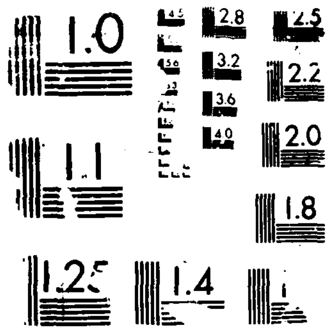
AD-A189 599 REDESIGN OF THE JOINT PLANNING EXERCISE (JPLAN)(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING J R JANSEN DEC 87 AFIT/GCS/ENG/87D-15 1/1

UNCLASSIFIED F/G 15/6 NL



MICROCOPY RESOLUTION TEST CHART

REDESIGN OF THE JOINT PLANNING EXERCISE (JPLAN)

THESIS

James R. Jansen
Captain, USAF

AFIT/GCS/ENG/87D-15

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/GCS/ENG/87D-15

REDESIGN OF THE JOINT PLANNING EXERCISE (JPLAN)

THESIS

James R. Jansen
Captain, USAF

AFIT/GCS/ENG/87D-15

AFIT/GCS/ENG/87D-15

REDESIGN OF THE JOINT PLANNING EXERCISE (JPLAN)

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science (Information Systems)

James R. Jansen, B.S., M.S.

Captain, USAF

December, 1987

Preface

The goal of this thesis was to design and implement a new database and user interface for the Joint Planning Exercise (JPLAN) using a microcomputer system and a microcomputer database management system and to port the simulation code from the mainframe to the microcomputer.

The Joint Operations Planning System (JOPS) is a Department of Defense system for the conduct of the joint planning process. JPLAN is an exercise which parallels JOPS and is used in education programs at the Air Force Institute of Technology and the Air Force Wargaming Center.

This thesis presents background on JOPS and JPLAN, problems with the current simulation, and a approach for a new microcomputer implementation which eliminates the "user hostile" interface for users and maintainers of the simulation.

I an deeply indebted to my thesis advisor, Captain Mark Roth, for his invaluable assistance during the development of this thesis. I also wish to thank the other members of my committee, Lt. Colonel Skip Valusek and Lt. Colonel Dennis Dragich, for their assistance. In addition, I wish to thank the Air Force Wargaming Center for their support of this thesis effort.

James R. Jansen

Table of Contents

List of Figures

List of Tables

AFIT/GCS/ENG/87D-15

Abstract

JPLAN is a simulation of the Joint Operation Planning System used by the various commands to develop operation plans. JPLAN allows student teams at the Air Force Institute of Technology, Air War College, and Air Command and Staff College to identify needed combat, combat support, and combat service support forces, build force lists, and simulate deployment of these forces in support of an operation plan. Transportation resources are provided to transport the forces to their destinations. Students must resolve conflicts in transportation availability and set priorities on forces and their destinations to ensure their forces arrive on time.

The existing JPLAN exercise is outdated both in terms of hardware and software which runs on hardcopy terminals connected to a large mainframe computer. It is not easily used nor maintained due to the use of a slow, unforgiving user interface, a fixed file management system, and fixed reporting capabilities. This thesis describes the redesign of JPLAN to operate on a microcomputer using state-of-the-art database and program design techniques and a fourth generation application development system.

The entity-relationship (E-R) data model was used to represent JPLAN's data entities, attributes and entity relationships. Then a relational database structure was used to implement the database design portrayed by the E-R diagram. The user interface was designed by using storyboards to illustrate menus and their function to the users, then user feedback was obtained to refine the design. The INGRES microcomputer database management system was chosen to implement the database and user interface designs. INGRES includes a complete array of modern application development tools. A screen editing tool and a fourth generation language (4GL) were employed to implement the user interface portion of the design, and a structured query language (SQL) was utilized for the database implementation.

The FORTRAN simulation was rewritten in Microsoft C and Embedded SQL. During the simulation coding and validation process several areas for improvement were identified and incorporated into the new simulation program, which will enable the exercise to parallel the "real world" more closely. A monitor program was written to give instructors the capability to easily edit the database data when changes are required. On-line user instructions were incorporated into the design to allow new users to quickly master the JPLAN exercise. System design information was provided to allow for the maintenance and improvement of JPLAN. The new JPLAN exercise is more "user friendly" than the existing exercise, runs on widely available microcomputers, and has become much easier to modify as requirements evolve.

REDESIGN OF THE JOINT PLANNING EXERCISE (JPLAN)

I. Introduction

1.1 Background

To be effective the four military services must take a joint, or team approach to attaining military objectives. This means that we must plan our operations together. As one can imagine, the job of fulfilling the requirements of each service is in itself a monumental task, and the process of integrating these operation plans throughout the four services is even more complex. The Joint Operation Planning System (JOPS) provides a standardized approach for planning and integrating these joint military operations. The following is an outline of the JOPS: [1]

1. INITIATION - The need for a plan is identified and CINCs are tasked.
2. CONCEPT DEVELOPMENT
   (a) Analysis of the Missions and Tasks
   (b) Preliminary Planning Guidance
   (c) Preparation of Staff Estimates
   (d) Preparation of the Commanders Estimate
   (e) Preparation of the Concept of Operations
3. PLAN DEVELOPMENT
   (a) Force Planning
   (b) Support Planning
   (c) Chemical and Nuclear Planning

(d) Transportation Planning

(e) Force, Movement and Support Shortfall Identification

(f) Transportation Feasibility Analysis

(g) Time Phase Force Deployment Data (TPFDD) Refinement

(h) Plan Documentation

4. PLAN REVIEW

5. SUPPORTING PLANS

1.1.1 Exercise Description. The Joint Planning Exercise (JPLAN) is a simulation of the Joint Operation Planning System (JOPS) used by the unified and specified commands to develop operation plans. The Rapid Deployment Exercise (RADEX) is a variation of JPLAN which adds sealift capability and is played at the Air War College. Throughout this thesis the term JPLAN will be used to refer to the exercise in general. JPLAN allows student teams to identify needed combat, combat support, and combat service support forces; to build force lists; and simulate deployment of these forces in support of an operation plan. Students first act in the capacity of logistics planners at the component level to select forces and build force lists. Then, they act in the capacity of unified command planners for airlift simulation.

JPLAN was originally designed by the HQ Air University, Data Automation Directorate Computer Science Division (now part of the Air Force Wargaming Center) for the Air Command and Staff College (ACSC) as part of their Theater Warfare Studies program. In that role, JPLAN supports up to 80 teams simultaneously plus a monitor program for instructors and system operators. The exercise currently runs on the Honeywell 6000 series computer with student teams making inputs with TI Silent 700 modem terminals.

JPLAN is also used at the Air Force Institute of Technology (AFIT) as part of the Combat Logistics graduate and professional continuing education curriculum. The exercise runs on the Air

Force Logistics Command (AFLC) Honeywell 6000 computer using the CREATE system dial up capability. Only four to six teams play at a time and no instructor monitor capability is needed. The AFIT course is also presented at many Air Force installations worldwide. To meet this requirement the exercise can run on a Honeywell computer at the installation, if one is available, or through a telephone link to the CREATE system at Wright-Patterson AFB, Ohio.

Neither of these alternatives is ideal. Running the exercise on a computer at the installation presents many problems, including allocating sufficient computer time, the availability of computer personnel to install the program, compatibility problems involved in getting the program up and running, and availability of terminals to use the exercise. The other alternative of using a telephone link to the CREATE system also has problems of finding sufficient terminals to use, losing the telephone link due to technical problems, and the high expense of time over telephone links.

The current implementation of the exercise is described as "user hostile" because both the hardware and software can be very frustrating to use. The exercise uses hardcopy thermal printer terminals (i.e., TI Silent 700) operating at 300 BAUD. Printing long listings of data on this device is quite common and consumes a large amount of the users' time. On the software side, the entire exercise is written in FORTRAN with data being stored in fixed flat files, which makes exercise data changes very difficult. All user inputs are line oriented since a screen doesn't exist, and all outputs are via the thermal printer. Since the implementation is closely coupled to FORTRAN conventions, the user is forced to enter lines of data with commas and spaces placed exactly as required or he must start over when an input error occurs. In summary, the students spend too much time wrestling with the current system when they could be accomplishing their learning objectives.

1.2 Problem Statement

The purpose of this thesis was to design and implement a new database and user interface for JPLAN using a microcomputer system (specifically, the Zenith Z-158 and Z-248) and a microcomputer database management system (MDBMS) and to port the simulation code.

1.3 Justification

This solution allowed the exercise to operate independently of a large computer system and eliminate the use of terminals for calling the CREATE system. This allowed the exercise to be used anywhere a compatible microcomputer can be found, which today means almost any government office or organization. The availability of portable microcomputers will allow the instructor to take all the necessary equipment with him to the installation. Also, by using a MDBMS the JPLAN exercise will be easily modified to meet changing learning objectives.

1.4 Results

The end product was a JPLAN exercise running on a microcomputer which maintains the overall integrity and operating accuracy of the current mainframe version, with minor changes to allow the exercise to mirror JOPS more closely .

1.5 Assumptions

The development of the JPLAN microcomputer system was based on the following assumptions:

1. That the current JPLAN exercise meets the learning objectives of the Air Force Wargaming Center and the Air Force Institute of Technology's education programs, and would not require modifications other than those required for redesign to a MDBMS.

2. The JPLAN microcomputer implementation would be considered valid if it produces output identical to the current implementation given the same input data.

1.6 *Objectives*

The following objectives were accomplished through the course of this thesis:

1. A database system was designed to handle all of the JPLAN information necessary to run the simulation.
2. The Dialog Component (user interface) was designed to allow users to interact with the system and run the simulation. Every effort was made to eliminate the current "user hostile" interface.
3. A monitor program was written to allow course administrators easy access to database tables so they can change the exercise conditions at any time [8].
4. The old simulation, which was written in FORTRAN IV, was rewritten in Microsoft C. Minor changes were made to allow the simulation to more closely simulate the current JOPS system. The simulation will operate on any IBM PC compatible microcomputer system (with sufficient memory and a hard disk drive), and was developed on a Zenith Z-248.
5. User instructions in the form of help menus were developed to aid in the operation of the system. Detailed student manuals were the responsibility of the sponsoring agencies.
6. Installation and system design information is provided to allow continued improvement and system maintenance.

1.7 *Software Development*

The following general software development methodology was used as a guide throughout the course of this thesis (see Figure 1): [4]

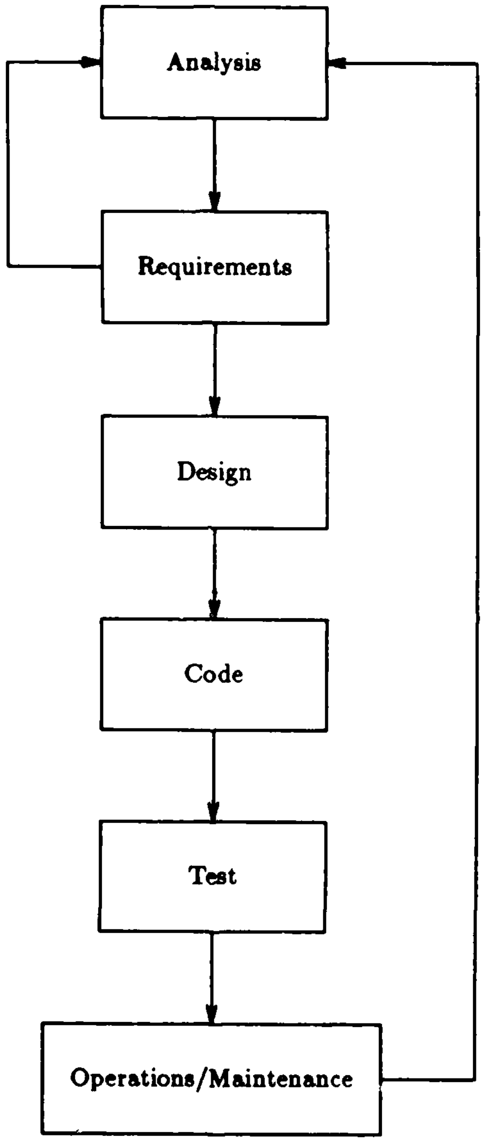


Figure 1. Software Development Cycle

- Analysis Phase – Define the system needs and development of system specifications, including data item identification for the database (data dictionary).
- Requirements Development Phase – Determine detailed requirements from the system specifications and create a conceptual view of the database. Develop screens for the user interface.
- Design Phase – Define the relationships among program components, complete the development specifications, and define and normalize the database relations. Refine screen designs and obtain user approval.
- Coding/Implementation Phase – Write/convert the programs and implement the database design and user interface.
- Testing Phase – Check that the system meets the original requirements.
- Operations and Maintenance Phase – Operation of the system, correction of deficiencies, and improvements.

1.8 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 presents a summary of database design methods and dialog design guidelines utilized throughout the design process. Then Chapter 3 presents the steps of the database and dialog design process for JPLAN. Chapter 4 discusses the implementation of the database and dialog designs from Chapter 3 using the MDBMS chosen. The JPLAN simulation conversion and its linkage to run under the new MDBMS is described in Chapter 5. Then methods used to test and validate the new JPLAN exercise are presented in Chapter 6. Chapter 7 presents conclusions and recommendations for enhancement of the new JPLAN exercise.

II. Database and Dialog Design Issues

2.1 Database Design Methodologies

To some the term database means a collection of files which contain some sort of data. However, a database is more than a collection of files. A database can be defined as a collection of interrelated data (files) stored together for the purpose of serving multiple applications which can be independent of one another, while avoiding unnecessary redundancy and data inconsistencies inherent in file systems [11]

Prior to the use of database techniques systems duplicated massive amounts of information by storing variations of data in different files and for different purposes. By having a large amount of data redundancy throughout these files the possibility of having inconsistent data was significantly increased. For example, a bank could have a file system for checking accounts and a separate file system for loans, each would have duplicate information on the customer (e.g., name, address). When the data (name, address) in the checking account changes the same change is not reflected in the loan account. Cross-checks had to be developed to ensure that when a data item was updated in one file the identical data item in another file was also updated. As systems became larger the problem of managing duplicate data compounded and data reliability became questionable.

2.1.1 Database Advantages. A properly implemented database system can eliminate these problems and provide for more flexibility in developing other applications to utilize the same data. Since data is often an organization's most vital resource the data must be reliable and consistent.

In reality some redundancy may exist in a database to improve access times to data items, simplify addressing schemes or provide for recovery in the event of an accidental loss of data [11]. However, in modern database systems this redundancy should not present a problem because automated cross-checking and verification are inherent in the system.

C. J. Date says that centralized control of data is one of the most valuable assets provided by a database and lists the following as advantages of this centralized control [5]:

1. Redundancy is reduced by not having separate files for each application as in non-database systems.
2. Most inconsistencies can be eliminated by reducing and controlling redundancies.
3. Data can be shared by present users and new users can be added without the addition of new files.
4. Standards are easier to enforce because of the centralized control of the database.
5. Security can be enhanced by ensuring the only access to the database is through the proper channels, and levels of access can defined for various types of data.
6. Integrity of the data is easier to maintain through the centralized control of a database than through decentralized file systems.

One of the major purposes of a database is to hide the implementation details from the user, thus providing an abstract view of the data [10]. The user need not be concerned with how the data is arranged (e.g., records, relations), how the data is physically stored on the disk (e.g., file structure) , or the retrieval techniques (e.g., indices, hashing algorithms) used to accomplish the user's request. Korth and Silberschatz break the abstraction into three levels (see Figure 2) [10, page 4]:

1. Physical level – The lowest level of abstraction which includes details on how the data are actually stored. Can be handled by the operating system or database.
2. Conceptual level – Describes what data are stored in the database and the relationships which exist among the data. Used by the Database Administrator to structure the database into schemas (a group of data).

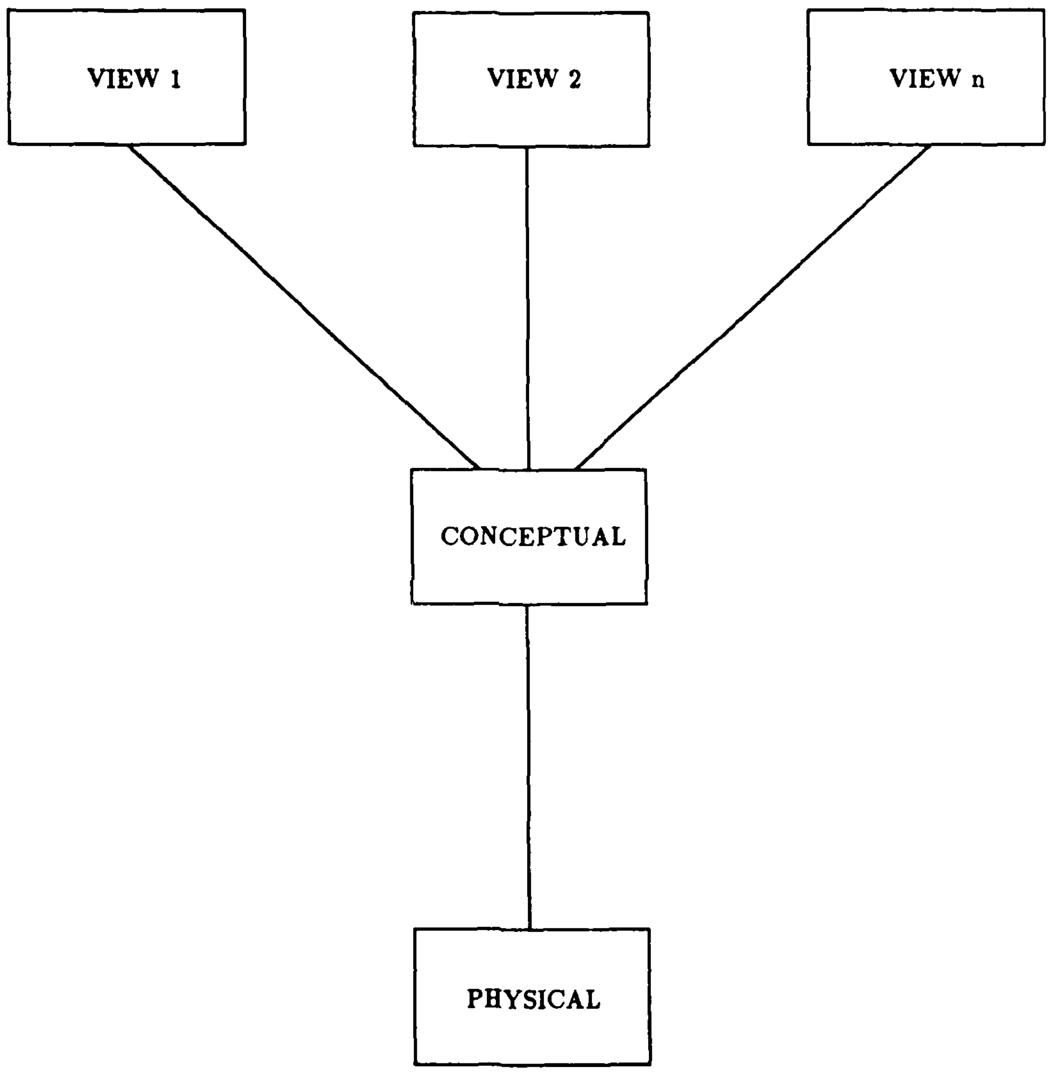


Figure 2. Three Levels of Data Abstraction

3. View level – This level describes a part of the database to a particular user or set of users. Only the information required by the particular case is described in a specific view. May be in the form of a set of screens on the user's application program(s).

