1

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

AFIT/GCS/ENG/88D-24

Extending the User Interface For The Theater War Exercise

THESIS

Kenneth Russell Wilcox
Captain, USAF

AFIT/GCS/ENG/88D-24

AFIT/GCS/ENG/88D-24

Extending the User Interface For The Theater War Exercise

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)

Accession For		
NTIS GRA&I		✓
DTIC TAB		☐
Unannounced		☐
Justification		
By		
Distribution/		
Availability Codes		
Dist	Avail and/or Special	
A-1		

Kenneth Russell Wilcox, B.S.

Captain, USAF

December, 1988

Preface

The goal of this thesis was to evaluate, enhance, and integrate the current Theater War Exercise (TWX) system. This thesis was a continuation of two previous efforts to develop TWX into an effective learning tool. The previous efforts rehosted TWX from a rigid flat file structure to a flexible relational database structure and developed a new user interface. This effort included further development of the TWX user and game controller interfaces using the integrated tools of the Ingres development system.

The Theater War Exercise is a wargaming simulation of airpower employment in a European theater conflict. TWX simulates a realistic five day conflict in which senior Air Force officers must make command decisions for airpower employment. The wargame serves as a learning tool to provide situations in which the officers apply airpower employment concepts and principles of war taught in the classroom.

This thesis presents the analysis and solution of specific problems in areas identified in the evaluation of the TWX system that require further development to improve the system's effectiveness.

I am extremely grateful to my thesis advisor, Captain Mark Roth, for his superb support, guidance, and patience during this endeavor. I wish to also thank my committee members, Dr B. Nagarsenker and Captain Nathaniel Davis, for their willingness to assist in my thesis work. I would also like to thank God for giving me the strength and perseverance throughout this thesis effort. Finally, I would like to thank my dear wife [redacted] for her loving support and understanding.

Kenneth Russell Wilcox

Table of Contents

List of Figures

AFIT/GCS/ENG/88D-24

Abstract

The Theater War Exercise (TWX) is a wargaming simulation of airpower employment in a European theater conflict. TWX simulates a realistic five day conflict in which senior Air Force officers must make command decisions for airpower employment. The wargame serves as a learning tool to provide an environment in which the officers apply airpower employment concepts and principles of war taught in the classroom.

The TWX system consists of the following subsystems: a) a user interface for the seminar students to retrieve and input data in the database; b) a controller interface which allows the game administrator to manage, monitor, and control seminar databases; c) land and air battle simulations which uses the data input by students and determines conflict results.

The current TWX system was the result of previous thesis efforts which rehosted TWX to a microcomputer environment and developed a flexible user interface. Both efforts were performed independently using a common commerical relational database management system. The completed works were to be combined into an integrated system.

The goal of this thesis was to complete the integration of the previous efforts and extend the capabilities of TWX to improve its effectiveness as a learning tool. This was accomplished by first evaluating the current system and identifying areas which required improvement. Specific areas that required attention included the need for better flexibility and efficiency in the user interface, an enhanced controller interface, an expert user mode for the user interface, statistical routines, and thorough integration of the TWX subsystems and enhancements. This was accomplished using the integrated tools of the Ingres development system in a multiuser computer environment.

All modifications and enhancements resulting from this thesis were related to the interfaces of the controller function and the seminar users. Though not all interface modifications are readily apparent to the user, the changes improved the user/TWX system interaction and increased the flexibility of the total system. Improvements to the TWX system resulted from a combination of changes to the database organization, modification of the application programs, reworking code to improve performance, and enhancements.

Extending the User Interface For The Theater War Exercise

I. Introduction

1.1 Background

The Theater War Exercise (TWX) is a wargaming simulation of airpower employment in a European theater conflict. TWX simulates a realistic five day conflict in which senior Air Force officers must make command decisions for airpower employment. As stated by the TWX user handbook, "TWX provides the educational opportunity to employ airpower strategies and doctrine and the principles of war in a simulated...situation" [3:2].

Originally programmed to run on a Honeywell H6000 mainframe computer, TWX has been rehosted to a microcomputer environment due to severe limitations and lack of flexibility of the original system. Rehosting the system to a new environment was the result of a previous thesis effort [6]. The environment combines standard PC-compatible microcomputers and a multiuser Micro VAX III microcomputer. Complete restructuring of the TWX system resulted from the rehosting work. The old flat file system, which was application dependent, was replaced with a commercial relational database management system. As a result of the rehosting, the TWX system has better portability and it's easier to maintain and enhance [6].

In conjunction with the rehosting work, another thesis effort concentrated on developing a new user interface [13]. The emphasis of the work was to replace a slow and difficult to use interface with one which is easy to use and flexible. The old interface used hard copy devices for all inputs and outputs. Lack of error correction capability when entering inputs made the process difficult and frustrating to the novice users [13].

The work of both theses took place during the same time period. Even though the components were developed separately, they were compatible. However, the components are not completely integrated together into a functional and flexible system. The integrated system must be flexible enough to function on various hardware configurations while maintaining the easy to use interface. Design and implementation techniques used to rehost the game and develop the user interfaces employed software engineering principles which allow for modifications without extensive changes to TWX [6,13]. Adhering to the same good software principles while integrating and enhancing the user interface will ensure continued maintainability and modifiability of the system [16].

1.2 Problem Statement

The previous thesis efforts to rehost the TWX system and develop a more friendly user interface were not fully developed and integrated into an effective learning tool. Specifically, the areas that require attention include better flexibility and efficiency in the user interface, an enhanced controller interface, an expert user mode for the user interface, statistical functions, and thorough integration of the TWX subsystems and enhancements.

Flexibility and Efficiency. The lack of flexibility and efficiency refers to problems in the area of human interaction with TWX. Problem areas include limiting the user to one type of terminal, the lack of flexibility to replay a portion of the game, slow user interface responses, and a requirement for a strict system structure.

The user interface developed for data entry by the seminar teams initially limited the user to using the applications on a microcomputer. Using the microcomputer to run the applications, the applications accessed the database residing on a central multiuser system, a Micro VAX III. This restricted users from accessing the application programs through other types of terminals which may be connected to the network.

The current TWX game does not have the means of resetting a seminar database to a previous point in the exercise. TWX spans several exercise days, Day 0 for prehostility planning, and Day 1 through Day 5 for mission planning and execution. If a problem occurs during the simulation which results in a corrupted database, there would be no way to reset the database to the point prior to the simulation. Without the capability, game adminstrators do not have the necessary control to reset to a particular point to conduct experiments with a specific part of the exercise. Currently the only way to restore a seminar database in the situations mentioned would be to recreate the seminar to Day 0 or attempt to fix the database manually. Recreating the database to Day 0 means starting from scratch. The other alternative of manually restoring the database tables requires that the previous values be known in each table that may have changed. Neither alternative has any appeal. Thus, a type of crash recovery capability is needed for TWX.

Controller Interface. The controller interface provided only the basic functions to manage, monitor, and control seminar databases. Functions of the controller have not been developed enough for the controller to perform effectively and efficiently.

A serious limitation of the controller is its ability to access only one seminar database at a time. This limitation is serious because the controller is responsible for managing control flags within the seminar databases. As the number of seminars increase, the task of managing these flags becomes more difficult and prone to error or neglect.

While the current function provides a means for backing and restoring a seminar on tape, the function is inefficient and time consuming. Backups are only performed as requested by the controller and not automatically. This in turn limits the point in time to which the seminar database can be restored.

Expert User Mode. The user interface has been developed with the characteristics of one type of user in mind, the novice. While this makes the interface easy to learn and use for the novice it can be source of frustration for the experienced user who knows what he or she wants to do. The functionality of the user interface needs to be developed further to also satisfy the needs of the more experienced user.

Faculty members which usually play the part of the opposing side are generally considered experienced users. Through playing experience, they may know exactly what data they want to load into the game. Following the step by step instructions of the current interface would be time consuming and frustrating. Therefore, an expert user mode is needed to provide experienced users greater control and flexibility.

Statistical Routines. The original TWX game provided some statistics which informed the game controller how well seminars were performing. When the game was rehosted, the statistical routines were not included. Therefore, if the game controller wanted statistics on a seminar, the statistics would have to be gathered manually using the seminar's daily reports. Such a task is better suited for a computer.

Since statistics were not kept on the seminars, no historical record was maintained for analysis of the team's progress. If TWX is to be an effective learning tool, then a means needs to exist to evaluate how well the students employ what they are taught. Statistics gathered on the seminar's performance would assist in such an evaluation.

Integration. The thesis efforts to rehost the TWX system to a new environment and develop a new user interface were performed during the same period of time. Due to a time restriction, both efforts were not fully integrated together into a consolidated system. In particular, seminars created by the controller required minor modifications prior to being usable. This included initializing control values properly and creating views required by the user interface applications. Most of the integration problems were minor compared to the complexity of the two efforts. However, this illustrates that integration is essential for the user interface to operate smoothly and the controller function portion of the TWX system to properly manage and control the seminar databases.

The importance of integration is reiterated as enhancements are developed to improve the game, they must be fully integrated into the overall system. Through proper integration unexpected and potentially disastrous side effects are avoided.

1.3 Proposed Solutions

All modifications resulting from this thesis were related to the interfaces of the controller function and the seminar users. Though not all interface modifications are readily apparent to the user, the changes improved the user/TWX system interaction and increased the flexibility of the total system. Improvements to the TWX system resulted from a combination of changes to the database organization, modification of the application programs, reworking code to improve performance, and enhancements.

User interface modifications were made following general interface design guidelines. According to Liang, there are three general guidelines which establish a foundation for good interface design. They are technological developments, requirements of the task, and characteristics of the user [14].

Liang considers an interface good if it satisfies the following requirements[14:182]:

1. Be diverse: support both inexperienced and experienced users.
2. Be forgiving: have good error recovering capabilities.
3. Be efficient: minimize the effort required to accomplish a job.
4. Be convenient: provide good accessibility to all operations.
5. Be flexible: provide multiple routes for accessing an operation.
6. Be consistent: minimize learning requirements and unexpected actions.
7. Be helpful: provide good help and error messages.

Flexibility and Efficiency. A designer must consider the environment and the situation to match the appropriate technology with the application. The newest technology may not always be the most appropriate for a given application [14]. In the case of TWX, the system must operate in a network environment. Within the network environment, the application programs must operate on the central system, which in the case of TWX is a Micro VAX III, or a PC. Applications on the central system must be configurable to operate with a variety of different types of terminals.

The user interface developed for data entry by the seminar teams initially limited the user to using the applications on a microcomputer, which accessed the database residing on a central multiuser system, a Micro VAX III. To build more flexibility into the applications, the applications were modified to run on either a microcomputer or the central system. Furthermore, the user can access the applications on the multiuser network through different types of terminals.

While the modifications increased the flexibility of the application, operation of the application remained consistent on the different terminals. User actions are the same for any of the different types of terminals available. The functions of the application screens remained constant while the function keys of the application screens are tailored to the specific terminal employed by the user.

TWX was further enhanced with the capability to restore a seminar to a particular point in the exercise. With the savepoints and restoration capability a database seminar is easily recreated should the need arise. The ability to reset to a previous point in the game increases the game's flexibility and value as a learning tool.

Controller Interface. Another important modifications to the system was further development of the user interface for the controller function to improve management and control of the active TWX seminars. The new interface provides simultaneous access to more than one seminar database in areas which are key to improving management tasks. Simultaneous access of more than one database required the functionality of a distributed database.

A distributed database refers to a database divided into distinct parts stored at separate locations. The different locations are connected together in a communications network by links. TWX has been developed using the relational database INGRES, which contains a subsystem called INGRES/NET that provides a communications network to access databases on the network. Another subsystem, INGRES/STAR, provides the distributed database functions [2]. Each location of the distributed database, also referred to as a site or node, resembles an individual, self-contained database [9,12].

A bottom-up approach was used to develop a distributed database system for the TWX controller function. In this approach, the existing seminar databases are aggregated to form the distributed database [8]. Similar to other distributed systems, each seminar database is autonomous. The current seminar databases contain all the information needed to function. Aggregating the seminar databases into a distributed database allows the controller concurrent access to the seminar databases.

Application programs for the controller interface controls access to the various seminar databases. Temporary and permanent links are created as required within the interface. The proper link name is determined based on the application and the seminar number. This relieves the user from have to keep track of the link name when executing an application. This is especially convenient to the user when simultaneously accessing multiple databases which requires multiple link names.

Implementing the controller function in a distributed database system provided the TWX system the capacity to expand as necessary to handle seminar databases as they are created. Since the controller distributed database system consists of links to seminar databases, the system can expand incrementally by adding new links as needed. Of course growth is limited to the network capacity.

Expert User Mode. Analyzing the tasks and the characteristics of TWX users resulted in the development of an expert user mode. The expert user mode is an alternative to the current easy to use, step by step interface. Instead of designing a completely new interface, the expert user mode was incorporated into the current interface.

The expert mode is provided as an alternative for data entry during the mission planning portion of the interface. This provides the user greater control. Of course with this increased control, the user accepts increased responsiblity for ensuring only valid data is entered. Error checking is still performed but only after all mission planning is completed. This differs from the current interface which performs error checking after the completion of each step.

Statistical Routines. An enhancement to the seminar user interface provided access to statistical routines which indicate the progress of the wargaming simulation. These routines aggregated selected TWX information to provide the seminar teams a broad overview of the effectiveness of executing their decisions.

