9950-1054

RDA-TR-185500-005

FINAL REPORT: AUTOMATED CPX SUPPORT SYSTEM PRELIMINARY DESIGN PHASE

MARCH 1984

(NASA-CR-175784) AUTOMATED CPX SUPPORT SYSTEM PRELIMINARY DESIGN PHASE Final Report (R and D Associates, Marina Del Rey, Calif.) 241 p HC A11/MF A01 CSCL 17B

N85-27091

Unclas
G3/32 22636

By:
T. A. BORDEAUX
E. T. CARSON
C. D. HEPBURN
F. M. SHINNICK

Submitted To:
JET PROPULSION LABORATORY of the
CALIFORNIA INSTITUTE OF TECHNOLOGY
4800 Oak Grove Drive
Pasadena, CA 91103

CONTRACT NO. 956622

R & D ASSOCIATES
Post Office Box 9695
Marina del Rey,
California 90291

4640 ADMIRALTY WAY • MARINA DEL REY • TELEPHONE: (213) 822-1715

RDA-TR-185500-005

FINAL REPORT: AUTOMATED CPX SUPPORT SYSTEM PRELIMINARY DESIGN PHASE

MARCH 1984

By:
T. A. BORDEAUX
E. T. CARSON
C. D. HEPBURN
F. M. SHINNICK

Submitted To:
JET PROPULSION LABORATORY of the
CALIFORNIA INSTITUTE OF TECHNOLOGY
4800 Oak Grove Drive
Pasadena, CA 91103

R & D ASSOCIATES
Post Office Box 9695
Marina del Rey,
California 90291

4640 ADMIRALTY WAY • MARINA DEL REY • TELEPHONE: (213) 822-1715

SUMMARY

RDA is developing the Distributed Command and Control System (DCCS) for the 9th Infantry Division at Ft. Lewis, Washington, which is also the headquarters of the I Corps, the parent organization. The development of an automated C^2 system stimulated additional interest at I Corps in developing an automated CPX support system in order to provide a more realistic stimulus to DCCS than could be achieved with the existing manual system. The Jet Propulsion Laboratory contracted RDA to perform a preliminary design of an automated CPX system to support corps-level exercises. The period of performance was two months, and the total effort corresponded roughly to one labor year, including a subcontract from RDA to Technical Solutions Incorporated of El Paso. The effort comprised four tasks: collecting and documenting user requirements, developing a preliminary system design, defining a program plan, and evaluating the suitability of the TRASANA FOURCE computer model.

We collected user requirements through technical discussions with I Corps and III Corps. We prepared draft descriptions that were modified in response to comments from these organizations. This cycle was repeated several times. We also considered the requirements described in a USREDCOM document. Our final report on this task, which is bound as the first part of this document, highlights points of similarity and difference among the sets of requirements. USREDCOM is primarily concerned with exercising staff procedures for joint missions. I Corps is concerned with issues of tactics, equipment, and procedures related to the High Technology Test Bed (HTTB). Therefore, some of the detailed I Corps requirements are not germane to the USREDCOM mission.

In the second task, we performed functional analysis of the requirements and derived a preliminary system design. We restricted this analysis to the I Corps subset of the requirements; for example, we did not examine the USREDCOM requirements to support an Air Force wing. We developed a functional design, defined a distributed processing architecture, estimated the number of work stations needed to support the various groups of role players, and selected example computer components as a basis for estimating costs. The final report for this task is bound as the second section of this document.

In the third task, we developed a preliminary program plan to aid decision makers in scoping the potential effort. We recommend an evolutionary approach that provides incremental levels of capability. In this way the users gain early experience in working with an automated system, thereby developing insights into the potential impact of such a system on their exercise plans and procedures. Furthermore, the developer benefits by using feedback from the users to improve the system design. The final report for this task is bound as the third section of this document.

In the fourth task, we evaluated the potential suitability of the Command, Control, Communications, and Combat Effectiveness (FOURCE) model as a kernel for the combat simulation program within a CPX support system. FOURCE is a systemic division-level model that has been used extensively for Cost and Operational Effectiveness Analysis (COEA) in both the United States and the United Kingdom. FOURCE differs from other models in its explicit representation of the message flow among echelons, which could be useful in developing the stimulus to the staff being exercised. We performed a high-level analysis of the FOURCE models and software structure, developed a technical approach for adapting FOURCE to an interactive environment, and evaluated in detail one example modification. Our final report on this task is bound as the fourth section of this document. RDA retained Technical Solutions Incorporated (TSI) to provide an independent assessment of the effort needed to modify FOURCE for interactive use and to extend it in selected areas. The TSI final report, which was produced under subcontract to RDA, is bound as a separate report.

The following reports are bound sequentially in this document:

1. Bordeaux, T. A., User Requirements for an Automated CPX Support System, RDA-TR-185500-001, October 1983.

2. Bordeaux, T. A., Carson, E. T., Shinnick, F. M., Preliminary Design for an Automated CPX Support System, RDA-TR-185500-002, October 1983.

3. Bordeaux, T. A., Carson, E. T., Shinnick, F. M., Preliminary Program Plan for Development of an Automated CPX Support System, RDA-TR-185500-004, October 1983.

4. Carson, E. T., Hepburn, C. D., Modifying TRASANA FOURCE for Interactive Use, RDA-TR-185500-003, October 1983.

RDA-TR-185500-001

USER REQUIREMENTS FOR AN AUTOMATED CPX SUPPORT SYSTEM

OCTOBER 1983

By:
T. A. BORDEAUX

Submitted To:
JET PROPULSION LABORATORY of the
CALIFORNIA INSTITUTE OF TECHNOLOGY
4800 Oak Grove Drive
Pasadena, CA 91103

R & D ASSOCIATES
Post Office Box 9695
Marina del Rey,
California, 90291

4640 ADMIRALTY WAY • MARINA DEL REY • TELEPHONE: (213) 822-1715

PREFACE

This is the first of a series of reports on a two-month effort to conduct a preliminary design of an automated CPX support system. The effort was performed for the Jet Propulsion Laboratory of the California Institute of Technology under Contract No. 956622.

PRECEDING PAGE BLANK NOT FILMED

CONTENTS

PRECEDING PAGE BLANK NOT FILMED

USER REQUIREMENTS FOR AN AUTOMATED CPX SUPPORT SYSTEM

1. INTRODUCTION

Current methods of conducting CPXs require inordinately large numbers of personnel to support the exercise relative to the numbers being exercised. Most of the support personnel get little (if any) mission training value from the exercise. Significant effort is devoted to interfacing with the computer system supporting exercise play. Another major deficiency in current CPX play is the low degree of realism in impacts of logistics and personnel decisions on the tactical situation. This results in inadequate attention being paid to logistical and personnel factors in tactical decision-making.

It is believed that the state of the art in computers and modeling of combat processes will support significant advances in the conduct of CPXs. An automated CPX support system (ACSS) could provide a realistic, economical capability to train personnel in proper staff procedures. Minimizing personnel support requirements can encourage more frequent exercises. These can be used not only for additional training but, depending on user interests and the design of the system, for evaluation of new equipment, tactics, organizations and operational concepts.

This report documents the results of one part of an effort to develop a preliminary design of an automated CPX support system. It contains a description of the requirements for such a system as perceived by three different potential users--U.S. Readiness Command, I Corps and III Corps. I and

III Corps requirements have been found to be virtually identical with some minor but not incompatible differences resulting from differences in experiences and orientation of the exercise staffs. Some USREDCOM requirements are significantly different from those of an Army Corps because it has additional units which must be exercised as part of their training for participation as part of a joint force and because I/III Corps have training requirements at a lower organizational level than does REDCOM. Where unique requirements are discussed herein, the affected organizations are indicated at the beginning of the paragraph in parentheses, e.g., (I/III Corps).

The user requirements as perceived by I and III Corps were developed in an iterative process of discussing them with Corps representatives, documenting the results of those discussions, having the Corps representatives review the documentation, and repeating the process. (There were three iterations with I Corps and one with III Corps on the final draft.)

USREDCOM had set forth their requirements in two documents: "Systems Requirements, Joint Exercise Support System (JESS), Operations/Combat Employment Modeling" and "Systems Requirements, Joint Exercise Support System (JESS) Logistics Module". The contents of these two documents were compared with the final draft of the I/III Corps user requirements to merge the ground force requirements which, while not identical, were complementary.

1.1 The ACSS Concept

The ACSS consists of a computer system with one or more processors, software, displays, and peripheral equipment such as printers. Simulation programs that are part of the computer software represent the events occurring on the battlefield and their outcomes. Events simulated for Blue and Orange include unit movements, combat actions with personnel and equipment losses, intelligence gathering, resupply, etc. The system provides appropriate activity and status reports to "player-controllers" who act as the subordinate units to the Blue echelons being exercised and as the Orange opposing force.* The player-controllers interact with the automated support system to receive activity and status information from the system and provide reports to the exercising units in accordance with field SOP. They also direct the actions of lower echelon units in the simulation in response to exercising unit direction. The players; i.e., the units being trained during the CPX, receive reports in SOP form, perform staff functions, make decisions and issue orders as though they were in an actual combat environment. It is emphasized that in this terminology, players interact with player-controllers but not with the automated support system.

There are additional players in a joint exercise not present in a Corps-level exercise. The major additional ones include the Joint Force Headquarters, the Air Force Navy and Marine elements of the joint force, and the Area Air Defense Command. Depending on the echelon of the Air Force element, e.g., numbered air force, there also could be subordinate echelon staffs to be trained such as air divisions and/or wings.

*Other player-controllers may be employed to represent superior echelons and/or adjacent forces; however, they do not interact with the simulation system. Still other player-controllers may be used to represent other participants such as Air Force elements. They may or may not interact with the simulation system.

2. REQUIRED SYSTEM CHARACTERISTICS

The following paragraphs delineate specific attributes required of a system for assisting staff training and for evaluating new operational and organizational concepts, new equipment, joint and combined tactics, techniques and procedures. Where requirements are not common to all three agencies, those that are unique are so identified.

2.1 Organization Level Supported

The system to support I/III Corps must be able to provide exercise stimulus for a Blue corps or for a Blue division. When used to support a REDCOM joint exercise it must be able to provide stimulus for an Army element up to corps size and an Air Force element up to numbered air force size.

2.1.1 Corps level exercises will consist of a corps CP and CPs for up to four subordinate divisions plus corps artillery and COSCOM. The system must provide combat reporting inputs from as many as four subordinate brigades per division to the division CPs. The capability to represent independent brigades and battalions including Army aviation units also must be included. (An echelon above corps normally will be represented by player-controllers to provide an appropriate higher headquarters environment.)

2.1.2 (I/III Corps) Division level exercises will consist of a division CP plus CPs for up to four subordinate brigades including a Combat Brigade (Air Assault) plus DIVARTY and DISCOM. Corps will normally be represented to provide an appropriate higher headquarters environment. The system must provide appropriate combat reports from subordinate battalions to the brigade CPs and to DIVARTY.

2.1.3 (I/III Corps) The system must accommodate a division-level exercise being played within a corps-level exercise.

2.1.4 (REDCOM) Joint Force exercises will consist of a Joint Command CP, a Corps level exercise force (see 2.1.1), a numbered air force CP and CPs for up to four subordinate air wings in order to accommodate for example a fighter wing, an attack wing, a tactical transport wing and perhaps a recce squadron.

2.1.5 The system must accommodate friendly national forces being played as part of a combined force.

2.1.6 The system must accommodate the opposing force for a corps-level exercise being up to the equivalent size of a Soviet Front. The system must accommodate the simulation of a range of enemy forces from Warsaw Pact, SW Asian, SE Asian, NE Asian, South/Latin American or African countries.

2.1.7 (I/III Corps) The opposing force for a division-level exercise will be up to the equivalent size of a Soviet Army.

2.2 Unit Resolution

It is required that the system represent the battlefield activities of units at echelons sufficiently low to be sensitive to organizational composition and the tactics employed.

2.2.1 The smallest Blue unit represented will be company-size for maneuver units.* Artillery, ADA and other units must be represented to a level consistent with their normal employment. The smallest opposing force unit represented will be the equivalent of a maneuver battalion or an artillery battery.

2.3 Exercise Preparation

The system must be capable of being made ready by trained personnel for a specific exercise in less than one month and without excessive numbers of personnel. This requirement may be relaxed for the first play in a new geographic area or for a major change in rules, TOE, or performance data.

2.3.1 The system must allow for prior preparation, storage and change of data for both sides concerning TOEs; along with weapon, vehicle and target characteristics; and geographic, topographical and climatological factors for several exercise scenarios. It must allow for easy extraction of data relevant to the exercise and its input for the application. It also must allow for input of units at less than full strength with no significant additional effort and must accommodate non-standard, modified TOEs.

2.3.2 (I Corps) The system must accept simulation of developmental and even future hypothetical weapon systems in addition to currently operational ones.

*USREDCOM requires the capability to maneuver battalion/ separate company size units.

2.3.3 The support system shall be able to accommodate player units (DTOCs, CTOCs, etc.) being fully deployed in the field with conventional communications back to the support system facility.

2.3.4 Communication links with the support system will be monitored and recorded for post exercise review, scoring, and analysis.

2.4 Headquarters Functions Supported

The ACSS in conjunction with the role-players must be capable of providing appropriate stimulus to the functions within a command post being exercised in a given CPX. These functions are described below by Blue staff officer position. Orange functions are similar, though staff organization may not be.[*]

2.4.1 G1 (Personnel). The ACSS must be able to report unit strength, by MOS and category (O, WO, NCO, E) and changes in unit strength due to casualties (KIA, WIA, MIA, non-battle injuries) and replacements.

2.4.2 G2 (Intelligence). The system must be able to accept tasking of all division and corps intelligence and target acquisition assets, simulating the accomplishment of those tasks and reporting the results thereof. Such assets shall include but not be limited to ground-based radars, airbourne MTI systems, reconnaissance aircraft and RPVs, counter-battery radars and unattended ground sensors. The system also must

[*] In the following, S or J can replace G, as appropriate.

be able to provide representative weather data. The system must be able to provide appropriate order of battle data to TACSIM.

2.4.3 G3 (Operations). The system must accommodate organizational changes during an exercise including task force organizations, cross-attachments and changes of operational control. Unit movement upon order, contact with the enemy and direct and indirect fire must be simulated. The results of such battlefield activities must be reported to the appropriate role-players. Reports shall include personnel and equipment losses and status as well as supply consumption and status. Combat operations must be realistically affected in the simulation by the availability of ammunition and POL and by status of forces and equipment on each side.

2.4.4 G4 (Logistics). The system is to be able to maintain the status of equipment and supplies of the units controlled and represented by the role-players. It also must simulate resupply operations at the division-brigade and corps-division levels. Ammunition and POL resupply operations must be simulated discretely. Other supplies may be aggregated. Resupply of combat and combat support units shall not be automatic; i.e., it must occur only as a result of direction by personnel of the exercised units. The resupply process must be sensitive to any damage that may be inflicted by enemy attacks.

Maintenance requirements for ground and air vehicles must be generated both as a result of combat action and random failures. Repair of equipment at various echelons and return to service must be simulated with realistic time factors.

2.4.5 Army Aviation. It is required that Blue aviation activities be simulated in greater detail than for the opposing force in order to exercise the relevant staff functions. These functions require the ACSS to simulate responses to requests for army aviation missions. These can be fire missions, supply missions, troop movements, or observation missions. Assets will be controlled at the platoon level for Army helicopters and at the individual aircraft level for other Army assets.

2.4.6 Air Force Operations. It is required that Blue air force activities be simulated in greater detail than for the opposing force in order to exercise the relevant staff functions. The ACSS must simulate the accomplishment of assigned reconnaissance, close air support, battlefield and deep interdiction, offensive and defensive counter air, and radar and communications jamming missions at the direction of the friendly and opposing forces. It must accommodate composite strike forces. Provisions must be made for reflecting the capability of reconnaissance aircraft to detect relevant types of vehicles and installations. The system must be able to provide information on mission accomplishment such as fuel and munitions expended, reconnaissance information obtained, targets damaged/destroyed and friendly aircraft damaged/lost. Degree of accomplishment of combat missions must be sensitive to the Orange defensive threat array, munitions employed, on-board ECM, flown aircraft defensive tactics, ingress and egress routes, air refueling, maintenance and turn-around times, radar and communications jamming by standoff and escort aircraft, support aircraft employed, terrain, lighting and weather. The system must assess and report damage resulting from such attacks.

It is expected that Air Force units as well as Army units would be subjected to training in a REDCOM exercise. The presence of Air Force players implies a need for a level of operations simulation detail comparable to that provided for the Army element of the joint force. It also implies the presence of Air Force player-controllers to play the role of units subordinate to the lowest echelon Air Force unit being trained. On the other hand, actual Air Force units are not expected to participate as such in a Corps exercise. However, the Air Force might provide personnel to act as player-controllers to represent Air Force functions.

2.4.7 Air Defense. The system must be able to simulate the operations of the air defenses of each side against the other's sorties. Blue air defenses must be treated in greater detail than the opposing force in order to exercise relevant staff elements. The system shall account for and report munitions expenditures and equipment losses and personnel casualties due to enemy suppressive action.

2.4.8 Helicopter-Helicopter Combat. (I Corps) Provisions must be made in the ACSS for aerial combat between helicopters.

2.4.9 Nuclear, Biological and Chemical (NBC). The ACSS must be able to simulate the results of employment of nuclear and chemical weapons by each side considering terrain, weather and degree of protective measures taken. It must provide NBC contamination and fallout regions as well as casualties in response to weapon usage. Reports of NBC incidents must be provided to Blue role-players.

2.4.10 Engineer. Engineer functions must be simulated by the ACSS, including movement of appropriate engineer troops and materials to the site, construction (bridge, barrier, building etc.), and return to base if appropriate. Engineer units shall be capable of joining in direct combat when combat is related to engineer tasks or when engineer units are reorganized as infantry. When the engineer function is a significant exercise element, Class IV (construction materials) logistic supply also must be played.

2.4.11 Field Artillery. The ACSS must simulate the action of field artillery batteries (platoons in the case of 8" SP howitzers, individual weapons in the case of CSWS and Pershing, battalions in the case of Orange). ACSS must generate calls for fire from subordinate units, simulate responses to fire orders by artillery units, including rounds fired, outcome at the target, observation of outcome by own troops, and increase in detectability of artillery unit. Response to movement orders must be simulated. Provisions must be made for employing all appropriate artillery munitions including smoke and mines.

2.4.12 Transportation. The ACSS must simulate the flow of traffic--especially logistics traffic--generated on both sides of the FEBA. It must be sensitive to interdiction attacks (ground and air) and traffic flow must be degraded appropriately as a result of such attacks.

2.5 Conducting the Exercise

The system must yield the maximum mission training value to all personnel involved in the exercise. There must be a minimum of personnel required to merely run the system.

2.5.1 The capability shall exist to deny role-players and exercising units access to any information they would not normally have on a real battlefield.

2.5.2 Communications between exercised command posts must be conducted in accordance with field SOPs. Communications to and from the exercised command posts must appear to them to be in accordance with field SOPs.

2.5.3 Each brigade represented in a corps-level exercise must provide for accomplishing the commander, S2/S3, S1/S4, ADA, engineer and other special staff functions plus those of up to five subordinate battalion commanders. The system must accommodate these functions being accomplished by six to ten people depending on number of people available.

2.5.4 (I/III Corps) The system must accommodate each battalion in a division-level exercise being represented by two to four people, depending on availability of personnel.

2.5.5 The opposing force in a corps-level exercise must be capable of being represented by 25 people or less.

2.5.6 (I/III Corps) The opposing force in a division-level exercise must be capable of being represented by 12 people or less.

2.6 Exercise Control Functions

During the conduct of the exercise, the exercise director must be able to influence and/or modify the tactical situation.

2.6.1 (I/III Corps) The system must accommodate "turning the clock back"; i.e., returning to a prior point in the battle so that it can be replayed with different tactical decisions and must have provisions for reviewing prior exercise events without affecting the progress of the exercise.

2.6.2 (I/III Corps) The system must accommodate stopping play and restarting with new initial conditions; for example, to represent skipping forward in time or to change positions of selected units.

2.6.3 (I Corps) It must be possible to make authorized changes in input data such as equipment performance during the course of the exercise.

2.6.4 The controller function must be able to override the outcome of other decision processes; for example, to assure the most training value of an otherwise random event.

2.7 Representation of the Battlefield

The system must present for both sides an accurate portrayal of locations of units and combat interactions and outcomes as affected by terrain, weather, and day versus night.

2.7.1 The system must be capable of portraying operations on an integrated battlefield. The effects of nuclear and chemical as well as conventional weapons must be included when played.

2.7.2 The capability of portraying effects of terrain, smoke, inclement weather, temperature, lighting conditions, and MOPP gear on unit, personnel, weapons, and vehicle performance including travel speed must also be included.

2.7.3 The system must be capable of portraying the locations of maneuver, fire support, combat support, and combat service support units plus selected facilities such as command posts, supply points and airfields.

2.7.4 Rear area protection operations must be capable of being included as part of division and corps-level exercises.

2.7.5 Provisions must be included for simulating target acquisition by relevant organic sensors including reconnaissance aircraft.

2.7.6 Provisions must be included for simulated input of intelligence from resources above the echelons being exercised including National assets.

2.7.7 Automated graphic representation of the battlefield shall be supported via computer displays.

2.8 Logistics Effects

Logistics actions (or inactions) that would impact on the tactical situation in real life must have a like effect during the exercise; e.g., units out of fuel must stop, units out of ammunition must not be able to shoot.

2.8.1 Improper planning must result in shortages of supplies at any echelon or unit being represented or exercised and must have a realistic impact on the tactical situation. The minimum logistics play shall model POL (Category III) and ammunition (Category V) supplies, supply points and transportation.

2.8.2 Attacks on Blue or Orange logistics must affect the course of the battle appropriately.

2.8.3 Effects of reliability-availability-maintainability on force capability must be reflected.

2.9 Exercise Output

The system must provide for recording and storage of exercise data to support post-exercise analysis and evaluations.

2.9.1 The positions of units on the battlefield must be able to be recorded at intervals of not more than 30 minutes during the exercise. The status of maneuver units including conditions of forces and level of supplies must be able to be recorded at the same interval.

2.9.2 Hard copies of situation reports for supported units must be available from the system on request.

2.10 Other Requirements

The ACSS must be flexible in application so that a wide variety of CPX training and evaluation needs can be supported with changes only to the database and not to system hardware/ software.

2.10.1 (I/III Corps) It must be able to support exercises in which maneuver unit operations are emphasized with only nominal (but appropriate) logistics representation and vice versa.

2.10.2 (I/III Corps) The system must support the use of surrogate units; i.e., the system must have provisions for representing the staff and command elements of selected maneuver units without the use of personnel.

2.10.3 (I/III Corps) The ACSS system must be compatible with the normal use by the exercising units of automated command and control equipment and other organic automated systems such as personnel and logistics systems.

3. DESIRED SYSTEM CHARACTERISTICS

The following system characteristics while not required would significantly enhance the utility of the system.

