;
    J = IP - (IS-1)*10;
    DO IE = 1 TO PARM_RC.NUM_ENV;
     EIAVAIL_RPT.AVAIL(J,IE) = ENVAVAIL(IP,IE) * 100;
     END;
    EIAVAIL_RPT.PDNO(J) = IP;
    EIAVAIL_RPT.SYS_AVAIL(J) = PAVAIL(IP) * 100;
    SS4_LOADLIST.LOAD((J-1)*6+1)=ADDR(EIAVAIL_RPT.PDNO(J));
    SS4_LOADLIST.LOAD((J-1)*6+2)=ADDR(EIAVAIL_RPT.AVAIL(J,1));
    SS4_LOADLIST.LOAD((J-1)*6+3)=ADDR(EIAVAIL_RPT.AVAIL(J,2));
    SS4_LOADLIST.LOAD((J-1)*6+4)=ADDR(EIAVAIL_RPT.AVAIL(J,3));
    SS4_LOADLIST.LOAD((J-1)*6+5)=ADDR(EIAVAIL_RPT.AVAIL(J,4));
```

```
      SS4_LOADLIST.LOAD((J-1)*6+6)=ADDR(EIAVAIL_RPT.SYS_AVAIL(J));
      END;

    PDISPLAY PNAME('SRPTAVAI') RSTATUS(RSTATUS) LDLIST(SS4_LOADLIST)
             ULDLIST(SS4_LOADLIST) DCURSOR(CURSOR);
    SELECT (RSTATUS);
     WHEN (PF7 ) DO;
      IF IS       1 THEN IS = IS - 1;
      ELSE IS = 1;
      END;
     WHEN (PF8 ) DO;
      IF IS < NG THEN IS = IS + 1;
      ELSE IS = NG;
      END;
     OTHERWISE DO;
      SELECT (RSTATUS);
       WHEN (PF1 ) SUMSC = 2;
       WHEN (PF2 ) SUMSC = 3;
       WHEN (PF3 ) SUMSC = 4;
       WHEN (PF4 ) SUMSC = 5;
       WHEN (PF5 ) SUMSC = 6;
       WHEN (PF6 ) SUMSC = 7;
       WHEN (PF9 ) SUMSC = 1;
       WHEN (PF10) DO;
        SCNUM = 14;
        MORE_SUMSC = NO;
        END;
       WHEN (PF11) DO;
        SCNUM = 13;
        MORE_SUMSC = NO;
        END;
       WHEN (PF12) DO;
        SCNUM = 1;
        MORE_SUMSC = NO;
        END;
       OTHERWISE SUMSC = 4;
       END;
     IS = NG + 5;
     END;
    END;
   END;
   PRELEASE;
   END SUMSC_4;

  /*****************************************/
  /**** WAREHOUSE SPACE REQUIREMENTS ***/
  /*****************************************/

  SUMSC_5: PROCEDURE;
  DECLARE
```

```
 1 SS5_LOADLIST,
  2 LOAD(40)          PTR,
  2 FENCE             BIT(32) INIT((32)'1'B);

IS = 1;
XNG = PARM_RC.RUN_LENGTH/10 + 1;
NG = CEIL(XNG);
DO WHILE (IS <= NG);
  DO IP = (IS-1)*10+1 TO IS*10;
    J = IP - (IS-1)*10;
    DO IE = 1 TO PARM_RC.REPAIR_LEVELS;
     SPACE_RPT.SQFT(J,IE) = SPACE(IP,IE);
     END;
    SPACE_RPT.PDNO(J) = IP;
    SS5_LOADLIST.LOAD((J-1)*4+1)=ADDR(SPACE_RPT.PDNO(J));
    SS5_LOADLIST.LOAD((J-1)*4+2)=ADDR(SPACE_RPT.SQFT(J,1));
    SS5_LOADLIST.LOAD((J-1)*4+3)=ADDR(SPACE_RPT.SQFT(J,2));
    SS5_LOADLIST.LOAD((J-1)*4+4)=ADDR(SPACE_RPT.SQFT(J,3));
    END;

  PDISPLAY PNAME('SRPTWHSE') RSTATUS(RSTATUS) LDLIST(SS5_LOADLIST)
           ULDLIST(SS5_LOADLIST) DCURSOR(CURSOR);
  SELECT (RSTATUS);
   WHEN (PF7 ) DO;
    IF IS       1 THEN IS = IS - 1;
    ELSE IS = 1;
    END;
   WHEN (PF8 ) DO;
    IF IS < NG THEN IS = IS + 1;
    ELSE IS = NG;
    END;
   OTHERWISE DO;
    SELECT (RSTATUS);
     WHEN (PF1 ) SUMSC = 2;
     WHEN (PF2 ) SUMSC = 3;
     WHEN (PF3 ) SUMSC = 4;
     WHEN (PF4 ) SUMSC = 5;
     WHEN (PF5 ) SUMSC = 6;
     WHEN (PF6 ) SUMSC = 7;
     WHEN (PF9 ) SUMSC = 1;
     WHEN (PF10) DO;
      SCNUM = 14;
      MORE_SUMSC = NO;
      END;
     WHEN (PF11) DO;
      SCNUM = 13;
      MORE_SUMSC = NO;
      END;
     WHEN (PF12) DO;
      SCNUM = 1;
```