2.1.2 Data Modeling. The following discussion of data models is based on Korth and Silberschatz. A data model is a conceptual tool used to describe data and the relationships among the data. There are three main groups of data models: object-based logical models, record-based logical models, and physical data models. As physical data models have limited usage in the design of database applications the following discussion centers on the object and record-based models.

Object-based logical models are used when describing data at the view and conceptual levels. They allow data constraints to be specified explicitly and permit users and designers to view the problem from an abstract level. This type of model is easier for the non-computer person to understand.

One such model is the entity-relationship (E-R) model which is based on the understanding of the real world which consists of objects called entities, and relationships which exist between these entities. Each entity is also described by a set of attributes or characteristics that describe the entity.

E-R diagrams provide a vehicle to describe the conceptual organization of a set of data irrespective of the proposed applications. It is imperative that the database design not be restrained to any one application and have extensibility inherent in the design. This technique is also easily understood by users, which enhances the communication between users and implementors when identifying requirements and specifications.

An E-R diagram uses the following components to describe the database from an abstract level:

1. Rectangles – indicate entity sets.

2. Ellipses – indicate an entities attributes/characteristics.

3. Diamonds – indicate relationships among other entity sets.

4. Lines – provide links among the above components.

5. Triangles – indicates an ISA (is a) relationship, which is a specialization or generalization of a higher level entity.

6. Arrows – indicate the types of "one" relationships between entities (e.g., many-to-one $\longrightarrow$ and one-to-one $\longleftrightarrow$).

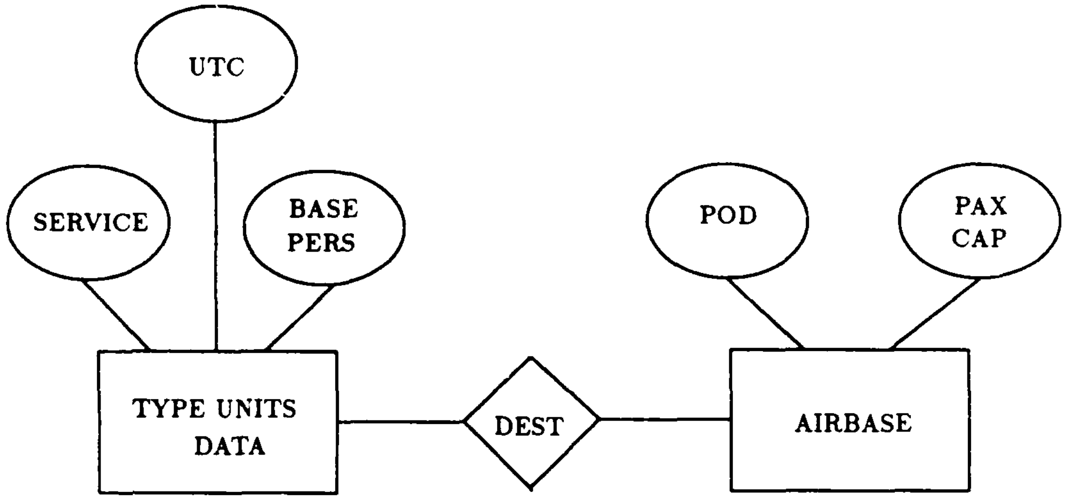


Figure 3. Sample E-R Diagram

Figure 3 illustrates a condensed subset of the JPLAN database consisting of Type Units Data (TUCHA) and Airbase entities along with some of their attributes. The Destination (DEST) relationship represents a Unit Type Code's (UTC) deployment to various bases, which are Ports of Debarkation (PODs). The relationship is many-to-many because a type of unit can be deployed to various bases (an entire unit goes to one particular base but there are *n* instances of that unit type) and a base has many units deployed to it. The graphical representation provided by the

E-R diagram allows the interrelationships among these portions of the JPLAN data to be easily understood.

Of course this technique is much more powerful when dealing with large quantities of data involving many entities, attributes, and relationships. Also, since this method is graphical it allows the designer to detect and eliminate redundant attributes and walk through examples of data access to help conceptualize the functioning of the database. As alluded to earlier, the ability to view the data at an abstract level and not be application dependent greatly increases the flexibility of the database.

Record-based logical models are also used to describe data relationships at the conceptual and view levels. They differ from object-based logical models by their use to describe the basic logical structure and a high level description of the database implementation. These models do not provide the abstract implementation independent view of the database like the E-R diagrams. They represent the methodology used when implementing a Database Management System (DBMS). The three most accepted models are:

1. Relational model (see Figure 4) – data and relationships are described by sets of tables (relations) with columns containing attributes and rows (tuples) containing instances of the data. The relational model is a direct implementation of the E-R diagrams which were used to graphically depict the relationships among the data.

2. Network model (see Figure 5) – data are described by collections of records with links representing relationships. The records are organized as a collection of arbitrary graphs.

3. Hierarchical model (see Figure 6) – similar to the network model except records are organized as trees and not arbitrary graphs.

Relationships among the data items can be one-to-one, one-to-many, many-to-one, and many-to-many. The various models described above support some or all of these relationships. The

Type Units Data (TUCHA)

UTC	*SERVICE*	*BASE PERS*	*POD*
A5223	ARMY	876	BAK
3M1AC	AIR FORCE	552	BUN
3M1AC	AIR FORCE	552	MOL
A5205	ARMY	354	MAX
A5205	ARMY	354	MOL

Airbase

POD	*PAX CAP*
BAK	1396
BUN	2500
MOL	3999
MAX	4000

Figure 4. Relational Database

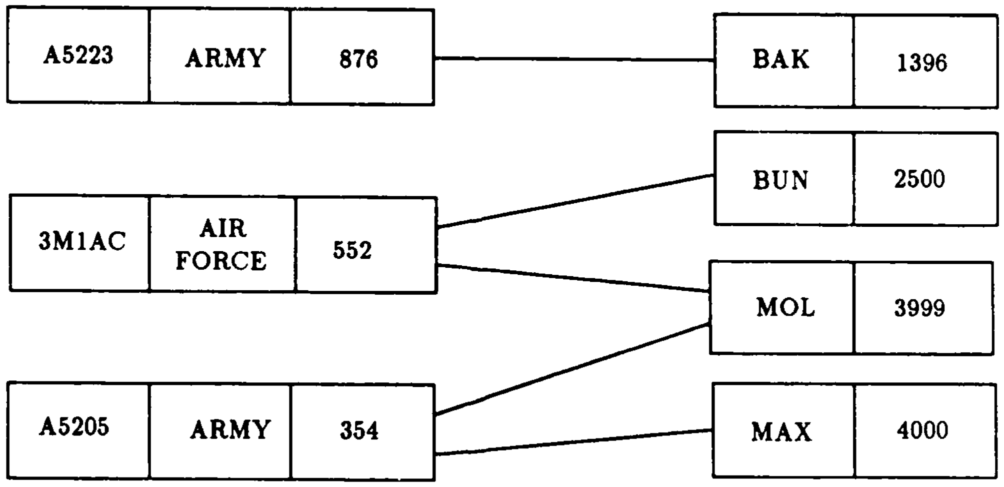


Figure 5. Network Database

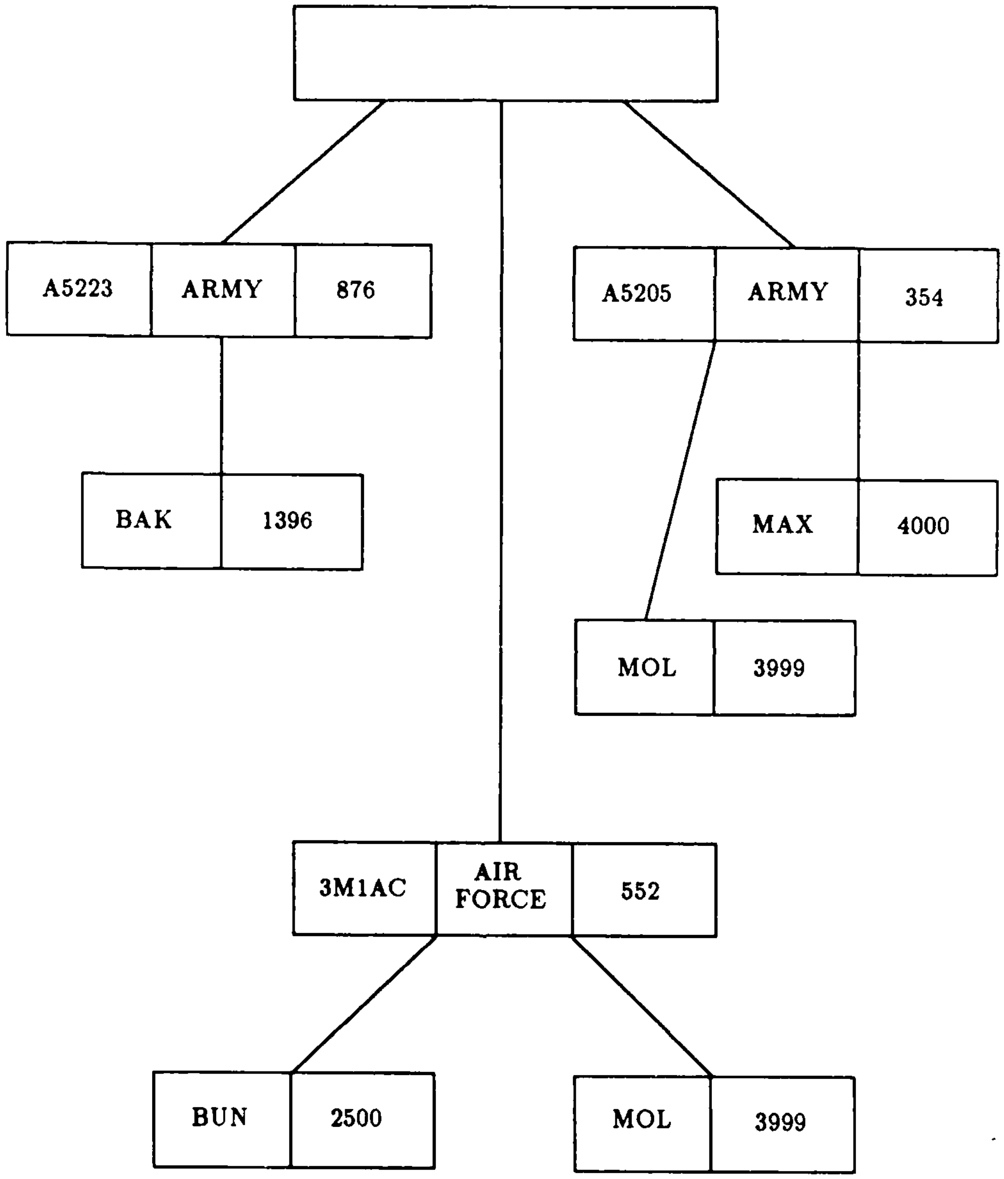


Figure 6. Hierarchical Database

relational model supports all of these relationships. The network model supports one-to-one, many-to-one, and many-to-many relationships between child and parent entities. And the hierarchial model supports one-to-one and one-to-many relationships between the parent and child.

The relational database methodology along with E-R diagrams was chosen for the JPLAN database design for the following reasons:

1. Provides a more logical view of the data and their interrelationships.
2. With the use of E-R diagrams to describe these interrelationships, the user and designer can easily understand the "big picture".
3. Provides a greater degree of symmetry, non-redundancy, and data independence.
4. The resulting design is easier to modify since the original design is not tied to any one application.
5. Relations are easier to understand than records and links.
6. Information is easier to retrieve by using data attributes and relationships rather than access paths as in the network and hierarchical methods.
7. Currently recognized as the best method for database implementation by experts in the field.

2.2 Dialog Design

Since the present JPLAN system has been declared "user hostile", every effort must be made to ensure that the new design is "user friendly". The term "user friendly" is very vague and situation dependent. Which means that what one person considers easy and a pleasure to use, another may have experienced great difficulty in using. For example, computer operating systems commands tend to be cryptic in nature, as a result an expert user may consider a system to be very "user friendly" because he can accomplish his task quickly and with a minimum of key strokes. However, a novice on the same system would probably use the term "user hostile" to describe its use because he must remember short cryptic commands and it takes him a long time to accomplish his task.

With the dilemma of designing systems for experts, novices, or somewhere in-between, a number of different alternatives are present for the designer to consider. Also, for each of these alternatives there are ways to optimize each design. The remainder of this section will focus on what some of the experts (primarily Shneiderman [12]) in the field of user interface design have found to be most effective.

2.2.1 Interactive Styles. The first step to designing the user interface is to identify the user, including his level of expertise and goals. The user's goals can then be broken down to specific tasks. When task identification is complete one of the following interactive styles can be chosen:

1. Menu selection – list of items to select from.
2. Form fill-in – used for data entry by filling in blank fields.
3. Command language – cryptic in nature and usually reserved for expert users.
4. Natural language – use of english-like commands which may require clarification queries if the computer doesn't understand the entry.
5. Direct manipulation – using visual representations (e.g., icons) for task accomplishment which are selected by a pointing device (e.g., mouse or highlighted bar).

When designing a system several of these interactive styles may be used at various points depending on the task type, user profile and design objectives. Table 1 illustrates some of the advantages and disadvantages of each style.

2.2.2 Rules of Dialog Design. By following the principles listed below the resulting system should be easier to use and more robust:

ADVANTAGES	*DISADVANTAGES*
MENU SELECTION	
shorter learning time	numerous menus possible
keystroke reduction	may hamper frequent users
structures decisions	requires a lot of screen space
easy error handling	
FORM FILL-IN	
simple data entry	requires a lot of screen space
little training	
easy assistance	
COMMAND LANGUAGE	
flexible	difficult to learn
good for expert users	requires more training
used to crea . macros	requires more memorization
NATURAL LANGUAGE	
no syntax to remember	may require clarification
	may require more typing
	could be unpredictable
DIRECT MANIPULATION	
presents tasks visually	might be difficult to implement
easy to learn and retain	might require special graphics
lower error rate	and input devices

Table 1. Interactive Styles Advantages and Disadvantages [12]

1. Be consistent – use of terminology, commands, functions keys, placement of text items on the screen, and colors or intensity levels should be the same throughout the system.

2. Build in shortcuts for expert users – use abbreviations, special key sequences, and macros to allow expert users to speed through menus.

3. Consistent feedback – every action by the user should have a response.

4. Closure or completion of events – the user should experience the scenario of INPUT $\rightarrow$ PROCESS $\rightarrow$ OUTPUT for his selections.

5. Simple error handling – don't allow the user to commit a serious error (e.g., crash the system), offer easy error correction, and simple error messages.

6. Allow the user to "Undo" – when possible the user should be able to either recover from an action or confirm the action.

7. Internal "locus of control" – the user should feel like he is in charge of the system.

8. Don't overload short-term memory – keep displays simple, codes and abbreviations should be minimal, and help should be available on-line.

2.2.3 Menu Systems. Menu selection systems are desireable because they eliminate memorization by the user since all alternative courses of action are presented on the screen. Complex command sequences are eliminated because menu items can be selected with one or two keystrokes (e.g., select item "1" press "RETURN").

The following are popular types of menus a designer can use when designing a system:

1. Pop-up or pull down – the menu text is normally placed along the top of the screen with key words or icons used as choices. The user places a highlighted bar, mouse icon, or similar pointing device on the choice to select an option. When activated that portion of the screen expands to reveal other choices, this system can be nested as far as practical.

2. Tree structures – partitions data and choices into similar groups and leads the user through a series of menus to perform the task. Tree structured menus are particularly good for novice and intermediate users.

3. Ring – with this type of menu the selection criteria are arranged along the top of the screen. When a highlighted bar is moved to the selection a brief explanation of that function is given on the next line. An alternate selection method is to have a unique letter, normally the first letter of the word, used to select the choice.

By using a unique and meaningful letter (mnemonic) to make a selection the system can be effective for both novice and expert users. By using the "typeahead" feature expert users can traverse multiple menus quickly without pausing to read the menu because they already know what their choice will be. By using "typeahead" users can not only traverse menus quickly, they can also execute a series of time consuming functions and do something else while those functions execute.