3.1 Utility in Training Geographically Separated Units

It is desired that the system be readily moveable to conduct an exercise at any base that has adequate facilities. It also is desired that it be adaptable, with proper communications, to the support of exercises involving multiple divisions of a corps that geographically are widely separated.

3.2 Training Time

The system must allow for exercise personnel, i.e., non-cadre personnel, to be easily familiarized with their system equipment and trained in its operation. The desired objective is a maximum of one day.

RDA-TR-185500-002

PRELIMINARY DESIGN FOR AN AUTOMATED CPX SUPPORT SYSTEM

OCTOBER 1983

By:
T. A. BORDEAUX
E. T. CARSON
F. M. SHINNICK

Submitted To:
JET PROPULSION LABORATORY of the
CALIFORNIA INSTITUTE OF TECHNOLOGY
4800 Oak Grove Drive
Pasadena, CA 91103

R & D ASSOCIATES
Post Office Box 9695
Marina del Rey,
California 90291

4640 ADMIRALTY WAY • MARINA DEL REY • TELEPHONE: (213) 822-1715

PREFACE

This is the second in a series of reports on a two-month effort to conduct a preliminary design of an automated CPX support system. The first report (RDA-TR-185500-001) defined user requirements collected from I Corps, III Corps, and USREDCOM for a CPX support system. This report develops a preliminary design based only on the corps subset of these requirements, and does not address the joint exercise requirements of USREDCOM. This effort was performed for the Jet Propulsion Laboratory of the California Institute of Technology under Contract No. 956622.

CONTENTS

ILLUSTRATIONS

TABLES

1. INTRODUCTION

RDA has performed a two-month effort to develop a preliminary design for an automated system to support corps-level command post exercises. As the first task in this effort, we identified and documented user requirements of I Corps, III Corps, and USREDCOM for such a system (Ref. 1). As part of the task reported in this document, we performed a functional analysis based on the corps subset of these requirements, and developed a preliminary functional design. In a parallel effort we have examined the suitability of the TRASANA FOURCE computer program (Ref. 2) for inclusion in the automated system. We have identified example hardware configurations and costs as a basis for development planning.

1.1 Background

Traditional methods of conducting command post exercises (CPXs) require a large number of personnel in addition to the command post staff. The extra personnel are needed to model lower-echelon activities, determine the outcome of combat processes, generate reports and requests for transmission to the CP, and receive and process orders or directions from the CP staff. In addition, the exercise director requires a staff to monitor the exercise and make sure that training objectives are accomplished. In a purely manual system, a large number of support personnel are needed to perform all of these functions, particularly for a corps-level CPX. These labor-intensive systems have several disadvantages. First, the labor requirements reduce the number of exercises that could otherwise be performed because so many people must be diverted from other duties. Second, some aspects of the exercise are unrealistic because of the limitations in a manual system, which reduces the potential training value of the exercise. For example some of the manual operations are so time consuming that the message rate to the CP staff is lower than it would be in actual combat, so that the load on the staff is unrealistic.

Recognizing the limitations of purely manual exercise support systems, the Army in recent years has developed systems that provide some degree of automated support. The Computer-Aided Map Maneuver System (CAMMS) supports an exercise primarily by calculating casualties and losses resulting from combat encounters, and by keeping track of the status of friendly and enemy units. CAMMS relieves the exercise support personnel of many dice-throwing and bookkeeping functions. CAMMS-II performs these same functions in greater detail, but also performs a number of simulation functions to stimulate CP staff in functions such as resupply, maintenance, and medical activities. Exercises conducted with these systems have indicated the potential benefits of automation, but have also shown some of the difficulties in achieving a satisfactory system. Exercises conducted with CAMMS and CAMMS-II have still required large support staffs and have not achieved satisfactory realism. There seem to be two major reasons for these limitations.

The first reason is that a large number of people are still needed, both to simulate operations not treated in the computer programs and to generate the inputs needed by the programs. For example, support staff use map tables and unit tokens to track the force status. Manual techniques are used to determine whether a line of sight exists between two units, and to compute the time required for a unit to move to a new location. Umpires are required to validate these judgments and in some cases to adjudicate disputes between the personnel representing opposing forces (whose motivation, after all, is to "win" the simulated combat). Another reason for heavy manpower usage is that the process of supplying data to the computer requires several manual operations. The support personnel must fill out forms describing the desired actions, and data entry clerks must translate these forms and key the information into the computer. For example, if the support personnel decide to fire artillery at an opposing unit, they must fill out an appropriate form, defining items such as the identifier of the firing unit, the identifier of the target, the munitions to be employed, and the distance between the units.

The second reason is that the current exercise process is slow and still contains unrealistic features that detract from training. The process described above for supplying input to the computer is obviously slow and error prone. Furthermore, both of the CAMMS programs are executed in batch mode on a remote computer, accessed through leased lines. The computer operations can lag behind the exercise because of input delays, hardware malfunctions, or communications losses. Some of the possible errors in filling out the input forms may not be detected until the batch run is completed, further impeding the flow of exercise events. The net result is that it is not unusual for exercises to run slower than real time. This is undesirable from a training standpoint because the tactical decision makers do not experience the stress of having to make timely decisions. Another unrealistic aspect of these exercises is that the support personnel typically have more information available than would be the case in actual combat, and there is always the risk that they will provide extra information to the CP staff, despite the exercise rules and the vigilance of the umpires. Therefore, the CP staff do not always experience the stress of making decisions under uncertainty, which is expected to be a key feature of modern combat.

The experience gained in using these semi-automated systems suggests a number of observations. First, automation has the potential to reduce exercise staff requirements and to enhance training. Second, an automated approach is apt to succeed only in the context of a balanced system that addresses training requirements and minimizes computer-peculiar operations. Third, an automated approach requires adequate hardware resources; it is not practical to rely on general-purpose leased equipment that is not dedicated to the exercise system.

1.2 Technical Approach

Our technical approach for designing the Automated CPX Support System (ACSS) comprised four steps:

- Collect exercise support requirements from users;
- Identify functions performed in a CPX;
- Select functions best suited to automation;
- Define an architecture to support the selected functions.

We collected requirements from various military groups, documented our findings, and submitted the documentation for review. (Ref. 1 contains the final results of this process.) Based on this interpretation, we proceeded to identify the functions currently being performed in an exercise facility by observing a number of exercises, augmenting our understanding through discussions with participating personnel and exercise managers. We have identified many functions that seem to be amenable to automation.

A major element of current exercises at division and corps level is that map boards and tokens are used to represent the status of the exercise. Support personnel, pretending to be the subordinate echelons of the units being exercised, move tokens corresponding to maneuver units. In most cases the map board includes tokens for all of the relevant Blue and opposing force (OPFOR) units. (In some cases the exercise director maintains map boards in separate rooms, but the umpires must expend extra effort to make sure that the two boards remain consistent.) The manual operations to maintain the map boards are labor-intensive and error prone. Furthermore, the support personnel can see the entire board, including all of the enemy tokens, and their decisions can be unrealistically good. In an automated system, it would be possible to use a computer to drive a display system showing the current deployments, thereby reducing labor requirements and eliminating some sources of error. Furthermore, by partitioning the exercise facility, it would be possible to provide separate but consistent displays for Blue and OPFOR, which opens the possibility of displaying only information to which the support personnel are entitled. For example, the display of friendly unit locations could be based on status reports, and the display of enemy unit locations could be based on target acquisition reports. Converting the current open game, in which everyone can see all of the forces, into a closed game in which support personnel can see only "their" forces, should increase the training value of the exercise, even for the support personnel.

The most promising technology for an automated display system is videographics, which blends pictures of military maps stored on videodisk with a computer graphics display of military unit symbols. This computer-based approach also lends itself to other functions that are currently time consuming, such as determining lines-of-sight, calculating movement times, and determining firing ranges. The interactive features of a videographics system can be exploited to reduce or eliminate special input forms and their associated clerical functions.

Developing an automated exercise support system can reduce but not eliminate support personnel. In a CP, the staff being exercised receives messages from lower echelons and responds with orders or questions. For the training to be realistic, the staff must communicate with lower-echelon personnel using SOP techniques and formats, including voice contact. Figure

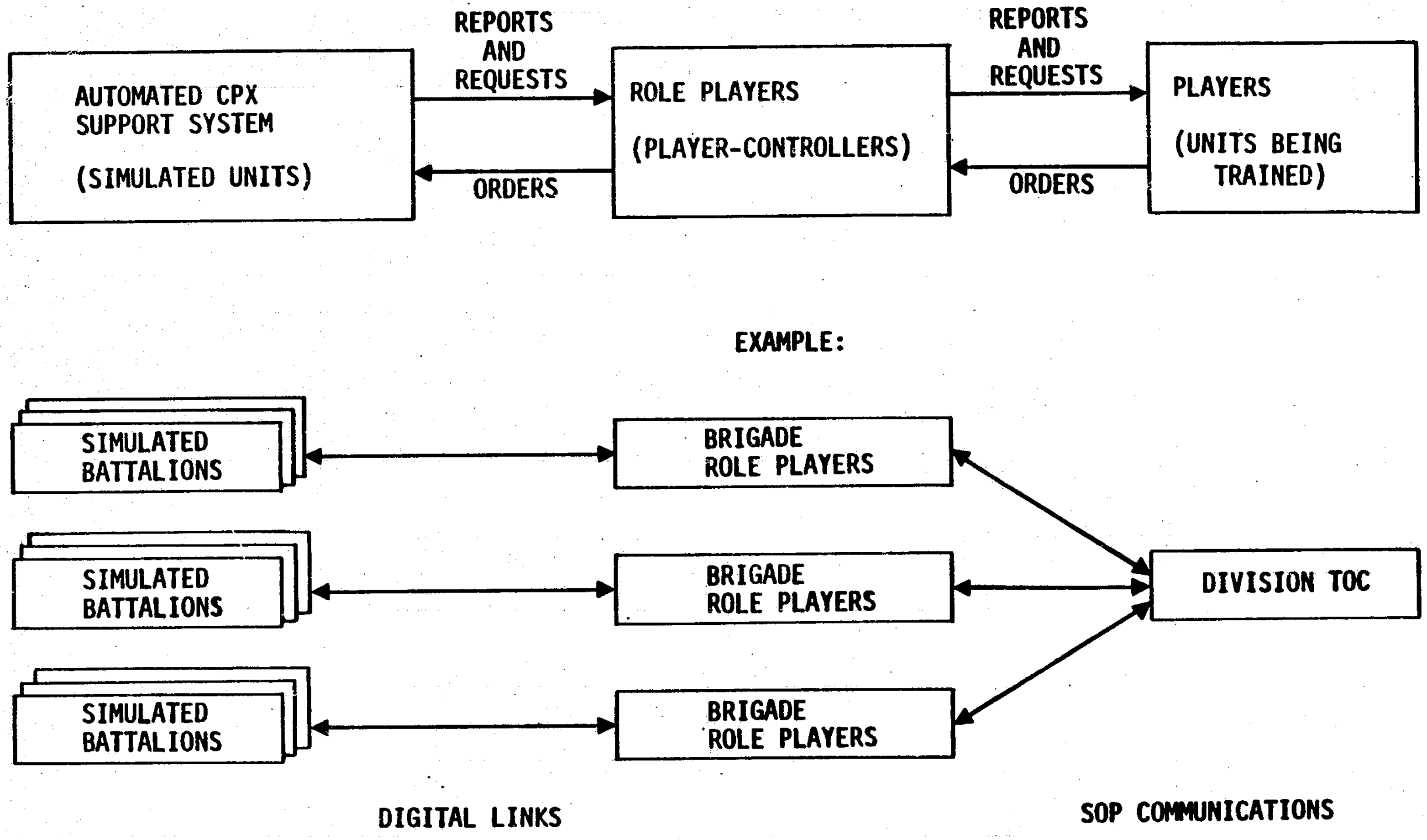
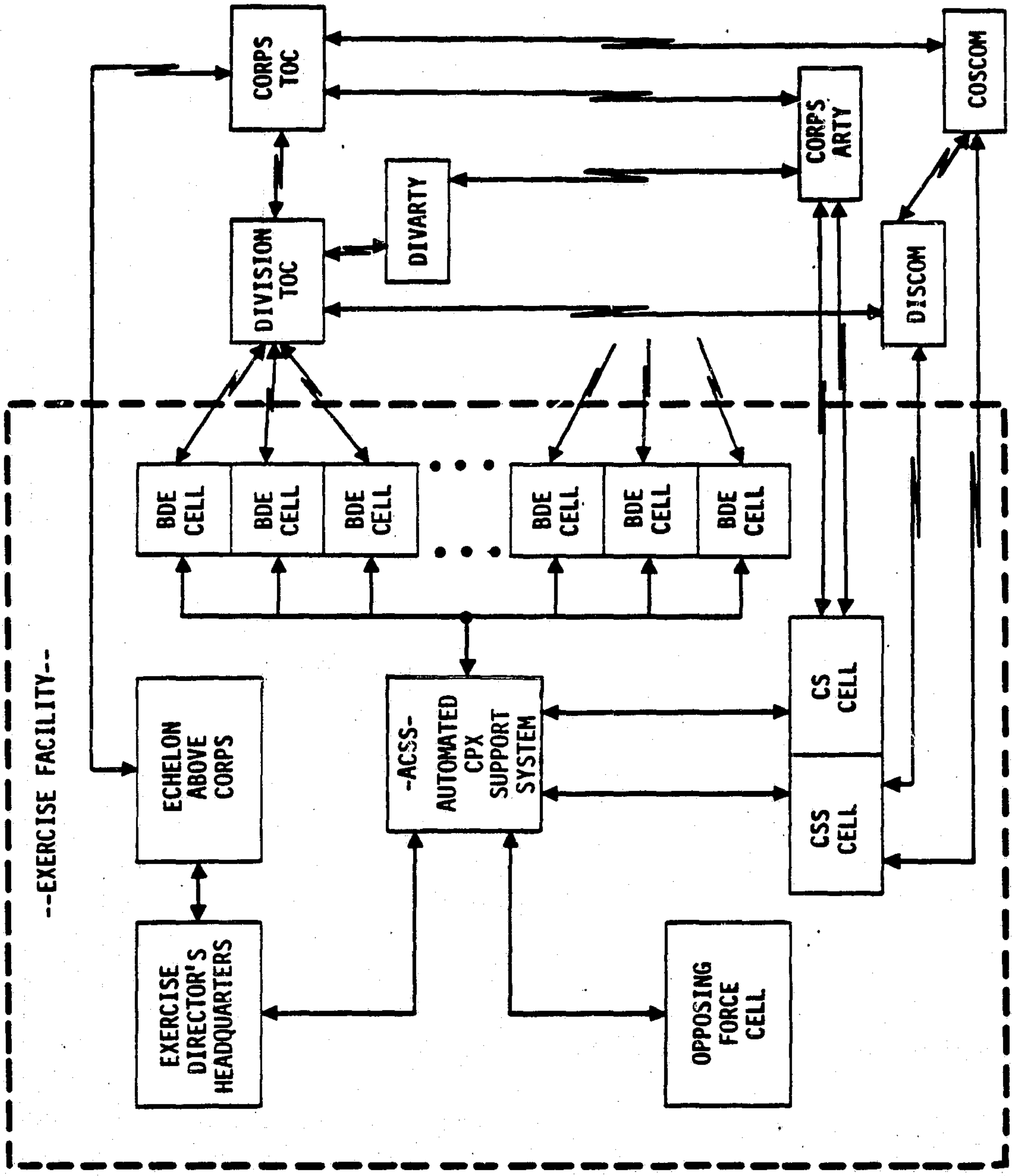


Figure 1-1. Example of exercise system relationships.

1-1 depicts exercise relationships, defines terms, and provides an example. In this view of an exercise system, the units being trained communicate with exercise support personnel who simulate the lower echelon units. In our terminology, the units being trained are 'players' and the personnel simulating lower echelon staffs are 'role players'. The automated system supports the role players, permitting a relatively small number of role players to simulate a much larger staff. The role players also include other groups, such as the personnel representing OPFOR and the exercise director's staff. In some contexts these other groups are referred to as player/controllers or reactors; for simplicity, we describe all such support personnel as role players. The key point in Figure 1-1 is that, in our definition, players do not have direct access to the automated system, but are always buffered by a layer of role players. The example reinforces this point. If the unit being exercised is a division TOC, then three groups of role players simulate brigade staff functions. They use reports and requests received from the battalions being simulated in the automated system to perform the normal functions of a brigade staff. They communicate with the division TOC using standard formats and techniques, preferably using field hardware if it is available.

Based on this philosophy, Figure 1-2 presents a sample functional diagram for a corps CPX. For simplicity, the various role player groups are all depicted inside the dashed line representing the exercise facility, although in practice they need not be colocated. The players, i.e. the units being exercised, are shown outside the facility. Some of the players (e.g., the other division TOCs) are not shown explicitly. Each player group is connected to a role player cell through SOP communications, indicated by a zig-zag line. Each role player cell that needs computer support is connected to the


CORPS TOC
COSCOM
CORPS ARTY
DIVARTY
DIVISION TOC
DISCOM
--EXERCISE FACILITY--
BDE CELL
BDE CELL
BDE CELL
BDE CELL
BDE CELL
BDE CELL
ECHELON ABOVE CORPS
-ACSS- AUTOMATED CPX SUPPORT SYSTEM
CS CELL
CSS CELL
EXERCISE DIRECTOR'S HEADQUARTERS
OPPOSING FORCE CELL

ACSS by a straight line that represents a digital link. The total number of cells may vary, depending on exercise objectives. For example, the distribution of players and role players for combat service support (CSS) would depend on the extent to which CSS functions are being exercised and on the number of personnel available for the exercise. As another example, the figure displays an exercise situation with division TOCs in the field and brigade role player cells; another configuration would place a brigade CP in the field, supported by battalion role player cells.

Figure 1-2 clarifies the relationship of the ACSS to the rest of the exercise system by emphasizing that it interfaces with role players rather than players. We have developed a preliminary design for the ACSS by defining a distributed architecture that includes a separate satellite system for each unique cell of role players, and by identifying the computer based functions needed in each satellite. Section 2 describes the architecture and presents an initial functional analysis. Section 3 describes the individual satellite systems and develops initial estimates of equipment requirements. Section 4 describes example computer configurations and estimated costs for planning purposes.

2. OVERVIEW OF SYSTEM ARCHITECTURE

The Automated CPX Support System (ACSS) is a computer based system that directly supports the exercise director, controllers, and other role players who provide the stimulus to the staffs being trained. Our preliminary analysis indicates that ACSS must accommodate large, calculationally-intensive functions (e.g., resolving combat) as well as highly interactive functions (e.g., mission planning). In our judgment, a single computer could not support both classes of function for a corps-level CPX. Therefore, our preliminary design envisions a distributed system with multiple computers to support different functions.

Figure 2-1 summarizes the architecture, which includes a central system (which may employ multiple computers) and multiple satellite systems. The primary function of the central system is to perform the modeling calculations needed to simulate those aspects of combat, combat support, and combat service support that are relevant to the exercise objectives. The primary function of the satellite systems is to support the intensive interactive functions such as providing videographic displays of the battlefield. The central system also maintains the master database and performs functions needed to manage the entire simulation. Each satellite system maintains a local database of information pertinent only to its particular operations, and also performs functions needed to control local operations and to interface with the central system. The composition of each satellite system depends on the exercise functions being supported (e.g., OPFOR or brigade role players), but in general the communication between central system and satellite systems takes the form of messages and reports flowing from the simulations in the central system to the role players in the satellite stations, and orders flowing

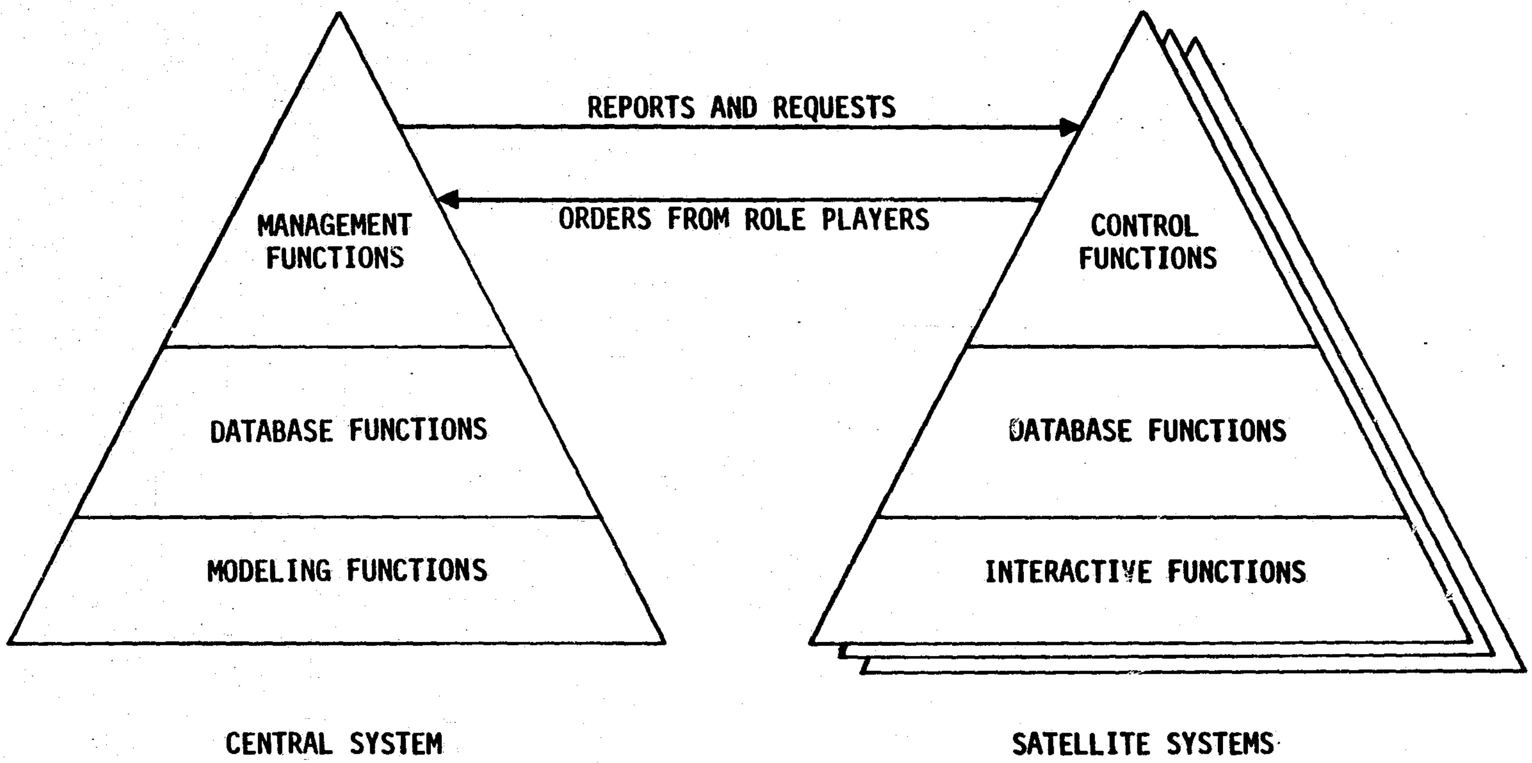


Figure 2-1. ACSS allocation of computer functions.

from the role players to the lower echelon forces being simulated in the central system.