Integration. Prior to enhancing either the independently developed user interface or the game controller interface, the TWX system was tested to ensure integration between the subsystems. Testing revealed only minor problems which required correcting. One reason for the problems was because the application programs for the user interface were developed on a microcomputer. The rehosted TWX database and the controller function were developed to operate on a Digital Equipment Company (DEC) Micro VAX III. Merging the two efforts into one system revealed the problems. Once integration was completed, enhancements to the user interface were designed and implemented to increase the effectiveness and functionality of the system.

Integration and testing were an essential part of the development process in extending the user interface. The application development environment of INGRES allowed incorporating integration and testing during development. This helped identify problem areas in early stages of development.

1.4 *Assumptions*

The following assumptions were made concerning the work related to this thesis:

1. The Air Force Wargaming Center is satisfied with the current structure of the database which resulted from the rehosting work of the TWX system.
2. The Air Force Wargaming Center is satisfied with the basic screen-oriented interface resulting from the work of developing a more friendly user interface.
3. After the TWX system was rehosted, the Wargaming Center validated the output of the system.
4. As long as modifications to the user interface does not change current functions or logic of the TWX system, then only the new output results will require verification and validation.

1.5 *Approach/Methodology*

The first task towards the accomplishment of this thesis work was a through understanding and evaluation of the previous thesis works related to the TWX system. This required retracing the process followed by each work as outlined by the theses.

The next task involved fully integrating the rehosted TWX system and the new user interface. At this point, only those modifications required to resolve integration problems were addressed and implemented.

The following tasks were accomplished after completing the necessary integration to provide TWX with capabilities needed to fully develop the system and make the it an effective learning tool:

1. The current user interface was modified to provide the capability to easily configure the application programs to allow the use of different terminals.

2. Further modifications to the user interface removed system dependent code, such as explicit path names, from the applications and incorporated the required information into the TWX database.
3. Additions to the database were designed and implemented following the same database design techniques used during the rehosting of TWX.
4. A distributed database system was designed and implemented to manage and improve the TWX controller function.
5. Considering interface guidelines, the controller and seminar user interfaces were modified to access the additions incorporated into the system.
6. The user's manual and the system integrator handbook were updated to reflect the changes to the system.

Application prototyping was the methodology used for this thesis. Certain assumptions about the thesis were made in connection with choosing the application prototyping methodology. The following is a list of the general assumptions and a brief explanation why these assumptions were valid for this thesis.

1. All requirements not prespecified: Discussions with the Air Force Wargaming Center provided general directions of what work needed to be accomplished. However, detailed requirements were not available.
2. Inherent communication gap: Communication of detailed requirements was hampered by a lack of understanding by the user of the development system's capability.
3. Availability of tools for quick building: The Ingres development system provides the necessary tools for rapid prototyping.
4. Active system required: The resulting TWX system will be used interactively.
5. Rigorous approach is correct once requirements known: Other, more rigorous approaches are applicable in different phases of the software development cycle.
6. Extensive iterations necessary: New requirements for the TWX system were identified with understanding of capabilities and limitations [5].

Determining the suitablity of application prototyping for this thesis was based on evaluation of a number of factors. The following is a list of the factors and a description of a type of system which is appropriate for application prototyping.

1. System Structure: Interactive and large amounts of database transaction processing.
2. Logic Structure: Very structured components.
3. User Characteristics: Uncertain about detailed requirements.
4. Application Constraints: Development time available to perform iterations.
5. Project Management: Confidence in the development system to perform application prototyping.
6. Project Environment: Prespecification difficult and capabilities unknown [5].

Based on the above factors, the identified problems were good candidates for this methodology.

1.6 Materials and Equipment

The equipment used for this thesis problem included one Digital Equipment Corporation Micro Vax III, one Zenith Z-158 microcomputer, one Zenith Z-248 microcomputer, a commercial relational DBMS, and other software development tools. All equipment listed was provided by the Air Force Wargaming Center. Source code and documentation of the previous thesis efforts were essential to the integration and further development of the TWX system.

1.7 Sequence of Presentation

Chapters II through VI detail the analysis, design, and solution to each of the problems presented in this chapter: flexibility and efficiency, controller interface, expert user mode, statistical functions, and integration. Each chapter begins with an introduction of the problem as it is related to the overall TWX system.

Next, the problem is analyzed following the fundamental software engineering principles for software requirements analysis. Relevant information from a review of current literature concerning the general problem area is included in this section.

The third section of these chapters discusses the solution to the specific problem. This includes a discussion of the software tools used, problems encountered, and testing procedures.

The fourth and final section of each chapter contains a summary of the area addressed and any recommendations for further work in the area.

Chapter VII completes the thesis with an overall conclusion and recommendations for further development of the TWX system.

II. Flexibility and Efficiency

2.1 Problem

The original development of an easy to use interface was performed on a microcomputer. Transitioning the interface software from the microcomputer to the multiuser system presented no compatibility problems with the underlying database management system, Ingres. However, the interface utilized function keys which were specific to the microcomputer. Working in the multiuser environment, the function keys were not applicable to the different types of terminals connected to the system.

Proper operation of a software system relies on the environment being properly configured. In the microcomputer environment, configuration information specified the system path to help files. This information was stored directly in the application code. Moving to the multiuser environment naturally required changing the system path to reflect the new environment. Furthermore, as enhancements are made to the system which allow the system to support a variety of devices then more configuration information must be maintained. Coding the configuration information directly into the application code to reflect the current operating environment makes the interface very restrictive and more difficult to manage.

Another restrictive and inefficient feature of the TWX game is the current backup mechanism. A seminar is only backed up when the action is manually initiated by the game controller. Backing up the database at specific intervals relies completely on the game controller to initiate the action. Furthermore, the process backs up the entire database, including those tables which do not change, such as constants tables. This means the process is wasting time unnecessarily backing up static tables. Backing up the exercise databases should be an automatic function performed at set intervals and should be efficient. Changes to the backup process will have a definite impact on the restoring process. Restoring the database is a problem which must be resolved by the controller function.

When the current user interface was developed the focus was on making it easy to use. While the design and implementation was performed in a logical manner, it may not have been done in the most efficient manner. Efficiency referred to here is in terms of the responsiveness of the interface. In portions of the interface, a user was required to wait up to 120 seconds while the interface performed calculations in response to the user's selection. Areas of the interface which have a response time greater than 15 seconds exceed the recommended response time to execute a problem. The code for these areas required review to determine if a more efficient means of implementation was available.

2.2 Analysis

In many ways the modular structure of the user interface easily supported modifications to make the system more flexible and efficient. Sisson referred to the benefits achieved by modular decomposition when modeling the "information flow during human-computer dialogue" [18:34]. These benefits are easily extended to show the advantages of modifying a complex system with a modular structure. These advantages are:

1. The overall system is easier to analyze and modify when it is comprised of small, relatively independent modules.
2. Modules are either independent or the interface between modules is well defined which supports independent testing.
3. Modules are easier to modify and unintended effects are minimized. [18,10]

The development environment of Ingres contributed significantly to easy modification and testing of the modules. All the modules, also referred to as applications, of the user interface were implemented using the Ingres subsystem, Application-By-Forms (ABF). Within applications are even smaller modules referred to as frames and procedures. Tools available to the developer under ABF include a fourth generation language (4GL), a means to compile 4GL code and a visual forms editor, VIFRED. ABF helped modify the user interface to increase flexibility and efficiency while maintaining a modular structure which will support further enhancements.

The requirement for flexibility has dictated that TWX be accessible by different type of terminals on a multiuser system. Since TWX is only accessible through the user interface, this means the user interface must support a variety of terminals. In the multiuser environment, users can access TWX through different type of terminals, such as VT100, VT200, or VT300 terminals, or microcomputers using software which emulate one of these types of terminals. The TWX user interface must reflect the correct function keys applicable to the type of terminal in use. No matter which type of terminal is used by a user, the interface should operate in a consistent manner.

Application programs respond to keyboard inputs from the user. Supporting different types of terminals meant the application programs have to distinguish between the different terminals to provide a correct response. Distinguishing between the terminals in application code would complicate the interface to an unmanageable level. Maintaining the terminal information separate from the application code would be far more manageable.

The information needed by the application programs to ensure a correct response is kept in a TWX table and a mapping file. The idea of separating control information from application programs was suggested in a paper written by Bass and Bunker. As pointed out by their paper, this method increases a system's flexibility. Changes can be made to the control information without disturbing the application programs [4]. As different terminals are added to the multiuser environment the necessary information would be added to the system.

A multiuser environment has several advantages over the standalone microcomputer environment. While microcomputer technology has advanced to provide greater processing speed and capability, it still does not match the power of the multiuser MicroVAX III. Also, the multiuser environment does not have the memory limitations that hamper the microcomputer. Finally, a multiuser environment allows more than one seminar team member access to different portions of TWX at the same time. The increased power and capability of the MicroVAX III allows TWX to develop and expand. Increased capability and the power to maintain an interactive environment augment motivation and provide players with a sense of realism [10].

Coupled with the increased capabilities is the requirement to improve management of the system. Maintaining backup copies of an exercise at predetermined intervals is an important management function. With the savepoints and restoration capability a database seminar is easily recreated should a situation occur that makes the current seminar database unusable or in need of resetting. One such situation could be that a seminar team needs to reaccomplish mission planning for a specific day because of gross deviations from doctrine.

In another situation the database may not be unusable, however, it may need to be reset back to a specific day for experimentation. For example, the same day could be replayed several times with changes only to the Blue side of the game while the Red side remains the same each time. This would allow comparison of the results from the changes to the Blue side while the Red side would act as a control factor. In both situations, being able to reset to a previous point in the game increases the game's flexibility and value as a learning tool.

The controller function has the primary responsibility for managing the exercise. However, the initiation of backing up an exercise at the savepoints is the responsibility of each seminar. The savepoints are automatically triggered at two points each day of the exercise. The first savepoint is after mission planning is completed by both sides. This provides a fallback position if something goes wrong during the simulation. The second savepoint is after the simulation which provides a fallback point in case a problem develops during mission planning. In either case the actions initiated by the savepoints are transparent to the user and do not effect the responsiveness of the interface.

Consistent feedback and responsiveness are very important characteristics of the user interface. The initial effort to improve the TWX user interface concentrated on developing an easy to use and flexible interface. There was little or no concern in the effort for speed. While the resulting interface satisfied the thesis objective, the transition delay between some of the screens was excessively long. Long response delays are disconcerting to all users, especially novices [11].

All INGRES applications of the TWX interface were evaluated to identify areas requiring improvements. Applications calculating available sorties for the AAFCE took an extra long time, between 105 and 120 seconds, due to a join of several tables and manipulation of the data in the tables. The join involved six comparisons of attributes from four tables. Four of the comparisons focused on the pair of attributes aircraft name and aircraft role.

One method considered to reduce the time to perform the join was introducing the use of a surrogate key [7]. The definition of a surrogate key, according to Cappelli, is a value "assigned to each occurrence of an entity in a database." The surrogate key has no relationship with the data but rather is just a pointer to the data.

To evaluate this method a surrogate key was assigned to a pair of attributes, aircraft name and aircraft role, of the aircraft relation. The aircraft relation is considered the kernel relation [7]. Cappelli refers to the kernel relation as "a top-level table for an entity set" [7:1]. This pair of attributes made a good candidate for a surrogate key because they formulated the primary key for the aircraft relation. This meant the pair can distinctively identify single tuples in the relation. These same attributes in other relations where replaced with the appropriate surrogate value. Surrogate keys were then used in joins rather than the attributes.

The other alternative considered to improve performance was rewritting the code. Code for the TWX application programs was written in Ingres' Fourth Generation Language (4GL). While the 4GL code was written in a logical manner it was not written efficiently in terms of processing speed. Changing the code to improve performance first required understanding the logic and objective of the code. The specific application that required improvement calculated the aircraft available to the seminar team for the different types of missions. A natural join of four tables was performed and results loaded into a temporary table of the form called a table field. The table field was then manipulated to deduct sorties of aircraft on bases which would not be available for missions on that particular day of the war game. This required updating the tuples in the table field which matched the aircraft name and aircraft role of the aircraft that could not fly. Even though only a very small number of tuples, less than 10 tuples out of a total of over 200 tuples, would require changing the code was such that the whole table was scanned sequentially. After the adjustments were made, the table field was finally loaded into the appropriate base table.