Function keys or other key sequences can also be associated with any of the above menu types for making selections.

For data entry applications form fill-in menus are the logical choice for most applications. The following are some guidelines that will make form fill-in menus easier to use:

1. Meaningful titles – avoid the use of computer jargon.
2. Understandable instructions – be brief, explicit, and use familiar terms.
3. Use logical groupings and field arrangement – follow patterns the user will be used to.
4. Aesthetics – the screen layout should be organized, not cluttered, and "pleasing to the eye".
5. Well known field labels – use labels the users are already familiar with.
6. Be consistent – use the same words and abbreviations for pieces of data throughout the application (e.g., don't use STREET, Street, street, St, etc. for the same data item).
7. Easy field traversal – provide simple method of moving about the fields (e.g., TAB key or arrow keys). Avoid auto-tabbing in most cases.
8. Easy error correction – backspace or type over to correct errors.
9. Meaningful error messages – indicate the type of error (e.g., character instead of integer, too many values, etc.) and give a list of permissible values if possible.
10. Indicate mandatory and optional fields – mark fields with color, brightness, etc. to distinguish fields.
11. Signal completion – avoid automatic completion, remind the user if a probable action has not been taken (e.g., save the information).

Most of these items appear to be common sense, however, may applications violate many of these rules. Reasons vary from too little time spent on designing the user interface to an inability to perform the checks with available hardware and software.

2.2.4 Color In Screen Design. Adding color to screens can add to the screen's useability, however, improper use of color can be worse than not using color at all [6]. Initially screens should be designed for monochrome because color may not be available everywhere the application is run, and the design will focus on logical layout instead of colors [12].

The following guidelines illustrate the use of color in screen designs: [6]

1. Formatting aid – to tie information together, differentiate between data, and emphasize certain fields.
2. Visual codes – can be used to identify kinds, status, and sources of data.
3. Be consistent – always use the same color for data in the same situation.
4. Too many colors – don't try to use different colors just because they are there. Doing so will only confuse the user. Four colors are about the maximum for one screen.
5. Separation – select colors widely spaced along the visual spectrum, these include; red, green, yellow, and blue.

This concludes the discussion of user interface design guidelines. In summary, all of the preceding points are good for the designer to utilize, however, the final determination on screen design comes down to determining who the user(s) is and to what the user wants and the available software and hardware available to implement the design.

III. Dialog and Database Design

3.1 The Dialog Component

To design the screens for the Dialog Component (user interface) a technique called storyboarding was used. Storyboards are used to represent the user's requirements using some type of graphics or form design tool to develop a sequence of displays which will represent a possible solution to the user's problem [9]. A set of storyboards were designed for JPLAN using a graphics tool to illustrate the screens, system flow, and reports necessary to translate the FORTRAN based exercise to a PC based implementation. Figures 7 and 8 represent the tree structure of these storyboards

The Dialog component was determined to be best represented as a Menu Dialog because this type of dialog is effective for inexperienced and infrequent users [9]. Since the users of this system are Logistic's School professional continuing education (PCE) students, and ACSC students, each of these groups (e.g., civilians, NCO's, and Officers) represent users who may be unfamiliar with computers in general. Therefore, the use of a highly structured and sequential menu system is appropriate. Also, this system mirrors the actual JOPS more closely by using the same module scheme.

The users will have a very limited exposure to the system (one day training and two days usage), so they will not gain a great deal of familiarity with it. To help them use and understand the system, each user is given a user's guide describing JPLAN and the scenario of the exercise. The user's guide provides the users a basic understanding of what the acronyms stand for, their definition, and purpose. To answer any additional questions, the course director presents a more detailed explanation of JPLAN and answers any questions the users may have during the training session. The high degree of structure used in the dialog, plus the help screens will further assist the users in learning and using the system.

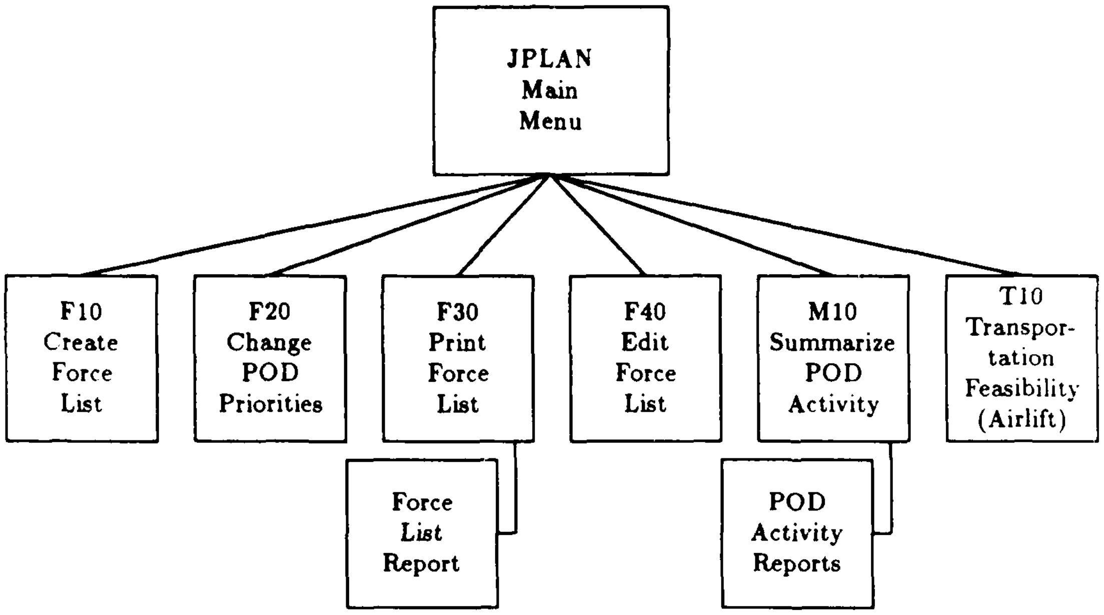


Figure 7. JPLAN Top Level Menu Overview

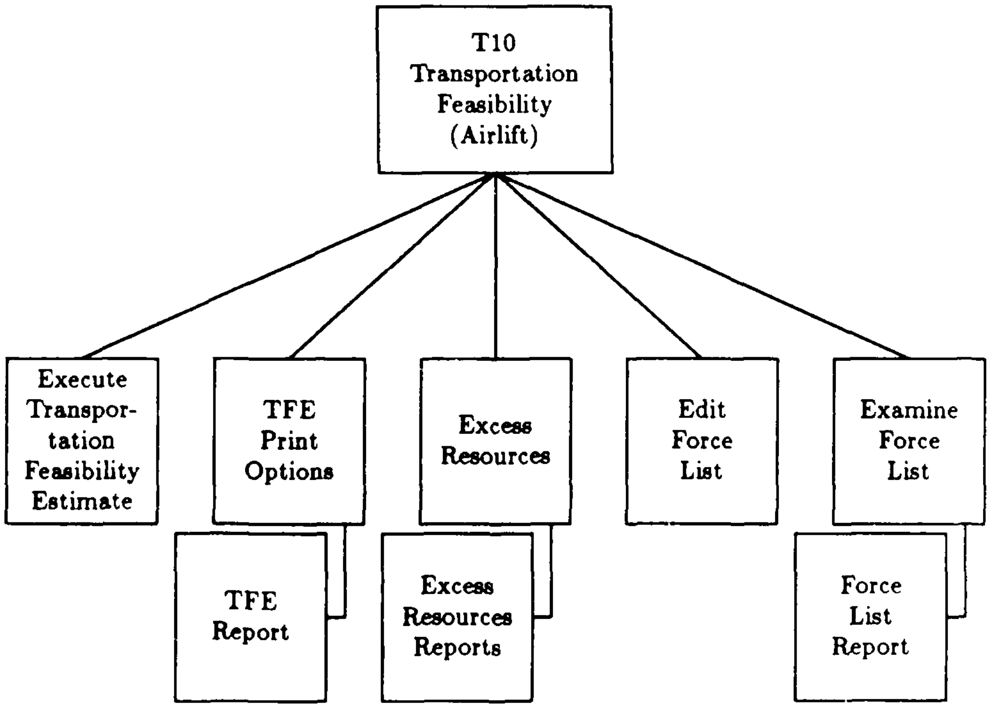


Figure 8. T10 Transportation Feasibility Estimate (Airlift) Menu Overview

To illustrate the method used and make a comparison between the old and new design a sample storyboard will be used, but first the data must be explained. When a user develops his Force List, which is a list of all the units he wishes to deploy, the data items in Table 2 are used to create the list.

In the FORTRAN based implementation all user entries to the system were line oriented, which means that all entries for editing data were limited to one line. FORTRAN uses commas as delimiters for fields, therefore, users also had to insert commas between multiple fields on a line. To help illustrate the improvement of the new screen oriented user interface over the FORTRAN based line entry system the two will be compared. For example, to change Line Number 119 to have a UTC of 4F9F9, a POD of PRM, a EMD of −7, and a LAD of −3 the following line entry in the old system would be required:

```
119,UTC,4F9F9,POD,PRM,EMD,-7,LAD,-3
```

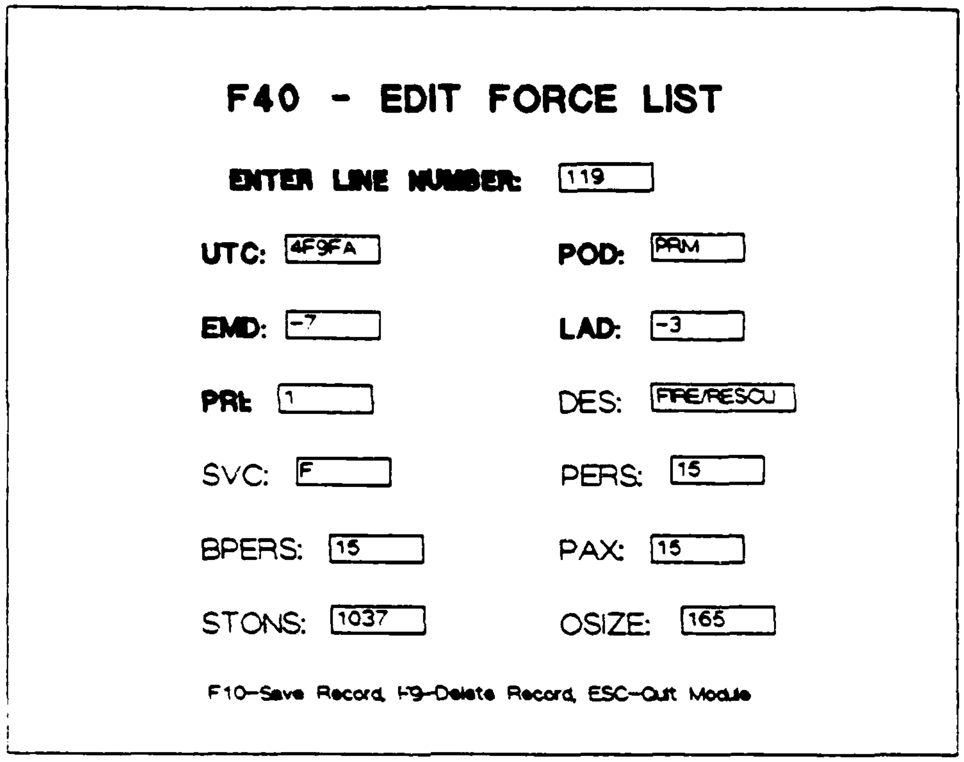
If an error occurred in the line the entire line would have to be repeated. Also, the user had to remember the abbreviations for the fields he wanted to change, because as shown the abbreviation had to precede the new data value. Now one of the storyboards (see Figure 9) will be used to illustrate the improvement obtained by the new user interface design.

When the user brings up the "F40 - EDIT FORCE LIST" screen he will input the line number he wishes to edit and the corresponding information will be presented. The only information the user is allowed to edit are the bold print fields; UTC, POD, EMD, LAD, and PRI because this information affects each deployment of a particular force. Information describing a particular UTC is constant regardless of the deployment date or location and is provided for informational purposes only.

The information across the bottom of the screen (F10 - Save Record, F9 - Delete Record, ESC - Quit) indicates the operations provided by this screen. Error checking is provided on each

ITEM	*DESCRIPTION*
Line Number	Uniquely identifies a particular deployment of a unit. A unit type may be deployed more than once and to more than one POD, because there are *n* instances of this UTC. For example, the UTC 3FJDC describes an F-16 unit, there may actually be 5 of these units stationed at various bases which can be individually deployed. Since the line number uniquely identifies an instance of a force deployment it may not be reused if that deployment is deleted. This is because in "real life" line numbers are used for cross-referencing between commands and services (e.g., Air Force and Army, or Military Airlift Command (MAC) and Strategic Air Command (SAC)).
UTC	Unit Type Code, uniquely identifies each unit type available for deployment and has the attributes of DES, SVC, PERS, BPERS, PAX, STONS, and OSIZE.
POD	Point of Debarkation, an abbreviation for the base to which the unit is being deployed.
EMD	Earliest Movement Date, a date (−10 to 20) chosen by the user for the earliest departure of the unit. NOTE: Dates are in reference to D-Day, which is the day an operation begins on.
LAD	Latest Arrival Date, a date (EMD to 20) chosen by the user for latest arrival of the unit.
PRI	Priority, the relative priority of this force deployment with other deployments.
DES	Description of Force Requirement, gives the user a short narrative on the units purpose.
SVC	Service the unit belongs to: A (Army) and F (Air Force).
PERS	The number of personnel assigned to that unit.
BPERS	The number of personnel that will require base support. For example, for an army unit none of the personnel will stay on base because they will deploy to the surrounding area.
PAX	The number of personnel that must be airlifted to the POD. This is usually the same as PERS, but could be different depending on the type of unit (e.g., some of the personnel may be flying aircraft to that base).
STONS	The short tons (2,000 pounds of generic cargo) of cargo belonging to the unit to be transported.
OSIZE	The portion of STONS that must be transported by C-5A because of size requirements (e.g., Tanks, and other heavy equipment).

Table 2. Force List Data Items



F40 - EDIT FORCE LIST

ENTER LINE NUMBER: 119

UTC:	4F9FA	POD:	PRM
EMD:	-7	LAD:	-3
PRI:	1	DES:	FIRE/RESCU
SVC:	F	PERS:	15
BPERS:	15	PAX:	15
STONS:	1037	OSIZE:	165

F10-Save Record, F9-Delete Record, ESC-Quit Module

Figure 9. Sample Storyboard F40 - EDIT FORCE LIST

field the user must input to ensure the data is valid. If invalid data is entered an error message will be displayed and the user is returned to the field, the user will not be allowed to go to the next field until a valid input is received. Also, if the user tries to quit without saving the data he will be prompted to verify his intentions. Clearly this method is much more "user friendly" than the equivalent entry in the previous implementation.

The other storyboards are quite similar to the sample shown. After these storyboards were developed user feedback was obtained and some minor changes were made. Of course actual appearance, menu placement, and function key assignments were implementation dependent. The use of storyboards to define user requirements was an excellent method and was well received by the users because they could see what the end product would look like. Also, by using storyboards development time is saved because time is not wasted developing what the user does not want or like.

3.2 Database Design

An object level design using entity-relationship (E-R) diagrams was used to develop a conceptual view of the database. This view showed the data items with their associated attributes and the interrelationships between the data items. Then a relational database design methodology was used in implementing the conceptual view of the database.

The first step in designing the database is to identify the objects (entities). The following are the objects referenced in the JPLAN simulation:

- Airbase
- Airlift Aircraft
- Tactical Aircraft
- Ship
- Seaport
- Time Phase Force Deployment Data (TPFDD)
- Type Units Data (TUCHA)
- Aerial Port

3.2.1 Database Data Specifications. After identifying the entities the accompanying attributes should be identified. The variable format (e.g. integer (I), real (R), and character (C)) and length was also identified at this step. Tables 3 – 9 represent the entities and attributes used in JPLAN. *NOTE: Ramp Space = SQ FT × 1000 and Cargo = Tons*

ATTRIBUTES	*FORMAT*
POD Abbreviation	C-3
POD Name	C-12
PAX Capacity	I-6
Cargo Capacity	I-6
Aircraft Accommodation	I-2
MAX Ramp Space	I-6
POD Priority	I-2

Table 3. Airbase Entity

ATTRIBUTES	*FORMAT*
Aircraft Name	C-6
Number Available	I-4
Pass Cap with 0 Cargo	I-4
Pass Cap with Max Cargo	I-4
Cargo Capacity	I-4
Ramp Space	I-5
Pass/Cargo Tradeoff	I-2
Priority of USE For:	
Outsized	I-6
Cargo	I-6
Passengers	I-6

Table 4. Airlift Aircraft Entity

ATTRIBUTES	*FORMAT*
Aircraft Name	C-6
UTC	C-8
Ramp Space	I-5
Number of Aircraft in UTC	I-2

Table 5. Tactical Aircraft Entity

ATTRIBUTES	*FORMAT*
Type	C-2
Number Available	I-2
Sailing Time (days):	
SUEZ	I-2
South Africa	I-2
Cargo Capacity	I-6
Load Rate	I-5
Unload Rate	I-5
Max Draft	I-2
Surface General Cargo Cap	I-1

Table 6. Ships Entity

ATTRIBUTES	*FORMAT*
POD Abbreviation	C-3
POD Name	C-12
Daily Cargo Capacity	I-8
Ship Accommodation	I-1
Controlling Depth	I-2
POD Priority	I-2

Table 7. Seaport Entity

ATTRIBUTES	*FORMAT*
Line Number	I-4
Description	C-16
Service	C-2
UTC	C-7
Mode of Transportation	I-5
POD for Force Requirement	I-5
Force Personnel	I-6
Base Personnel	I-6
Personnel Requiring Trans	I-6
Short Tons of Cargo	I-6
Outsized Cargo	I-6
Earliest Movement Date	I-3
Latest Arrival Date	I-3
Priority	I-3
Feasible Arrival Date	I-3

Table 8. Time Phase Force Deployment Data (TPFDD) Entity

ATTRIBUTES	*FORMAT*
UTC	C-5
Service	C-2
Port Code	C-3
Air Transportable	C-2
Description	C-16
Force Personnel	I-6
Base Personnel	I-6
Personnel Requiring Trans	I-6
Short Tons of Cargo	I-6
Outsized Cargo	I-6
Aerial Port	C-1
Passenger Capacity	I-6
Cargo Capacity	I-6

Table 9. Type Units Data (TUCHA) Entity

3.2.2 E-R Diagrams. Now the relationships between the various entities must be determined and the E-R diagrams can be drawn. The following explanation applies to Figure 10. *NOTE: Asterisks indicate keys for the entities.* By eliminating the duplicate attributes of (Description, UTC, Force Personnel, Service, Base Personnel, Personnel Requiring Transportation, Short Tons of Cargo, and Outsized Cargo) between Type Units Data and TPFDD entities, data redundancy in the TPFDD entity can be eliminated since these attributes appear in the Type Units Data entity. Therefore, the USES relationship exists between the Type Units Data and the TPFDD because the TPFDD uses data contained in the TUCHA to form the Force List. This illustrates one of the many benefits of using a relational database design to reduce data redundancy, enhance integrity, and parallels the real JOPS.