The following sections present a first-level decomposition of these functions, thereby providing a conceptual framework for subsequent design efforts.

2.1 Description of Central System Functions

The primary purpose of the central system is to support the modeling functions that involve intensive calculations, such as resolving ground combat between opposing units whose movements and actions are being simulated. In addition, the central system maintains a database defining the current state of the total exercise simulation. Finally, the central system also performs management functions that are needed to control the computer-based portion of the exercise. Table 2-1 presents a first-level decomposition of these functions, starting with the modeling functions.

TABLE 2-1. HIGH LEVEL FUNCTIONS SUPPORTED BY THE CENTRAL SYSTEM

- Modeling functions
  - Simulate ground combat
  - Simulate other processes
- Database functions
  - Prepare database for exercise
  - Maintain database during exercise
  - Support post-exercise analysis
- Management functions
  - Control simulation programs
  - Manage communications to satellites
  - Manage interfaces to external systems
  - Manage system resources

Subsequent sections describe these functions in more detail.

2.1.1. Modeling Functions Supported by the Central System

The central system bears the primary responsibility for running computer programs that model systems and functions that are not represented by role players. Table 2-2 summarizes the principal classes of models that must be included.

TABLE 2-2. MODELING FUNCTIONS SUPPORTED BY THE CENTRAL SYSTEM

- Simulate ground combat
  - Simulate battlefield environment
  - Model ground combat processes
  - Model effects of combat support
- Simulate other processes
  - Account for effects of combat service support
  - Model effects of NBC events
  - Model effects of electronic warfare
  - Simulate staff processes

The battlefield environment must be modeled in sufficient detail to support calculations of unit movement, sensor detection, and attrition. Examples of aspects of the environment that must be represented include terrain and foliage, existence of roads and bridges, principal obstacles, and weather and smoke effects on visibility.

The principal modeling function to be supported by the central system is treating ground combat processes for the units being controlled by the role players. The simulated units move in response to orders from the role players, engage in combat with opposing units, and send reports to the role players. The models, for example, may use threshold criteria such as time from last report, distance to enemy units, or supply status to trigger the generation of simulated reports.

An auxiliary function is to account for the effects of combat support on the ground combat processes. For example, the application of higher-echelon artillery assets must be reflected in the attrition experienced by the ground combat units. Furthermore, unless role players perform numerous manual calculations, the central system will have to model the constraints (and possibly some of the operations) for the combat support systems, such as limitations on range and munitions.

The central system must also model some of the aspects of combat service support. For example, the models must account for the consumption of key supply items such as fuel and ammunition by the combat units being simulated, and must replenish these items in a manner consistent with the overall treatment of combat service support in the exercise. For example, if the logistics staffs are not explicitly involved in the exercise, then the central system models must simulate, in a very simplified manner, the performance of their functions. On the other hand, if the logistics staffs participate extensively as either players or role-players then the ACSS models must perform only bookkeeping functions in response to the orders from the role players.

ACSS must be able to account for nuclear, biological, and chemical (NBC) events in order to provide the appropriate stimulus for staff training in dealing with these problems. The level of detail required in these models will depend on the exercise objectives. At the simplest level, the exercise director may choose to include NBC events in the script of scheduled events discussed in Section 2.1.3. In this case, the ACSS management function will process each event at its scheduled time, activating a modeling program to perform functions such as inflicting damage on units within a calculated region and triggering simulated reports to the relevant role players. At the next level of detail, the model would have to be expanded to simulate unit actions in response to commands from the role players (e.g., adopt MOPP procedures) and account for the effects on normal unit operations (e.g., reduced mobility). At a deeper level of detail, the model might be required to support the activation of NBC events directly by the role players, perhaps accounting for constraints on delivery systems.

ACSS must also account for aspects of electronic warfare on the conduct of combat operations, although it may not be necessary to model specific systems in engineering detail. Again, the scheduled event mechanism provides flexibility to introduce special effects under the control of the exercise director. For example, the director may wish to exercise the staff in dealing with degraded communications caused by jamming. Using the event mechanism, the director could supply a jamming event with a specified time, duration, and communication link. During the specified period, the appropriate role players would not receive the usual reports from the subordinate units being simulated, although they might receive a message indicating that the channel is being jammed. At a deeper level of detail, the exercise director might wish to stimulate staff training in the deployment and utilization of EW teams. In this case, the ACSS models would have to perform functions such as tracking the location of the EW teams, modeling the capabilities of the EW equipment, determining which enemy (and friendly) units would be affected by the emanations, and degrading communications capabilities in an appropriate manner. Similarly, if the function of the EW teams is to detect and identify enemy units, then the simulation models would have to represent the capability of their systems to intercept enemy transmissions and then generate intelligence reports to the relevant role players.

The final class of modeling functions identified in Table

2-2 treats staff functions. If staff functions could be modeled in ACSS, then it would be possible to reduce the number of role players required to support an exercise. For example, it might be possible to conduct a corps CPX using one or more surrogate divisions simulated in ACSS to generate messages to stimulate the corps staff.

2.1.2 Database Functions Supported by the Central System

The central system maintains the ACSS database for the entire exercise. The three primary classes of database functions are preparing the database for an individual exercise, maintaining the database during the exercise, and supporting post-exercise analyses. Table 2-3 presents a first-level decomposition of these functions, as discussed in the following paragraphs.

TABLE 2-3. DATABASE FUNCTIONS SUPPORTED BY THE CENTRAL SYSTEM

- Prepare database for exercise
  - Characterize battlefield
  - Prepare tables of organization and equipment
  - Collect parameter values for system models
- Maintain database during exercise
  - Initialize system at the start of the exercise
  - Update the database using the simulation output
  - Augment the database using the data files received from the satellite stations
  - Perform back-up and recovery functions
- Support post-exercise analysis

Preparing the database for a specific exercise can be extremely time consuming, particularly if the exercise is set in a geographic area not used previously in an exercise. If the exercise is a corps CPX, then the area of interest that must be modeled can be quite large. Most of the existing ground combat models treat the battlefield as a system of discrete grid elements (either rectangles or hexagons), and each element must be characterized in terms of terrain, foliage, and trafficability (e.g., existence of roads and bridges). If the battlefield region of interest is 100 km wide and 200 km deep, and if the grid element is a 1 km square, then 20,000 elements must be characterized, presumably by analysts using sources such as military maps. Various techniques may be developed to reduce the burden on the analyst (e.g., using larger grid elements in regions where resolution is less important, or providing computer-based tools to facilitate the analysis), but in any event the characterization of the battlefield must be entered in the exercise database, preferably in a form that facilitates review for analysis, retrieval for use in the exercise, and modification for use in other exercises. Other portions of the exercise database include the tables of organization and equipment for the specific exercise and parameter values for the various models of weapons, sensors, and other equipment. All of the data assembled for an exercise must be easily accessible to the exercise director for review and modification. In fact, the databases prepared for exercises should be accumulated in a library, providing an increasingly powerful tool for quickly preparing a new database. The scope and complexity of these requirements dictate using a commercial database management system, possibly augmented by special software tools.

During the operation of the exercise, ACSS maintains a database of simulation results and role-player actions. For example, at any given instant the database contains the true battlefield situation (location and status of all units) as well as the Blue and OPFOR perceptions of the battlefield. In addition, ACSS records all of the commands (e.g., movement orders) received from the role players. All of this information must be available for display or tabulation at the request of the exercise director. Furthermore, at periodic intervals the simulation results must be copied to long-term storage, both as a back up in case of system malfunction and to enable the exercise director to restore an earlier exercise situation or to replay portions of the exercise.

After an exercise has been concluded, the exercise director may wish to review the results in more detail in order to design future exercises. The replay capability previously mentioned will enable the director to display the evolution of the situation displays, but he may require additional computer support in order to determine the reasons for specific exercise results. For example, he may wish to examine the message traffic for a specific group of role players to verify that resupply requests were generated appropriately. The range of potential analysis functions is very broad; therefore, selected functions should be thoroughly defined so that the necessary data can be captured during the exercise.

2.1.3 Management Functions Supported by Central System

The central system supports management functions in four principal areas: controlling the simulation programs, managing the communications to the satellite systems, managing interfaces to external systems, and managing system resources. Table 2-4 presents a preliminary decomposition of these functions.

The principal management function is to support the exercise director in controlling the simulation aspects of the exercise. For example, the central system must control the execution of various computer programs or tasks, such as programs that model combat processes or programs that save and restore the database on mass memory devices. Another management function is to control the pace of the exercise in response to commands from the exercise director. This function includes controlling the simulation clock by setting the ratio

TABLE 2-4 MANAGEMENT FUNCTIONS SUPPORTED BY THE CENTRAL SYSTEM

- Control simulation programs
  - Control task execution
  - Control simulation clock
  - Process scheduled events
- Manage communications to satellites
  - Initialize configuration of satellites
  - Process messages received from satellites
  - Manage exchange of data files
  - Maintain transaction log
- Manage interfaces to external systems
- Manage computer system resources
  - Monitor system performance
  - Diagnose system faults
  - Perform configuration management

of game time to real time (typically 1:1), suspending the simulation at the director's command, restoring a previous state of the simulation, and replaying selected portions of the simulation. The final control function in Table 2-4, processing scheduled events, is a general mechanism to provide additional flexibility for the exercise director. In general, each event corresponds to an action that occurs at a definite time (and perhaps at a definite location) determined external to the ACSS simulation programs. Examples of such events might include changes in the weather or visibility, occurrence of nuclear or chemical attacks, and air drops of enemy troops in the rear area. The results of each event type on the simulated battlefield must be modeled by a separate program; the management function identified here consists of accepting events supplied by the exercise director, storing the events in time order, and retrieving each event at the appropriate game time. The concept of scheduled events is extremely powerful; for example, the list of scheduled events could correspond to a canned scenario of commands, thereby enabling the exercise director to test the simulation without needing a large staff of role players.

The second important type of management function performed by the central system is managing the communications to all of the satellite systems. The actual communications techniques to be used (e.g., networking methods, channel capacities) are issues to be resolved in the implementation design. For purposes of this preliminary design, it is simplest to imagine that the central system controls all exchanges with the satellite systems. For example, the central system could poll each satellite, sending it a data file containing the relevant modeling results (e.g., current unit locations) and receiving in return a data file containing the commands (e.g., movement orders) generated by the role players since the last communication exchange. This polling concept provides a simple model for identifying communications functions without restricting the actual implementation. (If polling were performed frequently enough, the simulation would appear continuous to the role players, avoiding the artifice of rigid game turns.)

Table 2-4 identifies four principal functions related to managing communications with the satellite systems. First, before an exercise begins, the central system must initialize the configuration of satellite systems by defining the nature and address of each satellite so that data files can be exchanged properly. For example, a given physical satellite might support role players for either a brigade staff or a battalion staff, depending on the specific exercise objectives. Second, the central system must process the messages received from the satellite that it is polling. In some cases these messages will co respond to orders from role players, and the central system must arrange for their transfer to the appropriate simulation model. In other cases the messages may be orders from the exercise director, for example to suspend execution of the simulation. Third, the central system must manage the transfer of a data file to the polled satellite. This function may require dividing the file into blocks and generating check sums, depending on the communications implementation. Fourth, the central system should maintain a log of all transactions, thereby facilitating post-exercise evaluations or reconstructions.

The third principal management function is managing interfaces to systems external to ACSS. Each external system may require different interfacing techniques, either completely or partially automated. One example of an important external system that requires an automated interface is the Distributed Command and Control System (DCCS) of the 9th Infantry Division. In order to provide the proper exercise stimulus to the division staff, ACSS must inject the simulation output using DCCS protocols. Another example of an important external system is TACSIM, which provides intelligence reports corresponding to assets above corps. In order to perform its modeling functions, TACSIM requires as output from ACSS a portrayal of the time-varying battlefield situation, including the location and status of all ground units. In return, TACSIM provides simulated intelligence reports which are distributed to appropriate high-echelon units using standard communications methods and formats. In principle, it might be possible for ACSS to include an automated interface to provide the input needed by TACSIM, but in practice it is likely (e.g., because of security considerations) that an electrical connection between the two systems will not be feasible, and arrangements must be made for ACSS to prepare a computer tape in a format acceptable to TACSIM. Another example of an external system might be the COSCOM Material Management Center, which might be employed in an exercise that emphasizes logistics training. In this case, some kind of electrical interface might be possible. Each case requires special consideration and imposes additional requirements on this management function for the central system.

The final class of management function supported by the central system is managing the computer system resources. Because ACSS will be developed over a period of years, it will be particularly important to monitor system performance during tests and exercises to identify bottlenecks (either in processing or in communications) that must be eliminated to achieve system growth. Special hardware devices and software probes will be needed for this monitor function. Furthermore, special software tools will be needed during the exercise to diagnose faults that may occur, such as the failure of a satellite system to receive central system output. Another system resource will be the software developed for ACSS. It is prudent to expect that the software will be very large, complex, and subject to frequent change. Therefore, the staff of the exercise director will need a suite of software tools to perform configuration management functions such as controlling changes to the code and preserving back-up copies. Finally, the central system will include mass storage systems such as disks and tapes whose usage must be controlled. The staff of the exercise director will need software tools (many of which might be supplied with the operating system) to diagnose device faults, free storage space by deleting unneeded files, move data from disk to long-term storage, and restore selected data sets.

2.2 Functions Supported by Satellite Systems

Each satellite system directly supports a group of role players (or controllers) participating in the exercise. The primary purpose of a satellite system is to support functions requiring frequent user interaction, thereby offloading the central system which is performing calculationally-intense modeling functions. In addition, each satellite station manages a local database of information pertinent to the operations being performed, and performs control functions such as managing data exchanges with the central system.

The hardware suite contained in each satellite depends on the number of role players and the nature of their activities. For example, a satellite station for brigade role players might need multiple work stations to control the maneuver units being simulated. Nevertheless, each satellite station must support three general classes of function, as listed in Table 2-5. Subsequent sections describe these functions in more detail.

TABLE 2-5 HIGH-LEVEL FUNCTIONS SUPPORTED BY A SATELLITE SYSTEM

- Interactive functions
  - Maintain situation displays
  - Support planning operations
  - Accept user input to control elements being simulated in central system
- Database functions
  - Initialize local database
  - Maintain local database
- Control Functions
  - Control local operations
  - Manage communications to central system

2.2.1 Interactive Functions

Each satellite system supports interactive functions to maintain situation displays, develop operational plans, and process orders from the role players to the simulated units. Table 2-6 presents a further analysis of these functions.

TABLE 2-6 INTERACTIVE FUNCTIONS SUPPORTED BY A SATELLITE SYSTEM

- Maintain situation displays
  - Control display of map background
  - Generate graphics overlay of unit locations
  - Control display list
- Support planning operations
  - Use data from simulated elements to generate reports in SOP format
  - Display current status of designated element
  - Perform calculations to assist planning operations
- Accept user input to control elements being simulated in central system
  - Display available options
  - Verify and accept user input

Each situation display consists of a map background drawn from a videodisk and combined with a computer-generated graphics overlay showing unit locations. The user interacts with the system to control the region being displayed, causing the computer to retrieve the appropriate image from the videodisk. Based on the region being displayed, the system determines which units are in the field of view and generates the corresponding symbols for the graphics overlay. The symbols used for the units conform to standard military usage as defined in FM 21-30. To facilitate role-player operations, the portion of the situation display that is derived using reports from simulated units is generated automatically, freeing the role players from the time-consuming operation of posting reports manually. In some circumstances additional user control of the display may be needed. For example, the display would typically show all of the perceived Blue and OPFOR units in the field of view, but this display might be too cluttered for some applications, so the role player may need the option to suppress portions of the display. For example, the role player might choose to display only those units within a specified chain of command,those units above a specified echelon, or those of a specified type.

The satellite system must also perform a variety of interactive functions to assist the role players in planning the operations of the units being simulated. The motivation for providing planning tools, which would not necessarily be available in a field situation, is to reduce the work load on the role players, thereby reducing the number of role players needed. For example, manual exercise systems require significant time and effort to capture and transcribe the results of combat being simulated on the battle board. In the ACSS design, the models being executed on the central system automatically generate combat results and send them to the appropriate satellite system. One of the functions of the satellite system is using the data received to create reports in standard format for use by the role players, thereby reducing the number of manual operations. The software at the satellite stations must include programs to interpret the data received from the central system and generate the corresponding report types, such as intelligence and status reports and requests for fire support and resupply.

The satellite system must also perform other functions to assist the role players in evaluating the current situation and planning future operations. For example, the system must display the current status of any element designated by the role player. One way for the role player to designate an element would be by positioning a cursor over the corresponding symbol on the situation display, or by pointing to the symbol if a touch-sensitive device is used. In either case, the system would respond by listing the current status of the indicated unit on an alphameric terminal. For example, if the element corresponds to a maneuver unit, the system would provide information such as its current strength, engagement status, supply status, and the time of the most recent report received. If the element corresponds to an artillery unit, the system would provide information such as current mission, engagement status, munition status, and time of most recent firing.

Each satellite system must also perform calculations to assist the role players in planning operations quickly. The nature of the calculations depends on the specific role player functions being supported. For example, the role player planning to move units needs interactive tools to assist him in estimating the time required for the movement. In this case, the role player might designate the unit to be moved by positioning a cursor, and then indicate a skeleton route by designating way points with the cursor. The system would then estimate the travel time, using standard Army planning factors and accounting for the terrain to be traversed. This capability would enable the role player to rapidly examine a number of candidate routes and select the one best suited to the situation. (Note that the actual time required to execute the movement may differ from the estimated time; in particular, the units may encounter OPFOR units whose presence had not been detected when the plan was created.) The role player controlling artillery operations needs different support functions. For example, the system might provide graphic displays of the coverage (or range to effect) for available munitions.

The final class of interactive functions identified in Table 2-6 concerns input of orders for the elements being controlled. The ACSS design contemplates maximum use of interactive graphics techniques to minimize the amount of typing needed and to simplify training the role players in using the system. One example is the graphics technique described above for defining a movement order; which frees the role player from the need to enter UTM coordinates using a keyboard. Once the role player is satisfied with the route as shown on the display, he needs to use the keyboard only to enter a few additional items, such as the required time of arrival or the time at which this movement order should be executed by the unit being simulated. The system will then automatically translate the route into a series of way points in the internal coordinate system and format it for transmission to the central system. Interactive menu techniques may be more effective for other role player operations. For example, the role player controlling artillery assets may need a menu to select munition and fuze options. Regardless of which input technique is used, the satellite system must, to the extent possible, prevent erroneous input. For example, the menu system should list only those options that are pertinent to the specific unit for which an order is being generated. In addition, the software should check the user's input for consistency. Errors should be detected as early as possible, preferably before the orders are transmitted to the central system. For example, a simulated artillery unit should not be instructed to fire more rounds than are currently available to it. Operator errors of this type can cause confusion and degrade exercise performance without compensating benefits in training.

2.2.2 Local Database Functions

Each satellite system initializes and maintains a local database containing information pertinent to the specific operations of the individual satellite. Table 2-7 presents a further analysis of these functions.

TABLE 2-7 DATABASE FUNCTIONS SUPPORTED BY A SATELLITE SYSTEM

- Initialize local database
- Maintain local database
  - Augment database with files received from central system
  - Augment database with user input
  - Delete items from database
  - Generate reports
  - Maintain record of user input sent to central system
  - Create back-up files

Before an exercise begins, the satellite system initializes the local database using information received from the central system. For example, the database contains the deployment and status of all elements at the specified game time that are being controlled by the role players in this satellite system. For example, the exercise might begin after the simulated fighting has already begun (e.g., a preparatory artillery barrage) so that some of the units have already experienced attrition. This initialization function also supports the restart capability required for ACSS, since the satellites can be initialized from any state recorded by the central system.

After an exercise begins, the satellite system maintains the local database to account for the results of the simulation. The principal function is to augment the current database using the messages received from the units being simulated on the central system. For example, in the local database the current location of a maneuver unit such as a battalion corresponds to the one contained in the status report most recently received. If the reporting interval is large, or if the simulated communications link is jammed, then the unit location in the local database may be quite different from the actual location. Therefore, the local database represents the battlefield as perceived by the role players, who must deal with these uncertainties in making decisions and in communicating with the higher-echelon units being trained.

Although most of the local database is generated automatically using output from the central system, in some cases it may be necessary to augment the database with direct user input. For example, in some cases the role players may receive intelligence reports from their superiors, who in turn may have received data from an external system such as TACSIM: the role players may need to enter this information in the database so that it can be reflected in the situation displays and used in the planning functions.

Depending on the exercise objectives, it may also be necessary to enable the role players to delete items from the database. For example, if spoofing is modeled in the exercise, then spurious intelligence reports can enter the database to stimulate the intelligence staff functions. The role player would then need to delete those items determined to be spurious, or at least mark them in some fashion to prevent them from being included in the situation display.

Another database function is generating reports to assist the role players. For example, a role player controlling artillery assets might need a report summarizing the status of all units. These reports would be generated at user request. In other cases, the role players need to send standard status reports at defined intervals to the higher-echelon staff; in these cases the system can automatically generate a printed report containing the necessary information.

The final database functions listed in Table 2-7 correspond to background tasks that are transparent to the role players. The system maintains a record of the orders generated by the role players and sent to the central system. These data support diagnostic and evaluation functions.

2.2.3 Control Functions

Each satellite station performs control functions to manage task execution and communications with the central system. Table 2-8 presents a further analysis of these functions.

TABLE 2-8 CONTROL FUNCTIONS SUPPORTED BY A SATELLITE SYSTEM

- Control local operations
- Manage communications with central system
  - Process messages from central system
  - Assemble data files for transmission to central system
  - Manage the transmission and receipt of files
  - Maintain a log of transactions

Each satellite station supports a variety of user-requested and automatic functions, implying a rich suite of computer programs whose execution must be controlled. For example, when the central system sends a status report from a simulated unit, the satellite system must automatically expand the received data into the appropriate standard format, produce a printed report for the role player, update the local database to reflect the new information, and update the graphics overlay for the situation display. The execution of these automatic functions must not conflict with user execution of other functions.

The other class of control functions is concerned with managing communications with the central system. Given the conceptual model of polling discussed in Section 2.1.3, the satellite functions are merely those required to respond to the central system. First, the central system may send control signals to the satellite, which must interpret the signal to determine which action to take. For example, the central system may indicate that the satellite should prepare to accept a new data file or may inquire whether the satellite itself has a file ready for transmission. Second, during the period between polls, the satellite communications function collects user input in a file for subsequent transmission. For example, various role players might submit movement orders and fire allocations for elements being controlled. Third, the satellite participates in managing the actual exchange of data files. The protocols for exchanging data depend on the implementation design, but typically involve blocking the data and computing checksums for error detection. Finally, the satellite station maintains a transaction log to aid in diagnosing system faults.