2.3 Solution

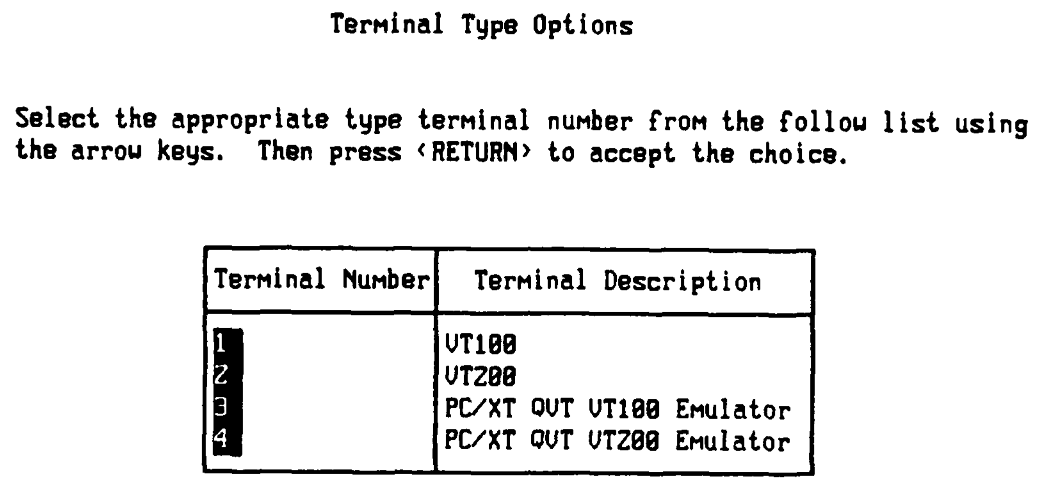
Several changes, visual and transparent, took place to enhance the user interface to support different terminals. The changes made did not affect the function of the application programs but rather configured the applications to reflect operating characteristics of the specific terminal.

A table was added to the seminar database to hold operating information on the terminals supported by the interface. Also, an attribute was added to tables which have other seminar control functions to keep track of the terminal currently being used. The user's controller table, either *bl_sem_con* or *rd_sem_con*, was changed to store the type of terminal used in the current session. This was later changed when the interface was modified to allow both sides access to the same applications. There is more discussion on this in Chapter VI.

The terminal constants table, *terminal_con*, associates the function key names of a specific terminal with functions used within the applications. These functions include exit, clear field, scroll, help, move, commit, delete, and print. Also stored in the table is a description of the terminal and the path and location of the mapping file for the terminal.

Mapping files contain the definition of active keys in Ingres's Forms Run-Time System (FRS) for a specific terminal. For example, in the mapping file for a VT200 terminal the key used to exit an application, frskey2, may be defined as the PF4 key on the keyboard. For another type of terminal this function could be mapped as another function key or control character. The interface can support additional terminals by appending the necessary information to the terminal constants table and creating an applicable mapping file. The application programs are independent of the type of terminal used and do not need to be changed when adding support for a new terminal.

User's select the type of terminal being used after logging into a session. Usually the seminar user will use the default terminal mapping which is saved in the database. For this reason the option to change the terminal type is not displayed. User's aware of this feature can press "T" or "t", referred to as the "hot key", when the introduction screen is



Terminal Type Options

Select the appropriate type terminal number from the follow list using the arrow keys. Then press <RETURN> to accept the choice.

Terminal Number	Terminal Description
1	VT100
2	VT200
3	PC/XT QVT VT100 Emulator
4	PC/XT QVT VT200 Emulator

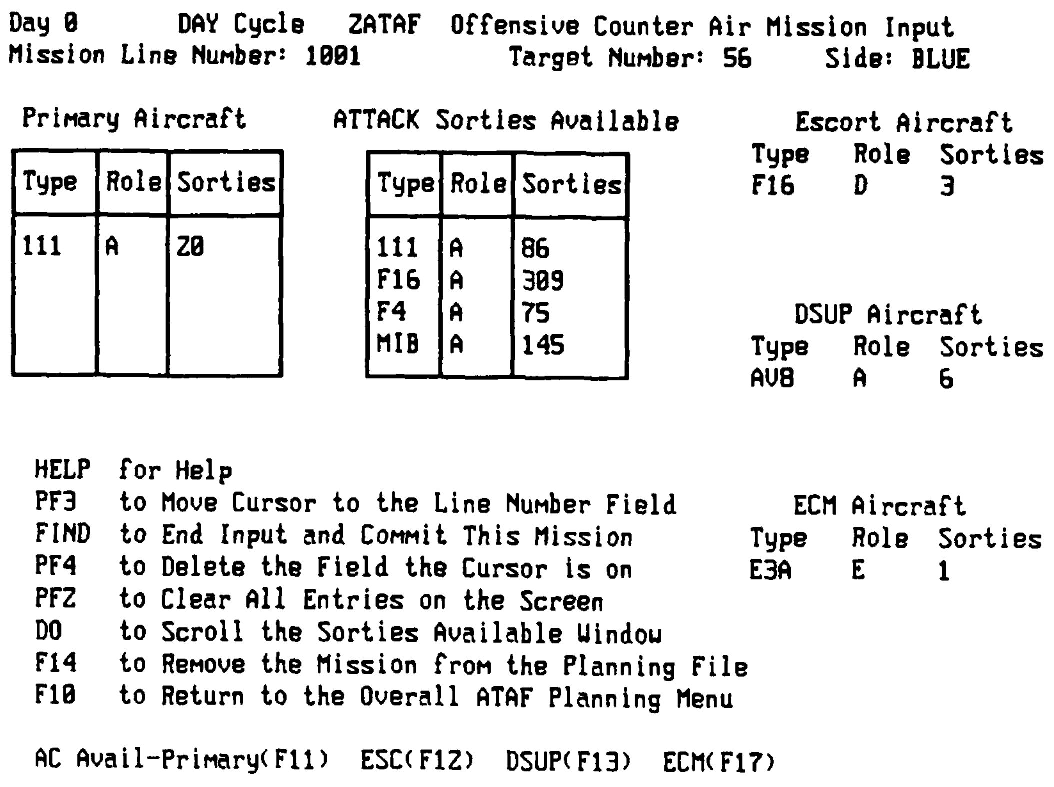
Figure 1. Terminal Options Menu

displayed. This calls the screen in which the user chooses his type of terminal from a list of supported terminals.

The menu, shown in Figure 1, displays a unique number to identify the type of terminal and a short terminal description. Using the arrow keys the user moves the cursor to the appropriate terminal number. Pressing the Return key commits the selection and the database table is updated. For the remainder of the session the application programs are adapted to reflect the function keys available on the specified terminal.

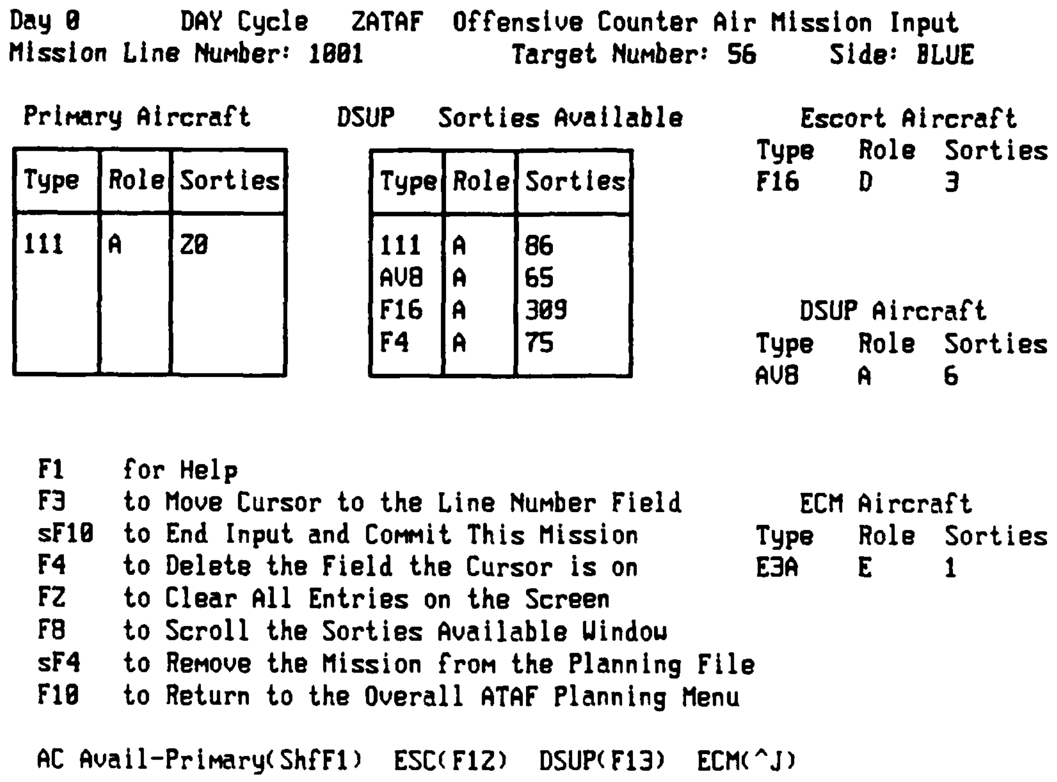
For each display, the function key names are retrieved from the database table with function key names for each terminal type. The names are inserted into the displays next to the description of their function. To illustrate this Figures 2 and 3 show the same display for different types of terminals.

Another enhancement not visible to the user but important in the exercise is the implementation of automatic savepoints. Savepoints are automatically initiated at the completion of two events, mission planning data entry and simulation execution.



```
Day 0       DAY Cycle   ZATAF  Offensive Counter Air Mission Input
Mission Line Number: 1001          Target Number: 56     Side: BLUE

 Primary Aircraft      ATTACK Sorties Available          Escort Aircraft
                                                       Type   Role  Sorties
+----+----+--------+     +----+----+--------+          F16    D     3
|Type|Role|Sorties |     |Type|Role|Sorties |
+----+----+--------+     +----+----+--------+
|111 |A   |20      |     |111 |A   |86      |
|    |    |        |     |F16 |A   |309     |
|    |    |        |     |F4  |A   |75      |           DSUP Aircraft
|    |    |        |     |MIB |A   |145     |         Type   Role  Sorties
+----+----+--------+     +----+----+--------+         AV8    A     6


  HELP  for Help
  PF3   to Move Cursor to the Line Number Field          ECM Aircraft
  FIND  to End Input and Commit This Mission           Type   Role  Sorties
  PF4   to Delete the Field the Cursor is on           E3A    E     1
  PF2   to Clear All Entries on the Screen
  DO    to Scroll the Sorties Available Window
  F14   to Remove the Mission from the Planning File
  F10   to Return to the Overall ATAF Planning Menu

  AC Avail-Primary(F11)  ESC(F12)  DSUP(F13)  ECM(F17)
```

Figure 2. Application Using VT200 Terminal



```
Day 0       DAY Cycle   2ATAF  Offensive Counter Air Mission Input
Mission Line Number: 1001          Target Number: 56     Side: BLUE

 Primary Aircraft       DSUP   Sorties Available        Escort Aircraft
                                                     Type    Role   Sorties
 Type  Role  Sorties     Type  Role  Sorties         F16     D      3

 111   A     20          111   A     86
                         AV8   A     65
                         F16   A     309                 DSUP Aircraft
                         F4    A     75              Type    Role   Sorties
                                                     AV8     A      6

  F1    for Help
  F3    to Move Cursor to the Line Number Field          ECM Aircraft
  sF10  to End Input and Commit This Mission         Type    Role   Sorties
  F4    to Delete the Field the Cursor is on         E3A     E      1
  F2    to Clear All Entries on the Screen
  F8    to Scroll the Sorties Available Window
  sF4   to Remove the Mission from the Planning File
  F10   to Return to the Overall ATAF Planning Menu

  AC Avail-Primary(ShfF1)  ESC(F12)  DSUP(F13)  ECM(^J)
```

Figure 3. Application Using PC/XT with VT200 Emulator

When a savepoint is reached a DIGITAL Command Language (DCL) file is submitted for batch processing. The DCL file contains instructions executed to copy out the tables which need to be saved. The files produced by the savepoint are stored under the directory of the specific exercise.

Every savepoint produces a group of files with the same file names as the previous group. Luckily, the multiuser operating system, VMS, does not overwrite files but saves the files under a new version number. Version numbers are critical when restoring a database to a specific savepoint.

Executing the savepoint instructions in a batch process eliminates the adverse effect of slowing down the user interface response. As pointed out by several authors the responsiveness of the user interface is important to an effective interface.

The code of those portions of the user interface which had slow response was analyzed to determine the bottlenecks. Output messages were inserted into the code at strategic locations to reveal areas to consider for recoding. A particularly significant problem area was identified as that portion of code which adjusted the number of available sorties for aircraft at airbases which can not fly on that particular day.

Over 200 tuples were created when calculating the available sorties. The tuples were held in a table field and a small number of tuples, less than 10, were changed before finally being loaded into the base table. To make the process more efficient the recoding changed the code to load the base table relation directly with the results of the calculations rather than storing them in the table field. The table field was then used to hold only tuples which identified tuples in the base table that require adjustment.

The assumption made prior to changing the code was that the current code produced valid results. Therefore, if the base table generated by the new code matches the base table generated by the old code then the code is valid.

Changing the code significantly improved performance. Originally, the time to calculate the available sorties was approximately 120 seconds when the system was operating with no other users. Under the same conditions, the new code calculated the available sorties in approximately 25 seconds or less.

Employing a surrogate key was another alternative considered to improve the efficiency. Using a surrogate key did not prove any more effective than just changing the code which calculated the available sorties. Both methods were tested to determine if either produced a significant improvement. Time did not allow for extensive testing, however, preliminary results showed no significant improvement in using the surrogate key method over recoding method.

A third alternative combined the two methods. Using a surrogate key with the rewritten code showed no significant improvement over the rewritten code only. Considering the additional overhead of creating and maintaining surrogate keys, the idea of implementing surrogate keys was dropped.