The attribute of transportation mode is implied by the ISA relationship to the Air and Sea TPFDD entity, thus saving space and increasing the logical structure of the data. The POD for Force Requirement is implied by the Destination relationships (Figure 11). Since not every UTC will consist of Tactical Aircraft, this information is best stored in a separate relation to eliminate duplication of information and storage of empty fields (nulls) when the UTC doesn't have Tactical

Aircraft assigned to it. The same is true of the Aerial Port entity. These storage methods also make changes to the database much easier because the data is stored in one place (e.g., to change the passenger and cargo capacities of an Aerial Port one only has to change one tuple, in the old system each force line that used that UTC would have to be edited). Similarly, the Number of Aircraft is an attribute of the Assigned To relationship since a different number of aircraft may be assigned to UTCs with the same type of Aircraft.

POD and transportation resources are represented in Figure 11. The Airlift Aircraft entity represents the various types of aircraft available to transport UTCs by air. The Airbase entity represents all the available Airbase Point of Debarkations (POD) available for UTC deployment. The Allowed At relation represents whether an Aircraft can land at that POD. UTCs from the Air TPFDD have a destination represented by the Air Destination relationship. The Seaport portion is essentially the same as the Aircraft section of the E-R diagram.

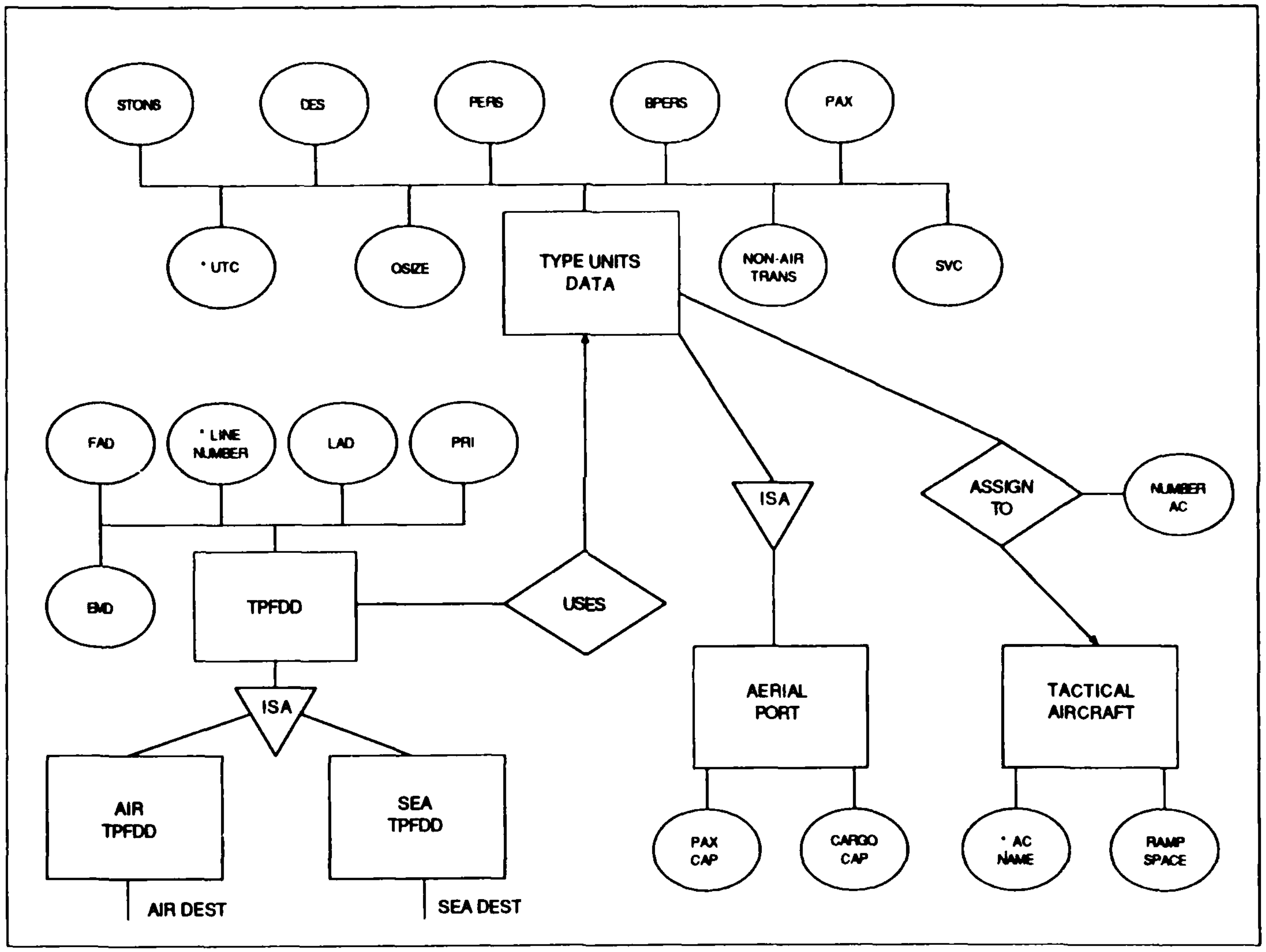


Figure 10. JPLAN/RADEX Database E-R Diagram (Part 1)

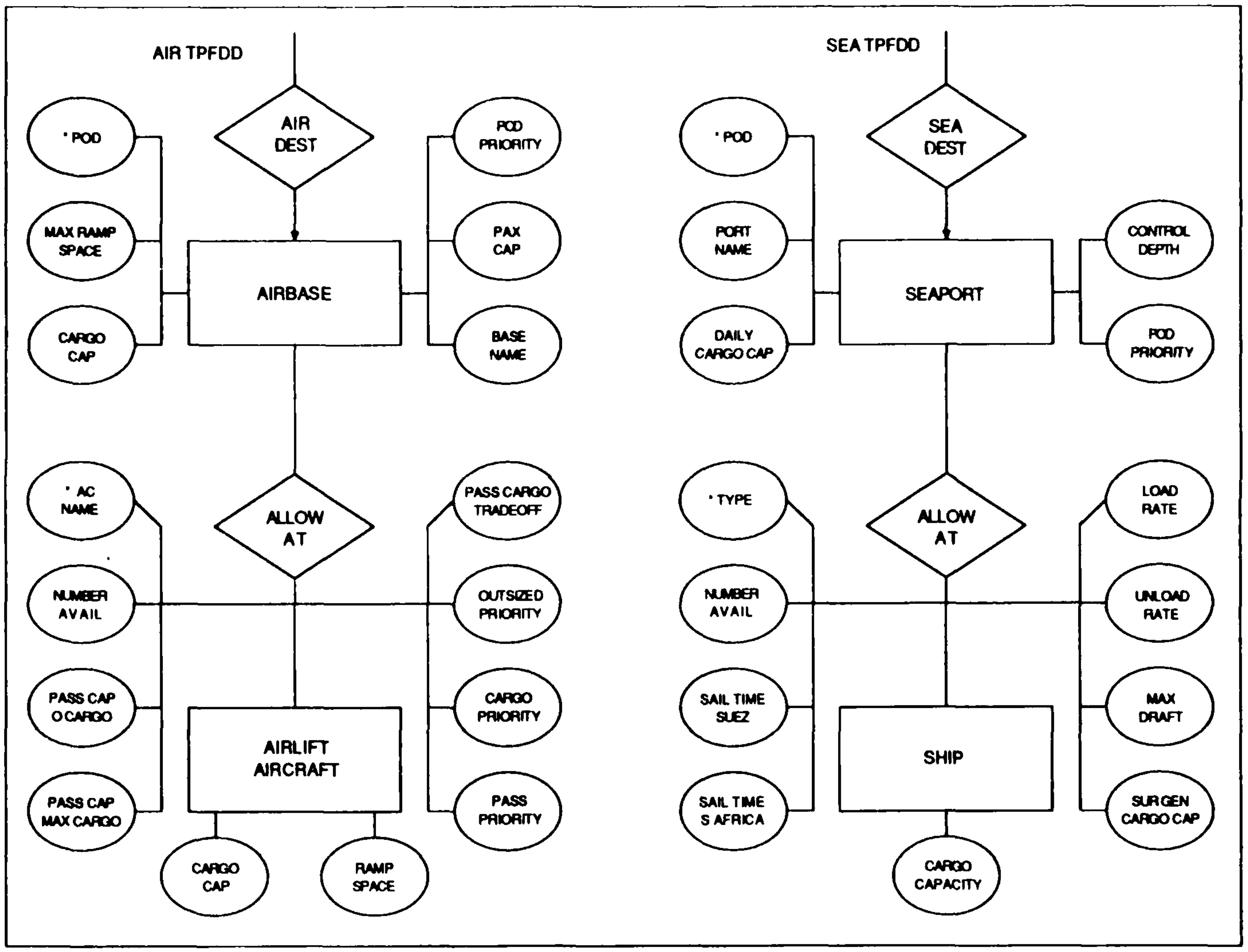


Figure 11. JPLAN/RADEX Database E-R Diagram (Part 2)

3.2.3 Database Relations. The following represent the relations taken from the E-R diagrams which will comprise the JPLAN/RADEX database. The keys to identify information within the relation are in *emphasized* print. NOTE: Tables 12 – 24 are located in Appendix B.

Unit (UTC) deployment information is represented by Tables 10 – 15. Note that the common element to tie these relations together is the UTC, which as stated earlier is one of the benefits of using a relational database schema. Without this structure there would be a lot of information duplicated. Also, one can easily change any of the UTC data using this structure (e.g., change Tactical Aircraft Assignments of a unit from F-4 to F-16 aircraft). These Tactical Aircraft also have certain requirements shown in Table 16.

POD data is represented in Tables 17 and 18. The types of transportation available to move forces are in Tables 19 and 20. These transportation resources have certain POD's they are allowed at, which is stored by Tables 21 and 22. For each day of the exercise there are *n* of these resources available in Tables 23 and 24.

COLUMN NAME	*FORMAT*
UTC	C-5
DES	C-16
PERS	I-6
BPERS	I-6
PAX	I-6
STONS	I-6
OSIZE	I-6
Non-Air Trans	I-6
SVC	C-2

Table 10. Type Units Data (TUCHA) Relation

IV. Implementation

This chapter will describe the implementation of the JPLAN database and user interface. The primary focus will be on the structure of each component and an overview of how the component was implemented. The MDBMS used to implement JPLAN was chosen by the Air Force Wargaming Center. The database system chosen was PC INGRES by Relational Technology. PC INGRES was chosen because the Wargaming center currently uses a mainframe version of INGRES, therefore, the personnel training required to maintain JPLAN would be minimal. Also, lower costs were associated with certain configurations of INGRES, and other projects under development were using INGRES.

Prior to proceeding with the implementation discussion a quick review of the development tools provided by INGRES is in order. The following are summaries of the various capabilities provided by PC INGRES [3]:

- INGRES Menu - the main menu which ties all of the components together and allows the developer to create an entire application through the functions provided.
- 4GL - this is a PASCAL like high-order language specially designed to work with the other INGRES components to speed application development.
- Visual-Forms-Editor (VIFRED) - a screen oriented editor which allows the developer to create the screens (i.e., implement the storyboards) the user will see. Specifically it provides data field editing, field placement, error checking, color specification, etc. for the form.
- Applications-By-Forms - provides an integrated environment to develop and test the 4GL to drive the application.
- Report Writer - allows the developer to format reports and retrieve data to be included in those reports.

- Interactive Structured Query Language (SQL) – provides an interactive mode for entry of SQL statements. SQL is a standard language designed for querying database tables.
- Embedded SQL – allows the use of SQL in a programming language to perform database queries. The language currently supported is Microsoft C version 4.
- Query-By-Forms – an interactive menu driven system for editing database data, running simple reports, etc.

4.1 *Database Implementation*

Once the relations have been defined the database implementation becomes a purely mechanical process of typing in the table definitions. The following is a summary of how this is accomplished in INGRES.

First, the database must be created. When INGRES creates a new database, required directories are automatically made, and a set of system tables are created for managing the database. Then data tables described in Chapter 3 were created using the INGRES Menu "TABLES" option (see [7] for descriptions of the INGRES data tables). When creating tables one must choose a data type for each field (i.e., integer, real, and character). In INGRES this equates to the following representations [3]:

- Integer – integer1 is 1-byte with a range of values -128 to +127, integer2 is 2-bytes with range -32,768 to +32,767, and integer4 is 4-bytes with range -2,147,483,648 to +2,147,483,647.
- Real – float4 is 4-bytes from 8.43×10^{-37} to 3.37×10^{38}, and float8 is 8-bytes from 4.19×10^{-307} to 1.67×10^{308}.
- Character – vchar(n) is a string of up to 2,000 ASCII characters.

When a table is created the default storage structure is "Cheap – Compressed Heap", which is an unordered random storage structure where new data is added at the end of the file with

TABLE NAME	*STORAGE STRUCTURE*
ab_ac_accomm	Cbtree
aerial_port	Cbtree
air_tpfdd	Cheap
airbase	Cbtree
al_ac	Cbtree
al_ac_avail	Cbtree
defaults	Cheap
ex_air_res	Cheap
ex_pod_res	Cheap
m10pod	Cheap
m10tot	Cheap
seaport	Cbtree
sea_tpfdd	Cheap
ship	Cbtree
ship_accomm	Cbtree
ship_avail	Cbtree
tac_ac_utc	Cbtree
tac_aircraft	Cbtree
tfe	Cheap
tucha	Cbtree

Table 11. Storage Structures Used For Database Tables

trailing blanks removed [3]. Storage structures in PC INGRES are Cheap and Cbtree, Cbtree is a compressed binary tree with trailing blanks removed. The first row automatically becomes the key or primary index field when the table is created, additional indexes can be created as desired. All index tables are stored as compressed binary trees [3].

In general a heap is best for a table that will experience a large amount of activity (i.e., additions and deletions) because of the low overhead required to maintain pointers. Conversely a binary tree structure has high overhead involved in maintaining pointers to records as they are added and deleted. Tables that are primarily "read only" in the JPLAN database are stored as compressed binary trees and tables used for repeated updates and deletions are stored as compressed heap (see Table 11).

4.2 Dialog Implementation

The Dialog implementation was the largest portion of this effort, and consists of forms for user input, the 4GL to drive those screens, and reports to format output data. Screens were implemented in VIFRED directly from the storyboards created in Chapter 3 (see Figure 9 for a sample storyboard). In most cases the storyboard implementation was the same as the storyboard, with minor cosmetic changes in some cases (e.g., boxes around fields or highlighting of fields).

INGRES provides a ring menu system at the top of the screen where each selection can have a function key assigned to it (10 keys are available). Therefore, menu placement was changed from the bottom of the screen as depicted in the storyboards to the top of the screen in a ring menu format.

This method provides the user with the following ways to choose an option from the ring menu:

1. Press the function key associated with the choice.
2. Press function key F2 to activate the ring menu, the cursor will go to the ring menu and the highlighted bar will be on the first choice and an explanation of that choice will be displayed below the menu. Now the user can use the arrow keys to move the highlighted bar from option-to-option and select by pressing RETURN when the highlighted bar is on the option he wishes to execute.

The following is a summary of the steps involved in implementing the Dialog Component in PC INGRES:

1. Create the Application - an application is a set of forms (screens), 4GL, and reports to be run against a database.
2. Create Forms - use the VIFRED to create the storyboard layout on the screen. Select field attributes (e.g., color, display intensity, boxed, etc.) and specify error checking on fields.

3. Write 4GL – use a text editor to write the 4GL code necessary to display the form, initialize fields, call other forms and applications, perform menu operations (e.g., save, delete, help, etc.), database computations, and error checking not provided by VIFRED (e.g., checking a UTC the user inputs against the database).
4. Write Reports - use the Report-Writer to create report specifications to satisfy the various reports required by the modules.
5. Test – perform testing on each module as development progresses.

4.2.1 Implementation Decisions. As stated in Chapter 2 screens should be designed for monochrome first and then color. To comply with this guidance display intensity was increased and boxes were used on fields which are more important to the user (e.g., menu titles and fields the user must fill in). The four recommended colors (red, green, yellow, and blue) were used to differentiate between categories of information. All menu titles are in red to catch the user's attention and let him know where he is, menu options and fields that require user input are green, blue and yellow are used for other information throughout the various screens (For example, on the F10 - CREATE FORCE LIST screen the title is red, current line number is yellow, information entered by the user is green (UTC, POD, EMD, LAD, PRI), and information associated with the UTC is blue (DES, PERS, etc.).