3. DESCRIPTION OF SATELLITE SYSTEMS

The ACSS architecture comprises a central system and multiple satellite systems. Each satellite system provides direct support for a single cell of role players. Our approach for determining the number of satellite systems is to provide a satellite for each group of role players that need access to a common database and can share computer resources, including hardware such as videographics monitors and software such as planning tools. Based on this approach, we have identified five unique classes of satellite system. Figure 3-1 summarizes our preliminary view of the ACSS architecture.

The staff of the exercise director requires access to the entire database, including the actual unit locations, the Blue unit locations as perceived by OPFOR, and the OPFOR locations as perceived by Blue. No other group of role players is allowed access to all of this information. Therefore, we assign a separate satellite station for the exercise director's staff.

The OPFOR role players are responsible for controlling a large number of units on the simulated battlefield. In principle, it would be possible to partition the OPFOR role players into multiple cells, each with access to a separate portion of the database. In practice, since the OPFOR role players are controllers rather than trainees, we believe that it is appropriate to provide a single satellite so that all role players have access to the same information. We hope that this approach will increase operational efficiency and minimize the number of role players required for this function.

Each cell of brigade role players needs access to a database of information pertinent to the brigade area of interest, such as the location and status of maneuver units and direct support units, the reported location and strength of OPFOR

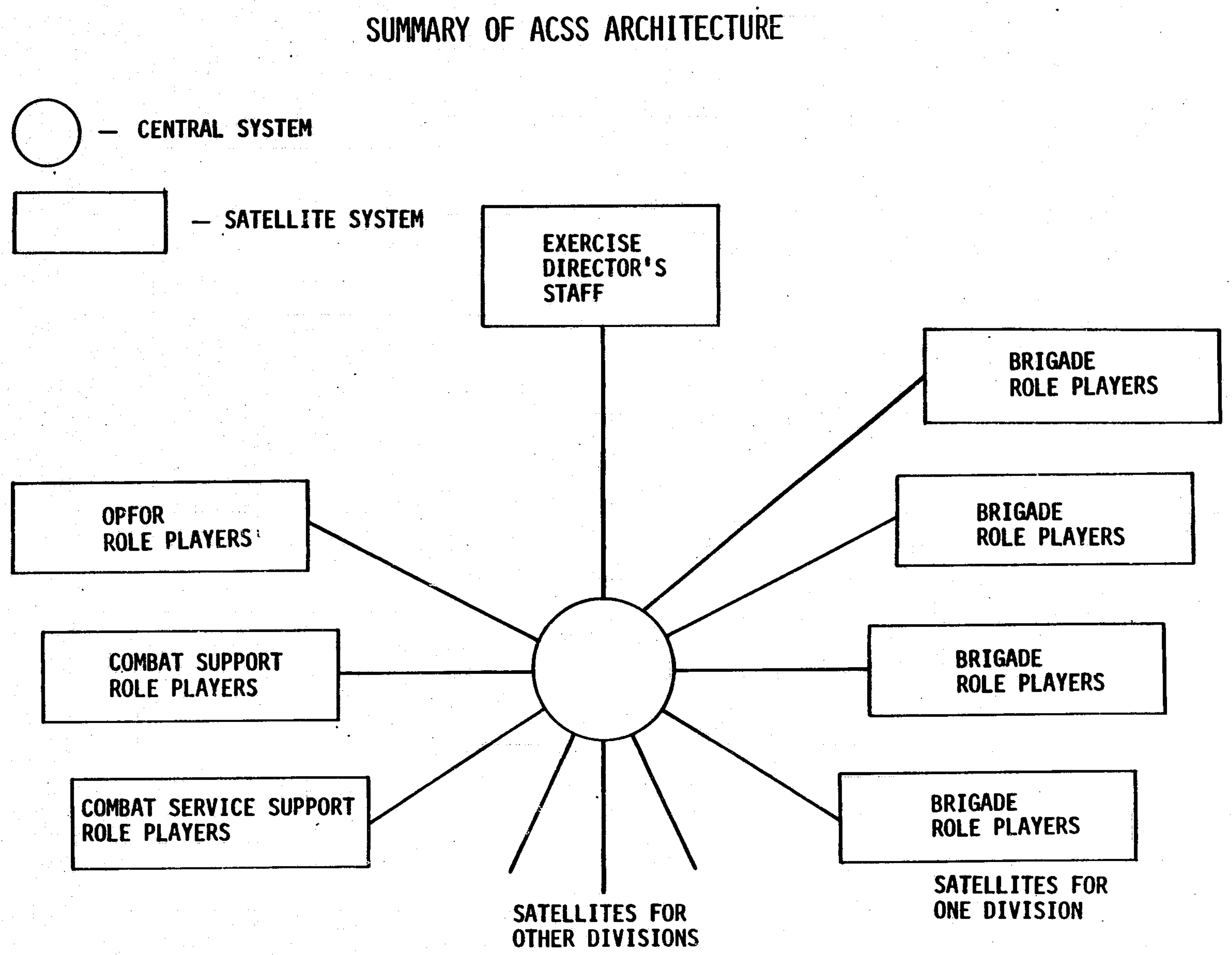

SUMMARY OF ACSS ARCHITECTURE
— CENTRAL SYSTEM
— SATELLITE SYSTEM
EXERCISE DIRECTOR'S STAFF
BRIGADE ROLE PLAYERS
BRIGADE ROLE PLAYERS
BRIGADE ROLE PLAYERS
BRIGADE ROLE PLAYERS
SATELLITES FOR ONE DIVISION
OPFOR ROLE PLAYERS
COMBAT SUPPORT ROLE PLAYERS
COMBAT SERVICE SUPPORT ROLE PLAYERS
SATELLITES FOR OTHER DIVISIONS

Figure 3-1. Summary of ACSS architecture

units, and the location of general support units in the area. Many of the planning and control operations will use the same videographics display (typically derived from a 1:50000 map). Therefore, we assign a single satellite station to each cell of brigade role players. The total number of satellites of this type depends on the scope of the exercise; we expect that at least ten satellites of this class would be needed to accommodate a corps CPX with three divisions and an independent brigade being exercised.

The combat support role players need access to data for all division and corps assets that are not deployed in direct support roles, including artillery, aircraft, and air defense units. These role players also use planning tools (e.g., to generate missile strikes) that are not needed by the brigade role players. Because of the probable commonality in displays and tools, we believe that one satellite station can serve a single set of role players representing combat support for both division and corps assets. It is possible, however, that it might turn out to be beneficial to divide this satellite.

Similar arguments apply for the combat service support elements. The role players will typically use the videographics system to display the corps or division rear area, using a 1:250000 map as background. Role players for corps and division will all be concerned with supply depots, transportation routes, and rear area protection. We therefore postulate a single satellite system to support all role players for combat service support, although it may be necessary to modify this design to accommodate exercises with heavy emphasis on logistics training.

Table 3-1 summarizes the factors we have considered in trying to estimate the equipment requirements for the five classes of satellite systems. A single satellite supports a

TABLE 3-1 SATELLITE SYSTEM DESIGN CONSIDERATIONS

- Satellite system: supports a group of role players who need access to a common database and can share system resources.

- Work station: supports user interaction with a videographics monitor and an alphameric terminal.

- Each satellite system may require multiple work stations to support simultaneous interactions.

- Each work station can support any function associated with the parent satellite system.

- Satellite system includes additional hardware such as large screen displays, printers, and communications devices.

- Estimates of hardware requirements are based on minimum configuration consistent with a corps-level CPX.

group of role players, who may not be colocated, and therefore may require sufficient hardware to support simultaneous interactive functions. The essential hardware for supporting an interactive user includes a videographics monitor and an alphameric terminal, so we denote this combination as a work station, and attempt to estimate the number of work stations needed in each class of satellite system. Our estimate is based on the number of functions that we believe might have to be performed simultaneously by the role players during extended portions of the exercise. For example, if it seemed likely that role players would have to control maneuver units and process fire mission requests at the same time, then we would conclude that two work stations would be needed. One of the ground rules of our design, however, is that the work stations within a satellite should be interchangeable rather than dedicated to specific tasks. The flexibility to use a given work station for any task provides important flexibility for the role players to adapt to fluctuations in work load during the exercise. In the example already cited, during some portions of the exercise it might be desirable to use both work stations for controlling maneuver units.

The Blue satellite descriptions in Sections 3.3-3.6 are not intended to reflect a particular Corps or division capabilities. The assumed scenario is of a three division Corps, with one division heavily involved in combat, one lightly involved to the side of the main OPFOR attack, and the third in reserve. Corps assets have been assumed to be fairly heavy from a TOE standpoint; in particular they are heavier than in a high mobility Corps. Two brigade satellites are described: one for an armor rich brigade and the other for the Combat Brigade (Air Assault), representing widely diverse mobilities. It was found that the same hardware configuration should suffice for both. It is assumed the HTLD brigade also can be accommodated in this configuration, although the evolution of HTLD operational concepts and doctrine may change this assumption.

The following subsections present our initial results for the five classes of satellite system. The estimates correspond to the minimum configuration that we think would be adequate for a corps-level CPX.

3.1 Satellite System for Exercise Director's Staff

The staff of the exercise director performs multiple functions. They generate the scenario for the exercise, prepare the system, develop any material needed to train the role players, manage the system during the exercise, and give presentations. Some of these functions must be performed concurrently during the exercise. Therefore, even though all of these functions require access to the complete exercise database, we believe that the staff would be best served by separate facilities. Table 3-2 summarizes the preliminary design.

One facility is primarily devoted to the functions of preparing for the exercise and managing its conduct. The staff members will need access to the true battlefield situation as well as the Blue and OPFOR perceptions. They may also need to examine the current tactical plans as input by the role players. They will need to monitor the computer system performance, and may need to diagnose and correct system faults. The exercise director may ask this part of the staff to inject special events into the exercise (e.g., a change in weather); the staff may need software tools to make sure that the special events will have the desired effects. We estimate that this portion of this satellite system needs a minimum of two videographics work stations (for monitoring the exercise and planning special events) and a system console (for controlling computer operations). We also think that a single large screen display would be helpful for reviewing the status on extended portions of the battlefield.

The second facility is primarily devoted to monitoring the

TABLE 3-2 SATELLITE SYSTEM FOR EXERCISE DIRECTOR'S STAFF

- Staff requires separate facilities to perform exercise control functions and briefing functions.
- Facility for exercise control functions:
  - Two work stations for monitoring the exercise, analyzing simulation results, and planning special events.
  - One large screen display for monitoring extended portions of the battlefield.
  - One printer for generating simulation histories and program listings.
  - One system console for controlling computer system resources.
- Facility for briefing functions:
  - One work station for interrogating system status, preparing the presentation, and controlling displays.
  - One large screen display for audience.
  - One printer/plotter for generating tables and graphs.

status of the exercise and giving presentations. This facility needs at least one large screen display. The exercise director will need to control the portion of the battlefield and the level of unit resolution in the display. In many situations the director will wish to compare the true and perceived battlefield situations, and special software tools will be needed. For example, the director may need a single composite display that shows both true and perceived unit locations, using special display techniques (e.g., blinking or special colors) to point out the discrepancies. In order to control the large screen display, a videographics work station will also be needed. The director's staff may also use the work station for preparing a presentation (e.g., developing visual displays) and for controlling the actual presentation (e.g., bringing up the displays in the predefined sequence).

Figure 3-2 presents a sketch of the two facilities in this satellite. The components are not drawn to scale.

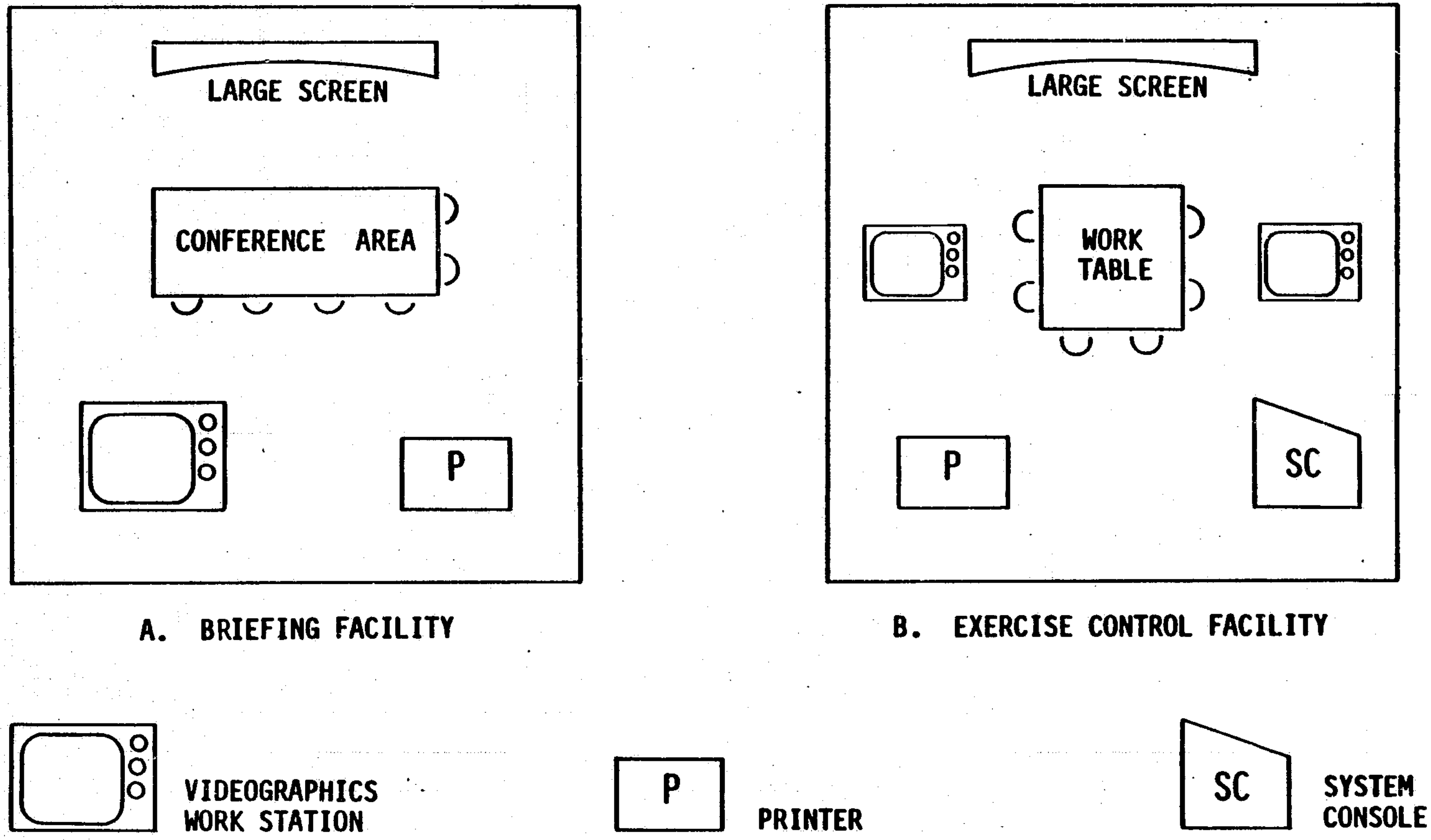


Figure 3-2. Sketch of facilities for exercise director's staff.

3.2 Satellite System for OPFOR Role Players

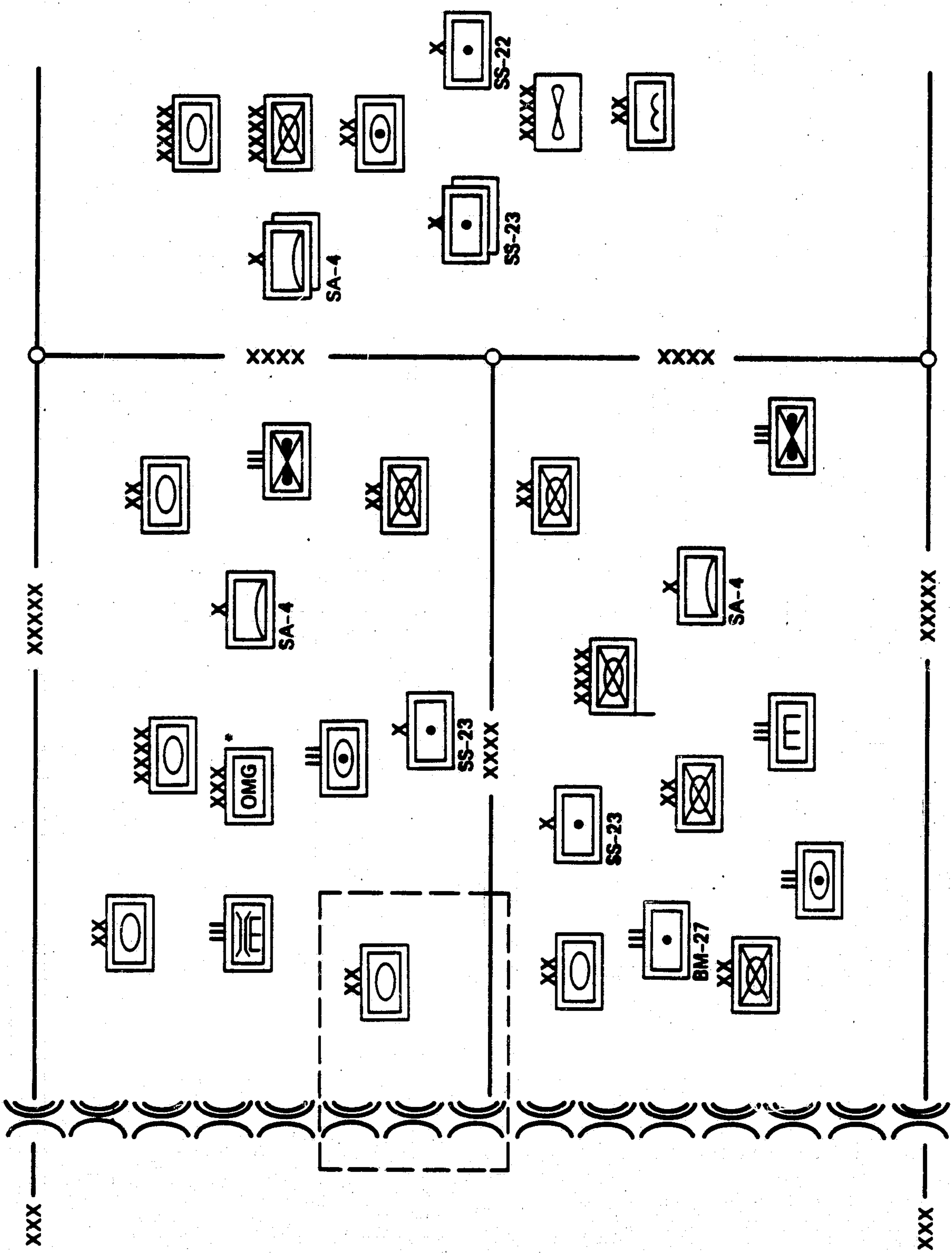
The OPFOR role players, in response to the exercise director, control all of the elements in the force opposing the Blue units in the exercise. The requirements (Ref. 1) state that for a corps CPX the exercise system must be capable of treating an opposing force comparable in size and composition to a Soviet front. The following discussion of such an organization provides a basis for estimating the number of work stations needed by OPFOR role players in a corps CPX.

Fronts do not have fixed TOEs like divisions, and assets may change somewhat because of current changes relative to TVD and TacAir. Nevertheless, it is appropriate to consider the following assets in determining the types and levels of action the OPFOR cell must be capable of performing. Figures 3-3, 3-4 and 3-5 present schematic diagrams for a front, one forward tank division of the front, and one tank regiment of the division. The unit locations are schematic and are not intended to show Soviet or Warsaw Pact doctrine or positioning. Units are assumed to be at full strength.

Fig. 3-3 gives typical aggregate frontal assets for our purposes. While heavier fronts exist they are uncommon. In the rear are shown two second echelon armies, not further delineated, a tactical air army, an artillery division and three theater ballistic missile brigades, two frontal long range SAM brigades and an attached airborne division. Army assets not shown are rear services, CSS brigades, radio electronic warfare regiments, signals, etc. Deep insertion forces (diversionary, air and air mobile assault brigades) and the intelligence regiment's assets are not shown; neither are most engineer assets.

In the forward portion there are two armies, broken down to divisional main units with brigade/regimental assets shown

FRONT ASSETS



*OPERATIONAL MANEUVER GROUP

in some cases. Additionally, an operational maneuver group (OMG) of approximately corps size is shown in the center of the upper forward area. It would represent roughly two tank divisions, reinforced with air, air defense, artillery, engineer and logistic elements and would be tasked to range deeply in Blue's rear.

The area occupied by a front depends on its mission and its size in divisions. A typical area for a front tasked to destroy a U.S. corps enroute to its final objectives would be roughly 100 km wide by perhaps 200 km deep. Its immediate objectives would be in the U.S. corps rear, perhaps 150 km on the Blue side of the FEBA.

A typical attack against a prepared U.S. defense might involve two thrusts by army sized units,(one a feint) one deep penetration by the above mentioned OMG, and a reinforced regimental sized airborne assault in the division or corps rear. Additional major decisions by the front commander (or the exercise director) would include use of artillery/missile/ air strike assets and their coordination, use of second echelon forces, major allocation of supplies among the several thrusts, and possibly a major rear area protection operation against a Blue air assault unit. This paragraph represents a level of activity fairly close to peak for the front commander and staff. The subordinate commanders for the army, OMG and airborne thrusts may have roughly similar task loads.

The army commander as an example will have (typically) two divisions attacking, two in a second echelon and (possibly) a company-to-battalion sized helicopter-inserted force operating in Blue's division rear. An OMG of reinforced regimental size is also possible. He will be attacking across (perhaps) a 40 km front, with objectives in the Blue division rear, 60 km beyond the FEBA. His depth may be 100 km. His rear area concerns will be similar (and smaller) to those of his commander.

Figure 3-4 displays a tank division in greater detail, showing four maneuver regiments, one regiment and two missile battalions of artillery, an air defense regiment, and two special battalions (attack helicopter and reconnaissance). Figure 3-5 provides the final description at battalion level by indicating the makeup of a forward tank regiment as it closes with the Blue forces: four maneuver battalions, three batteries of artillery, one air defense battery, one reconnaisance company, and one chemical defense company. Therefore the role player must monitor and control at least eleven units, possibly more if CSS units are also displayed. Additionally, coordination with higher echelon units operating in the division area is required. Thus, representing a division in which two lead regiments must be detailed will require approximately 35 tokens. These symbols may be distributed over differing areas, but 15 by 30 km would not be unusual. OPFOR divisions typically attack on a 15 km front, are 30-45 km deep, and have objectives about 30 km away (in the Blue brigade rear).