2.4 Summary

The problems addressed in this chapter center on the tolerance of the TWX user interface to environmental changes. This is considered as one of the principles of ergonomic software. Ergonomics associated with computers have generally focused on computer hardware. The goal of ergonomics is to improve performance by reducing negative factors (e.g., fatigue, frustration, boredom) and increase positive factors (e.g., motivation, satisfaction). All these factors have an impact on the results achieved by the user [11].

The principle, "Maximize Tolerance for Environmental Change", recognizes the importance of a software ability to adapt to new technology and take advantage of the technology's capabilities [11]. Problems dealt with in this chapter were examples of the interface's tolerance to change.

III. Controller Interface

3.1 Problem

The game controller performs exercise management functions to ensure the seminar exercises operate smoothly. These functions include creating new seminars, maintaining, monitoring, and controlling existing seminars, and backing up and restoring exercises. A basic, easy to use interface was employed to access the various controller functions. However, the controller interface provided only the rudimentary functions, which were not developed enough for the controller to perform effectively and efficiently.

A significant drawback affecting the controller's efficiency and effectiveness was the limited capability to access only one database at a time. Some of the controller functions were performed regularly on all the databases. This means repeating the same actions for each database. If the controller was allowed access to several databases simultaneously, redundant actions would be eliminated and controller efficiency improved. The controller's effectiveness to manage and control the seminar databases is also influenced by this limitation. As the number of seminars increase, the task of managing seminars individually becomes more difficult and prone to error or neglect.

Another area in which the controller is underdeveloped is in the area of backing up and restoring exercises. Backups of an exercise are manually performed by the game controller. Thus restoring an exercise is limited to whenever a backup was performed. The manual backup process is also inefficient because every table, including the static tables, are backed up in the process.

Finally, the area of managing and monitoring has been neglected in the game controller interface. The only monitoring capability provided by the controller was the ability to determine the current function an exercise was executing and the total time in that function. This is inadequate to manage and monitor exercises effectively.

3.2 Analysis

The initial TWX controller function consisted of the fundamental functions that were required to manage, and control the database seminars. These functions and functions not currently incorporated in the controller were analyzed to identify areas requiring improvement.

A major area of improvement was transforming the controller's access capability from one database to multiple databases. Converting the controller from accessing seminar databases one at a time to simultaneous access of several databases required that two essential conditions exist. First, the controller and seminar databases must exist on a network to allow access to the databases by the controller. This means that the seminar databases can exist on any computer as long as an access channel is available from the controller to the database. Secondly, the DBMS used by the seminar databases must support the access of local databases by a remote site. In other words, the DBMS must support the functions of a distributed database.

A distributed database refers to a database divided into distinct parts stored at separate locations. The different locations are connected together in a communications network by links. Each location of the distributed database, also referred to as a site or node, resembles an individual, self-contained database. The seminar databases represent the nodes of the distributed database.

A special node on the network, called the coordinator node, maintains all information pertaining to the distribution of data in the system. The coordinator node also controls addition and deletion of nodes and controls access of information between nodes. The controller database represents the coordinator node of the distributed database.

From the user point of view the overall distributed database should appear no different than a centralized database. When accessing data, it is the responsibility of the system to know where the data is located and not the user. This is referred to as location transparency. The method used to process requests is determined by the system and transparent to the user. Since the underlying database system controls query processing, then application programming is simplified. Location transparency also allows for reorganization of data without the need to change application programs.

The distributed database is managed at two levels, the distributed level and the local level. Each site has a local database manager that is responsible for managing the local information. The distributed database manager is responsible for managing the communication between the sites of the system. Queries for information at remote sites are converted to subqueries by the distributed database manager and sent to the local database manager of the site where the information resides. The local database manager executes the subquery and passes the results back to the distributed database manager. Subquery results received by the distributed database manager are consolidated and passed to the user [2].

The requirement dictates that the controller have access to the seminar databases. However, the seminar databases do not require access to the controller or other seminar databases. Therefore, links are created for access in only one direction, from the controller to the seminar databases.

Links to the seminar databases are vital to the proper management and control of the exercises. Consequently, the necessary links are automatically created when a seminar database is created. Conversely, when a seminar database is destroyed the links are no longer needed, so the links are also automatically destroyed when a seminar database is destroyed.

Another seminar management function of the controller is backing up and restoring of databases. Originally, the controller manually backed up and restored seminars. This function was expanded to restore databases which are automatically backed up at predetermined savepoints. Databases backed up at the savepoints differ from the manually backed up databases. Only those database tables which change are backed up at a savepoint, which means the restore process will also differ. Restoring to a savepoint requires that values of tables that change are replaced with the values at the time the savepoint was executed.

While the current controller interface provided a monitoring function, it did not provide the necessary information to analyze the progress of the seminars. A submenu was added to the controller which provided the game controller access to information on a seminar's air effectiveness and apportionment history. This information provides the game controller an overview of the seminar's progress and strategy. The importance and function of these statistical functions are discussed fully in the chapter on statistical functions.

3.3 Solution

A bottom-up approach was used to develop a distributed database system for the TWX controller function. In this approach, the existing seminar databases are aggregated to form the distributed database [8]. Each seminar database is autonomous and contains all the information needed to function. Aggregating the seminar databases into a distributed database was implemented without altering the seminar database structure.

The DBMS chosen for the TWX system was Ingres. Ingres is a well developed relational database system which consists of many subsystems. Two Ingres subsystems vital to developing a distributed database controller are Ingres/Net and Ingres/Star. Ingres/Net provides the means to communicate over the network [1]. While Ingres/Star provides the mechanisms to establish and use a distributed database [2].

The controller user interface consists of applications developed using the Fourth Generation Language (4GL) of the Ingres Application Development system. Figure 4 shows a feature chart of the controller interface. To maintain a modular structure, each application addresses a general area. Within an application there are frames which perform specific functions in that general area. Some of the functions required access to only one database at a time, while others required simultaneous access to more than one database. One such specific function was required to set the access flags simultaneously in all the seminar databases.

In order for the controller to access the other database in a distributed fashion, the controller had to be associated with a database. The controller database is a small database which maintains operating information. This includes the current databases and links to access them. Links between the controller database and the seminar databases were required for the controller to access the other databases. Figure 5 shows the configuration of the distributed database with links to seminar databases.

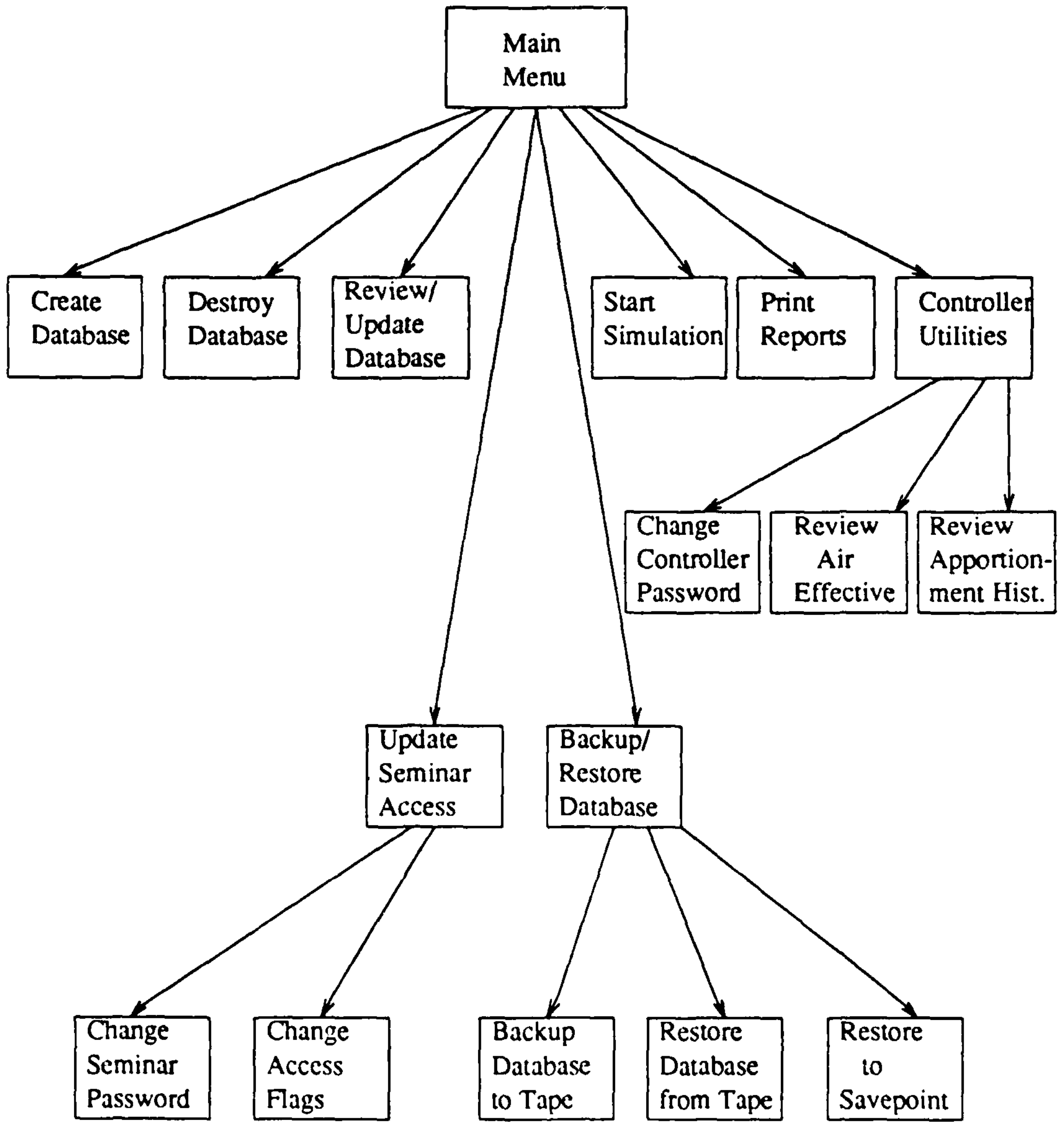


Figure 4. Feature Chart of the Controller Interface

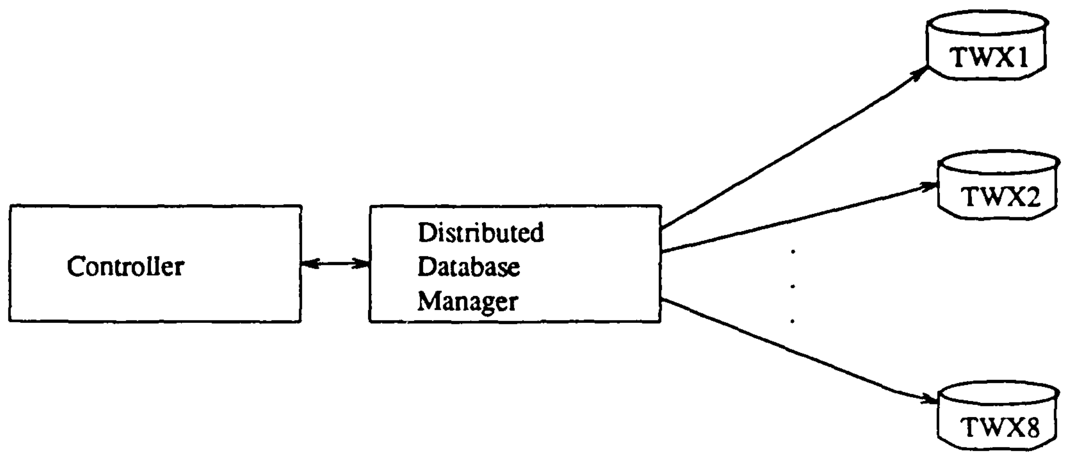


Figure 5. Distributed Database Controller Configuration

The links were created using Ingres/Star by specifying the node location of the database, the database name, and the name of the table to access. One link accesses one table of one database. Thus, links were only created for those tables which the controller must access to manage and control the seminar. Of course, the controller can always access any table of a seminar database by using the Query-By-Forms (QBF) subsystem, but this limits the controller access to only one database at a time.