Due to memory limitations JPLAN had to be broken up into several applications which are called by the main menu application. Since each of the applications called is exited when control is returned to the calling application there are at most two applications in memory at any one time (i.e., the main menu and the called application).

The storyboard for the "F20 – Change POD Priorities" module called for the user to input the POD he wanted to change the priority for, edit the priority and then save the change. To save the user's time and make this module easier to use, the PODs and their priorities are arranged in a table on the screen (see Appendix A) which allows the user to scroll through and make changes.

This allows the user to see up to 6 PODs at one time and scroll through the rest, rather than one-at-a-time in the original design.

The original storyboards called for graphs of excess resources and shortfalls in the T10 module, however, the PC version of INGRES does not have any graphing capability. Graphs could be accomplished by exporting data to another package such as Lotus 1-2-3. To export data the Visual-Query-Language (VQL), which is another INGRES module, would have to be purchased.

At any time the user can return to the previous screen by pressing F10. In cases where the user may have made changes to data or entered new data he is prompted to verify his intentions before exiting to the previous screen. For example, if the user is editing the POD priorities in the F20 module and decides not to save the changes he has just made, he will press F10 to quit, then he will be prompted to verify quitting, if he answers Y (Yes) he will be returned to the previous menu without the changes being saved.

After all of the applications have been developed and tested image (executable) files are made. This enables the developer to provide a "run time" only package to the end user, thus preventing them from accessing other database utilities to perform functions other than those provided by the application.

V. Transportation Feasibility Simulation

5.1 Simulation Description

The exercise structure and how the simulations, database, dialog, and user interact are shown in Figure 12. Note that the user does not interact directly with the models but uses the models as an extension of the database. This restriction exists because of the classroom environment the model is used in. To insure each group is using the same criteria, no group is allowed to manipulate the model in any way. The only interaction the users have with the model is through the dialog and any changes in the input parameters they make.

The simulation portion of the JPLAN exercise is the Transportation Feasibility Estimator (TFE). A TFE exists for both airlift (JPLAN and RADEX) and sealift (RADEX only), with the following description being oriented towards airlift because sealift is essentially the same (i.e., identical methodology and purpose). The TFE takes the various unit logistics data (i.e., personnel and materials) and simulates the movement of those forces from their Point of Embarkation (POE) to the Point of Debarkation (POD) by using several types of airlift and sealift resources made available. Each UTC (Unit Type Code) is assigned an Earliest Movement Date (EMD), Latest Arrival Date (LAD), and Priority (PRI). This data along with POD and transportation data are used to simulate deployment and compute a Feasible Arrival Date (FAD). The FAD is a key result of the model because it indicates the feasibility of the current transportation plan. If all units arrive on time (i.e., each FAD on or before its LAD), the plan is considered transportation feasible [2].

Any late arrivals are shortfalls and may be resolved by the adjusting the EMD, LAD, or Priority, moving the UTC to another POD, or changing the POD Priority. Other considerations include plan requirements for force deployment and available resources (i.e., available aircraft, ships, ramp space for aircraft, and port capacity for ships) at the POD.

More specifically the JPLAN simulation must perform the following steps one day at a time between the start and end days specified in the T10 Module: [1]

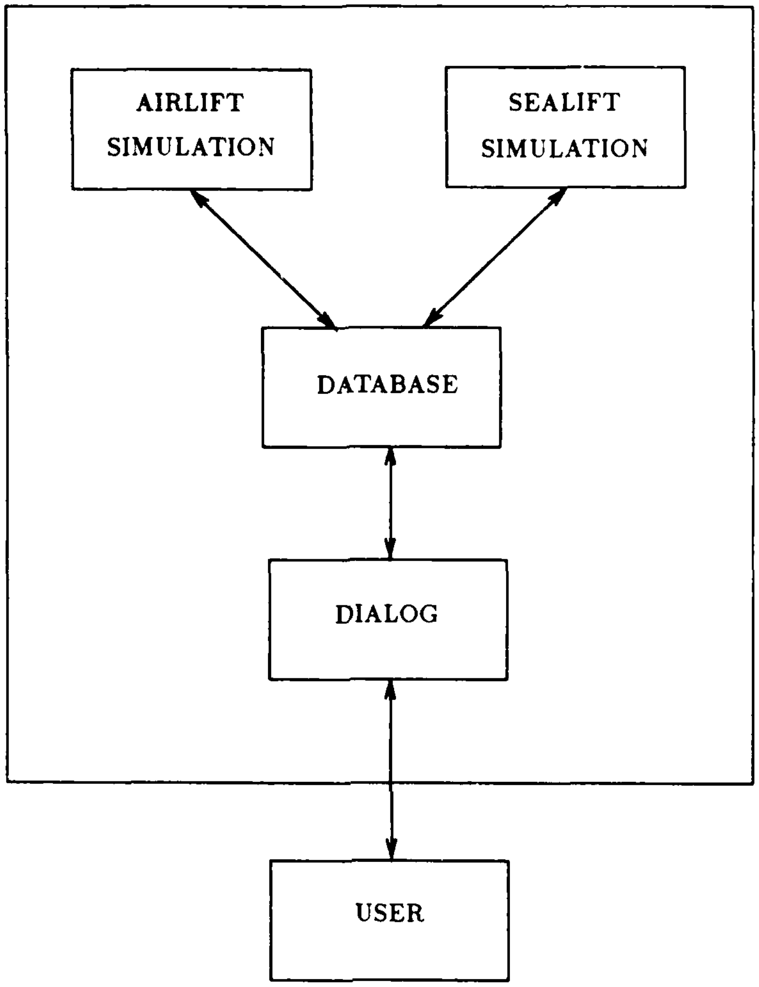


Figure 12. Data Flow - System Integration Diagram [9]

1. Sort the Force List by LAD, UTC priority within LAD, and POD priority within UTC priorities.
2. If the force has Tactical Aircraft, ramp space is deleted at the POD (if available). This is because TAC aircraft fly to the base themselves, so ramp space is deleted up front.
3. Assign C-5As to carry outsized cargo.
4. If these C-5As are not full do the following in order·

   (a) Move outsized cargo from other UTCs with the same POD.

   (b) Move short tons of cargo from UTCs with the same POD.

   (c) Move passengers from UTCs with the same POD.

5. C-5As have space for a small number of passengers in addition to a maximum cargo capacity which must can be filled if passengers are available.

6. Assign C-141s to carry a UTC's short tons of cargo.

7. If all C-141s assigned are not full do the following in order:

   (a) Move short tons of cargo from UTCs with the same POD.

   (b) Move passengers from UTCs with the same POD.

8. After the preceding steps have been accomplished additional attempts are made to use any available aircraft resources which have not yet been assigned to move cargo and passengers left. The previous steps concentrated on moving cargo, and passengers only on those C-141s not filled with cargo. Now these steps are taken to move any remaining cargo that is not outsized by C-5A and remaining passengers:

   (a) Move short tons of cargo (other than outsize) with C-5As. Note: This differs from 4(b) because these C-5As do not carry any outsized cargo.

   (b) Move passengers with C-141s.

   (c) Move passengers with C-5As.

9. If any passengers remain use CRAF aircraft (any other available aircraft) to move passengers until CRAF resources expire.

This completes the simulation of one day's transportation of forces, this process is repeated for each day of movement. Upon assigning a load for a UTC to an aircraft the following checks are performed against data in the database:

1. A load cannot be assigned to an aircraft prior to the UTC's EMD.

2. The type of aircraft must be allowed at that POD.

3. The aircraft resources must be available.

4. Ramp space at the POD must be available.

5. The daily POD capacities for cargo and passengers must not be exceeded.

The following considerations apply to ramp space at a POD:

1. Once a unit with Tactical Aircraft deploy to a POD the ramp space required by those aircraft is decreased for remainder of the exercise.

2. When an airlift aircraft (e.g. C-5A, C-141, or CRAF) lands the ramp space required by that aircraft is taken for the entire day.

The preceding explanation applies to the way JPLAN is presently played, the actual implementation of the simulation is not written for specific aircraft, PODs, etc. The actual program operates on rules for allocating resources and the resource attributes like priority of use for outsized cargo, short tons of cargo and passengers. Therefore, one can add additional types of aircraft with different priorities for use and aircraft capacities. The program will use those additional aircraft using the same rules described above.

In summary, the simulation allocates transportation resources (i.e. aircraft and ramp space) to move selected forces into the theater of operation according to the base and unit priorities selected by the student. The primary limiting factor in the simulation is the amount of ramp space available at each POD to park aircraft. Since the student normally has plenty of aircraft available, the solution lies in the arrival timing, priorities, and reallocation of forces (Army are the largest) among the PODs.

5.2 *Simulation Implementation*

As stated earlier the old JPLAN simulation was written in FORTRAN IV. Since FORTRAN is not supported by PC INGRES the simulation was rewritten in Microsoft C, which is the only language supported by PC INGRES at this time. By using C the use of embedded SQL is also made available to perform database accesses.

Throughout the code conversion process the basic structure (i.e., subroutines and procedures) was maintained to speed the conversion process, ease the transition of code maintenance personnel at the Wargaming Center, and assist in the simulation verification process between old and new. In the FORTRAN simulation data checks were performed by using indexes to arrays and by storing information in certain "words" of memory. Also, any character data items were converted to integer indexes (e.g., the POD would be found by an integer index instead of the character string BAK).

The new implementation eliminated word indexing to make the simulation less machine dependent. All information is read from the database using embedded SQL statements to read the data into arrays of structures (records). As the simulation executes any information required is retrieved from these arrays using indexes. Indexes for information in arrays are set by doing a sequential look-up for a match on the appropriate field. As an example, when a Force List line is looked at, the program must determine if the UTC has Tactical Aircraft associated with it. To determine this, a subroutine is executed to try and find a match on UTCs between the Force List array and Tactical Aircraft UTC array. If a match is found the index is set and the program can determine how many aircraft the UTC has and how much ramp space they will require.

When the simulation has completed, information required to produce post simulation reports is written from the arrays to the database. This information includes the percentage of tons and passengers delivered by the LAD, the FAD of the Force List line, and all of the excess resources (i.e., aircraft, ramp space, passenger capacities, and cargo capacities) at the completion of the simulation.

In the old simulation the user had to enter the starting and ending day of the simulation (e.g., -10 and 15). According to the course administrators this caused some confusion for the students, because one would normally want to run the simulation for the duration of the time period that forces are moving. Therefore, a flag was added to the "defaults" table (see [7]) in the database to allow the course administrator to have the option of activating the starting and ending days.

VI. Testing and Validation

The most important part of any software project is the testing and validation phase. If the end product does not solve the problem or operate properly, the project has been a failure. Also, one should not wait until the end of the project to perform testing and validation. If the design will not meet the users' needs, or serious implementation problems exist, they must be corrected immediately if the project is to succeed. Therefore, during the development of JPLAN testing and validation was accomplished parallel to the implementation process.

6.1 Dialog Component

The dialog component was compared with the old JPLAN to ensure the same functions and operations were available to the user and the overall purpose of the JPLAN exercise was maintained. Some error checking is automatically provided by INGRES (e.g., field type checking) and those checks related to JPLAN were either performed in 4GL or the Visual-Forms- Editor. Specifically the following areas were tested to ensure the integrity of the new JPLAN exercise:

1. Standard Errors – these include entering characters in numeric fields, entering values to test the upper and lower bounds of fields, and making a blank entry.
2. Invalid Data – some of these errors are covered by standard errors, however, in the context of JPLAN this refers to testing invalid UTCs, PODs, etc.
3. Calculations – the M10 module calculates the movement of passengers and tons of cargo at each POD for each day in the exercise. This data was validated by comparing a run from the old and new systems which exercised all values that could be computed by this report.
4. System Crash – this area is automatically covered by the INGRES database. When the system is turned off or rebooted during the middle of running an application INGRES automatically rebuilds any tables that were affected by the crash. Messages are displayed by INGRES

throughout this process to inform the user on what is occurring. This function was thoroughly exercised throughout the development process and no data was lost.

6.2 Simulation

6.2.1 Problem Areas. During the simulation testing and validation phase a few inaccuracies were discovered between the way the simulation was operating and the "real world". The following paragraphs describe the problems found in the old simulation program.

The old simulation moves outsized cargo twice, because short tons (STONS) of cargo is a combination of regular cargo and outsized (OSIZE) cargo. During the simulation OSIZE cargo must be moved by C-5A and the rest of the cargo is moved by C-141s, then C-5As are used to carry the cargo once all C-141s have been committed. The old simulation moved all the OSIZE cargo and then all of the STONS of cargo. Therefore, OSIZE cargo was moved twice, once as OSIZE and once as a part of STONS. For example, if STONS is 154 and OSIZE is 11, the old simulation moved the 11 tons of OSIZE and 154 STONS of cargo, the simulation should move 11 tons of OSIZE cargo and 143 STONS of cargo since OSIZE is a subset of STONS. The effect on the simulation was that more aircraft than necessary were used to move STONS of cargo. The problem was corrected by subtracting OSIZE from STONS when the data is read from the database into the simulation arrays. This does not affect what the student sees on any of his reports showing STONS and OSIZE cargo (i.e., he still sees 154 STONS and 11 OSIZE).

The next problem deals with Tactical Aircraft assigned to a UTC. When a UTC has Tactical Aircraft assigned, the ramp space required by those aircraft must be accounted for prior to moving the force, because, if those aircraft are not allowed on the POD or there is not enough ramp space for them, the force cannot be delivered. The old simulation only accounted for Tactical Aircraft when the force line was initially looked at for assignment of aircraft to move the force. However, if the force line was looked at to fill aircraft capacities remaining from a previous line's aircraft assignment, Tactical Aircraft were not checked. Therefore, if a force line was completely moved by filling aircraft from previous lines Tactical Aircraft would never be accounted for. The effect on the simulation was that not all ramp space for Tactical Aircraft was accounted for. The problem was corrected by adding the code to account for Tactical Aircraft in the subroutine which attempts to fill excess space on aircraft.

Two other areas where the exercise could be changed were discovered. These did not involve errors in the program, but areas that could be changed to make the exercise simulation more realistic. By making the exercise more realistic, problem resolution (e.g., shortfalls and force timing) may become more difficult without enhancing learning objectives. The two areas involve Aerial Ports and the rule for adding additional Airlift Aircraft to move a Force List line.

An Aerial Port is a UTC which includes the equipment and personnel necessary to support off-loading a certain amount of passengers and cargo (e.g., 1100 tons/day and 2500 passengers/day) from Airlift Aircraft. When the exercise starts the PODs are bare bases (a bare base only has a runway and a few buildings), therefore, personnel and equipment necessary to off-load aircraft must be delivered prior to other forces. In the current exercise these capacities are preloaded for each POD, and the course administrator must check to see if appropriate UTCs (Aerial Ports) were sent prior to other forces. This could be changed to start each POD with a zero capacity and a maximum capacity, then students would have to ensure that the first UTCs to arrive were Aerial Ports to provide off-loading capacity for the other forces. The users decided not to incorporate this change because doing so would make the exercise more difficult and not enhance learning objectives.

The last area involves the rule for adding additional Airlift Aircraft. When a Force List is assigned aircraft to move cargo or passengers the following procedure is followed:

1. Divide the amount to be moved by the aircraft's capacity, which equals the number of aircraft.
2. Then if the remainder is greater than 1/1000th (.001) of the aircraft's capacity another aircraft is added.

3. For example, if the amount to be moved is 201 STONS and the aircraft's capacity is 200, the remainder is 1 and another aircraft would be added because 1 is greater than 0.20 (.001 × 200).

By using .001 another aircraft is always added when there is a remainder, as long as the aircraft's capacity is greater than 1, which is true of any aircraft used. Two possibilities exist for this added aircraft; if no other Force List lines meet the criteria (i.e., EMD $\leq$ current_day, the same POD, and the appropriate item to move) the aircraft will only transport the 1 ton or passenger, or the aircraft is filled to some degree with other Force List lines meeting the criteria. A detailed analysis was not performed, however, this rule appears to waste aircraft resources. Also, in reality an aircraft would not be utilized to deliver a small quantity of passengers or cargo. If this rule were changed to only add another aircraft when at least 50% of an aircraft's capacity were needed the number of shortfalls may increase and further complicate the exercise. Since the use of aircraft resources is transparent to the student and changing the rule would only complicate the exercise and not enhance learning objectives, the users decided to use the old rule of 1/1000th of an aircraft.

6.2.2 Validation. As stated in Chapter 1 if the new simulation results were the same as the old simulation results for identical inputs, the new simulation would be considered valid. Since some problems were discovered in the old simulation the validation of the new JPLAN simulation was accomplished in two parts.

First, all errors were included in the new simulation so the logic of the two programs was identical. Then a sample input set of 145 Force List lines [1, pages 89-92] which exercise every aspect of the simulation was run on both systems. Transportation Feasibility Estimate and Excess Resource reports from the T10 module were compared and were identical

Then, the two errors (i.e., moving OSIZE cargo twice and accounting for Tactical Aircraft) described earlier were fixed. The new simulation was run and the values which changed (primarily ramp space due to Tactical Aircraft) were verified by "hand calculations". Fixing the error of moving OSIZE cargo twice only involved one statement to subtract OSIZE from STONS, therefore, a detailed verification was not required.

VII. Conclusions and Recommendations

7.1 Conclusions

This thesis has described the JPLAN Exercise and how the exercise is used to model the JOPS. The use of JPLAN for education at ACSC and AFIT was discussed, including some of the problems of the current implementation. The problem was specified along with the results to be obtained from this thesis, and the following objectives were accomplished:

1. Database design and implementation.
2. Dialog component design and implementation.
3. Monitor program for course administrators.
4. Simulation description and code conversion.
5. User instructions (help menus) were written.
6. System design and installation documentation.