This description of potential OPFOR assets indicates that, for example, more than 300 maneuver battalions must be deployed. In order to manage these assets with a reasonably small number of role players, the system must provide a number of independent work stations and planning tools. For example, the system must allow the role player to generate movement orders in an aggregated fashion, using templates (defined by standard doctrine and procedures) to determine the relative locations of subordinate units.

Table 3-3 summarizes the results of our evaluation of work station requirements. Because of the large number of units involved, we believe that one work station would be needed to control the simulated units in each front-line division. The videographics display would typically cover a region 15 by 20

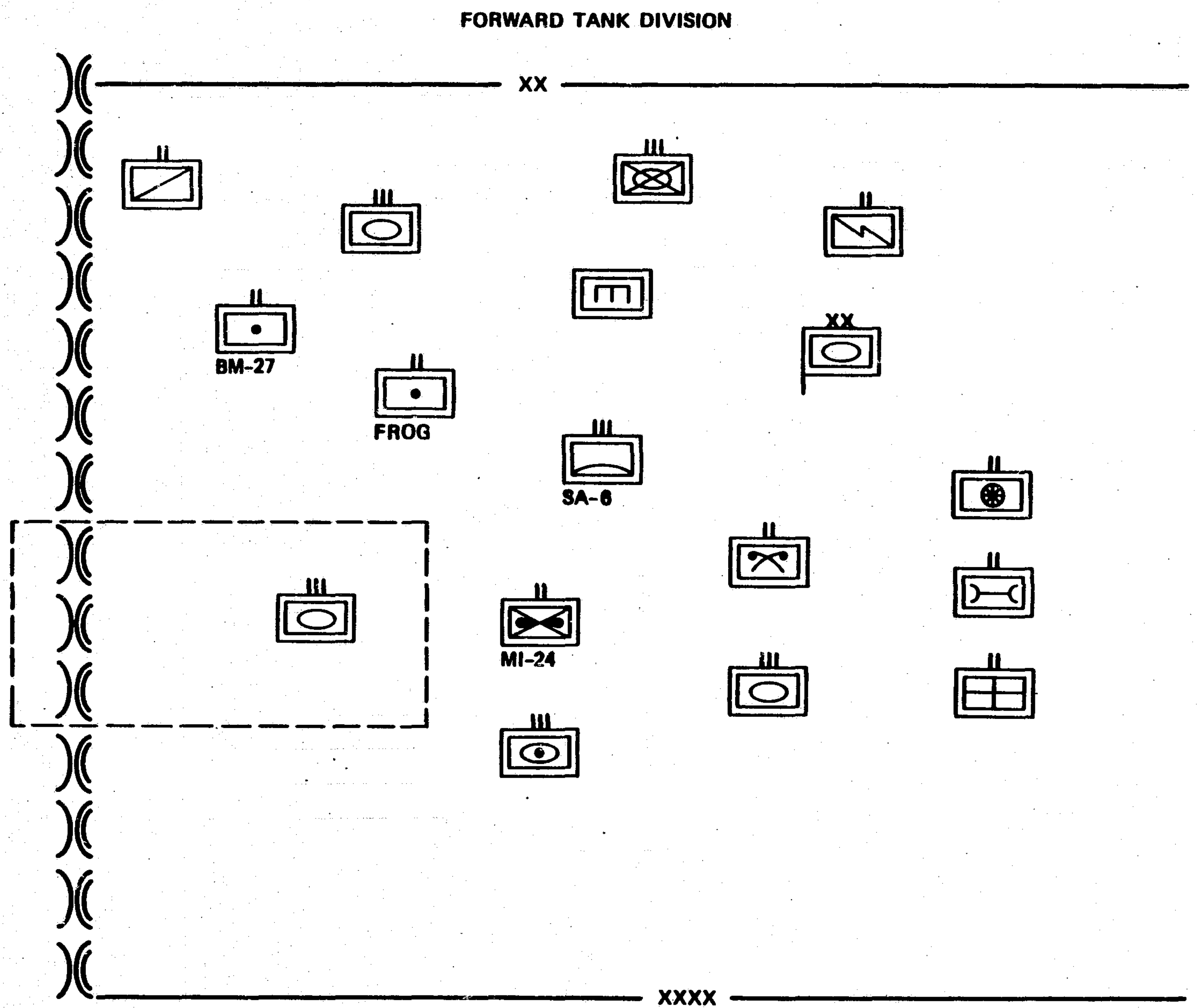


Figure 3-4. Schematic diagram of forward tank division within OPFOR front

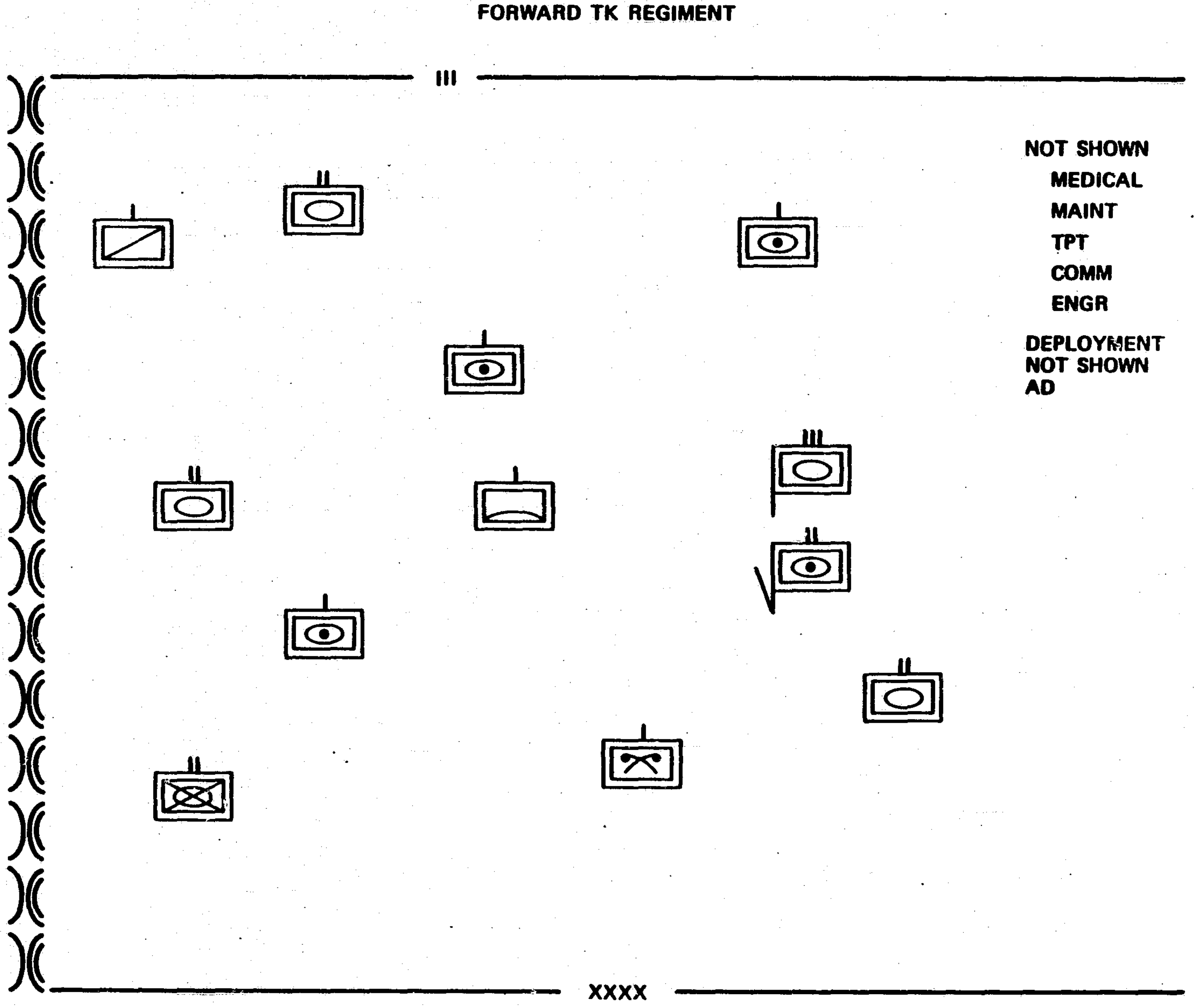


Figure 3-5. Schematic diagram of forward tank regiment withing OPFOR division

TABLE 3-3 SATELLITE SYSTEM FOR OPFOR ROLE PLAYERS

- Work stations for front-sized OPFOR:
  - Three for controlling maneuver units within front-line divisions.
  - Two for controlling lead army operations
  - One for controlling all frontal 2nd echelon units.
  - One for controlling artillery operations.
  - One for planning air operations and deploying air defense elements.
  - One for planning, intelligence, and logistics operations.
  - One for special functions such as conducting a deep insertion strike.
- Large screen displays provide overviews of entire area of conflict.
  - One for area from in front of FLOT to OPFOR rear.
  - One for area from FLOT to Blue rear.
- Four printers to generate reports.
- Fewer work stations would be required for OPFOR opposing a division instead of a corps.

km, with forward maneuver units represented to battalion size and combat support units (such as engineers, artillery, radio electronic warfare, and air defense) possibly represented to company size. The lead armies in the attack deploy significant assets in addition to those of the front divisions, so we believe that an additional work station is needed for each lead army. As the elements of a lead army move toward the FLOT, their symbols will automatically appear on the division displays, implying the need for coordination of these role players. Because of the lower level of activity in the rear, we believe that one work station will be adequate for controlling all 2nd echelon units. It would display approximately 75 by 100 km. We visualize separate work stations being needed for operations related to artillery, intelligence, and air operations. Finally, we recommend an extra work station which could be used to plan special operations (such as a deep strike) or to support any of the other functions during periods of stress.

Figure 3-6 presents a sketch of the OPFOR facility with ten videographics work stations, organized in a manner consistent with their primary functions, although any work station can be used for any function (e.g., the work station marked 'air operations' could also be used for planning artillery fire). The facility also includes two large screen displays to show an overview of the battlefield situation and support general planning. Despite the large number of OPFOR functions to be supported, we believe that this equipment suite will be adequate if full advantage is taken of the natural exercise opportunities for advance OPFOR planning and scripting.

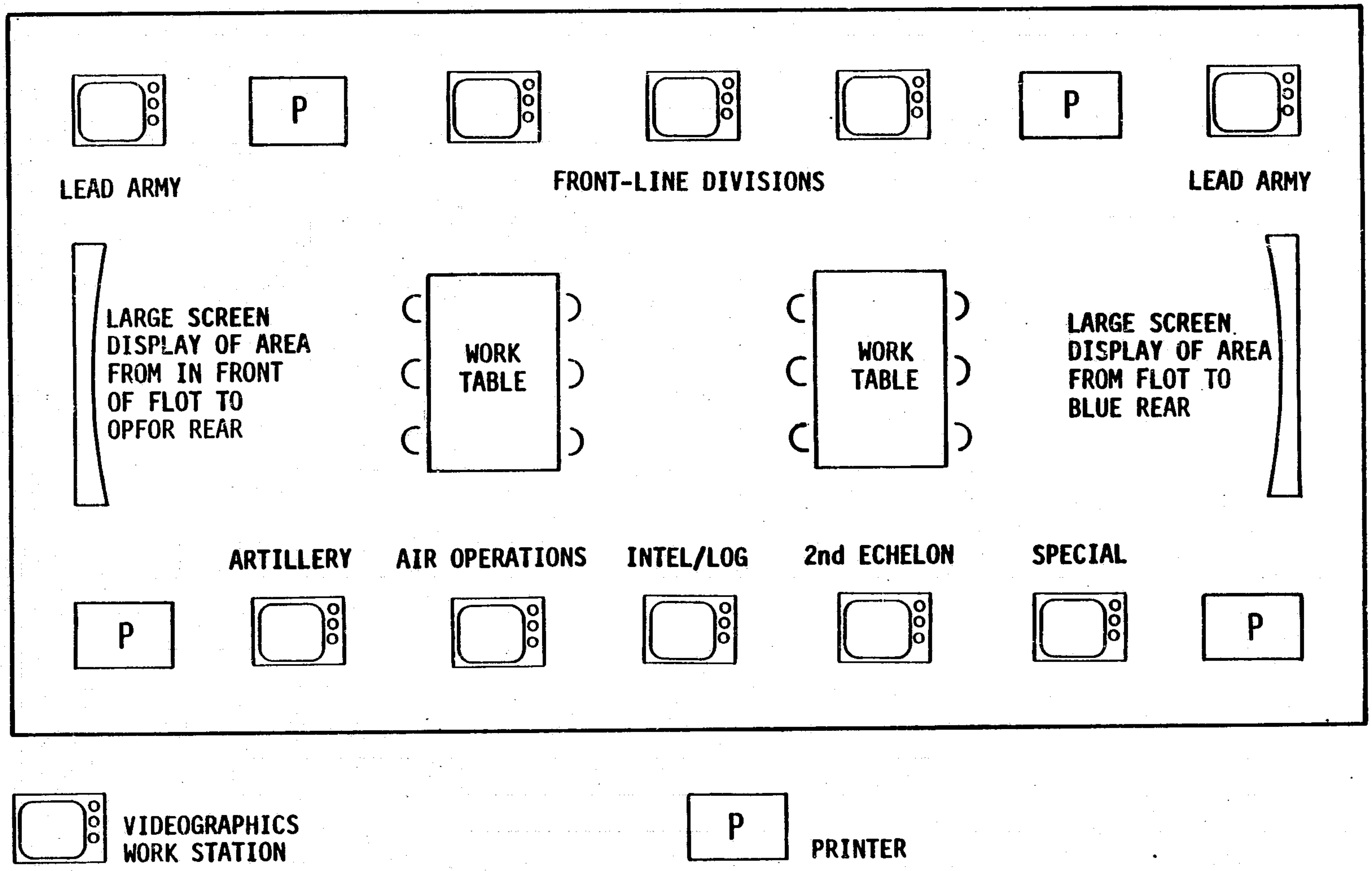

LEAD ARMY
FRONT-LINE DIVISIONS
LEAD ARMY
P
P
LARGE SCREEN DISPLAY OF AREA FROM IN FRONT OF FLOT TO OPFOR REAR
WORK TABLE
WORK TABLE
LARGE SCREEN DISPLAY OF AREA FROM FLOT TO BLUE REAR
ARTILLERY
AIR OPERATIONS
INTEL/LOG
2nd ECHELON
SPECIAL
P
P
VIDEOGRAPHICS WORK STATION
P
PRINTER

Figure 3-6. Sketch of OPFOR facility.

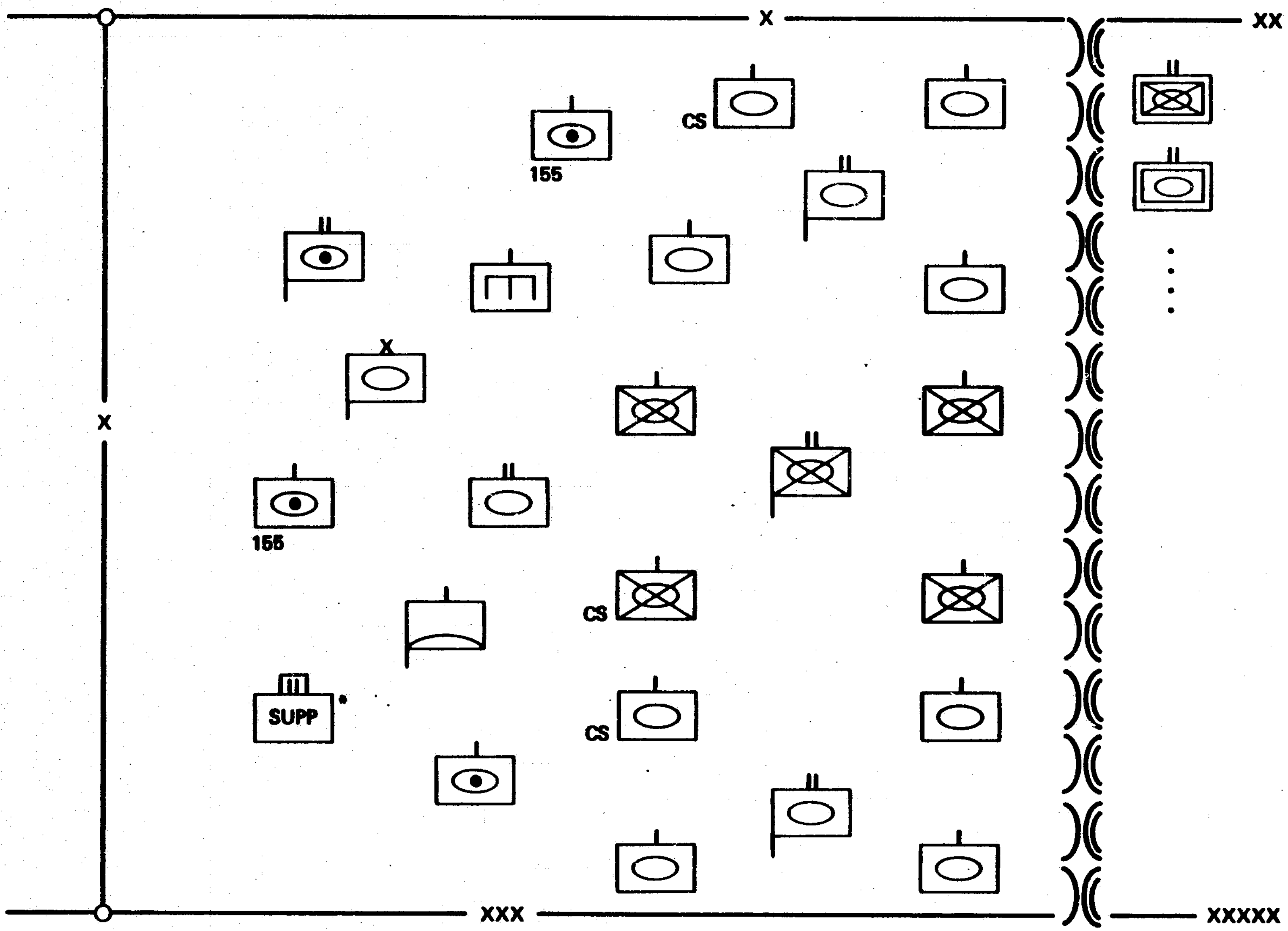
3.3 Satellite System for Brigade Role Players

The ACSS architecture provides separate satellite systems for the individual brigades reporting to a division TOC in a CPX so that the division staff can experience a realistic loading. All of the brigade satellites can communicate simultaneously with the division staff, each satellite submitting reports based solely on the information naturally available to the brigade being simulated.

The ACSS requirements (Ref. 1) state that the system must support simulation and display of Blue maneuver units at company resolution, largely because of the need to treat special battalion organizations with mixed companies. Figure 3-7 is a schematic diagram to clarify a discussion of brigade role player responsibilities; the unit locations are not intended to represent doctrinal deployments, and the battalion compositions are notional. The brigade role players are responsible for maneuvering as many as twenty companies (fifteen maneuver units and five combat support units)* as well as a battalion of artillery (organized in three batteries), a battery of air defense assets, and the equivalent of a combat service support battalion. There can be five HQ units for combat and combat support. The brigade area is approximately 15 km by 30 km. (At 1:50000, a 19" videographics display covers 15 km x 19 km, which may be sufficient). Its region of influence extends about 15 km beyond the FLOT and twelve hours into the future, while its region of interest may extend as much as 70 km beyond the FLOT and 24 hours into the future. An armor heavy brigade is shown; such an organization can be constructed from mechanized infantry or armored division assets, but not out of light infantry division assets. Additional divisional and corps units (such as armored cavalry or engineer) can be operating in the brigade area, and coordination will be required to a greater extent than in the

* This number corresponds to the doctrinal maximum of five maneuver battalions per brigade. The illustration shows four.

ARMORED BRIGADE DEPICTED AT CO/BTRY LEVEL



*TEAM EQUIVALENT TO ORGANIC BN IN SEP. BDE

Figure 3-7. Schematic diagram of forward BLUE brigade

equivalent OPFOR cell. In addition to the normal missions of active defense, the brigade operations may include special tasks such as using air defense assets against a flight of enemy transport and attack helicopters, or attacking the flanks of an OPFOR assault.

Tab:.e 3-4 summarizes the preliminary design for the satellite system to support brigade role players. The design includes three work stations, each with a videographics monitor, because we anticipate the need to support three simultaneous functions. One work station is needed to control the maneuver units, principally by generating movement orders and missions. Another work station is needed to control the direct support artillery assets, principally by directing fire and by generating movement orders. The third work station supports special functions, such as air defense operations and general planning. Each work station can be used for any of these functions, allowing the role players to tailor the usage to exercise requirements. For example, during periods of heavy engagement it may be necessary to use two work stations for controlling the maneuver units.

Figure 3-8 presents a sketch of the facility for the brigade role players, which must include sufficient equipment for the role players to communicate with the relevant exercise players.

TABLE 3-4 SATELLITE SYSTEM FOR BRIGADE ROLE PLAYERS

- One work station (with videographics) primarily for controlling maneuver units (e.g., generating movement orders).
- One work station (with videographics) primarily for controlling direct support artillery (e.g., processing fire mission requests).
- One work station (with videographics) primarily for general planning and special functions such as intelligence and personnel.
- One printer to generate hard copies of the reports and requests from the simulated units.
- Same architecture will support battalion role players reporting to a brigade TOC.
- Architecture can be expanded by adding work stations (e.g., to provide a separate work station for each battalion in the brigade).