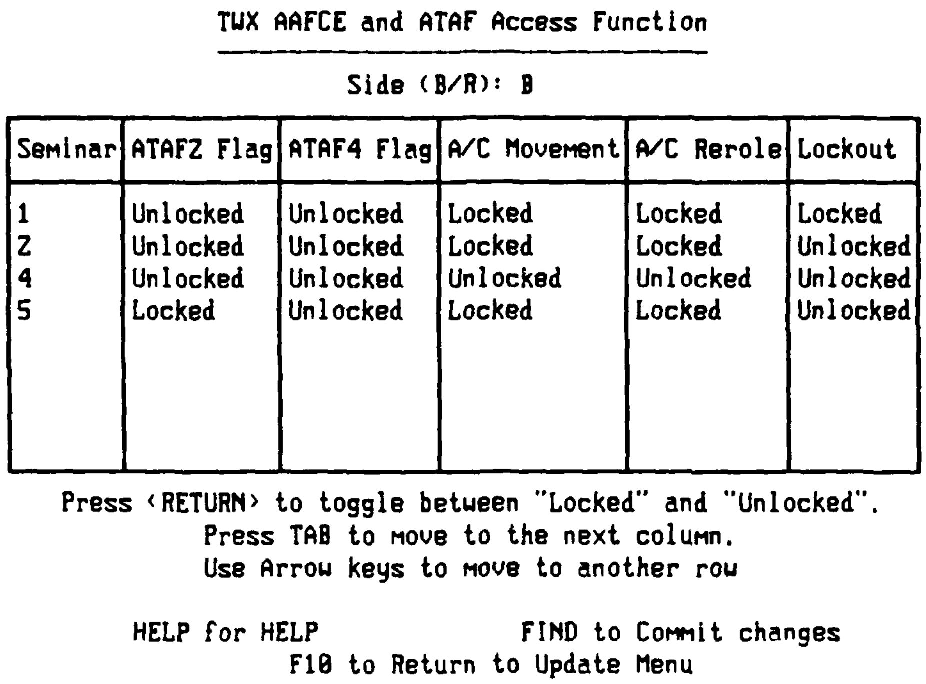
Even though the query language SQL (Structured Query Language) was used for most of the application, the Ingres query language QUEL was required to communicate through the established links. SQL would not allow the use of variables in the applications to specify links. This was essential, otherwise the link names would have to be "hard coded" into the 4GL code. Therefore QUEL was used to develop applications because it did allow the flexibility to use variable to define linknames. Within the applications the linkname to a specified seminar database was constructed in the code based on known information. A naming standard for the linknames was employed to ensure the correct linkname was constructed by the application. For example, links were necessary to access the seminar control table of each side in a seminar. These control tables are *bl_sem_con* for the Blue side and *rd_sem_con* for the Red side. The naming standard employed for these linknames requires the first letter designating the side, either "R" or "B". The next set of letters designate the purpose of the link, which in this case is "ACCS". "ACCS" means access because through this link the controller manages user access to portions of the seminar. Finally, the linkname ends with the number of the seminar. All seminar numbers are unique and a seminar consists of only one Red side and one Blue side. Therefore, the link names created will be unique.

One of the management functions of the controller is to govern access of the teams to the two levels of a seminar, the Allied Air Forces Central Europe (AAFCE) level and the Allied Tactical Air Force (ATAF) level. Objectives and strategic planning are performed at the AAFCE level. Mission planning to meet objectives determined at the AAFCE level is performed at the ATAF level.

The function for controlling access to these levels was developed to allow the controller to simultaneously monitor and control the flags of all the seminars. The game controller accesses this function from the main controller menu. Another menu is displayed which provides the user the option to change AAFCE and ATAF access flags, or the seminar access flag.

If the user selects to change the AAFCE and ATAF access flags a form is displayed with all the current seminars and the current status of the access flags. The access flags are the ATAF2 input flag, the ATAF4 input flag, the AAFCE aircraft movement flag, the AAFCE aircraft rerole flag, and overall access to the seminar. On the form displayed (Figure 6), the user controls the access flag by first positioning the cursor on the row of the selected seminar using the arrow keys. Next, the user positions the cursor on the desired column using the TAB key. Finally, the user presses the RETURN key to toggle the access between "Locked" and "Unlocked". When a flag is "Unlocked" the seminar user can then access the level associated with the flag. On the other hand, when a flag is "Locked" the seminar user is locked out of that level.

While most of the flags control access to only portions of the seminar, the lockout flag controls access to the overall seminar. When locked, all seminar users for that particular side are denied access to the seminar. When finished, this flag and all the other flags are



TWX AAFCE and ATAF Access Function

Side (B/R): B

Seminar	ATAF2 Flag	ATAF4 Flag	A/C Movement	A/C Rerole	Lockout
1	Unlocked	Unlocked	Locked	Locked	Locked
2	Unlocked	Unlocked	Locked	Locked	Unlocked
4	Unlocked	Unlocked	Unlocked	Unlocked	Unlocked
5	Locked	Unlocked	Locked	Locked	Unlocked

Press <RETURN> to toggle between "Locked" and "Unlocked".
Press TAB to move to the next column.
Use Arrow keys to move to another row

HELP for HELP   FIND to Commit changes
F10 to Return to Update Menu

Figure 6. Seminar Access Menu

updated in the seminar databases when the game controller commits the locks and exits the function.

Controller functions, such as creating or destroying a database, take several minutes to complete. Instead of performing such functions interactively, these functions utilized the batch processing capability of the Micro VAX III system. Once the task is initiated, the game controller is allowed to carry out other tasks. The instructions for these tasks, creating or destroying a database, must be directed to a specific database. The VAX/VMS DIGITAL Command Language (DCL) was used to "build" the specific instructions to carry out the task. DCL allows symbols, which are the same as variables, to be used in a command line which is stored in a file with a .COM extension. The arguments which represent the values of the symbols are specified when the file is executed or submitted for batch processing.

The command procedure which creates a new seminar database performs several functions besides just executing a command instructing Ingres to create the new database. Commands are executed to set up the directory and subdirectory to store files specific to the database. It also constructs QUEL script files which when executed establish the necessary links with the controller distributed database. Finally, the command procedure deletes files which are no longer needed.

Another function which used DCL files was the restore function. This function restores databases to one of the savepoints, previously discussed in the chapter concerning flexibility. During restore, only the controller should be able to access the database to ensure the restored database is in a consistent state. This means all other users need to be locked out of the database and the controller must have an exclusive lock on the database.

Without an exclusive lock on the database, users could be attempting to change tables that would be destroyed or making changes to new tables which are not appropriate.

The restore function gives the controller the flexibility to restore both sides or just one side of a database. However, there are restrictions on restoring only one side of a database. To ensure consistency when restoring only one side, the savepoint must be the same day as the current day. For example, after mission planning one of the seminar sides may contain gross planning errors. In this case only one of the sides needs to be restored to reaccomplish mission planning if the simulation has not run yet.

Using interface design guidelines, each function offers the user the option to abort the operation. Furthermore, changes made by the controller are not executed without the explicit instruction of the user. This prevents unexpected and undesirable results.

3.4 Summary

Distributed database systems have the capacity to expand as necessary to meet the needs of the users. Centralized databases are limited to the capacity of the system on which it resides. Since a distributed database system consists of a network of systems, the system can expand incrementally by adding systems to the network as needed. Of course growth is limited to the network capacity. This is essential for the controller to effectively manage and control the exercises.

The enhanced design of the controller user interface provides greater flexibility, efficiency, and control. Modular design of the interface provides a modifiable strucrture for further enhancements.

IV. Expert User Mode

4.1 Problem

Considering the characteristics of the user is very important when designing a user interface. The original user interface had been developed with the characteristics of one type of user in mind, the novice. While this makes the interface easy to learn and use for the novice it can be a source of frustration for the experienced user who knows what he or she wants to do. The functionality of the user interface needs to be developed further to also satisfy the needs of the more experienced user.

Faculty members which usually play the part of the opposing side are generally considered experienced users. Through playing experience, they may know exactly what data they want to load into the game. Following the step by step instructions of the current interface would be time consuming and frustrating. Therefore, an expert user mode is needed to provide experienced users greater control and flexibility.

4.2 Analysis

The way the program operates should be compatible with several user characteristics. Program results and the user's expected results should be compatible. A useful program is compatible with the user's needs. The needs of an experienced TWX user, such as a faculty member, are different from those of the novice user on a seminar team. Interaction with the application should be at a level compatible with the user's level of experience and understanding [19].

Consideration of the experience level of the projected user and the task to be performed is essential. The type of user and the type of task are factors which will determine the interface design [19]. An important part of the user interface which should accommodate the different levels of experience is the data entry phase of mission planning. For the novice, this phase is a step by step process with extensive error checking and easy error correction. However, this may not be too cumbersome for the experienced user. Therefore, the interface design should include an expert mode for the data entry phase.

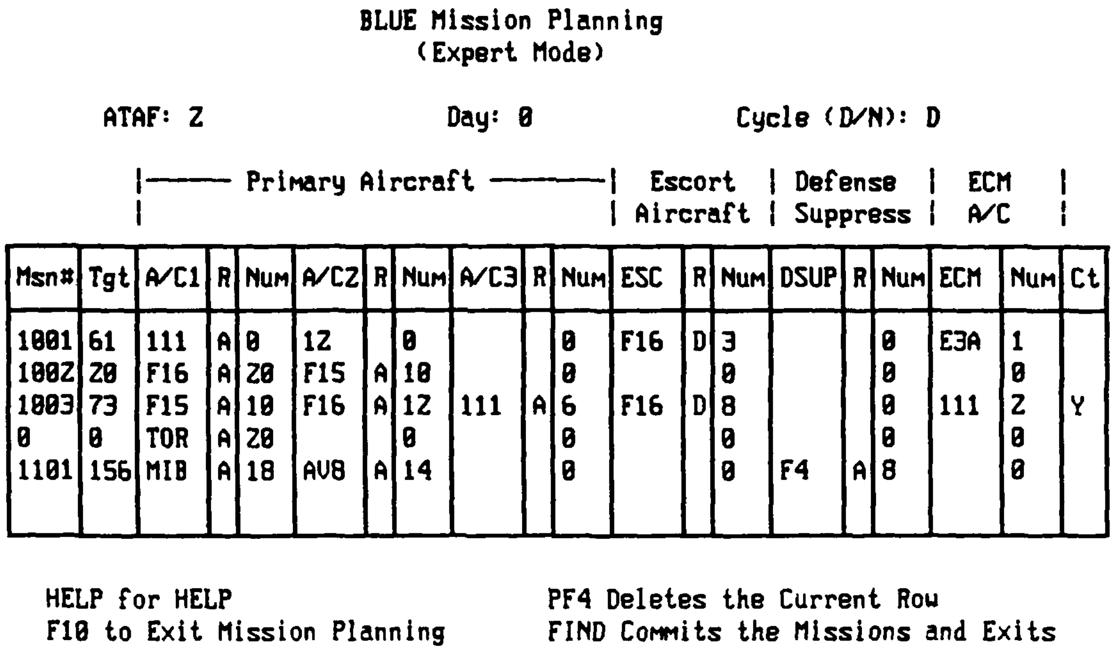
An important design decision influenced by these factors is selecting the appropriate dialogue (form fill-in, menu-driven, or command line) to use in the interface. Matching the dialogue, also known as the interaction style, to the type of user guards against frustrating the novice and boring the experienced user [19]. TWX actually employs a combination of menu-driven and fill-in dialogues. Though, as mentioned earlier, the fill-in portions of TWX used by less experienced users have extensive error checking. The expert mode also uses fill-in, but differs because the error checking is not performed until all the data is entered. Furthermore, the error checking is not as in depth in the expert mode.

The form fill-in interaction style displays labels with fields in which the user enters the appropriate data corresponding to the label. Form fill-in interfaces require the user to have an understanding of the label's meaning and knowledge of what values are valid for a given field [17]. Experienced users do not need as much information in the label as the novice. More information can be entered by the user on a screen with abbreviated labels. This in turn allows the experienced user to complete the data entry task quicker.

4.3 Solution

The current user interface was further modified to allow optional access to an expert mode during mission planning. Under normal circumstances seminar participants are not experienced enough to use this mode. The expert mode is reserved for the experienced user, such as faculty members, because the error checking is not as extensive as the normal user mode. Also, error checking that is performed is only rendered after all data entry is completed.

In the expert user mode, the user inputs mission data into a table as shown by Figure 7. Valid input for each portion of the table is the responsibility of the user. For example, it is the responsibility of the user to know the range of valid mission numbers for each type of mission. When a mission is initially entered the whole row is active. If the user requires more than three different primary aircraft than this is indicated in the last column of the row with a "Y". The cursor drops down to the next row, positioned at the first primary aircraft column. Mission number and target number are skipped since this is just a continuation of the mission. While the user is able to enter more types of



BLUE Mission Planning
(Expert Mode)

ATAF: 2 Day: 0 Cycle (D/N): D

Primary Aircraft											Escort Aircraft			Defense Suppress			ECM A/C		
Msn#	Tgt	A/C1	R	Num	A/C2	R	Num	A/C3	R	Num	ESC	R	Num	DSUP	R	Num	ECM	Num	Ct
1001	61	111	A	0	12		0			0	F16	D	3			0	E3A	1	
1002	20	F16	A	20	F15	A	10			0			0			0		0	
1003	73	F15	A	10	F16	A	12	111	A	6	F16	D	8			0	111	2	Y
0	0	TOR	A	20			0			0			0			0		0	
1101	156	MIB	A	18	AV8	A	14			0			0	F4	A	8		0	

HELP for HELP PF4 Deletes the Current Row
F10 to Exit Mission Planning FIND Commits the Missions and Exits

Figure 7. Expert Mode Mission Input Table

primary aircraft, the user can not enter additional different support aircraft. Therefore, these columns are skipped also.

The form was easily developed with the Visual-Forms-Editor (VIFRED) of Ingres. Using VIFRED, the developer visually creates the form and assigns attributes to the form fields. Associated with the form is 4GL code which defines the function of the form. This includes controlling the positioning of the cursor.

When mission planning in the expert mode is completed the user selects the commit function to insert the information into the database. Prior to committing the information, validation checks are made to ensure legal data is inserted into the database. Errors are flagged and no data is committed to the database until either the errors are corrected or the mission with the errors is deleted from the form. Once the data is committed, the user is returned to the main menu of ATAF mission planning.

Currently the expert mode is limited to input of new data only. However, this does not deny the user from making changes to data once the missions are committed to the database. The user can retrieve and modify missions input with the expert mode in the normal mission planning interface.