```
      MORE_SUMSC = NO;
      END;
     OTHERWISE SUMSC = 5;
     END;
   IS = NG + 5;
   END;
  END;
 END;
 PRELEASE;
 END SUMSC_5;

/*****************************/
/**** LABOR REQUIREMENTS ***/
/*****************************/

SUMSC_6: PROCEDURE;
DECLARE
 1 SS6_LOADLIST,
  2 LOAD(40)          PTR,
  2 FENCE             BIT(32) INIT((32)'1'B);

IS = 1;
XNG = PARM_RC.RUN_LENGTH/10 + 1;
NG = CEIL(XNG);
DO WHILE (IS <= NG);
  DO IP = (IS-1)*10+1 TO IS*10;
    J = IP - (IS-1)*10;
    DO IE = 1 TO PARM_RC.REPAIR_LEVELS;
     LABOR_RPT.HOUR(J,IE) = LABOR(IP,IE);
     END;
    LABOR_RPT.PDNO(J) = IP;
    SS6_LOADLIST.LOAD((J-1)*4+1)=ADDR(LABOR_RPT.PDNO(J));
    SS6_LOADLIST.LOAD((J-1)*4+2)=ADDR(LABOR_RPT.HOUR(J,1));
    SS6_LOADLIST.LOAD((J-1)*4+3)=ADDR(LABOR_RPT.HOUR(J,2));
    SS6_LOADLIST.LOAD((J-1)*4+4)=ADDR(LABOR_RPT.HOUR(J,3));
    END;

  PDISPLAY PNAME('SRPTLABR') RSTATUS(RSTATUS) LDLIST(SS6_LOADLIST)
           ULDLIST(SS6_LOADLIST) DCURSOR(CURSOR);
  SELECT (RSTATUS);
   WHEN (PF7 ) DO;
    IF IS       1 THEN IS = IS - 1;
    ELSE IS = 1;
    END;
   WHEN (PF8 ) DO;
    IF IS < NG THEN IS = IS + 1;
    ELSE IS = NG;
    END;
   OTHERWISE DO;
    SELECT (RSTATUS);
```

```
      WHEN (PF1 ) SUMSC = 2;
      WHEN (PF2 ) SUMSC = 3;
      WHEN (PF3 ) SUMSC = 4;
      WHEN (PF4 ) SUMSC = 5;
      WHEN (PF5 ) SUMSC = 6;
      WHEN (PF6 ) SUMSC = 7;
      WHEN (PF9 ) SUMSC = 1;
      WHEN (PF10) DO;
       SCNUM = 14;
       MORE_SUMSC = NO;
       END;
      WHEN (PF11) DO;
       SCNUM = 13;
       MORE_SUMSC = NO;
       END;
      WHEN (PF12) DO;
       SCNUM = 1;
       MORE_SUMSC = NO;
       END;
      OTHERWISE SUMSC = 6;
      END;
    IS = NG + 5;
    END;
   END;
  END;
  PRELEASE;
  END SUMSC_6;

/*****************************************/
/***** COST SUMMARY BY ENVIRONMENT ***/
/*****************************************/
SUMSC_7: PROCEDURE;
DECLARE
 1 SS7_LOADLIST,
  2 LOAD(81)          PTR,
  2 FENCE             BIT(32) INIT((32)'1'B);

XNG = PARM_RC.RUN_LENGTH/10 + 1;
NG = CEIL(XNG);

IE = 1;
DO WHILE (IE <= PARM_RC.NUM_ENV);
 IS = 1;
DO WHILE (IS <= NG);
  DO IP = (IS-1)*10+1 TO IS*10;
   J = IP - (IS-1)*10;

   IF IP <= PARM_RC.RUN_LENGTH THEN DO;
    COST_RPT.TOT_COST(IP,IE) = COST_RPT.CAP_COST(IP,IE)
     + COST_RPT.REP_COST(IP,IE) + COST_RPT.PUR_COST(IP,IE)
```

```
                                + COST_RPT.INV_COST(IP,IE);
   COST_RPT.UNITS_START(IP,IE) = 0;
   COST_RPT.UNITS_END(IP,IE) = 0;
  DO JS = 1 TO NUM_STREAMS;
   IF STREAM.FAIL_ENV(JS) = IE THEN DO;
   COST_RPT.UNITS_START(IP,IE) = COST_RPT.UNITS_START(IP,IE)
                                   + STRQTY.BEG_PD(IP,JS);
   COST_RPT.UNITS_END(IP,IE) = COST_RPT.UNITS_END(IP,IE)
                                   + STRQTY.END_PD(IP,JS);
   END;
  END;
END;
COST_RPT.PDNO(J,IE) = IP;