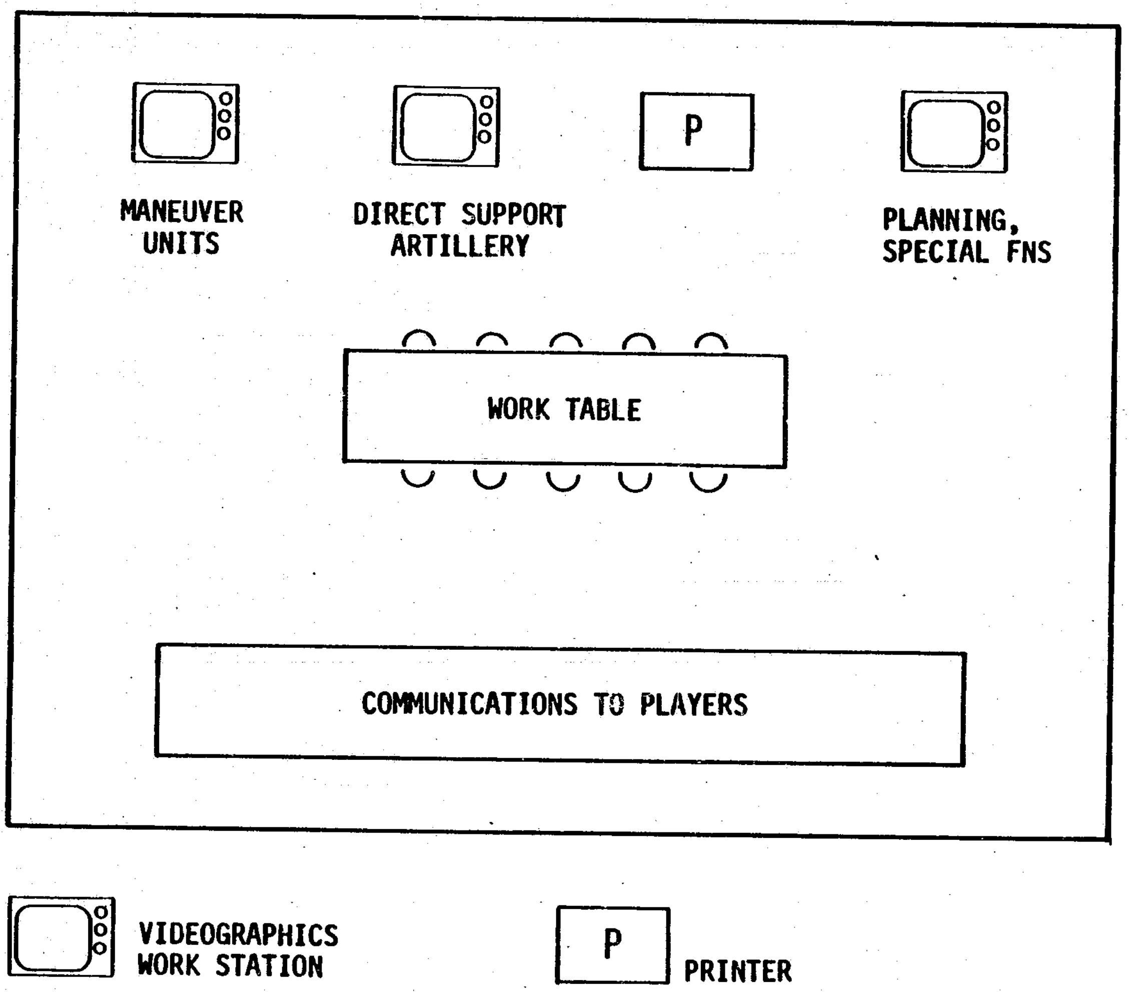


Figure 3-8. Sketch of brigade facility.

3.4 Satellite for Combat Brigade (Air Assault) Role Players

Divisional air assets can be organized in three ways: the Combat Brigade (Air Assault), currently unique to the HTLD, the similar Division 86 Air Cavalry Attack Brigade which can be either a divisional or a non-divisional asset, and the earlier arrangement of an aviation battalion and an air cavalry squadron, which together provide somewhat similar aviation assets. The CB(AA) organization is assumed in the discussion which follows.

The CB(AA) is organized in two attack battalions, a cavalry squadron (which has two troops of ground cavalry as well as two air cavalry troops), a combat support air battalion (which also handles aviation-peculiar maintenance and supply), and a medium lift helicopter company. Units are usually deployed by troop/company with supply points at battalion level.

The CB(AA) is treated as a single maneuver brigade when it exists in role player status. Its satellite has similar duties and responsibilities to those for the other maneuver brigades, with a few exceptions that result from the CB(AA) containing essentially all of the divisional air assets, and being almost exclusively an air unit.

The first difference is in the region of concern, which extends over the entire division area and well forward of the FEBA, both for direct fire attack helicopter missions and for insertions using assault nelicopters. The second difference is in the absence of direct support artillery, which reduces some aspects of the workload, though CB(AA) units, because of their forward position, may be expected to generate a fairly large number of calls for fire. The third difference is that the CB(AA) satellite handles CSS functions that use aviation assets or are unique to aviation assets.

A feature of the CB(AA) cell is the need for unusually close coordination with all ACSS cells, and with all player cells, because of geographic and functional relationships. A particular need is to coordinate with the CS cell, since the Division and Corps Air Management Elements, which manage air space, are in the CS satellite as part of the Air Defense Artillery function.

The size of the CB(AA) satellite is not expected to change from the brigade satellites described in section 3.3 above, and its physical configuration will be similar to that of a maneuver brigade satellite as described above.

3.5 Satellite System for Combat Support (CS) Role Players

The ACSS architecture provides a satellite system for role players performing non-played CS functions. The CS functions are:

Air Defense
Army Aviation (Corps)
Artillery
Chemical Defense
Communications
Engineers
Rear area operation control
Tactical air (USAF)

Generally these functions are performed by division assets of battalion size which are appropriately dispersed within the division area. Certain functions will be augmented from corps and some will be attached downward to brigade (e.g. artillery shows both cases). In addition Corps has retained assets.

Air Defense Artillery: (ADA) The divisional ADA assets are organized in a single battalion of five batteries tasked with the SHORAD mission. (HIMAD coverage comes from EAC.) Three batteries contain DIVAD guns, and are deployed in support of the brigades. Two contain improved Chapparral missiles and protect critical assets in the division rear. Each battery has a platoon of MANPADS (Stinger) which is deployed by sections (18 sections overall) generally in defense of specific points, convoys, and formations. The ADA battalion also has the division airspace management function as well as threat detection assets (FAAR). Corps has an organic capability responsible for protecting the corps rear area, using I-Hawks and SHORAD assets. The Corps ADA assets include the Corps Air Management Element.

Army Aviation (Corps): While divisional assets are played as a maneuver element, the roughly equivalent sized Corps aviation assets are played as combat support. These assets typically consist of 4-5 battalion level units: one or two attack squadrons, employed in general support, two battalions of lift assets, one Chinook and one lighter utility unit (Huey or Blackhawk) and a command battalion, serving the corps command element. The use of these elements also will be controlled from the ADA work station.

Artillery: Divisional fire support assets typically consist of four artillery battalions, three (155 mm) being assigned to the brigades for direct support and a heavy battalion (8" SP and MLRS) retained for general support artillery; a heavy division typically would receive four battalions organized as a brigade: one 155 mm SP, two 8" SP, and MLRS. These usually would be maneuvered by batteries; target acquisition assets would be dispersed among the forward command posts of supported units and would move typically with them. Artilley assets retained by corps are variable; they tend to be battlefield missiles, e.g. Lance or the future CSWS.

The direct support battalions would be operated from the maneuver brigade satellites, while the general support battalions would be operated from the combat support satellite.

Firing batteries will be located 3-10 km behind the FEBA depending on caliber and posture. Headquarters and organic support are further to the rear and will include both radar and RPV acquisition means.

Engineers: The division has an engineer battalion which is composed of four engineer companies and a bridging company. Typically three companies are in direct support to the brigades, the others being in general support to the division. Corps assets will vary; a heavy corps is claimed to have four battalions per division available. Use of these assets would be under the control of the division engineering officer. The most urgent needs will be near the FEBA; other engineering tasks will exist toward the rear.

Critical engineering tasks are removing and constructing barriers, combat bridging, and fighting as infantry when needed. Engineers take a leading role in military operations in urban terrain, both offensive and defensive.

Other assets played from the combat support satellite are the chemical defense, communications, MP (RAOC) and (via the DAME in air defense) the USAF TacAir activity. Generally these activities do not require action by role players. When they do, it is in response to an intervention in the game on the part of OPFOR or the exercise director for the purpose of exercising the players. The principal duty of the CS cell is to detect the activity, take local action as appropriate, forward to player echelons the impact of such action, and react as ordered.

The description indicates a large number of functions must be performed, each of which is, in itself, fairly infrequent. Overlaid on this background are the higher frequency artillery function (minus direct support) and at a lesser frequency, the engineer function. The functions have in common being played over a division area rather than a brigade area (some corps assets may require a larger field of action still, CSWS being one such case, but these can be expected to be infrequent.)

Table 3-5 shows our evaluation of work station requirements assuming one division and corps are played heavily and two divisions are played lightly (i.e. the intensity and tempo of combat is much less for the two supporting divisions than for the "star"). We expect the highest intensity CS function will be the general support and reinforcing artillery functions (direct support being handled out of the brigade satellites). Based on the number of units to be deployed and fired, three workstations are allocated to this function. A separate work station is dedicated to the air defense function because, in large part,of its division air management and Air Force Tac Air liaison responsibilities. The remaining functions require a large variety of interactions at fairly low frequency. Three work stations are allocated to these functions to reduce the impact of shifting from task to task, although additional analysis and exercise experience are required to validate this estimate.

The consolidation of division and corps CS functions in the same satellite is to allow efficient use of staff and resources in all loadings, regardless of organizational attachments to corps or division, or the changing workload as the CS planning activity works its way down the chain of command. Figure 3-9 depicts one such satellite configuration, which includes a large screen display for a coordinated view of the area of responsibility.

TABLE 3-5 SATELLITE SYSTEM FOR COMBAT SUPPORT ROLE PLAYERS

- Three work stations for general support, general support reinforcing and reinforcing artillery.
- Work station for DAME function, air defense and Corps aviation.
- Three work stations for engineer and "other" activities
- One large screen display for overview of CS operations
- Three printers for reports
- This system performs Combat Support for one heavily played division, two lightly played divisions, and corps.

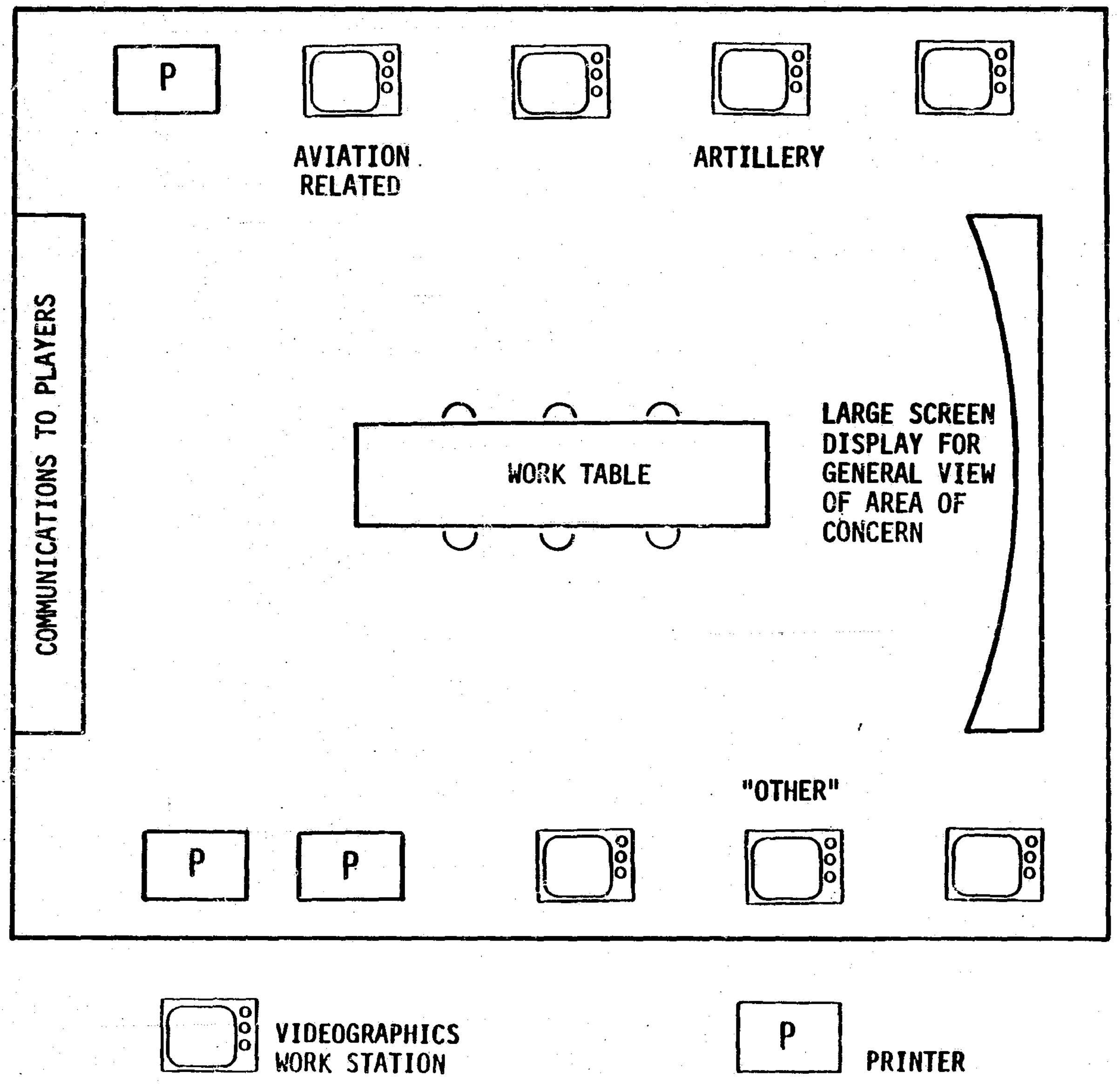

P
AVIATION RELATED
ARTILLERY
COMMUNICATIONS TO PLAYERS
WORK TABLE
LARGE SCREEN DISPLAY FOR GENERAL VIEW OF AREA OF CONCERN
"OTHER"
P
P
VIDEOGRAPHICS WORK STATION
P
PRINTER

Figure 3-9. Sketch of combat support facility.

3.6 Satellite System for Combat Service Support (CSS) Role Players

The ACSS requirements report indicates that unless appropriate actions in the requisition channels occur, resupply, reinforcement, evacuation, and repair activities should not occur, and that appropriate reductions in unit effectiveness should be levied. Also, damage to CSS facilities should reduce stockpiles and repair rates. From this we may infer that the CSS satellite has four basic functions: positioning and movement of facilities to support the changing battlefield, the allocation of resources within the role played units, the forwarding to players of requests generated by role played units, and the insertion into the model of the player responses to those requests.

The forward support battalion attached to the brigade represents the lowest echelon where this process comes under role player purview.

The number of CSS locations that might be targeted, and whose movement should be controlled, appears to be about four per battalion, eight in the brigade rear, sixteen in the division rear and twenty five in the corps area. Higher echelon targets will be larger, more lucrative, and less frequently moved.

In general the requisitioning process moves from the battalions up, with requisitions filled at the lowest possible level and, if not filled, consolidated and sent on up the chain. Initially three classes of supply will be considered: POL (Class III), ammunition (Class V), and everything else.

The medical and maintenance functions operate in analogous fashion, with wounded soldiers and broken material being shipped back to progressively more elaborate facilities, where they either are healed (repaired) and returned to duty or sent back further to the rear.

The requests that can be handled forward (essentially organic to maneuver battalion) are done internal to the simulation. The role players are provided automated aids to perform the consolidation function and satisfy routine requests at higher echelons. The intent is to retain the consolidation process, so that the role players get training value, while reducing the time required to perform it and cutting manpower. Depending on the degree of manning at DISCOM and COSCOM player cells, a point is reached where the aggregation of requisitions is converted to SOP format and sent to the players. The players react (possibly performing consolidations and forwardings of their own) and responses return to the role players. The role players act on the responses and put the appropriate information back into the simulations.

The discussion above suggests a four workstation satellite. One station will be required for site and route relocation, one will be used as a general purpose spare, and two for supply, personnel and maintenance, where training benefit ensues from modeling the movement of requisitions up the chain by sending from workstation A to B, and sending back to A. The number of workstations will grow if more units are modelled or if the number of supply categories grows substantially.

Table 3-6 summarizes the satellite configuration for combat service support, and Figure 3-10 provides a sketch of the facility.

TABLE 3-6 SATELLITE SYSTEM FOR COMBAT SERVICE SUPPORT ROLE PLAYERS

- Four work stations minimum (three divisions plus corps).

- Three printers to handle larger than ordinary report load.

- Videographics not required on all workstations as system grows (will be required on first four for redundancy).

- Skeleton player manning of DISCOMs and COSCOM assumed.

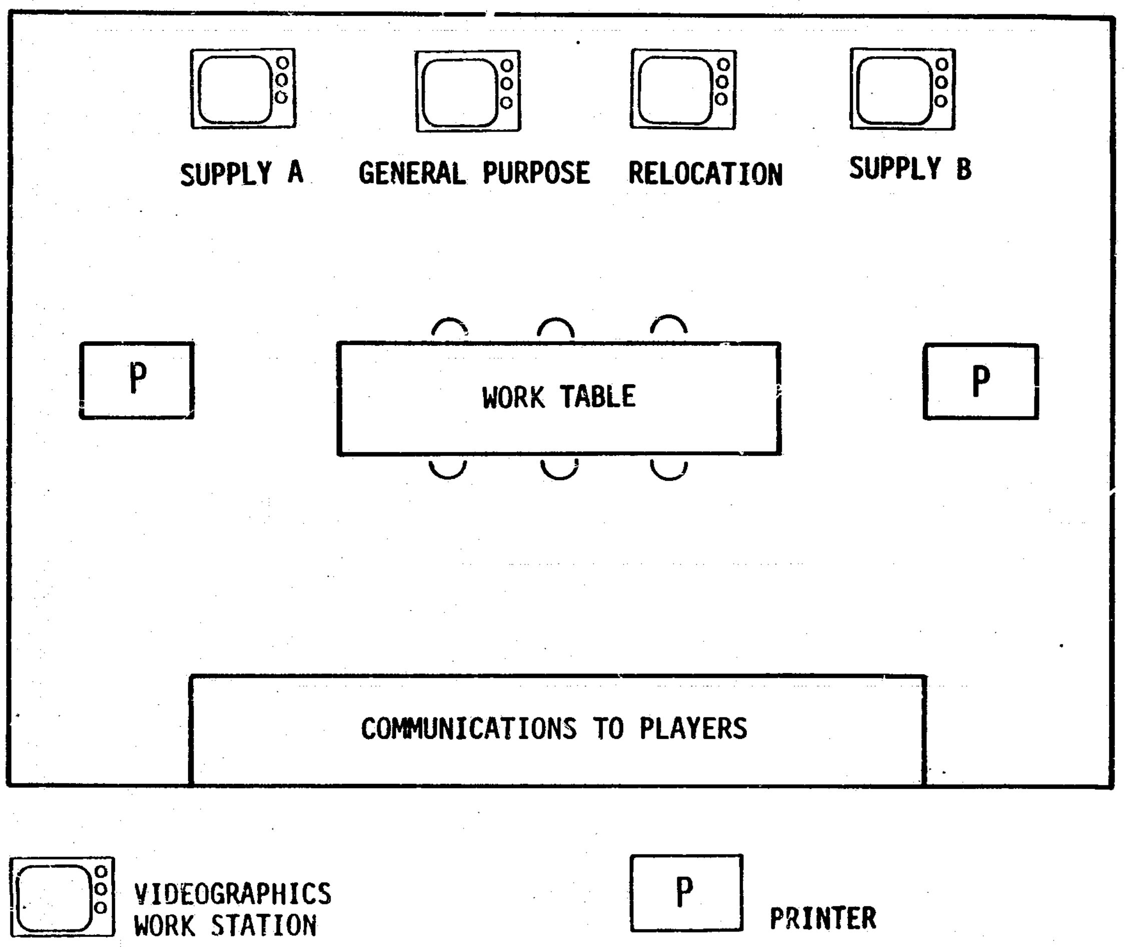


Figure 3-10. Satellite system for combat service support role players.

4. EXAMPLE COMPUTER SYSTEMS

The ACSS functions discussed in previous sections could be implemented in a variety of computer architectures. Figure 4-1 summarizes some of the most prominent candidates. A centralized system, consisting of terminals connected to a powerful minicomputer or mainframe computer, would have the advantage of low complexity. Nevertheless, while a single powerful computer might be able to support all of the simulation functions needed for a CPX, it would be difficult to drive independent graphics displays and support multiple users. A networked system, consisting of multiple microcomputers linked together with broadband communications, would support a large number of interactive users, but at the cost of high complexity, particularly if the models and database are partitioned among the processors. The distributed architecture uses microcomputers to handle the interactive and display functions and a minicomputer to handle the simulation and centralized database functions. The hardware cost is apt to be higher for the distributed architecture than for the network architecture, but this design can provide more flexibility and may be less complex to implement. Therefore, we have selected this architecture as the baseline for the preliminary ACSS design.

We have chosen representative computer systems consistent with the distributed architecture in Figure 4-1 to serve as examples and as a basis for planning. We recommend that the actual development of an automated CPX support system should start with a detailed design phase that would include identifying and evaluating hardware options. Furthermore, because of the uncertainties inherent in developing a corps-level system, we recommend an evolutionary development approach to minimize risk and maximize system performance. The following paragraphs describe the example computer systems

- CENTRALIZED
  - LOW COMPLEXITY
  - CENTRALIZED DATABASE
  - HIGH SPEED COMPUTATION
  - HIGH HARDWARE COST
- NETWORK
  - HIGH COMPLEXITY
  - DISTRIBUTED DATABASE
  - PARALLEL COMPUTATION
  - LOWER HARDWARE COST
- DISTRIBUTED
  - MEDIUM COMPLEXITY
  - CENTRALIZED DATABASE
  - DISTRIBUTED COMPUTATION
  - HIGHER HARDWARE COST
  - MORE FLEXIBILITY

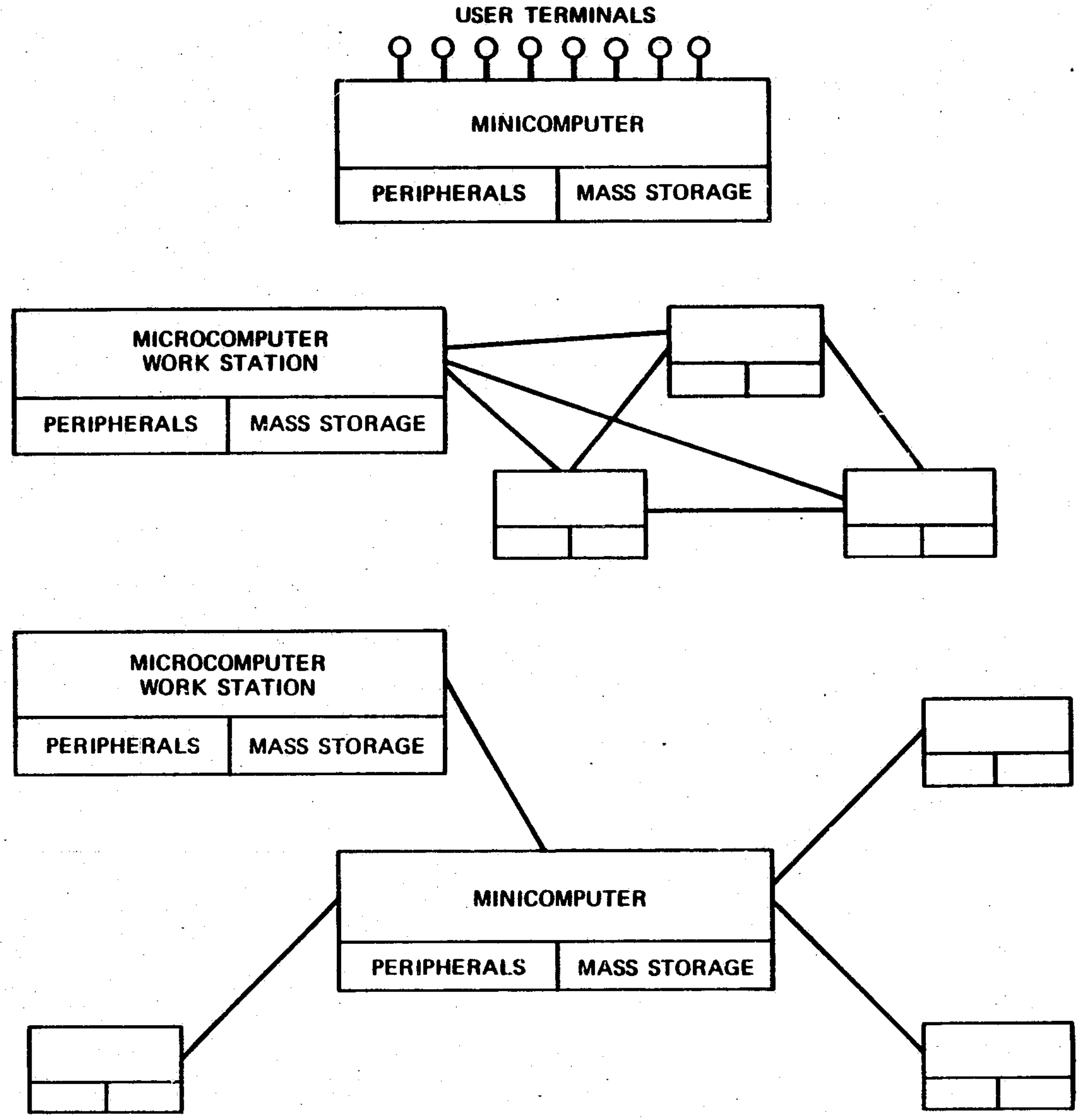


Figure 4-1 Spectrum of candidate architectures

C-2

and provide preliminary cost estimates for a full system to assist planners in determining the potential scope of effort. The prices listed are nominal prices from the current vendor catalogs, and do not reflect discounts or special government rates that might be available. Computer hardware changes quickly in prices and availability; many of the examples described here were not available six months ago, and it is reasonable to expect that many new options will appear during the next six months.