4.4 Summary

A good design for a user interface takes into consideration the level of experience of the user. The TWX design required modifications to the area where users desire more control as they develop more experience. Specifically, this is in the area of entering mission planning data.

The expert mode of mission planning allows the user greater control but also places more responsibility on the user. Players on the Red side are usually experienced in TWX and need a fast, simple method to enter mission data.

V. Statistical Routines

5.1 Problem

Prior to rehosting the TWX game to a new environment, game controllers were provided with an aggregation of relevant information on the seminar's progress. The statistical routines which aggregated the information from seminar data were not included in the rehosting effort. Therefore, statistics would have to be gathered manually, if at all. This would be a labor intensive task better suited for a computer.

TWX is an environment for students to apply force management decisions based on principles of doctrine and strategy learned in the classroom. Maintaining historical data on a seminar provides important information for analysis of a team's progress and a means to evaluate effectiveness of the teaching. Aggregation of carefully selected areas of information would provide the necessary information needed in such an evaluation.

5.2 Analysis

The incorporation of statistical routines into TWX has two major objectives. First, to maintain statistical data on selected portions of a seminar to provide measurements for evaluation of a seminar team's performance. Secondly, to provide data to the team in an aggregated format in much the same way a decision maker may receive data in a real situation.

The statistical routines of the TWX user interface interactively support seminar teams during force employment. Effective statistics require defining the type of information that is important to the decision making process. Not only must the statistic be effective, it must also be valid for the environment portrayed by TWX. In other words, only statistics which might realistically be available in a conflict environment would be available.

Adapting Fox's paradigm for model usage to statistic usage, Figure 8 shows the phases for using TWX results to aid the decision making process [10]. Once the TWX simulation has run, pertinent results, along with previous results, are aggregated into statistics. In the next phase the statistics are interpreted by the seminar teams. The

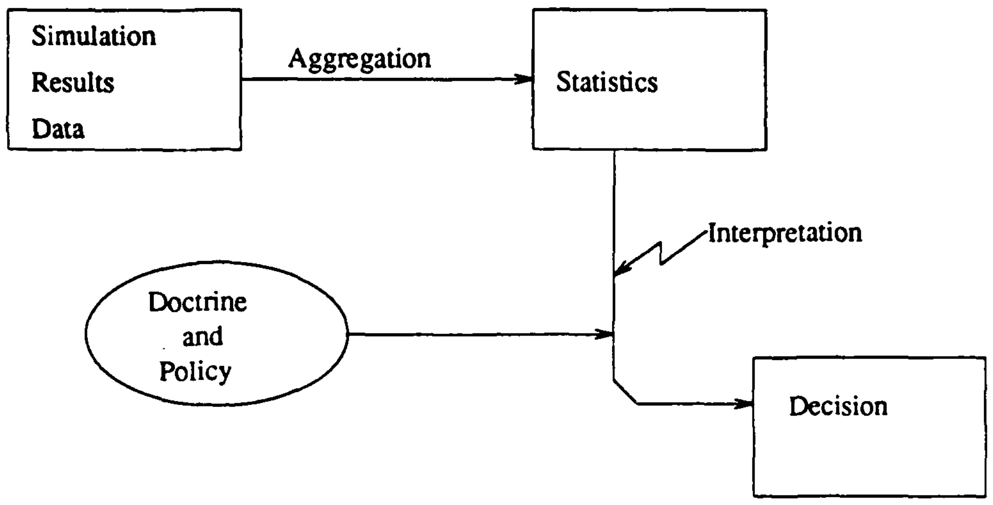


Figure 8. Paradigm of Statistical Usage

interpretation process is influenced by the doctrine and policies taught which govern force employment. Decisions and choices result from the interpretation process.

While several statistics were considered, only four were implemented because of the time constraint. These statistics cover several techniques of implementation in the user interface. They provide examples for incorporating additional statistics into the interface. The statistics implemented were an aggregation of the status of friendly and enemy airbases, air effectiveness ratio, and apportionment. The following is a discussion of the statistic's benefit and the process used to calculate each statistic.

Base Status. Base status statistics cover two areas, status of friendly bases and status of enemy bases, which are very similar. Both statistics maintain a count of the number of bases in three damage categories, light, moderate, and heavy. The difference lies in the criteria used for calculating which category an airbase falls into based on the base status.

Light damage of friendly airbases is considered any airbase with a base status of .66 or greater. Airbases with a base status between .35 and .65 are categorized as having moderate damage. While airbases with a base status between 0 and .34 are categorized as heavy damage.

The status of the enemy's airbases is calculated in much the same way. The only differences between the two are that enemy airbases with a base status between .40 and .65 have moderate damage and between 0 and .39 have heavy damage. Since determining the enemy's airbase status relies on reconnaissance intelligence the values may or may not be totally accurate, at least that is what is conveyed to the seminar team. Accuracy of the values is based upon comparison of the airbase intelligence index and a reliability factor.

Air Effectiveness. The air effectiveness statistic is used by the seminar administrators to evaluate how well seminar participants are employing air power against the opponent. This statistic maintains air effectiveness information on the Red and Blue forces of a seminar.

Before the conflict starts the total number of aircraft for each side is determined. This includes aircraft on airbases and available for augmentation.

Each day, including Day 0, the force ratio of Red aircraft available to Blue aircraft available is calculated. The following is an example of how the force ratio is calculated:

```
Red aircraft available:  5634
Blue aircraft available:  3234
Ratio: ((5634/3234) / 1)  =  1.74 / 1
```

This ratio indicates that the Red side has an aircraft advantage of 1.74 aircraft to every 1 aircraft of the Blue side.

During the exercise, other than Day 0, the number of aircraft available and the number of aircraft losses for each side are recalculated for each day. Using the new aircraft figures and the previous day's force, the Daily Exchange Ratio (DER), the Daily Air Exchange Effectiveness Index (DAEEI), and force ratio are determined. Figure 9 shows the formulas used to calculate these ratios.

The DER is a ratio of the number of Red aircraft losses and the number of Blue aircraft losses for that particular conflict day. DAEEI provides a quick check of how well the war is going in respect to the how the losses of each side compare to each other. If the DAEEI is equal to 1.00 then the losses are such that each side will run out of aircraft on

```
DER = Red losses / Blue losses
DAEEI = DER / Previous force ratio
Force ratio = Red aircraft / Blue aircraft
```

Figure 9. DER, DAEEI, and Force Ratio Formulas

the same day. If the DAEEI is greater than 1.00 then Red is losing aircraft at a rate faster than the Blue losses. On the other hand, if DAEEI is less than 1.00 then Blue losses are at a greater rate then Red.

Apportionment. Apportionment is an important concept in mission planning. Apportionment of the forces should reflect an effort to achieve the planned objectives. The Theater Warfare Exercise handbook defines apportionment as:

> "the determination and assignment of the expected effort by priority and/or percentages devoted to the various air missions." [3:19]

This statistic maintains a history of the projected apportionment, the scheduled apportionment, and the effective apportionment for each day. Apportionment history allows the seminar team to evaluate how well their mission planning and execution matches their planned objectives.

Projected apportionment is entered prior to mission planning and reflects the percentage of missions to be scheduled to meet air directive objectives. Scheduled apportionment indicates the actual percentage of missions scheduled in each apportionment category. Finally, effective apportionment indicates the actual percentage of missions flown that reached their target.

5.3 Solution

Incorporating statistics into TWX required modifying both the user interface and adding statistic gathering subroutines to the simulation. Modifications to the interface were made to input projected apportionment values and to display statistical information in a usable format.

The statistic gathering subroutines are a set of subroutines which are executed at the end of the simulation program. However, the subroutine are independent of the simulation because the subroutines do not use the global variables of the simulation. There are two reasons for developing the statistical gathering subroutines independent of each other and separate from the simulation. First, separating into independent modules follows software engineering principles for software development. Secondly, without direct ties to the simulation the subroutines were developed and tested without having to execute the simulation.

For consistency with the simulation program of TWX, the subroutines were written in Fortran. However, the emphasis in the subroutines was on using embedded SQL to retrieve data as needed rather than maintaining data in global arrays. Using this approach supports maintainability and will simplify conversion to another language at a later date, if necessary.

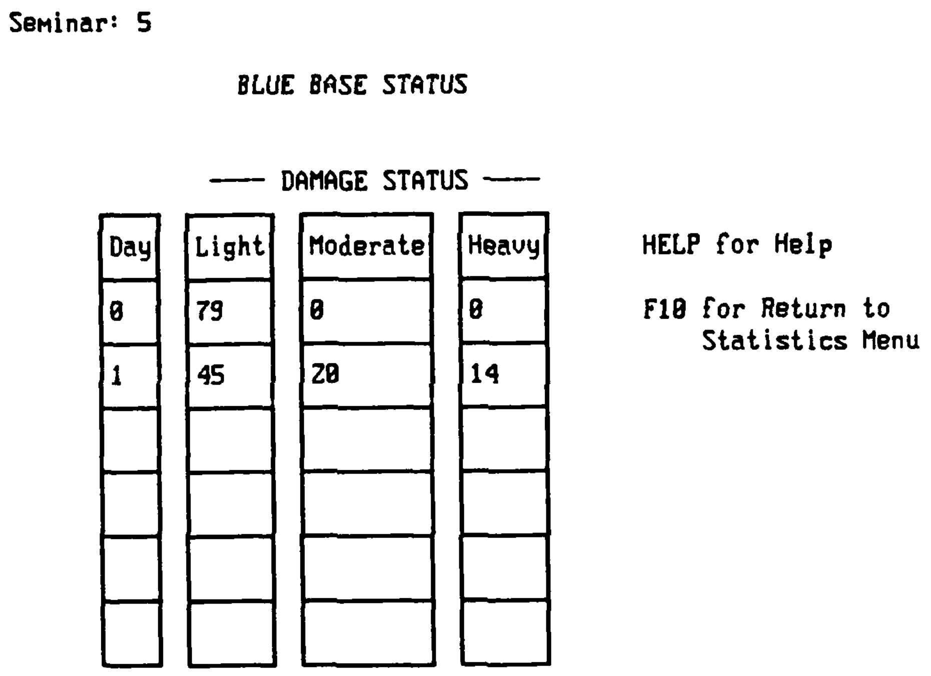
Base Status. The aggregated base status information for each side is maintained in tables added to the database, *bl_base_stat* for the Blue side and *rd_base_stat* for the Red side. Other tables added to the database, *bl_enmy_abst* and *rd_enmy_abst*, maintain a status history on the enemy's airbases.

The subroutine which inserts base status information into the base status table extracts and aggregates the information from the airbase tables. A tuple is inserted into the base status table for each of the categories, light, moderate, and heavy.

The user interface has been modified to allow the user to review the aggregated base status information for each conflict day up to the present. Figure 10 shows the base status display the user can access from the AAFCE main menu.

The subroutine which calculates the enemy's airbase status extracts and aggregates the information from the opponents airbase table. Information on the enemy's airbase differs because the accuracy of the information is based on the airbase's intelligence index and a reliability factor.

An airbase's intelligence index is a numeric value on a scale of 0 to 2. The value 1 means the intelligence on that airbase is totally accurate. If the intelligence index is



Seminar: 5

BLUE BASE STATUS

—— DAMAGE STATUS ——

Day	Light	Moderate	Heavy
0	79	0	0
1	45	20	14

HELP for Help

F10 for Return to
Statistics Menu

Figure 10. Friendly Airbase Status Display

less than 1 than the airbase's capabilities are underestimated. Likewise, if the intelligence index is greater than 1 then the airbase's capabilities are overestimated.

The reliability factor is a constant of the TWX game which is set by the game controller. It determines the range around totally accurate intelligence (1) in which the information is considered to be reliable. For example, if the reliability factor is set at .2 then the range of the intelligence index in which information is considered reliable would be .8 to 1.2.

To determine the accuracy of information on an airbase, the airbase's intelligence index is first normalized by subtracting 1 from the value. The absolute value of the result is then compared to the reliability factor. If the absolute value of the result is less than or equal to the reliability factor then the information is considered accurate.