   SS7_LOADLIST.LOAD(1) = ADDR(ENV_DATA.ENV_DESC(IE));
   SS7_LOADLIST.LOAD((J-1)*8+2)=ADDR(COST_RPT.PDNO(J,IE));
   SS7_LOADLIST.LOAD((J-1)*8+3)=ADDR(COST_RPT.CAP_COST(IP,IE));
   SS7_LOADLIST.LOAD((J-1)*8+4)=ADDR(COST_RPT.REP_COST(IP,IE));
   SS7_LOADLIST.LOAD((J-1)*8+5)=ADDR(COST_RPT.INV_COST(IP,IE));
   SS7_LOADLIST.LOAD((J-1)*8+6)=ADDR(COST_RPT.PUR_COST(IP,IE));
   SS7_LOADLIST.LOAD((J-1)*8+7)=ADDR(COST_RPT.TOT_COST(IP,IE));
   SS7_LOADLIST.LOAD((J-1)*8+8)=ADDR(COST_RPT.UNITS_START(IP,IE));
   SS7_LOADLIST.LOAD((J-1)*8+9)=ADDR(COST_RPT.UNITS_END(IP,IE));
   END;

PDISPLAY PNAME('SRPTCOST') RSTATUS(RSTATUS) LDLIST(SS7_LOADLIST)
         ULDLIST(SS7_LOADLIST) DCURSOR(CURSOR);
SELECT (RSTATUS);
 WHEN(PF7 ) DO;
    IF IS <= 1 THEN DO;
      IF IE <= 1 THEN DO;
        IE = NUM_ENV + 1;
        IS = NG + 1;
        END;
      ELSE DO;
       IE = IE - 1;
       IS = NG;
       END;
      END;
    ELSE IS = IS - 1;
    END;

 WHEN(PF8 ) DO;
    IS = IS + 1;
    IF IS      NG THEN DO;
      IE = IE + 1;
      END;
    END;

 OTHERWISE DO;
```

```
      SELECT (RSTATUS);
       WHEN (PF1 ) SUMSC = 2;
       WHEN (PF2 ) SUMSC = 3;
       WHEN (PF3 ) SUMSC = 4;
       WHEN (PF4 ) SUMSC = 5;
       WHEN (PF5 ) SUMSC = 6;
       WHEN (PF6 ) SUMSC = 7;
       WHEN (PF9 ) SUMSC = 1;
       WHEN (PF10) DO;
        SCNUM = 14;
        MORE_SUMSC = NO;
        END;
       WHEN (PF11) DO;
        SCNUM = 13;
        MORE_SUMSC = NO;
        END;
       WHEN (PF12) DO;
        SCNUM = 1;
        MORE_SUMSC = NO;
        END;
       OTHERWISE SUMSC = 7;
       END;
     IS = NG + 5;
     IE = PARM_RC.NUM_ENV + 5;
     END;
    END;
   END;
   END;
   PRELEASE;
   END SUMSC_7;

  END RUN_SUMRY_REPORTS;

END CODA;
```

Vita

Roderick J. Reasor was born on April 8, 1953. He grew up and went to school in Hampton, Virginia, graduating with honors from Hampton High School in 1971. Mr. Reasor entered Virginia Polytechnic Institute and State University (VPI&SU) that same year to pursue a career in engineering. As an undergraduate, he was a co-op student at NASA's Langley Research Center. Mr. Reasor completed his Bachelor of Science degree in Industrial Engineering and Operations Research (IEOR) in 1976.

Immediately following graduation, Mr. Reasor began work on his Master of Science degree. He completed his course work before accepting a position as an Industrial Engineer at Eastman Kodak, Tennessee Eastman Company in Kingsport, Tennessee. At Tennessee Eastman, Mr. Reasor was primarily responsible for the design and analysis of materials management systems supporting production planning, inventory control, warehousing, and physical distribution. During this time, Mr. Reasor completed his thesis and received his Master of Science degree in IEOR from VPI&SU in 1981.

In 1983, Mr. Reasor accepted a position as an Instructor in the IEOR Department at VPI&SU. In addition to his undergraduate teaching responsibilities, Mr. Reasor was involved in funded research of decision support systems at the Management Systems Laboratories for three years. He is currently an Assistant Professor in the IEOR Department at VPI&SU in Blacksburg, Virginia. Mr. Reasor has been a registered professional engineer in the State of Tennessee since 1982. He is married to the former Anita Knibb. They have three children: Rebecca, Matthew, and Christopher.