The example minicomputer chosen for the central computer system in the ACSS preliminary design is the VAX 11/780, made by the Digital Equipment Corporation(DEC). Although there are other minicomputers (e.g., Prime or Perkin-Elmer) that could be considered, the VAX is a well-established product with broad support from DEC and other vendors, it is widely used by the Army and other government agencies, and RARDE has adapted FOURCE (Ref. 3) to execute in this environment. The current experience is that FOURCE runs three to four times faster than real time on a VAX 11/780. This fact implies that a less powerful (and less expensive) computer such as the VAX 11/750 might be adequate for a division-level CPX. TRASANA, however, believes that a corps-level CPX would require approximately five times as much computational power as a division-level CPX, implying that even an 11/780 might be inadequate. Nevertheless, we believe that the 11/780 is a reasonable choice for planning purposes because it is clearly adequate for a division CPX, potentially adequate for a corps CPX, and it has growth capability if computational requirements are greater than expected. First, the 11/780 can be upgraded to an 11/782, which is a dual-processor system. Second, the 11/780 can be upgraded to a VAX cluster, which allows multiple VAX computers to operate cooperatively. Third, the 11/780 can be augmented with slave processors to perform simulation functions (e.g., a separate slave processor to execute FOURCE for each division). The slave processor might be a relatively inexpensive microcomputer such as the new DEC MICRO-VAX, which uses a 32-bit processor and is compatible with VAX systems. For these reasons, we believe that a central system based on DEC equipment would have enough growth capability to handle the needs of a CPX system.

Table 4-1 summarizes the example computer hardware for the central system, based on discussions with the vendor and data derived from current catalogs (Refs. 4 - 6). The basic VAX 11/780 system comes with 2 M bytes of main memory, which can be expanded to 16 M bytes within the same cabinet or to 32 M bytes by adding another cabinet. We believe that 4 M bytes is the minimum needed for this application. Furthermore, a floating-point accelerator is needed for the numerically-intensive simulation programs. We have selected one of the new UNIBUS disk subsystems that provide large capacity at low price, although a removeable disk system might facilitate security procedures. We include a programmable communications processor because of the large number of satellite systems in the ACSS architecture. The VT102 alphameric terminals, which will be used by the systems staff, provide more features and are less expensive than the VT100 terminals (which used an out-sized power supply to support optional processor boards that are no longer available). The laser printer is included because it is fast, quiet, and produces report-quality masters. It might be desirable also to include a regular line printer for producing program listings, but the laser printer may be sufficient for all applications. The software includes the VAX/VMS virtual memory operating system and a Fortran compiler, which is needed for FOURCE. The nominal purchase price is approximately $300,000, and the associated mainten-

TABLE 4-1 EXAMPLE MINICOMPUTER COMPONENTS FOR CENTRAL SYSTEM

	Element	Price	BMC
1)	VAX 11/780 with 2MB of main memory (780XA-AE)	145,000	407
2)	2MB of additional main memory (MS780-FA)	9,000	58
3)	Floating-point accelerator (FP780-AA)	11,200	50
4)	Fixed disk subsystem with 456 MB capacity (RUA81-CA)	26,000	120
5)	Magnetic tape subsystem; 1600/800 BPI (TEU77-FB)	36,800	259
6)	Communications interface (DZ11-DP)	2,175	33
7)	Communications processor (FEPCM)	20,000	100
8)	Console Subsystem (LA120-DA)	2,800	32
9)	2 Alphameric terminals (VT102-AA)	3,420	44
10)	Laser printer (LN01-CA)	19,995	320
11)	VAX/VMS operating system (QE001-AZ,HM)	10,000	NA
12)	VAX-11 FORTRAN 77 Compiler (QE100-AY)	8,700	NA
	Total:	$ 295,090.	$ 1423./month

ance price or basic monthly charge (BMC) is slightly less than $1500 per month.

Each satellite system contains a number of work stations that support interactive videographics functions. A microcomputer is needed in the work station to generate the graphics overlays and respond to user input. The example work station in the ACSS preliminary design is based on a new DEC product, the Interactive Video Information System (IVIS). IVIS, which will be released in December of 1983, is explicitly designed for interactive videographics applications, and hence provides an integrated package well-suited to ACSS. IVIS is based on the DEC Professional 350 microcomputer, which uses the PDP-11/23-PLUS 16-bit processor. Table 4-2 describes the example work station. The basic IVIS package includes both hard and floppy disk systems, a floating point adapter, and a color monitor, so only a videodisk player and a communications package (to exchange files with the VAX) are needed as extras. IVIS includes a multi-tasking operating system, which is required in the ACSS design so that communications and database tasks can execute in the background without interfering with the user's interactive tasks. We have inspected IVIS in several demonstrations, and we were favorable impressed with the display quality, particularly its stability; the IVIS display exhibits essentially no flicker, which is a problem for many of the existing videographics systems. Nevertheless, IVIS is a new product and not enough information is available at this time to verify its adequacy for ACSS. Table 4-3 itemizes the principal considerations. Our summary is that we believe that IVIS would be a cost-effective component for ACSS, but there is some risk because it has very limited growth capability. If further analysis indicates that IVIS is not adequate, then it would be necessary to design a work station using a separate microcomputer

TABLE 4-2 EXAMPLE WORK STATION

DEC IVIS (Interactive Video Information System)

- Professional 350 microcomputer with 512 KB of main memory
- Hard disk subsystem (10 MB)
- RX50 floppy disk subsystem (800 KB)
- Floating point adapter
- Videodisk interface
- Extended bit-map module
- 13-inch RGB monitor
- P/OS operating system

DEC PRO/Communications package (QB005-C3)

DEC NPL Database Software (QA117-C3)

DEC Special Extended Warranty (12 months)

Sony LDP1000 Videodisk System
Nominal prices:

IVIS:	12,600
PRO/COMM:	195
Database software	400
Special Warranty:	200
Videodisk player	2,475
Total	$ 15,870

TABLE 4-3 IVIS CONSIDERATIONS

Advantages:

- Lower cost than other systems with comparable capability.
- Integrated package (microcomputer, videographics mixer, and display generator) simplifies set up and use.
- Flicker-free RGB display.
- Hardware and software are compatible with central minicomputer, simplifying file transfers.

Disadvantages:

- New product, although used by DEC for training programs during past year.
- Vertical resolution marginal (240 lines); circles have visible stair steps.
- Single-user operating system restricts flexibility.
- Memory limited to 512 KB until DEC upgrades to higher-density boards.
- Communications channel capacity limited to RS232 asynchronous in current hardware configuration.

and display generator. For example, the microcomputer could be the DEC MICRO-11 (which has a multi-user operating system and the capability to use large memory systems) or one of the many systems (e.g., Wicat, Pixel, Charles River) based on the Motorola 68000 processor, and the display generator could be a device such as the Symtec PGS-III or the New Media Graphics GraphOver 9500. This design would increase the price of the work station by perhaps $10,000, but would also provide much more capability. For planning purposes, we assume that the IVIS design is sufficient.

Application software needed for the work stations could be developed either directly on the microcomputers or on the minicomputer; in the latter case, the developed software would be down-loaded to the microcomputers for testing and use. Developing the source code on the minicomputer facilitates configuration management by keeping all code in a single library. Furthermore, the current DEC pricing policies discourage using microcomputer-based tools for systems such as ACSS with multiple computers because the tools must be bought separately for each computer. Therefore, we have selected the minicomputer software listed in Table 4-4 to support the development of application software for the work stations.

Tables 4-5 and 4-6 contain additional examples of candidate software for the central minicomputer system to support the database management and configuration management functions. The examples include products from both DEC and other vendors. For planning purposes, we have selected the Software House System 1032 database management system and the Softool Corporation configuration management software for the baseline. Table 4-7 describes an example uninterruptible power system (UPS), which we believe has the capacity to protect the minicomputer and the disk subsystem from power anomalies.

TABLE 4-4 EXAMPLE DEC MINICOMPUTER SOFTWARE

● Professional Developer's Tool Kit QJ071-AM	Price:	$4,000
● FORTRAN - 77 Compiler for Tool Kit QJ074-AM	Price:	$2,185
● Pascal Compiler for Tool Kit QJ082-AM	Price:	$2,300
● IVIS Software for Videographics	Estimated Price	$15,000
	Total Price:	$23,485

TABLE 4-5 EXAMPLES OF DATABASE MANAGEMENT SYSTEMS

Example DEC software:

- VAX-11 Forms Management System ($7,500).
- VAX-11 Common Data Dictionary ($3,000).
- VAX-11 Datatrieve ($15,000).
- File management system with a central repository of data definitions.
- Interfaces with application languages.
- Report and query facilities.

Example non-DEC software:

- Software House System 1032 ($40,000).
- Integrated software system.
- Relational database management system.
- Designed for optimum performance in VAX environment.
- Interfaces with application languages.
- Report generator.
- Query capability with extensive help and prompt features.

TABLE 4-6 EXAMPLES OF SOFTWARE CONFIGURATION MANAGEMENT SYSTEMS

Example DEC software:

- Code Management System ($15,000).
- Module Management System ($3,300).
- Maintain multiple files as a project library.
- Generate reports on changes in code.
- Build system by updating only modules that have changed.
- Collection of separate utilities accessed from command level.

Example non-DEC softwares:

- Softool Corporations's Softool for FORTRAN ($80,000).
- Integrated set of tools for all phases of software development.
- Preprocessors for developing structured code.
- Special tools for static and dynamic testing.
- Satisfies most MIL.STD requirements for software configuration control.

TABLE 4-7 EXAMPLE OF UNINTERRUPTIBLE POWER SYSTEM (UPS)

Gould Model 6156-1 UPS

- Rated at 15 KVA output.
- Input 208 VAC, 60 Hz.
- Output 120/208 VAC, 60 Hz.
- Provides line isolation as well as power conditioning.
- Protects against undervoltages as well as spikes.
- Battery bank protects against blackouts.

List Price: $26,900.

We believe that a system of this type would be a prudent investment, given the cost of the computer system and the potential implications of a disruption during an exercise.

Tables 4-8 and 4-9 provide examples for peripheral devices needed in the satellite systems. The Electrohome ECP1000 is a new large screen display expressly designed to operate with computer-generated images. The Tally printer has many desirable features and is much less expensive than the standard DEC printer.

Table 4-10 summarizes the list prices for the elements in the central system. The total price is about $550,000, including an allowance for items such as taxes and spares, but excluding site preparation (e.g., power distribution and air conditioning).

Table 4-11 summarizes the list prices for the elements in the satellite systems. For planning purposes, we have assumed a full corps CPX system that supports three divisions with four maneuver brigades in each division. The total price is about $1,300,000. Again, we have not attempted to estimate the costs associated with site preparation.

TABLE 4-8 EXAMPLE LARGE SCREEN DISPLAY SYSTEM

Electrohome Electronics ECP1000:

- Designed for computer graphics displays (RGB); composite video optional.
- Adaptable to front and rear projection screens, 5 ft. through 7 ft. diagonal.
- Single lens system and range finder lights simplify alignment.
- Optical resolution of 600 lines.
- Aspect ratio of 3:4.
- Ceiling installation optional.

Nominal prices:

ECP 1000 projection system	$14,800
Ceiling mount	200.
Cart	550.
VS6-56 curved screen (6 ft. diagonal)	499.
Total price, including cart instead of ceiling mount:	$15,849.

TABLE 4-9 EXAMPLE PRINTER FOR SATELLITE WORK STATIONS

Mannesmann Tally MT 160L:

- Dot matrix printer:
  - 7 x 9 (data processing mode).
  - 20 x 18 (correspondence mode).
  - 128 x 133 (per inch in graphics mode).
- Multiple character sets:
  - 96 ASCII characters.
  - 11 Graphic symbols.
  - 31 Scientific characters.
- Print speed
  - 160 cps (data processing mode).
  - 40 cps (correspondence mode).
- RS 232C interface, 150-9600 baud.
- Tractor feed optional at extra cost.

List price: $798.

TABLE 4-10 SUMMARY OF CENTRAL SYSTEM

ELEMENT		PRICE
DEC VAX 11/780 system with a 4 MB main memory, 456 MB disk system, peripherals, and software.		295,090
Software tools for developing work station programs.		23,485
Database management system.		40,000
Configuration management system.		80,000
Uninterruptible power system.		26,900
Allowance for taxes, supplies, spare parts, and miscellaneous items		80,000
	Total	$545,475

TABLE 4-11 SUMMARY OF SATELLITE SYSTEMS FOR A FULL CORPS CPX SYSTEM

Satellite	IVIS work stations	Printers	Large screen displays	Price
1) Maneuver brigades (12)	36	12	0	$580,896
2) Independent brigade	3	1	0	$48,408
3) Combat support	7	3	1	$129,333
4) Combat service support	4	3	0	$65,874
5) OPFOR	10	4	2	$193,590
6) Exercise director's staff	3	2	2	$80,904
7) Allowance for taxes, supplies, spare parts and miscellaneous items	-	-	-	$200,000
Total:	63	25	5	
Total price:	$999,810	$19,950	$79,245	$1,299,005

5. FREEDOM HALL IMPLICATIONS

The following preliminary comments on physical layout assume the use of Freedom Hall, Ft. Lewis , WA. Maximum use is made of existing spaces, additional partitions being added as required and minimal changes being made to existing partitioning.

Figure 5-1 is a 2:1 reduction of an architectural sketch of Freedom Hall as of 21 July 1983. The lettered notations are for identification and refer to present usage as appropriate.

Figure 5-2 indicates proposed usage, including thirteen brigade spaces, a combat support space, a combat service support space, the OPFOR space, the exercise director's facility, the briefing facility, and the facility for echelons above corps (EAC).

5.1 Use of Present Space

Storage room A will continue to be used for storage. Rooms B-H will be used for up to seven of the brigade cells. These rooms vary somewhat in size, G and H being marginal, and therefore used for less active units. When shelves are set up in G and H, all of these rooms have four to five feet of space for each of three workstations, plus room on the opposite wall for communications to the player cells. Room I will continue to be used in its present capacity, as will the OPFOR cell, room J. The large screen displays for OPFOR will be set up on opposite short walls, while the ten workstations will be set up along the side walls with between five and six feet per workstation. (The indicated 20' 6" dimension appears to be a typo; 29' 6" is the correct value). The exercise director's staff will use the suite of rooms K-Q, evicting the EAC. The briefing area rooms K and L will continue to be used as such with the proposed briefing workstation in K. Spare room in L can be used for data and records storage. The main area, M, provides

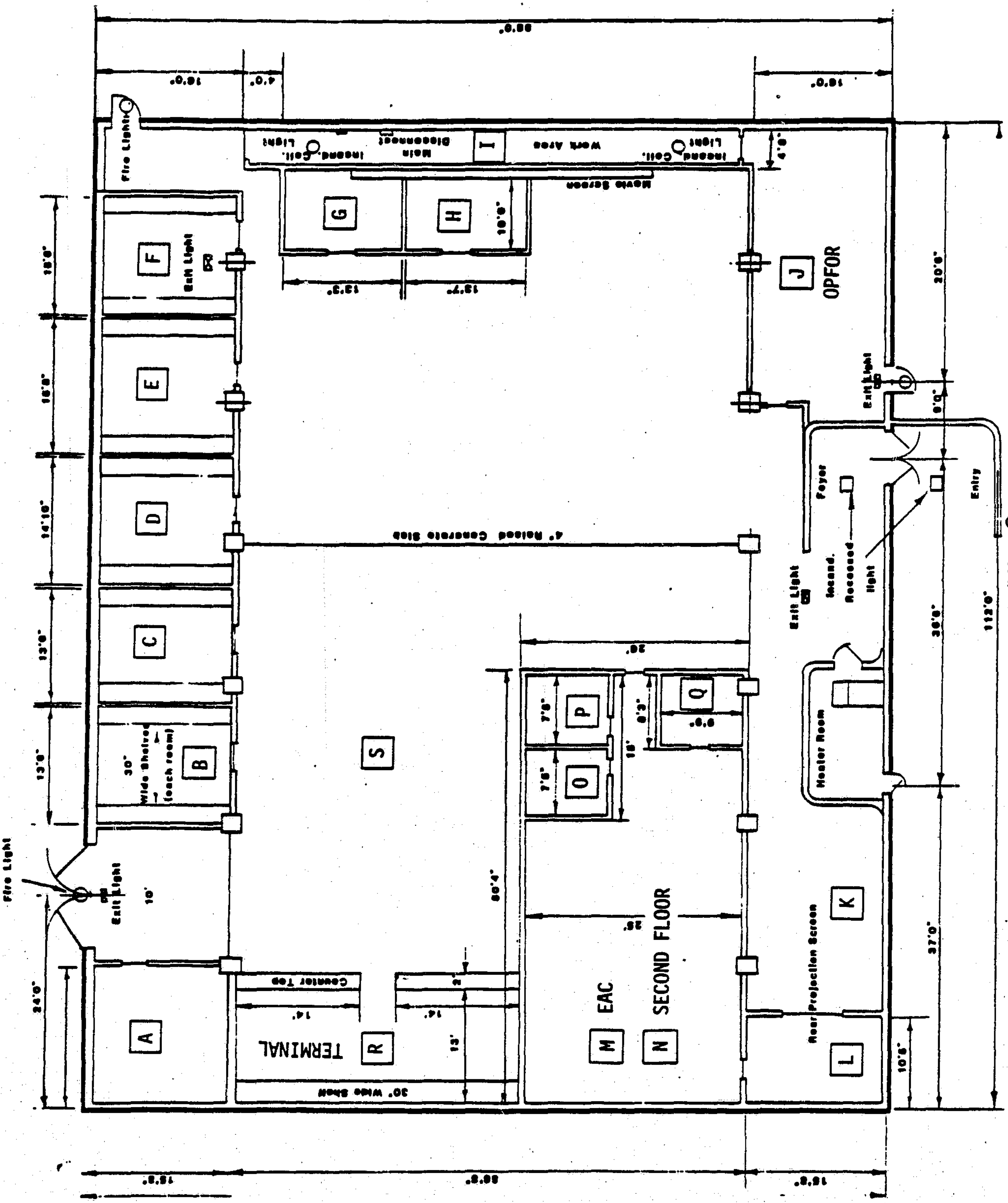


Figure 5-1. Freedom Hall mid CY 1983

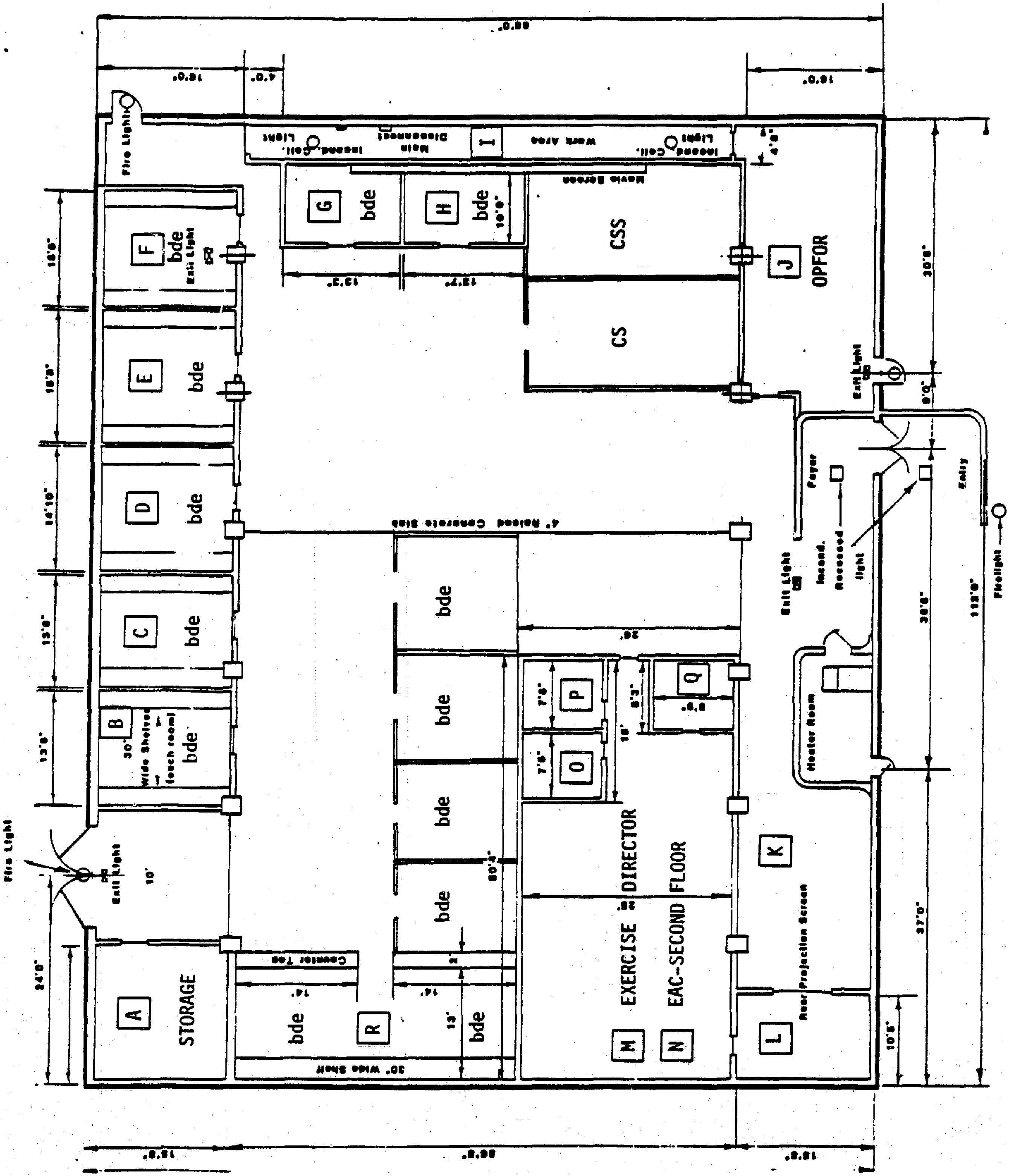


Figure 5-2. Freedom Hall modified for 13 BDE exercise

the space for the exercise director's staff workstations and for the central computer facility. Raised flooring, air conditioning and a new wide door facing the existing exterior wide door should be expected.