Since the focus of the thesis was the Database and Dialog Design of JPLAN, a review of database methodologies and user interface design guidelines was given. Database advantages included reduced redundancy, removal of inconsistencies, data sharing, centralized control, enhanced security, and increased integrity. The concept and importance of data abstraction was reviewed, with emphasis on removing implementation details from the user's environment.

Data models used for data representation were introduced. The object-based logical model called the Entity-Relationship model was described using example entities and attributes from JPLAN. The relational, network, and hierarchical record-based models were described using sample data from JPLAN.

User interface guidelines included; types of interactive styles, menu selection criteria, rules to follow when designing screens, and the most effective ways to use color in screens were given

Some key points were to use menu driven systems for first time or infrequent users, design screens to yield closure, and design for a monochrome system first.

Storyboards were then used to represent the user's requirements and give a representation of what the new JPLAN exercise might look like. These storyboards were presented to the user for review and required changes were made. A sample storyboard was used to compare the old and new JPLAN systems. A relational database design was chosen to store the JPLAN data. The design process was described from identifying the data items up to making the database tables from the E-R diagrams.

The MDBMS chosen by the users was PC INGRES. An overview of PC INGRES was given prior to describing the implementation of the Dialog Component and database. To implement the new design database tables were created, storyboards were converted to computer screens, and the 4GL programs and reports necessary to tie them together were written. A number of implementation decisions and rationale for each were given.

The Airlift Transportation Feasibility simulation was described in detail along with how the simulation, database, and dialog fit together. An overview of the code conversion process was given with some of the implementation decisions to utilize a database system to store the data.

After all components were implemented the entire system was tested to ensure the operation paralleled the old JPLAN exercise. Some problems discovered in the old simulation were discussed, with reasons for correcting or not correcting them. Finally, criteria and methods used to validate the new JPLAN exercise were explained.

In summary, the new JPLAN exercise is more "user friendly" than the old exercise, runs on a IBM PC or compatible computer, and has become much easier to modify as requirements evolve.

7.2 Recommendations

As with any system or program change and evolution are continuous. Since the purpose of this thesis was to redesign JPLAN to run on a PC, time was not available to design and implement some other improvements to JPLAN. Now that the framework has been built, extensions can be added and JPLAN can be enhanced.

The next step in the development of JPLAN is to build interfaces to other exercises used by the Wargaming Center. This is made relatively easy because of the relational database structure. Since all information is easily retrieved and manipulated through the use of the INGRES toolset the "hooks" to attach other exercises are already present.

One of these exercises is FAST STICK, this exercise involves the application of Tactical Aircraft planned by JPLAN. The following are some of the objectives of FAST STICK: [1, pages 155-157]

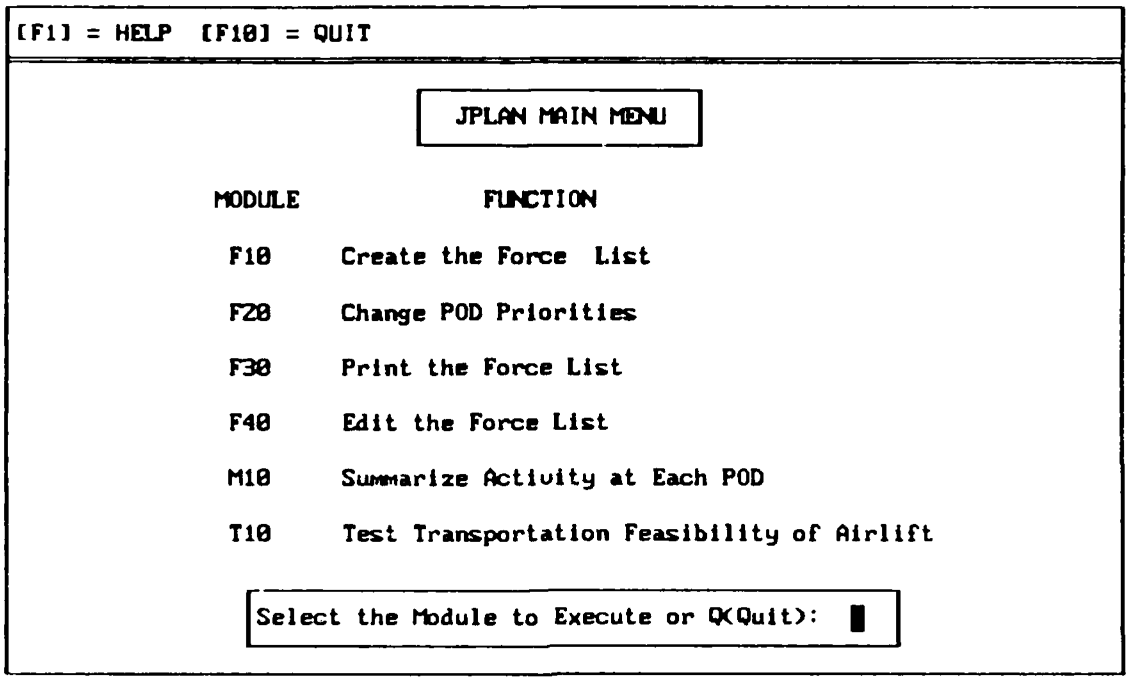
1. Plan and support attack and reconnaissance sorties.
2. Select air defense and close air support aircraft to counter enemy attacks and provide support to ground units.
3. Revise target selections to assure a successful operation.

Another high priority extension is to automate the evaluation done by the course instructors at the completion of each seminar. Some of this checking involves verifying that the students have sent the appropriate UTCs to each base, along with the proper support for a unit when required (e.g., the students can send F-16 support for a wing of F-4s). Another function could be an analysis of the shortfalls (i.e., number of days, tons, passengers, etc.). This extension would probably be in the form of a model involving a set of rules to check the students data files.

An ongoing consideration is to keep the exercise current with the "real world" planning system. As modifications are made and new systems are developed their impact on the way JPLAN is played must be evaluated and incorporated into the exercise.

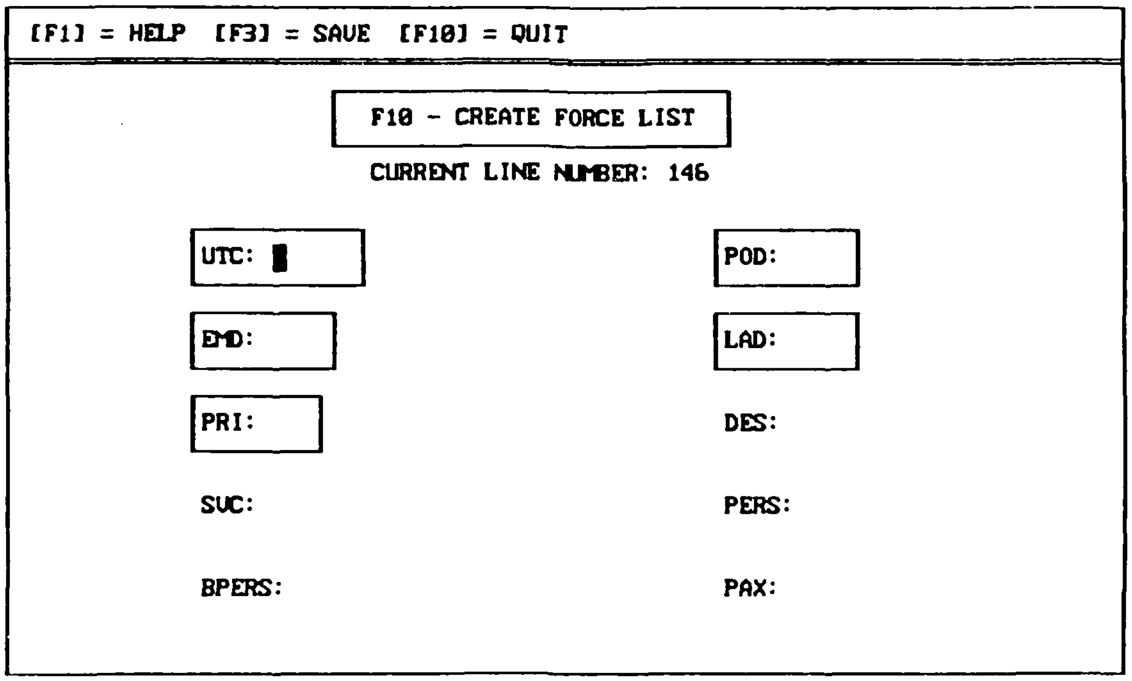
Appendix A. *JPLAN Screen Displays*

The following figures represent the actual screens used in the JPLAN exercise.



```
[F1] = HELP  [F10] = QUIT

                    JPLAN MAIN MENU

        MODULE              FUNCTION

         F10     Create the Force  List

         F20     Change POD Priorities

         F30     Print the Force List

         F40     Edit the Force List

         M10     Summarize Activity at Each POD

         T10     Test Transportation Feasibility of Airlift

     Select the Module to Execute or Q(Quit):
```

Figure 13. JPLAN Main Menu



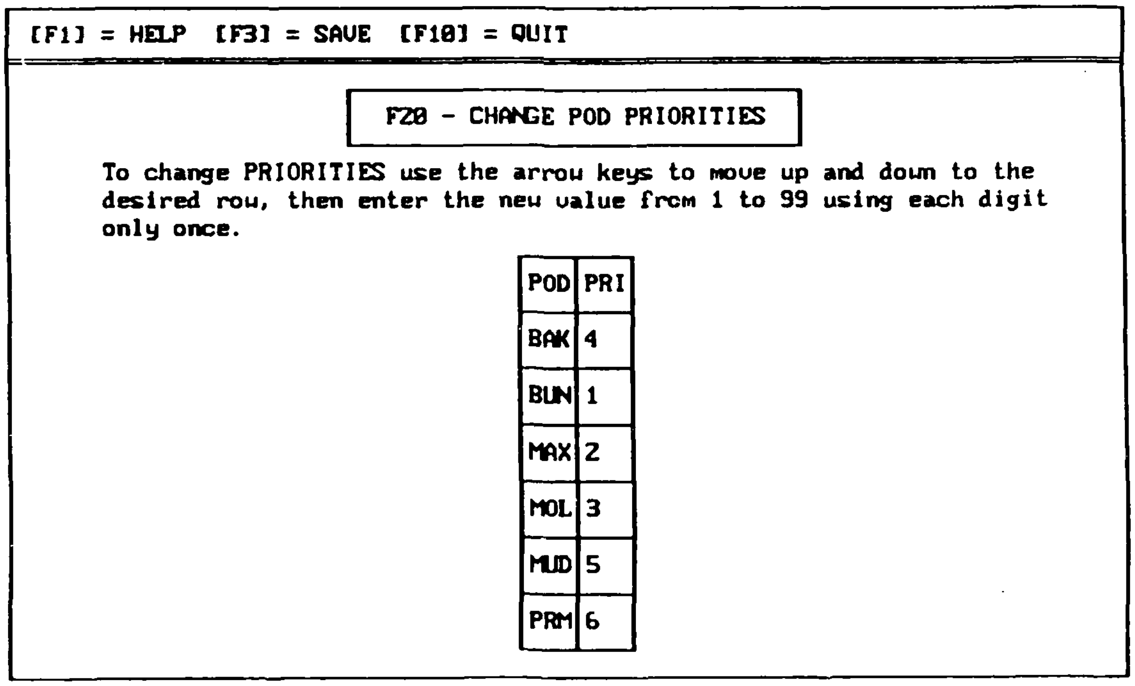
[F1] = HELP [F3] = SAVE [F10] = QUIT

F10 - CREATE FORCE LIST

CURRENT LINE NUMBER: 146

UTC: █	POD:
EMD:	LAD:
PRI:	DES:
SVC:	PERS:
BPERS:	PAX:

Figure 14. F10 - Create Forcelist Menu



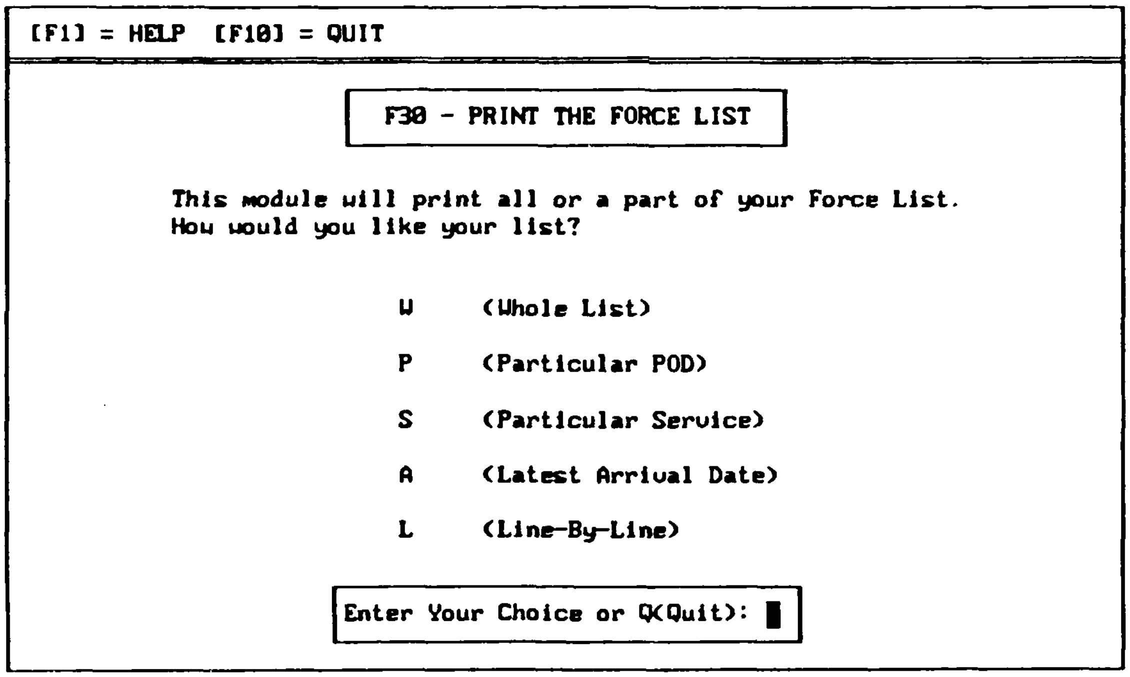
[F1] = HELP [F3] = SAVE [F10] = QUIT

F20 - CHANGE POD PRIORITIES

To change PRIORITIES use the arrow keys to move up and down to the desired row, then enter the new value from 1 to 99 using each digit only once.

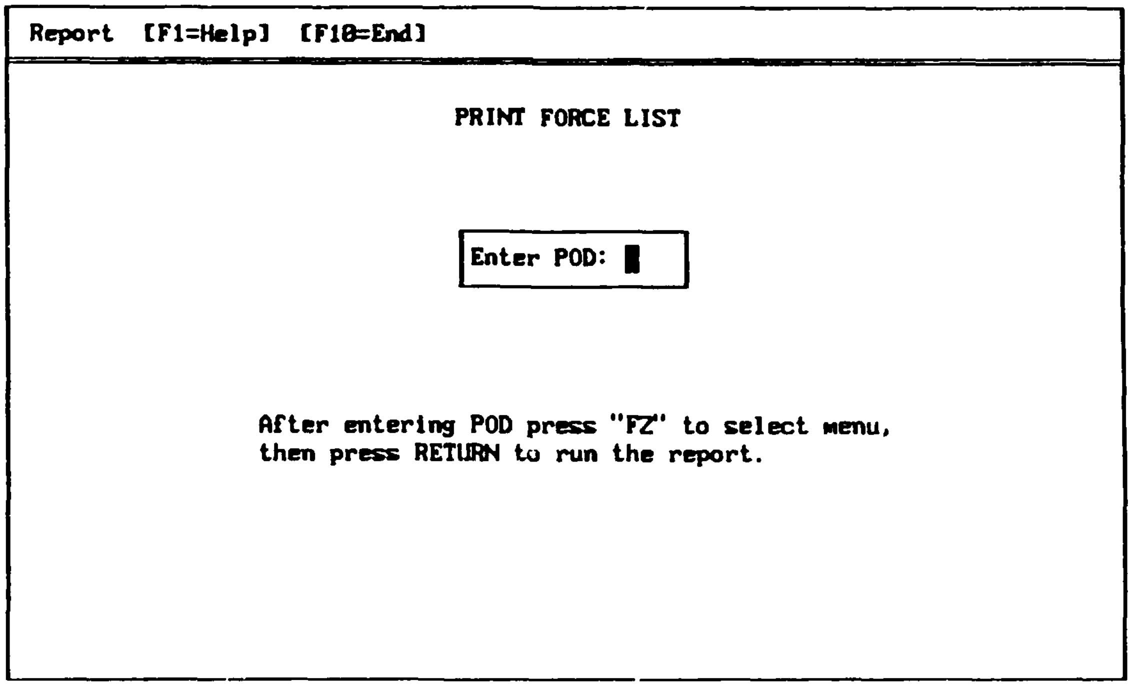
POD	PRI
BAK	4
BUN	1
MAX	2
MOL	3
MUD	5
PRM	6