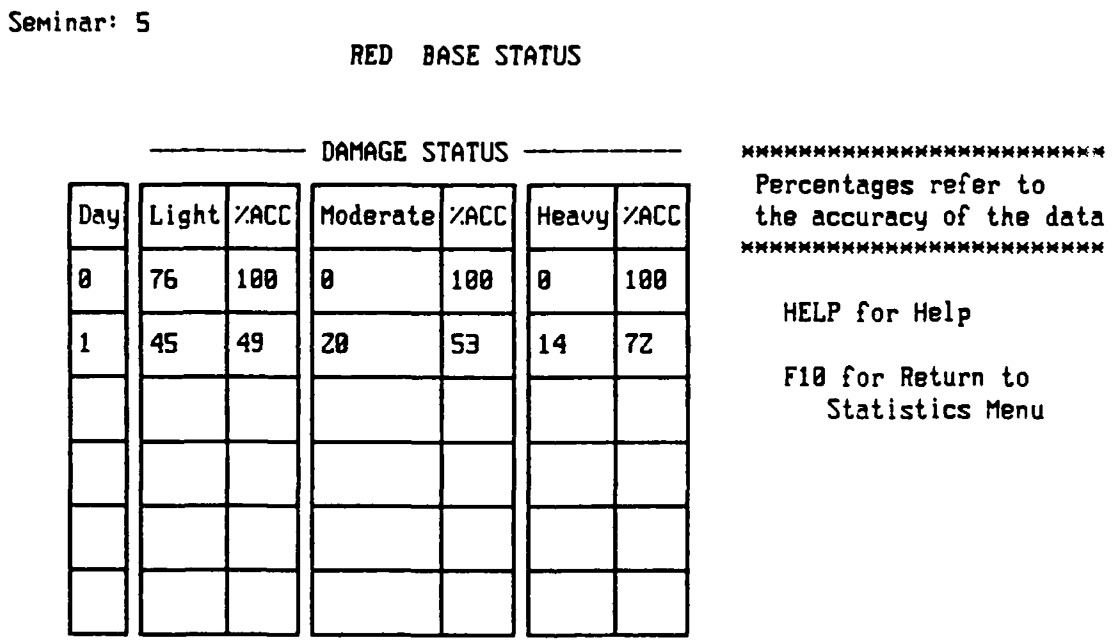
If the condition is true then the information is considered accurate, otherwise the information is unreliable. For example, if the intelligence index is 1.040 and the reliability factor is 0.200 then using the above method the absolute value of .040 is less than the reliability factor. This indicates the information on the base is reliable. An example of unreliable information would be if the intelligence index is .6. In this case, the absolute value of the normalized result is .4 which is greater than the reliability factor.

The count of the number of true conditions for each damage category is determined to indicate what percentage of each category is considered accurate information. For example, if the subroutine calculated that 50 airbases had light damage but reliability condition was true only 25 times then the result would have an accuracy of only 50%.

The user can access the aggregated information on the opponent's airbases (Figure 11) from a statistical information menu. This is the same menu which allows the user to review information on friendly airbases and apportionment history.

Air Effectiveness. This is a statistic gathered on each exercise but used only by the controller. Therefore, the information is gathered and maintained with the seminar and accessed through the controller user interface.

Each seminar has a table in the database which maintains the number of aircraft available and the losses for each conflict day for its own side. The tables are named



Seminar: 5

RED BASE STATUS

DAMAGE STATUS

Day	Light	%ACC	Moderate	%ACC	Heavy	%ACC
0	76	100	0	100	0	100
1	45	49	20	53	14	72

****
Percentages refer to
the accuracy of the data
****

HELP for Help

F10 for Return to
Statistics Menu

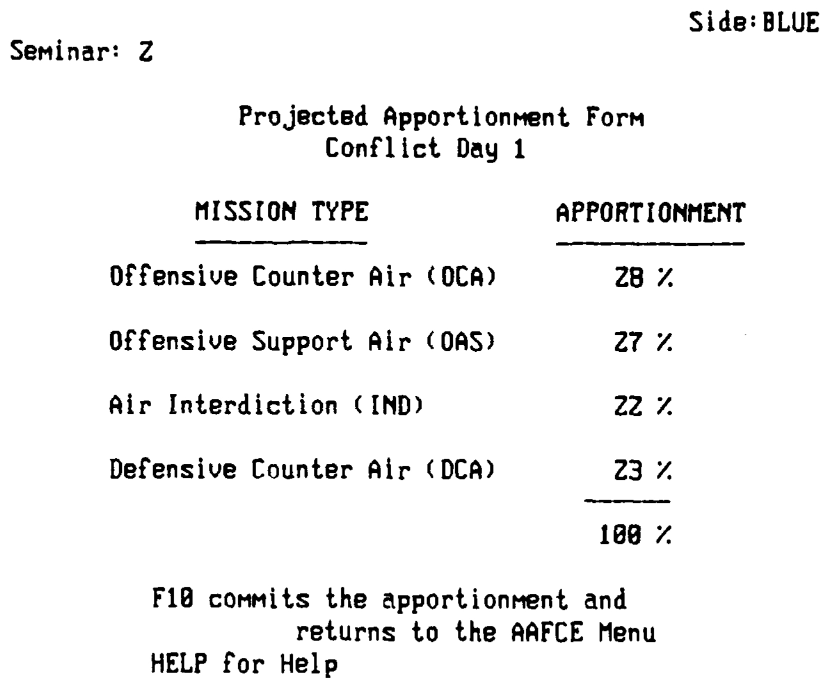
Figure 11. Enemy Airbase Status Display

bl_air_eff for the Blue team and *rd_air_eff* for the Red team. An application within the controller function uses the information from the tables to calculate the DER, DAEEI, Force Ratio, and indicate which side has the numerical advantage.

A new tuple with the appropriate conflict day, the remaining number of aircraft, and the number of aircraft lost that conflict day is added to the tables by a Fortran subroutine. The subroutine is executed after the simulation. First the air effectiveness subroutine calculates the total number of aircraft available for each side by summing the number of aircraft on each airbase and the number of aircraft available for augmentation. Since the subroutine executes after the simulation program is over, the database tables which hold information on the number of aircraft on each base have already been updated to reflect any losses. Losses for that day are calculated by using the total number of aircraft value stored in the air effectiveness database table for the previous conflict day. The new total number of aircraft is deducted from the previous day's total resulting in the number of aircraft lost. Finally, a new tuple with the calculated values is inserted into the air effectiveness database table.

The game controller accesses the information in the tables from the controller utilities option. In this option the user specifies which seminar to review. Because of the controller's distributed database capabilities, described in Chapter 3, the air effectiveness information is retrieved from the seminar database and displayed.

Apportionment. As decribed earlier, apportionment goes through three phases in a conflict day. First, each day of the exercise students are required to input the projected apportionment for that day. Seminar users access the projected apportionment form (Figure 12) from the AAFCE main menu. The form lists the four apportionment categories: Offensive Counter Air (OCA), Offensive Air Support (OAS), Air Interdiction (IND), and Defensive Counter Air (DCA). Apportionment percentages are entered for each category such that the total adds up to 100running total is maintained for the user. Users are not permitted to commit the apportionment unless the total is exactly 100the user can change the projected apportionment values only as long as mission planning has not begun. Students can not enter either ATAF for mission planning unless projected apportionment values are entered.



Side:BLUE

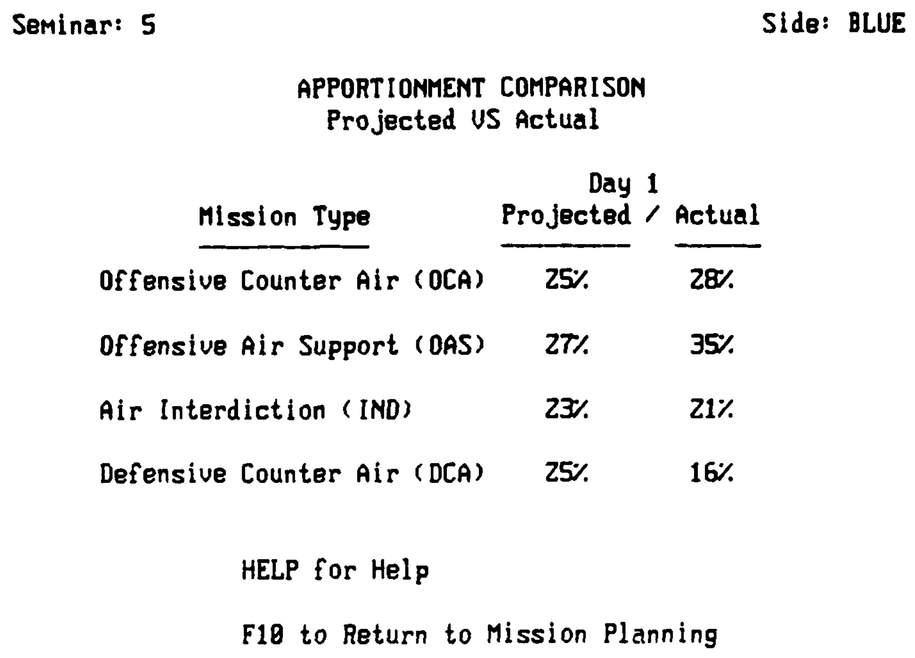
Seminar: 2

Projected Apportionment Form
Conflict Day 1

MISSION TYPE	APPORTIONMENT
Offensive Counter Air (OCA)	28 %
Offensive Support Air (OAS)	27 %
Air Interdiction (IND)	22 %
Defensive Counter Air (DCA)	23 %
	100 %

```
F10 commits the apportionment and
         returns to the AAFCE Menu
HELP for Help
```

Figure 12. Apportionment Input Form



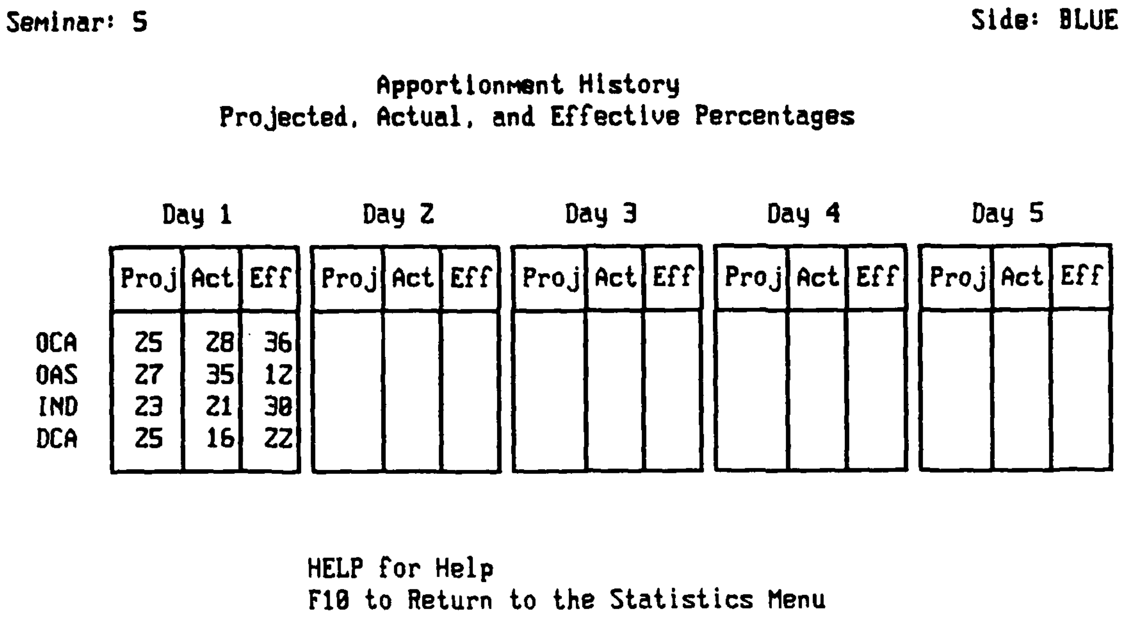
```
Seminar: 5                                             Side: BLUE

                    APPORTIONMENT COMPARISON
                       Projected VS Actual

                                          Day 1
              Mission Type          Projected / Actual
              ____________          _________   ______

     Offensive Counter Air (OCA)       25%        28%

     Offensive Air Support (OAS)       27%        35%

     Air Interdiction (IND)            23%        21%

     Defensive Counter Air (DCA)       25%        16%


                HELP for Help

                F10 to Return to Mission Planning
```

Figure 13. Apportionment Comparison Display

A flag is set in the *bl_sem_con*, or *rd_sem_con* for the Red side, to prevent going back into the AAFCE menu to change apportionment percentages. The flag is also used to indicate whether the projected values have been entered. The seminar team would be restricted from doing mission planning at the ATAF level until the projected values are entered.

The second phase of apportionment is during mission planning. In this phase actual apportionment values are calculated as the seminar team enters missions. Any time during mission planning a seminar team can invoke a display (Figure 13) which compares the projected apportionment percentages with the actual apportionment percentages.

The third phase of apportionment is the effective apportionment which is calculated after the simulation has executed and the seminar database updated with the results. Effective apportionment differs from actual apportionment in that missions which were aborted in any way are not included in the apportionment calculation. The aborts deducted



Seminar: 5 Side: BLUE

Apportionment History
Projected, Actual, and Effective Percentages

	Day 1			Day 2			Day 3			Day 4			Day 5		
	Proj	Act	Eff	Proj	Act	Eff	Proj	Act	Eff	Proj	Act	Eff	Proj	Act	Eff
OCA	25	28	36												
OAS	27	35	12												
IND	23	21	30												
DCA	25	16	22												

HELP for Help
F10 to Return to the Statistics Menu

Figure 14. Apportionment History Display

from the number of sorties scheduled are regular aborts, pms aborts, weather aborts, and jettisons. This provides an accurate picture of the apportionment for the four mission categories.

A submenu of the AAFCE main menu allows the user to view a history of the apportionment categories for each day of the exercise (Figure 14). Here users can compare the projected, scheduled, and actual apportionments.

5.4 Summary

Implementation of statistical routines in TWX involved two changes to TWX. The changes include incorporating statistical gathering subroutines into the simulation and modifying the user interface to display the statistical information.

While the information is displayed mostly in table format, graphical display of the information was considered. After performing tests with the business graphics portion of Ingres the idea was discarded. Severe limitations were discovered when attempting to tailor the graphics to the TWX application.