Figure 15. F20 - Change POD Priorities Menu



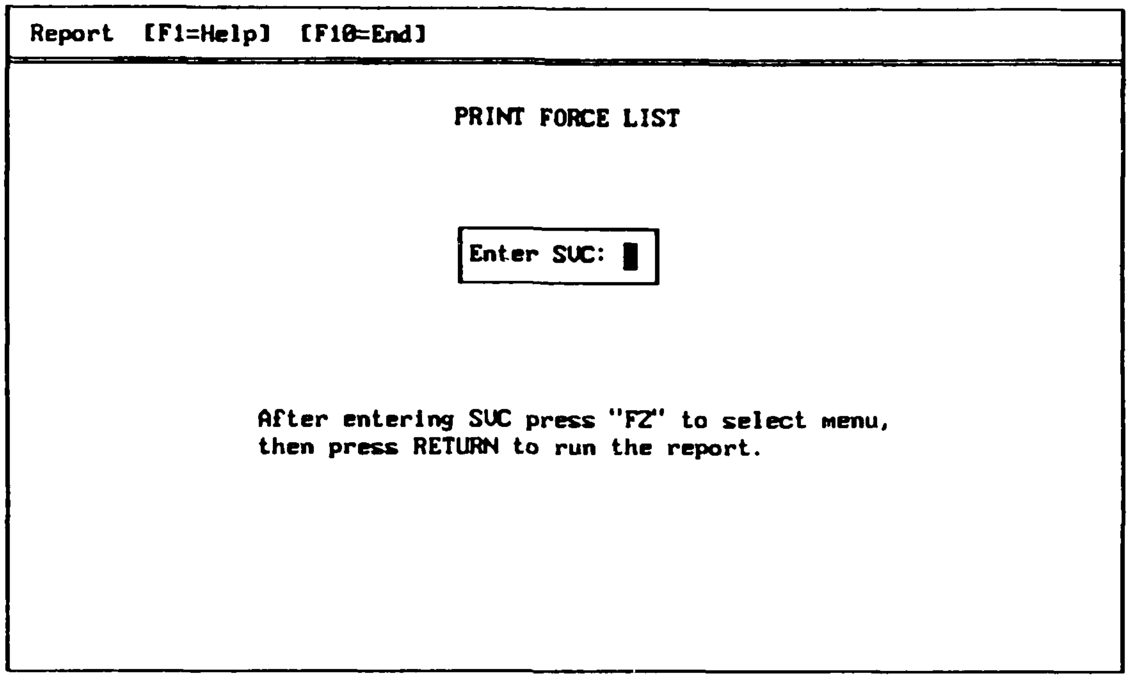
```
[F1] = HELP  [F10] = QUIT

              F30 - PRINT THE FORCE LIST

      This module will print all or a part of your Force List.
      How would you like your list?

              W     (Whole List)

              P     (Particular POD)

              S     (Particular Service)

              A     (Latest Arrival Date)

              L     (Line-By-Line)

            Enter Your Choice or Q(Quit): █
```

Figure 16. F30 - Print Forcelist Menu



```
Report  [F1=Help]  [F10=End]

                 PRINT FORCE LIST

                 Enter POD: █

      After entering POD press "FZ" to select menu,
      then press RETURN to run the report.
```

Figure 17. F30 - Print Forcelist, POD Entry Menu



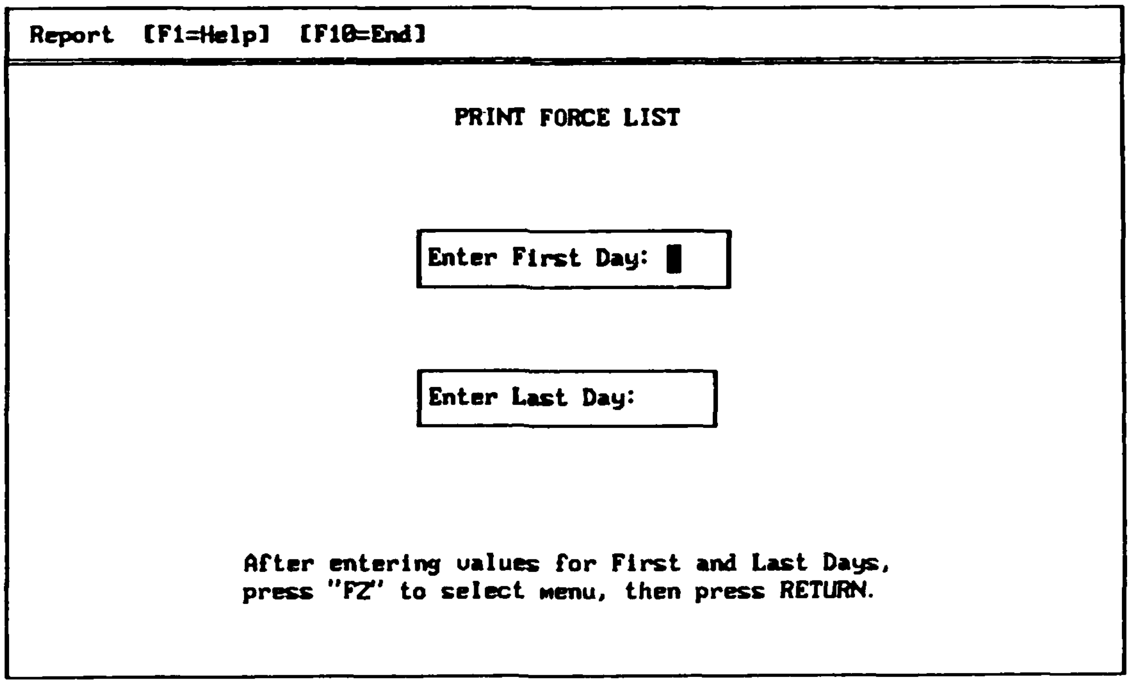
Report [F1=Help] [F10=End]

PRINT FORCE LIST

Enter SVC: █

After entering SVC press "F2" to select menu,
then press RETURN to run the report.

Figure 18. F30 - Print Forcelist, SVC Entry Menu



Report [F1=Help] [F10=End]

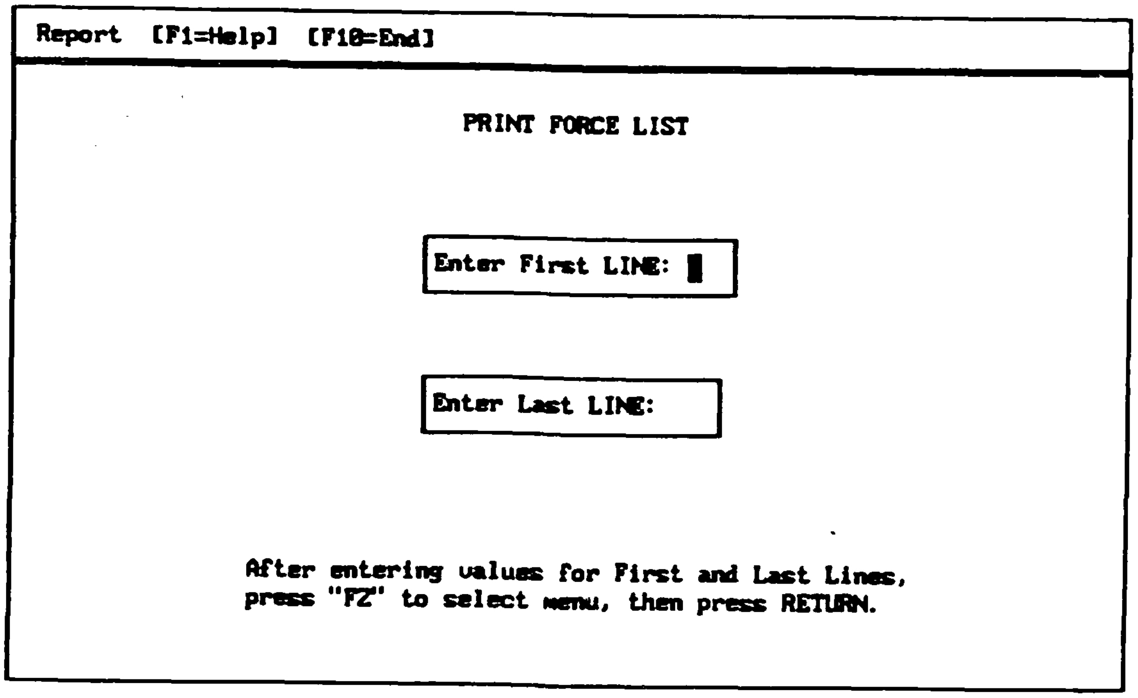
PRINT FORCE LIST

Enter First Day: █

Enter Last Day:

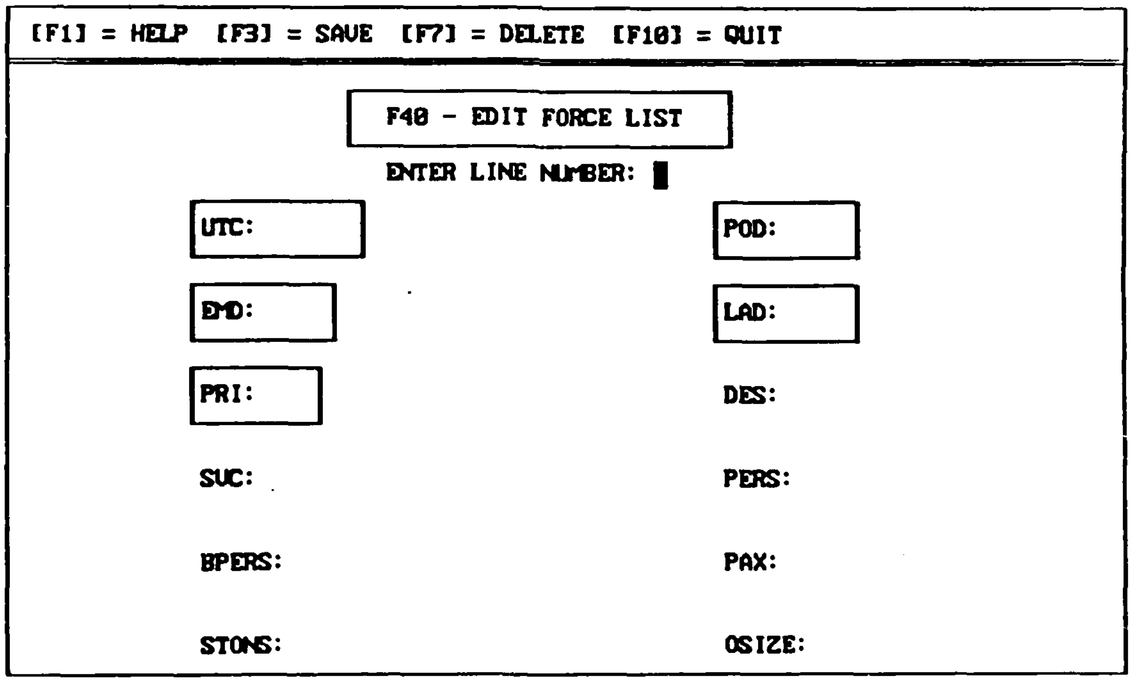
After entering values for First and Last Days,
press "F2" to select menu, then press RETURN.

Figure 19. F30 - Print Forcelist, LAD Entry Menu



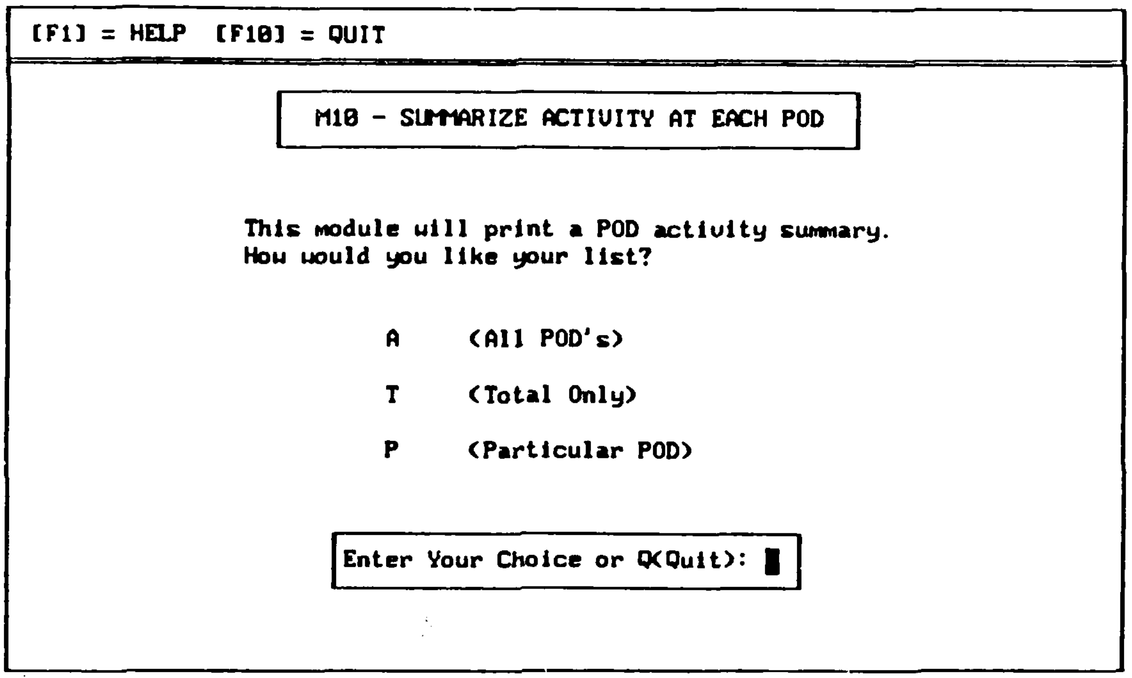
```
Report  [F1=Help]  [F10=End]

                    PRINT FORCE LIST

                 Enter First LINE: █

                 Enter Last LINE:

       After entering values for First and Last Lines,
       press "F2" to select menu, then press RETURN.
```

Figure 20. F30 - Print Forcelist, Line Entry Menu



```
[F1] = HELP  [F3] = SAVE  [F7] = DELETE  [F10] = QUIT

              F40 - EDIT FORCE LIST
             ENTER LINE NUMBER: █

    UTC:                           POD:

    EMD:                           LAD:

    PRI:                           DES:

    SUC:                           PERS:

    BPERS:                         PAX:

    STONS:                         OSIZE:
```

Figure 21. F40 - Edit Force List Menu


[F1] = HELP [F10] = QUIT

M10 - SUMMARIZE ACTIVITY AT EACH POD

This module will print a POD activity summary.
How would you like your list?

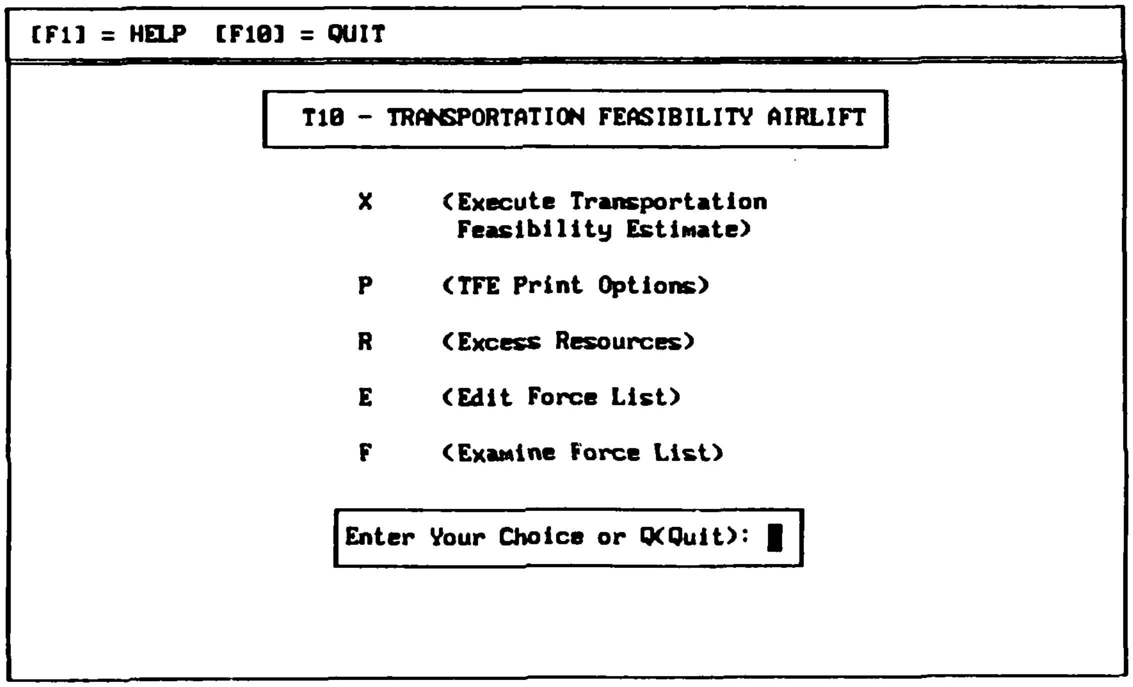
A (All POD's)

T (Total Only)

P (Particular POD)

Enter Your Choice or Q(Quit):

Figure 22. M10 - Summarize Activity At Each POD Menu


[F1] = HELP [F10] = QUIT

T10 - TRANSPORTATION FEASIBILITY AIRLIFT

X (Execute Transportation Feasibility Estimate)

P (TFE Print Options)

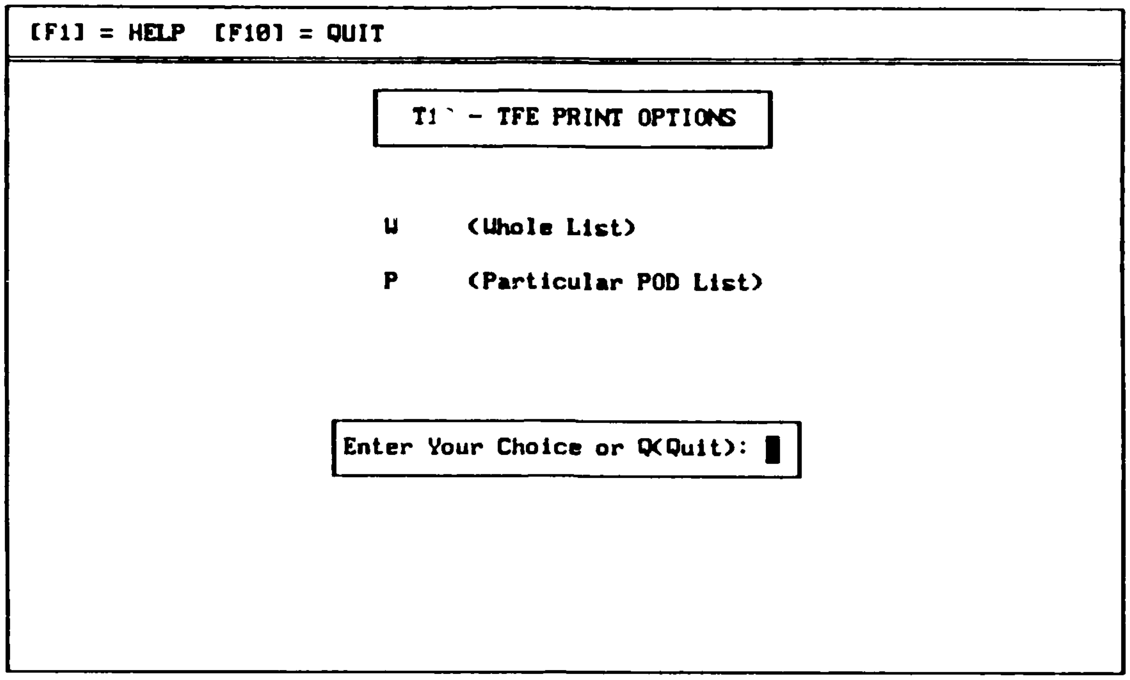
R (Excess Resources)

E (Edit Force List)

F (Examine Force List)

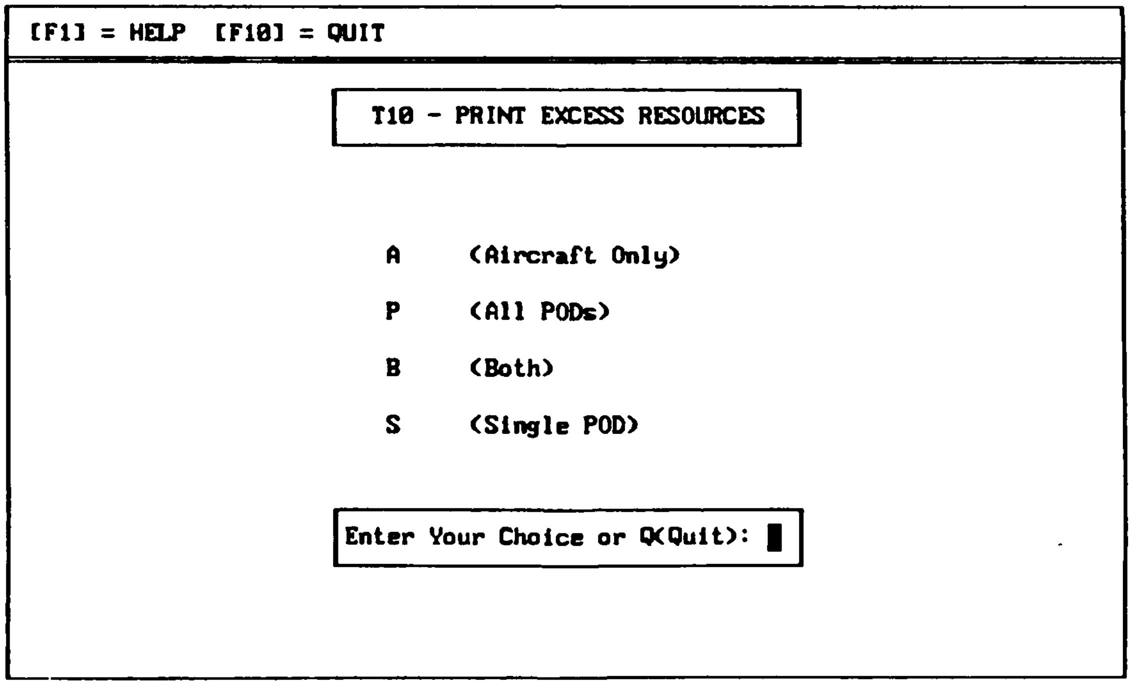
Enter Your Choice or Q(Quit):

Figure 23. T10 - Transportation Feasibility Airlift Menu



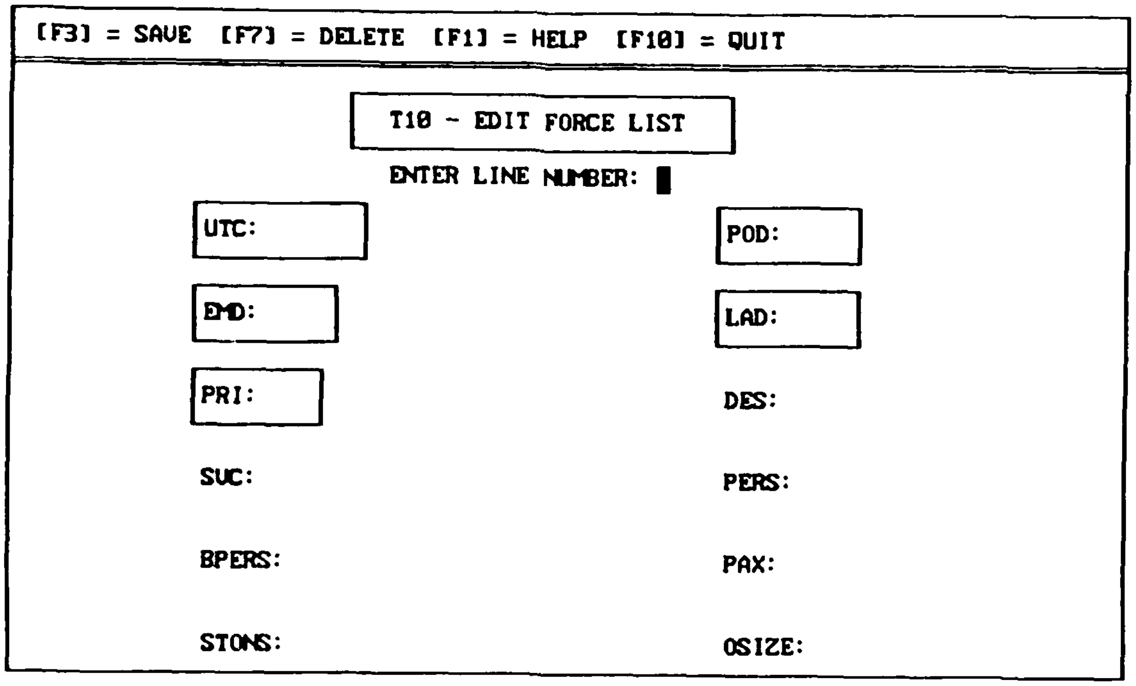
```
[F1] = HELP  [F10] = QUIT

T1` - TFE PRINT OPTIONS

W     (Whole List)
P     (Particular POD List)

Enter Your Choice or Q(Quit): █
```

Figure 24. T10 - TFE Print Options Menu



```
[F1] = HELP  [F10] = QUIT

T10 - PRINT EXCESS RESOURCES

A     (Aircraft Only)
P     (All PODs)
B     (Both)
S     (Single POD)

Enter Your Choice or Q(Quit): █
```

Figure 25. T10 - Print Excess Resources Menu



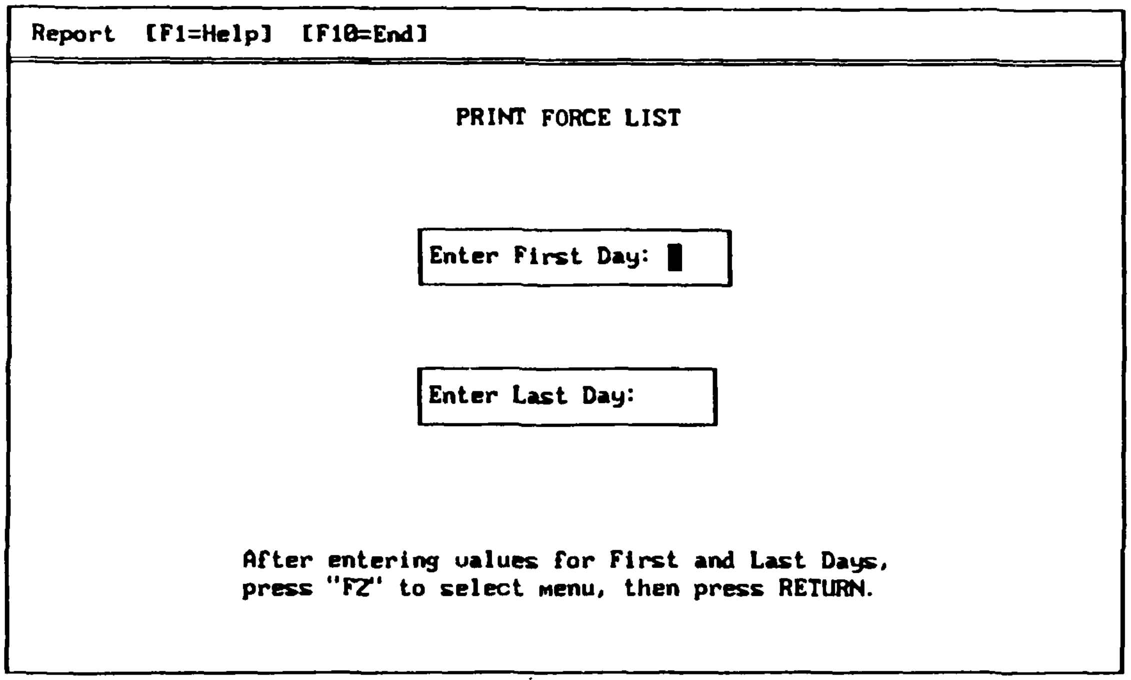
[F3] = SAVE [F7] = DELETE [F1] = HELP [F10] = QUIT

T10 - EDIT FORCE LIST

ENTER LINE NUMBER:

UTC:	POD:
EMD:	LAD:
PRI:	DES:
SUC:	PERS:
BPERS:	PAX:
STONS:	OSIZE:

Figure 26. T10 - Edit Forcelist Menu



Report [F1=Help] [F10=End]

PRINT FORCE LIST

Enter First Day:

Enter Last Day:

After entering values for First and Last Days,
press "F2" to select menu, then press RETURN.

Figure 27. T10 - Print Forcelist Menu

Appendix B. *Database Relations*

The following are the remaining database relations taken from the E-R diagrams in Chapter 3.

COLUMN NAME	*FORMAT*
Line Number	I-4
UTC	C-5
POD	C-3
EMD	I-3
LAD	I-3
PRI	I-3
FAD	I-3

Table 12. Air TPFDD Relation

COLUMN NAME	*FORMAT*
Line Number	I-4
UTC	C-5
POD	C-3
EMD	I-3
LAD	I-3
PRI	I-3
FAD	I-3
Trans Mode	C-2
Sail Route	C-2

Table 13. Sea TPFDD Relation

COLUMN NAME	*FORMAT*
UTC	C-5
PAX Cap	I-6
Cargo Cap	I-6

Table 14. Aerial Port Relation

COLUMN NAME	*FORMAT*
UTC	C-5
Number AC	I-2
AC Name	C-6

Table 15. Tactical Aircraft Assigned to UTCs Relation

COLUMN NAME	*FORMAT*
AC Name	C-6
Ramp Space	I-5

Table 16. Tactical Aircraft Data Relation

COLUMN NAME	*FORMAT*
POD	C-3
Base Name	C-12
PAX Capacity	I-6
Cargo Capacity	I-6
Max Ramp Space	I-6
POD Priority	I-3

Table 17. Airbase Relation

COLUMN NAME	*FORMAT*
POD	C-3
Port Name	C-12
Daily Cargo Cap	I-8
Control Depth	I-2
POD Priority	I-2

Table 18. Seaport Relation

COLUMN NAME	*FORMAT*
AC Name	C-6
Pass Cap w/0 Cargo	I-4
Pass Cap w/Max Cargo	I-4
Cargo Cap	I-4
Ramp Space	I-5
Pass Cargo Tradeoff	I-3
Outsized Priority	I-3
Cargo Priority	I-3
Pass Priority	I-3

Table 19. Airlift Aircraft Relation

COLUMN NAME	*FORMAT*
Type	C-2
Sail Time Suez	I-2
Sail Time S Africa	I-2
Cargo Capacity	I-6
Load Rate	I-5
Unload Rate	I-5
Max Draft	I-2
Sur Gen Cargo Cap	C-2

Table 20. Ship Relation

COLUMN NAME	*FORMAT*
AC Name	C-6
POD	C-3

Table 21. Airbase Aircraft Accommodation Relation

COLUMN NAME	*FORMAT*
Type	C-2
POD	C-3

Table 22. Ship Accommodation Relation

COLUMN NAME	*FORMAT*
AC Name	C-6
Day	I-3
Number Avail	I-4

Table 23. Airlift Aircraft Available Relation

COLUMN NAME	*FORMAT*
Type	C-2
Day	I-3
Number Avail	I-4

Table 24. Ships Available Relation

Bibliography

1. *Joint Operation Planning and Execution*. Department of the Air Force, Air University, Maxwell AFB, AL.
2. *Joint Staff Officers Guide, AFSC PUB 1*. National Defense University, Armed Forces Staff College, Norfolk, Virginia, July 1986.
3. *PC INGRES Reference Guide*. Relational Technology, Inc., May 1987.
4. Grady Booch. *Software Engineering with Ada*. The Benjamin/Cummings Publishing Company, Menlo Park, California, 1983.
5. C. J. Date. *An Introduction to Database Systems*. Addison-Wesley, Menlo Park, California, 1982.
6. Wilbert O. Galitz. *Handbook of Screen Format Design*. QED Information Sciences, Inc., Wellesley, Massachusetts, 1985.
7. James R. Jansen. *JPLAN Documentation Manual*. Air Force Institute of Technology, Department of Electrical and Computer Engineering, Wright-Patterson AFB, Ohio, 1987.
8. James R. Jansen. *JPLAN Monitor User's Guide*. Air Force Institute of Technology, Department of Electrical and Computer Engineering, Wright-Patterson AFB, Ohio, 1987.
9. Ralph H. Sprague Jr. and Eric D. Carlson. *Building Effective Decision Support Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.
10. Henry F. Korth and Abraham Silberschatz. *Database Systems Concepts*. McGraw-Hill Book Company, New York, 1986.
11. James Martin. *Computer Data-Base Organization*. Prentice-Hall, Englewood Cliffs, New Jersey, 1975.
12. Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, Menlo Park, California, 1987.

Vita

Capt James R. Jansen was born on 13 August, 1957 in Hazel Green, Wisconsin. He graduated from Hazel Green High School in 1975 and enlisted in the U. S. Air Force, he served until 1979 when he received an ROTC scholarship to attend Metropolitan State College in Denver, Colorado. He graduated in 1981 from Metropolitan State College with a B.S. in Computer and Management Science. Upon graduation he received a commission in the U.S. Air Force and attended Missile Initial Qualification Training (IQT) at Vandenburg AFB, California. After completion of IQT he was assigned to the 10th Strategic Missile Squadron, Malmstrom AFB, Montana where he served as a Missile Launch Officer and Crew Commander. During the last year of his tour at Malmstrom he was an Instructor Crew Commander for the 341st Strategic Missile Wing. While stationed at Malmstrom he also earned a M.S. in Systems Management from the University of Southern California. In 1986 he was assigned to the Air Force Institute of Technology.

Permanent address: 1410 36th Street
Wichita Falls, TX 76302

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/87D-15	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL *(If applicable)* AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State, and ZIP Code)* Air Force Institute of Technology Wright-Patterson AFB, OH 45433-6583	7b. ADDRESS *(City, State, and ZIP Code)*
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Air Force Wargaming Ctr 8b. OFFICE SYMBOL *(If applicable)* AU CADRE/WG	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State, and ZIP Code)* Maxwell AFB, AL 36112	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO | WORK UNIT ACCESSION NO

11. TITLE *(Include Security Classification)*
REDESIGN OF THE JOINT PLANNING EXERCISE (JPLAN)

12. PERSONAL AUTHOR(S)
James R. Jansen, M.S., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
MS Thesis	FROM ________ TO ______	1987 December	80

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
05	02	
14	02	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Data Bases, Simulation, User Needs

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

Approved for public release: IAW AFR 190-1,
[illegible] 22 Feb 88
LYNN E. WOLAVER
[illegible] and Professional Development
[illegible] (AIC)
[illegible] OH 45433

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Mark Roth, Capt, USAF	22b. TELEPHONE *(Include Area Code)* (513) 255-3576 22c. OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

(19) JPLAN is a simulation of the Joint Operation Planning System used by the various commands to develop operation plans. JPLAN allows student teams to identify needed combat, combat support, and combat service support forces, build force lists, and simulate deployment of these forces in support of an operation plan. Transportation resources are provided to simulate transportation of the forces to their destinations.

The existing JPLAN exercise is outdated both in terms of hardware and software which runs on hardcopy terminals connected to a large mainframe computer. It is not easily used nor maintained due to the use of a slow, unforgiving user interface, a fixed file management system, and fixed reporting capabilities. This thesis describes the redesign of JPLAN to operate on a microcomputer using state-of-the-art database and program design techniques and a fourth generation application development system.

The entity-relationship (E-R) data model was used to represent JPLAN's data entities, attributes and entity relationships. Then a relational database structure was used to implement the database design portrayed by the E-R diagram. The user interface was designed by using storyboards to illustrate menus and their function to the users. A screen editing tool and a fourth generation language (4GL) were employed to implement the user interface portion of the design, and a structured query language (SQL) was utilized for the database implementation.

The FORTRAN simulation was rewritten in Microsoft C and Embedded SQL. During the simulation coding and validation process several areas for improvement were identified and incorporated into the new simulation program, which will enable the exercise to parallel the "real world" more closely. The new JPLAN exercise is more "user friendly" than the existing exercise, runs on widely available microcomputers, and has become much easier to modify as requirements evolve.

END
DATE
3-88
DTIC