VI. Integration

6.1 Problem

The current TWX system was a result of two thesis efforts which rehosted the TWX system to a new environment and developed a new user interface. Both efforts produced complex systems which were consolidated into a single system. However, due to time constraints both efforts were not fully integrated together. Most of the integration problems were minor compared to the complexity of the two efforts. For example, at one point the user interface attempted to access an attribute of a database table that was not in the proper format. Even these minor problems illustrate the necessity of integration. Besides the user interface not functioning properly without full integration, the controller function portion of the TWX system could not properly manage and control the seminar databases. A lack of full integration contributes to results which are unpredictable and uncontrollable.

Another integration problem requiring resolution was the incorporation of a user interface for the Red side. Interface support for the Red side of the exercise was required to make the overall system usable. The thesis which developed the current seminar user interface concentrated only on an interface for the Blue side. The interface for the Red side was to be a copy of the Blue interface with all references to Blue database tables changed to Red. While this initially appeared satisfactory, after further consideration this method presented some problems. First, making copies of the interface programs for the Red side would double the amount of storage occupied by the interface. This leads to the second problem, which is interface code that becomes more difficult to maintain. Changes to the code would have to be done twice, once for the Blue side and once for the Red side, rather than once if Red and Blue share the same interface code.

Using the same interface code to support both sides presents a problem in itself. The code must be able to identify the user when applications are entered without having to ask the user to reenter their seminar side. Furthermore, the user's identification must be unique to keep track of the type of terminal being used to take advantage of the increased flexibility.

6.2 Analysis

Consolidating TWX into an operational system required the full integration of a user interface for both sides, the seminar database, the controller function, and the air battle simulation programs. The modular design structure of TWX not only reduced the system to manageable modules but also reduced the complexity of integration. Any integration problems discovered were easily narrowed down to specific modules.

The process of full integration required modifying the user interface to support both sides of the seminar. Each side, Blue and Red, performed the exact same steps in exercise planning and implementation. This meant that the current Blue interface could be modified to include access by the Red side.

Working with existing software rather than developing a new system from scratch involved the software maintenance aspect of software engineering. Marca refers to maintenance as "successive repetitions of software development activities" [15:21].

Since software maintenance is considered an iteration of the software development cycle, then all the phases of software development apply when making modifications. Modifications to the interface were analyzed, designed, implemented, and tested. The quality of the system after changes is definitely affected by the maintenance approach. If each iteration follows a structured software methodology, then the quality of the software will not deteriorate. As stated by Marca, "Poor maintenance practices can turn a good system into a bad one" [15:22].

Two types of software maintenance activities were undertaken in this thesis problem. The first activity, called adaptive maintenance, refers to modifying the software to accommodate changes in the computing environment. Perfective maintenance, the second activity, refers to modifications of a software system to provide enhancements and new capabilities [15].

All maintenance activities involved using the Ingres application development system, Application-By-Forms (ABF). ABF promoted a modular structure because it allowed independent development and testing of small tasks called frames. Related frames were grouped together to form a specific application area. To maintain modularity, applications are restricted from passing data by means of a global variable to other applications. The visual forms editor and 4GL of ABF significantly contributed to successfully implementing a maintainable and modifiable system.

Screen displays were easily developed using the visual forms editor. This tool allows the programmer to place fixed text and variable fields directly on the screen. The programmer is relieved of the tedious task of writing the screen display code. Values placed in fields on the screen and operations executed by specific function keys was controlled by the 4GL code.

Ingres 4GL is a powerful programming language used in creating the database applications. 4GL provides many of the same features found in other programming languages, such as comparisons and arithmetic operations. However, it significantly differs from other languages because commands specific to 4GL support a higher level of abstraction for development of an interactive system. Solutions are easier to develop in 4GL and existing code is easier to understand.

6.3 Solution

The interface code was modified to allow both sides to use the same interface programs. However, the biggest problem was how to determine which side was accessing an application. Solving this problem required being able to distinguish the user from all other users. When a user logs into a session, a process is started on the system. This process has a unique number, referred to as the process identification, associated with it. VMS has a system command which will provide the process identification of the current process. However, as revealed by testing, each time a new application is entered a new process is spawned and consequently a new process identification. Fortunately, VMS has a system command which provided identification of the master process.

The application programs of Ingres were required to retrieve the master process identification from the system. This required the programs to call an external procedure to get the value from the system. The value must then be passed back to the application. Ingres' Application-By-Forms (ABF) allows external procedures of high order language to be called and values passed into applications from the procedures. The high order languages supported by ABF include Fortran, Pascal, C, COBOL, PL/1, and Ada. Since the simulation was already written in Fortran, this high order language was chosen for the subroutine.

Within all applications a call is made to the Fortran procedure for the master process identification. The Fortran procedure retrieves the information from the system using the lexical function LIB$GETJPI. This information was then passed back to the application.

After a user has successfully logged into a session, the master process identification and the side specified by the user are written to a special database table. Whenever the user enters a new application the master process identification retrieved from the system determines which side the interface should service. The type of terminal being used is also kept track of with the other information to identify the user's type of terminal and ensure the function keys of each application adapt for the specific type of terminal.

All references in the application programs to database tables for the Blue side were duplicated to include the Red side tables. Conditional statement were added to the programs which tested which side was using the program and control which tables to access. Even though code which referenced a specific side had to be duplicated, the significant portion of the code was side independent and did not require duplication.

Master process identifications are unique and usually different every time a session is started. The information stored in the database on a process identification is useless when the user quits a session. Therefore, whenever a user logs out of session the tuple with the information is removed from the database table.

6.4 Summary

Integration is essential for the complex TWX system to function properly. Applying the software engineering principles throughout the process of requirements analysis, design, and implementation helps achieve the essential integration.

VII. Conclusion and Recommendations

7.1 Summary

The main focus of this thesis has been on extending the user interface of TWX. Extending the TWX user interface was based on five areas of the user interface identified as problems: flexibility and efficiency, controller interface, expert user mode, statistical functions, and integration.

The TWX user interface included two types of interfaces, the seminar interface and the game controller interface. The seminar user interface is used to analyze, plan, and implement strategies for airpower employment in the exercise environment. The game controller interface is used to manage and control the exercises. While each interface accomplished different tasks, many of the same user interface design guidelines were used in both. Such guidelines included consistent and logical displays, consistent language, accurate instructions, simple error recovery procedures, and helpful error messages.

Extending the user interface involved changing the existing user interface rather than developing an interface from scratch. The original structure design of the database and user interface resulting from the work of Brooks and Kross proved to be easily expanded in the Ingres development environment. The modular design of TWX provided an easily maintainable and modifiable structure for performing software maintenance. Using proven software principles in modifications minimized effects on other portions of the system.

While the design of the database and user interface clearly plays the most significant role in maintainability and modifiability, the availablity of tools also contributed to successfully expanding TWX. ABF provided an environment which enabled rapid development, testing and debugging. The programming language of Ingres, 4GL, has many high level functions built into the language which made it powerful and easy to use. The conclusion is that the availability of integrated tools in the development environment plays an important part in the development of a complex system.

7.2 Recommendations for Further Work

Moving the application programs of the user interface to the Micro VAX III provides an opportunity to further expand the capabilities of TWX. Three specific areas of expansion are the automating of the Red side, developing an interface to allow input for the land battle, and incorporating warnings to the seminar team about exercise conditions based on the current situation.

There may be two levels of automation of the Red side considered. First, the Red side may be automated to relieve the Air Force Wargaming Center faculty of the repetitive task of reentering the same data for each exercise. This type of automation would follow a predetermined plan for each day of the conflict.

The second level considered would incorporate artificial intelligence in the automated Red side. Objectives and implementation strategy could be determined by artificial intelligence based on Red doctrine and the current situation. Mission planning would then reflect the results of the determination.

A user interface needs to be developed for the recently upgraded land battle simulation. The interface would perform the planning and execution of objectives for the land units. This would further enhance TWX's role as a tool to learn employing airpower in a theater level conflict.

Another suggested area of research would be incorporating the ability to warn the seminar team of a potentially disastrous situation. Warnings such as logistic shortfalls, surrounded or out numbered land units, or airbases in danger of being overrun would be based on the current data and situation. The seminar team is then put in the situation of deciding a course of action or accepting the risk.

Bibliography

1. *Ingres/Net Reference Manual.* Relational Technology Inc., Alameda, California, 1986.
2. *Ingres/Star Reference Manual.* Relational Technology Inc., Alameda, California, 1986.
3. *Theater War Exercise User's Handbook.* Air Force Wargaming Center, Maxwell AFB, AL, 1987. Unpublished Manual.
4. Leonard J. Bass and Ralph E. Bunker. A Generalized User Interface for Applications Programs. In *Tutorial: End User Facilities in the 1980's*, pages 465–469, IEEE Computer Society Press, December 1982.
5. Bernard H. Boar. *Application Prototyping.* John Wiley & Sons, New York, 1984.
6. Captain Michael D. Brooks. *Developing a Database Management System and Air Simulation Software for a Theater War Exercise (ADA189681).* Master's thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1987. AFIT/GCS/ENG/87D-6.
7. Dawn M. Cappelli. *The Key to the Future: Implementation of Surrogate Keys in Relational Database Design.* Technical Report, Westinghouse R&D Center, Engineering Service Bureau, Westinghouse Generation Technology System Division, Pittsburgh, November 1987.
8. Stefano Ceri, Barbara Pernici, and Gio Wiederhold. An Overview of Research in the Design of Distributed Databases. In *IEEE Database Engineering*, pages 227–232, IEEE Press, 1984.
9. C. J. Date. *An Introduction to Database Systems, Vol II.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.
10. Lt. Col. Daniel B. Fox. *A Conceptual Design for a Model to Meet the War-Gaming Needs of the Major Commands of the United States Air Force.* Technical Report AU-ARI-84-8, Airpower Research Institute, Air University Press, Maxwell AFB, AL, July 1985.
11. Diana L. Knittle, Stephen Ruth, and Ella Paton Gardner. Establishing User-Centered Criteria for Information Systems: A Software Ergonomics Perspective. *Information & Management*, 11:163–172, November 1986.
12. Henry F. Korth and Abraham Silberschatz. *Stabase System Concepts.* McGraw-Hill Book Company, New York, 1987.
13. Captain Mark S. Kross. *Developing New User Interfaces for the Theater War Exercise (ADA189744).* Master's thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1987. AFIT/GCS/ENG/87D-19.
14. Ting-peng Liang. User Interface Design for Decision Support Systems: An Adaptive Approach. *Information & Management*, 12:181–193, April 1987.
15. David Marca. *Applying Software Engineering Principles.* Little, Brown and Company, Boston, 1984.

16. Roger S. Pressman. *Software Engineering: A Practitioner's Approach.* McGraw-Hill Book Company, New York, 1987.

17. Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1987.

18. Norwood Sission. Dialogue Management Reference Model. *SIGCHI Bulletin*, 18:34–35, October 1986.

19. Jean-Claude Sperandio. Software Ergonomics of Interface Design. *Behavior and Information Technology*, 6:271–278, 1987.

Vita

Captain Kenneth R. Wilcox was born o[redacted]to Mr and Mrs Joseph Wilcox. He graduated from South Dade Senior High School in 1971. In 1976 he enlisted into the AIr Force. After 4 years on active duty, Captain Wilcox was selected for AFIT's Airman Education Commissioning Program. He attended the University of South Florida and received a BS in Information Systems in 1983. Following Officer Training School, he was assigned to Tactical Air Command Headquarters in the Tactical Communication Division. Captain Wilcox served as an information systems requirements staff officer.

[redacted]

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

A202 726

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/88D-24	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 Center; 8b. OFFICE SYMBOL (If applicable) AUCADRE/WG	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Maxwell AFB, AL 36112-5532	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
EXTENDING THE USER INTERFACE FOR THE THEATER WAR EXERCISE (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Kenneth R. Wilcox, Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS thesis	FROM ____ TO ____	1988 December	68

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	User Interface, Wargaming Database, Prototyping
12	05		

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

Thesis Advisor: Mark A. Roth, Captain, USAF
Assistant Professor of Electrical Engineering and Computer Science

[illegible]
10 Jan 89

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT.] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Mark A. Roth, Captain, 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

The Theater War Exercise (TWX) is a wargaming simulation of airpower employment in a European theater conflict. TWX simulates a realistic five day conflict in which senior Air Force officers must make command decisions for airpower employment. The wargame serves as a learning tool to provide an environment in which the officers apply airpower employment concepts and principles of war taught in the classroom.

The current TWX system was the result of previous thesis efforts which rehosted TWX to a microcomputer environment and developed a flexible user interface. Both efforts were performed independently using a common commerical relational database management system. The completed works were to be combined into an integrated system.

The goal of this thesis was to complete the integration of the previous efforts and extend the capabilities of TWX to improve its effectiveness as a learning tool. This was accomplished by first evaluating the current system and identifying areas which required improvement. Specific areas that required attention included the need for better flexibility and efficiency in the user interface, an enhanced controller interface, an expert user mode for the user interface, statistical routines, and thorough integration of the TWX subsystems and enhancements. This was accomplished using the integrated tools of the Ingres development system in a multiuser computer environment.

All modifications and enhancements resulting from this thesis were related to the interfaces of the controller function and the seminar users. Though not all interface modifications are readily apparent to the user, the changes improved the user/TWX system interaction and increased the flexibility of the total system. Improvements to the TWX system resulted from a combination of changes to the database organization, modification of the application programs, reworking code tc improve performance, and enhancements.