5.2 New Space

The entities that remain to be accommodated are six brigade satellites, the combat support and combat service support satellites, and the echelons above corps cell.

Figure 5-2 shows one way of placing these entities. By walling up the counter space at room R, space for two brigade satellites is provided. The other four brigades would occupy the four spaces created by use of flexible partitions along the wall of the exercise director's suite. Combat support and combat service support satellites would be placed adjacent to the OPFOR room, again using flexible partitions. Depending on communication requirements the combat support area can grow at the expense of the combat service support area. Echelons above corps would be moved to the second floor area over the exercise director's suite. Walls and a stairway will be needed, and some care should be given to the satellite wiring from the central computer so as not to interfere with EAC facilities.

5.3 Other Features

The partitions should be head high and should be acoustically treated. (Cloth is preferred to perforated vinyl - notices can be pinned up.) Whiteboards (not chalkboards) should be available for hanging, as should be desk space for the work station terminals. Power, computer, and communications lines would be routed overhead and dropped to workstations for maximum flexibility.

The critical unanswered questions in determining whether this arrangement will work are: 1) the amount of additional space per workstation required for manuals, procedures, and reference material, 2) the number of communicators required on the other side of the unit cell, and 3) the need to retain maps in the cells. The flexible arrangements made possible by the partition approach will materially assist in arranging equipment and personnel as needed to support the actual work load.

APPENDIX A. REFERENCES

1. Bordeaux, T.A., User Requirements for an Automated CPX Support System, RDA-TR-185500-001, October 1983

2. Parish, R.M., Modeling Command, Control, and Communications, U. S. Army TRADOC Systems Analysis Activity, May 1979.

3. Carson, E.T., Hepburn, C.D., Modifying TRASANA FOURCE for Interactive Use, RDA-TR-185500-003, October 1983.

4. Digital Equipment Corporation, VAX Systems and Options Catalog, July-September, 1983.

5. Digital Equipment Corporation, PDP-11 Systems and Options Catalog, July-September, 1983.

6. Digital Equipment Corporation, VAX Systems Site Preparation Guide, April 1982.

RDA-TR-185500-003

MODIFYING TRASANA FOURCE FOR INTERACTIVE USE

OCTOBER 1983

By:
E. T. CARSON
C. D. HEPBURN

Submitted To:
JET PROPULSION LABORATORY of the
CALIFORNIA INSTITUTE OF TECHNOLOGY
4800 Oak Grove Drive
Pasadena, CA 91103

R & D ASSOCIATES
Post Office Box 9695
Marina del Rey,
California 90291

PREFACE

This is the third in a series of reports on a two-month effort to develop a preliminary design of an automated CPX support system. This effort was performed for the Jet Propulsion Laboratory of the California Institute of Technology under Contract No. 956622.

CONTENTS

ILLUSTRATIONS

TABLES

1. INTRODUCTION

RDA is developing a preliminary design for an automated corps-level command post exercise (CPX) support system. The support system models combat processes and generates data for exercise support personnel (role players), who stimulate the command post staffs (players) who are being exercised. We have collected user requirements for such a system (Ref. 1) and have defined the computer-based functions that would be needed (Ref. 2). Some of the important computer requirements include modeling processes such as target acquisition, unit movements, and combat resolution. Existing computer programs developed to support military analysis include models of these types; based on this observation, we are examining the feasibility of an evolutionary approach as shown in Figure 1-1 for developing an automated support system.

Assuming the existence of an appropriate systemic division-level model, then the first step in developing the CPX support system would be to modify the program for interactive use, allowing the exercise role players to control the units being simulated. One potential advantage of such an approach is reduced time and effort to develop an initial capability, which can then serve as a test bed for refining the requirements for a complete system. It is clear, however, that the division-level model would have to be augmented, both to account for non-divisional assets and to treat additional processes. The resulting corps-level CPX support system might then provide the foundation for a systemic corps-level model. There are significant potential advantages in having compatible models for both systemic and exercise applications. For example, the systemic models typically use modeling assumptions about staff performance (e.g., delay times); if a compatible model is used in exercises, then data can be

EVOLUTION OF INTEGRATED CORPS-LEVEL MODELS

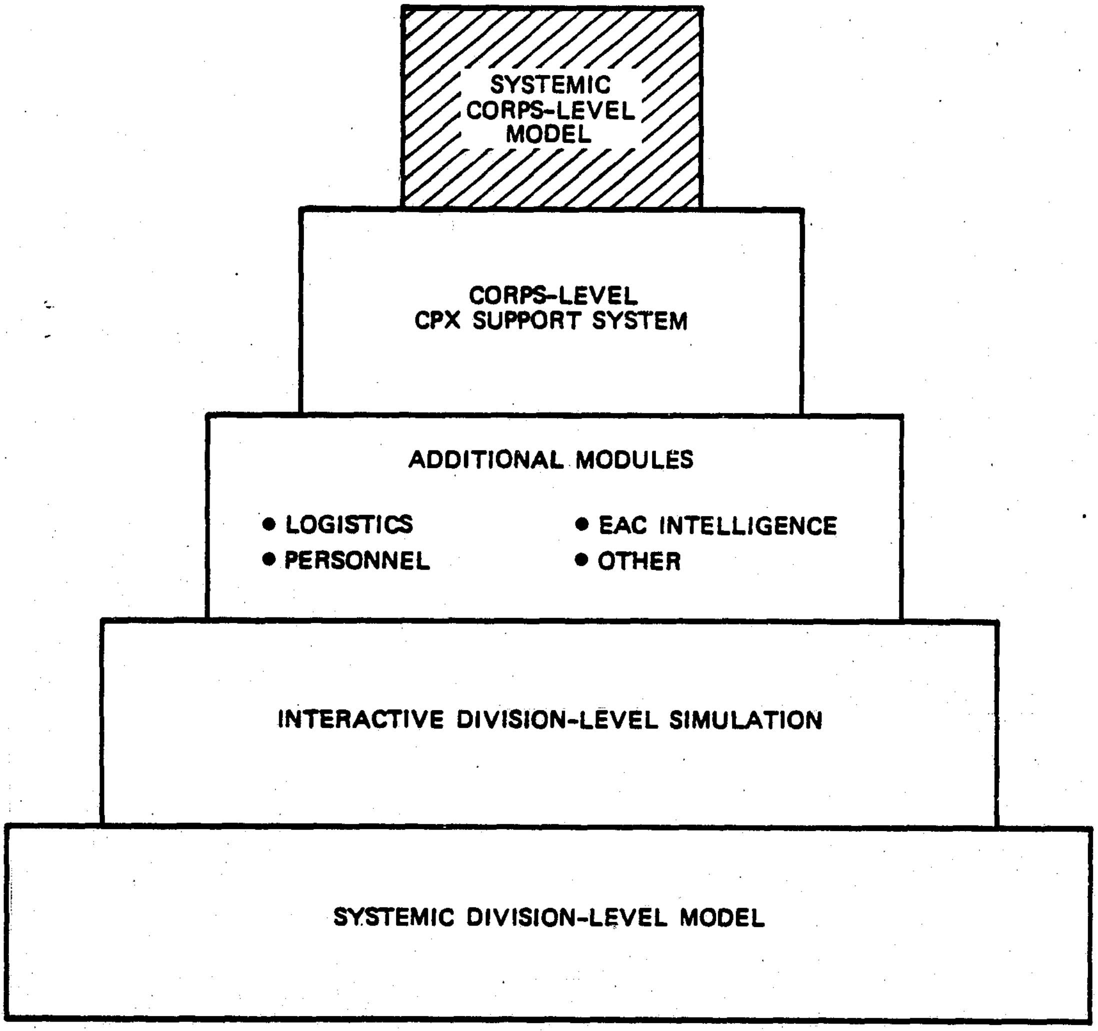

SYSTEMIC CORPS-LEVEL MODEL
CORPS-LEVEL CPX SUPPORT SYSTEM
ADDITIONAL MODULES
● LOGISTICS
● PERSONNEL
● EAC INTELLIGENCE
● OTHER
INTERACTIVE DIVISION-LEVEL SIMULATION
SYSTEMIC DIVISION-LEVEL MODEL

Figure 1-1. Potential evolution of integrated corps-level models

collected on actual staff performance and used to improve the representations in the systemic models. As another example, if the systemic models are demonstrated to be compatible with actual exercise experience, then it is possible that the systemic models could be used in a larger-scale exercise as surrogate staffs, reducing the personnel requirements for role players.

One prominent candidate for use as the systemic division-level model is the Command, Control, Communications, and Combat Effectiveness Model, commonly referred to as FOURCE, developed by the TRADOC Systems Analysis Activity (TRASANA). This model was originally developed to support a cost and operational effectiveness analysis (COEA) for an automated command and control system, and hence differs from other systemic models in having a detailed representation of command and control in addition to the models of sensors and weapons for an Army division. FOURCE (Refs. 3-6) has been used extensively by TRADOC to support subsequent COEAs, including a joint study with the Royal Armament Research and Development Establishment (RARDE). FOURCE was adopted by RARDE, who have rehosted the software from a Univac mainframe computer to a DEC VAX 11/780 minicomputer. For clarity, we refer to the RARDE version as UKFOURCE (Refs. 7-9).

FOURCE has many of the modeling capabilities that would be required in an exercise support system. FOURCE contains explicit models of the battlefield, including smoke effects; target acquisition by a variety of sensors, accounting for constraints on intervisibility; movement by Red and Blue units, treating factors such as obstacles (e.g., minefields), terrain, and trafficability; and communications networks connecting the Blue staff echelons from battalion to division. The Red and Blue forces develop perceived views of the battlefield based on target acquisition reports and status reports.

These battlefield perceptions then influence tactical decisions such as whether to withdraw or reinforce specific units, taking into account factors such as the distance to the closest enemy unit and the perceived relative combat effectiveness. The combat resolution models account for aspects such as the allocation of fire power, the number of targets within range, and the vulnerability of each target type to each weapon type.

In an exercise environment, the tactical decisions would be based on role player judgment rather than predefined rules, as in the current FOURCE. Nevertheless, FOURCE has many ingredients that make it especially attractive as a point of departure. In particular, FOURCE explicitly represents the generation of reports for staff processing, and thus may be well-suited to generating reports from the simulated battalions to the role players representing the brigade staff. Furthermore, the units in the simulation take action upon receipt of orders, so in principle it might be possible to inject orders from the role players.

In order to evaluate the suitability of FOURCE for use in an interactive environment, we contracted with Technical Solutions, Inc. (TSI) to perform an independent assessment of the effort required to modify the software. TSI selected the RARDE version (UKFOURCE) as the basis for this evaluation, partly because UKFOURCE is written entirely in Fortran and includes enhancements (such as a restart capability) that foster its application in the exercise environment. TSI performed a detailed analysis, itemizing changes to individual subroutines. Their results appear in Ref. 10. In parallel with the TSI effort, we examined both FOURCE and UKFOURCE, based on the referenced documents and on program listings and cross-reference tables supplied by TRASANA. Our objectives were to understand the model's capabilities and software structure, to identify characteristics suited to interactive use, and to find out what types of problems might be encountered in modifying the code.

UKFOURCE is a medium-sized Fortran program, as summarized in Table 1-1. It contains approximately 50 more subroutines than FOURCE (which is described in Appendix B). Many of the additional subroutines support the backup and recovery functions, although in some cases RARDE has improved FOURCE structure by breaking a very large subroutine (e.g., CMDBDE, which performs many command functions) into a number of smaller and more tractable subroutines. We have not found any significant differences between the two versions in modeling capability. The tactical decision and staff processing modules account for a large fraction of the code; many of these subroutines would still be needed even in an interactive version because they deal with battalion operations such as sending reports to brigade. Therefore, it is reasonable to hope that it might be more efficient to modify UKFOURCE than to attempt building an equivalent software system from scratch.

Both FOURCE and UKFOURCE are time-step simulations that advance combat time in terms of minor and major cycles. Figure 1-2 presents a very simplified view of the simulation control logic. The program performs a sequence of distinct calculations using separate software modules controlled by an executive. Within each major cycle, UKFOURCE begins by determining the current movement status for each unit. For example, a unit which was ordered to advance might have encountered a previously undetected enemy unit that blocks it. Using the current destination, the program computes the direction of movement for those units that have movement orders and are able to move. (Although UKFOURCE uses a

TABLE 1-1 SUMMARY OF UKFOURCE MODULES

	Module	Number of Subroutines	Total Statements
1)	Executive	12	1252
2)	Battlefield Representation	32	3919
3)	Tactical Decision	40	5361
4)	Direct Fire	20	3355
5)	Combat Support	20	3074
6)	Staff Processing	39	3317
7)	Artillery Staff Processing	47	4654
8)	Target Acquisition	35	3220
9)	Utilities	23	1920
10)	Backup and Recovery	33	2744
	Total	301	32820

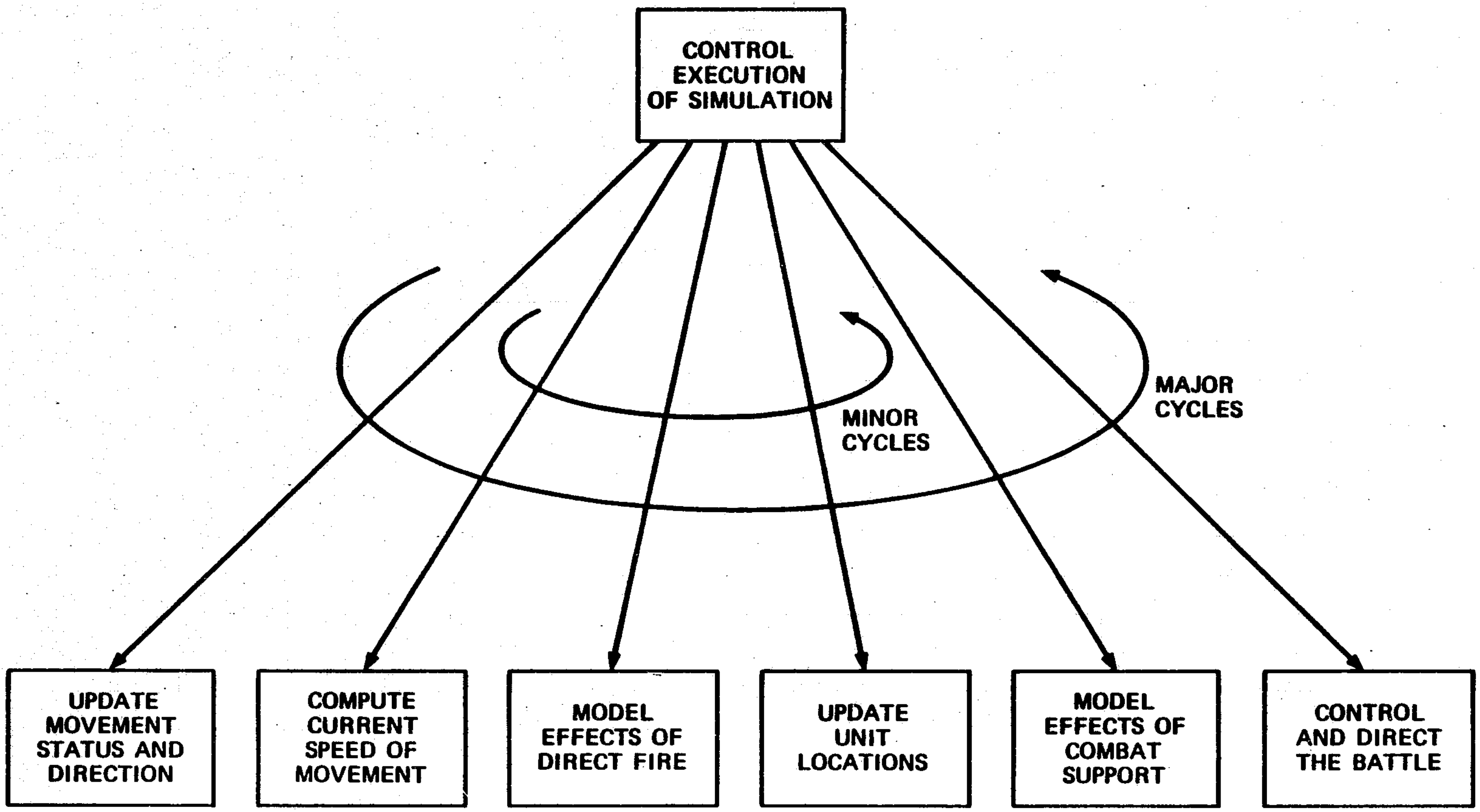

CONTROL EXECUTION OF SIMULATION
MAJOR CYCLES
MINOR CYCLES
UPDATE MOVEMENT STATUS AND DIRECTION
COMPUTE CURRENT SPEED OF MOVEMENT
MODEL EFFECTS OF DIRECT FIRE
UPDATE UNIT LOCATIONS
MODEL EFFECTS OF COMBAT SUPPORT
CONTROL AND DIRECT THE BATTLE

Figure 1-2. Simplified functional summary of UKFOURCE

rectangular grid to represent the battlefield, units can move in arbitrary directions.) UKFOURCE then performs three functions in sequence on a minor cycle basis, iterating until one major cycle has been completed: it computes the current speed of movement, taking into account a number of factors that could degrade mobility; it models the effects of direct fire; and it computes new locations for the moving units. The program then accounts for the effects of combat support such as artillery, models the processes of controlling and directing the battle, advances the combat time by one major cycle (typically set to one minute), and then repeats the sequence.

The fact that UKFOURCE performs distinct functions using separate modules encourages the hope that the program can be modified for interactive use. Furthermore, the various Fortran subroutines usually communicate through named COMMON blocks, which provides a mechanism for intercepting the data flow. Nevertheless, UKFOURCE is a complex program (because it models very complex processes), so a detailed analysis would be needed to determine what modifications would be appropriate. In the remainder of this document we present a technical approach for designing the modifications (Section 2), a sample analysis of modifying the program to accept movement orders from role players (Section 3), and a summary of our analysis of the software structure of the existing program (Section 4).

2. ADAPTING UKFOURCE TO AN INTERACTIVE ENVIRONMENT

The current version of UKFOURCE is a systemic simulation that is executed in batch mode on a DEC VAX minicomputer. In order to use UKFOURCE as part of an automated exercise support system, the software must be adapted to execute in an interactive environment where role players enter commands to govern the activities of entities such as maneuver units and artillery batteries that are being simulated. This adaptation requires changes in three functional areas:

- Controlling the simulation;
- Accepting orders from the role players;
- Generating reports for the role players.

The current simulation program runs faster than real time - that is, three to five hours of combat can be simulated in roughly one hour of computer time. While this speed is highly desirable for analysis purposes, it is clearly inappropriate for an exercise application because it would place an unrealistic load on the role players and players. In order for UKFOURCE to be used in an exercise environment, the logic that controls the pace of the simulation must be modified to enable the exercise director to set the ratio of combat time to exercise time (typically the desired ratio will be 1:1, but in some cases the director might wish to accelerate the exercise).

The systemic version of UKFOURCE is noteworthy for its explicit representation of command and control, a feature that distinguishes it from most other models of ground combat. In an exercise environment, however, most of the command and control functions must be performed by role players. Therefore the software must be modified for accepting orders generated by role players rather than using the current code that embodies tactical decision rules.

The systemic version of UKFOURCE models the generation of reports from lower to higher echelons, accounting for time delays caused by staff processing and communications. Each echelon develops a perception of the battlefield situation based on the reports received. In an exercise environment, forwarding reports and developing perceptions of the battlefield are responsibilities of the role players and players. Therefore, the software must be modified to provide the relevant reports to the role players rather than using them internally. Furthermore, the simulation within UKFOURCE of higher-echelon staff processing can be suppressed since role players will be doing the processing.

We have identified several objectives to guide the process of adapting UKFOURCE to operate in an interactive environment. The principal design goals are:

- Insulate role players from details of UKFOURCE operation;
- Maintain compatibility with systemic version;
- Minimize software changes.

The first goal is to minimize the need for the role players to understand the inner workings of UKFOURCE. In the ideal situation, the role players would not even be aware that they are interacting with a computer simulation. If the role players have to learn special commands or supply unusual data just in order to drive the simulation, then they are becoming computer operators rather than exercise role players.

The second goal, maintaining compatibility with the systemic version, is motivated by two desires. First, it may be desirable to update the exercise version at various times to incorporate improvements or corrections developed for the systemic version. (Conversely, the detailed design for the exercise version might identify improvements for the systemic version.) Second, a goal for the overall exercise support system is to permit simulation of surrogate divisions, thereby reducing the labor requirements for a corps exercise. The systemic version may lend itself to this application.

The goal of minimizing software changes is motivated by the desires to reduce the cost of developing the exercise support system and to facilitate configuration management. UKFOURCE is a medium size Fortran program; expanding it into a large program would present more problems than developing supplemental programs because of the internal couplings (e.g., existing UKFOURCE subroutines are coupled through calling sequences and through access to common data structures).

Based on these general considerations, our technical approach is to use the existing UKFOURCE, with minimum software changes, as the kernel of an automated exercise support system rather than to expand UKFOURCE itself to perform all of the extra functions that are needed. With this approach, the role players interact with input/output processors that encapsulate the UKFOURCE kernel. The input processors accept commands in standard military form from the role players and expand and translate these commands into the format needed by UKFOURCE. The output processors retrieve data in the compact format generated by UKFOURCE and produce expanded reports to the role players in standard formats.

Figure 2-1 presents a preliminary approach for using a UKFOURCE kernel within an automated exercise support system. Each rectangle represents a software module, lines connecting retangles represent calling sequences, and loops represent iterative execution. The exercise system executive controls the execution of all of the separate models or programs, but the figure depicts only those tasks relevant to UKFOURCE. The executive initiates each task in sequence, iterating until the exercise director indicates that execution should be suspended or terminated. The first task invoked by the executive collects

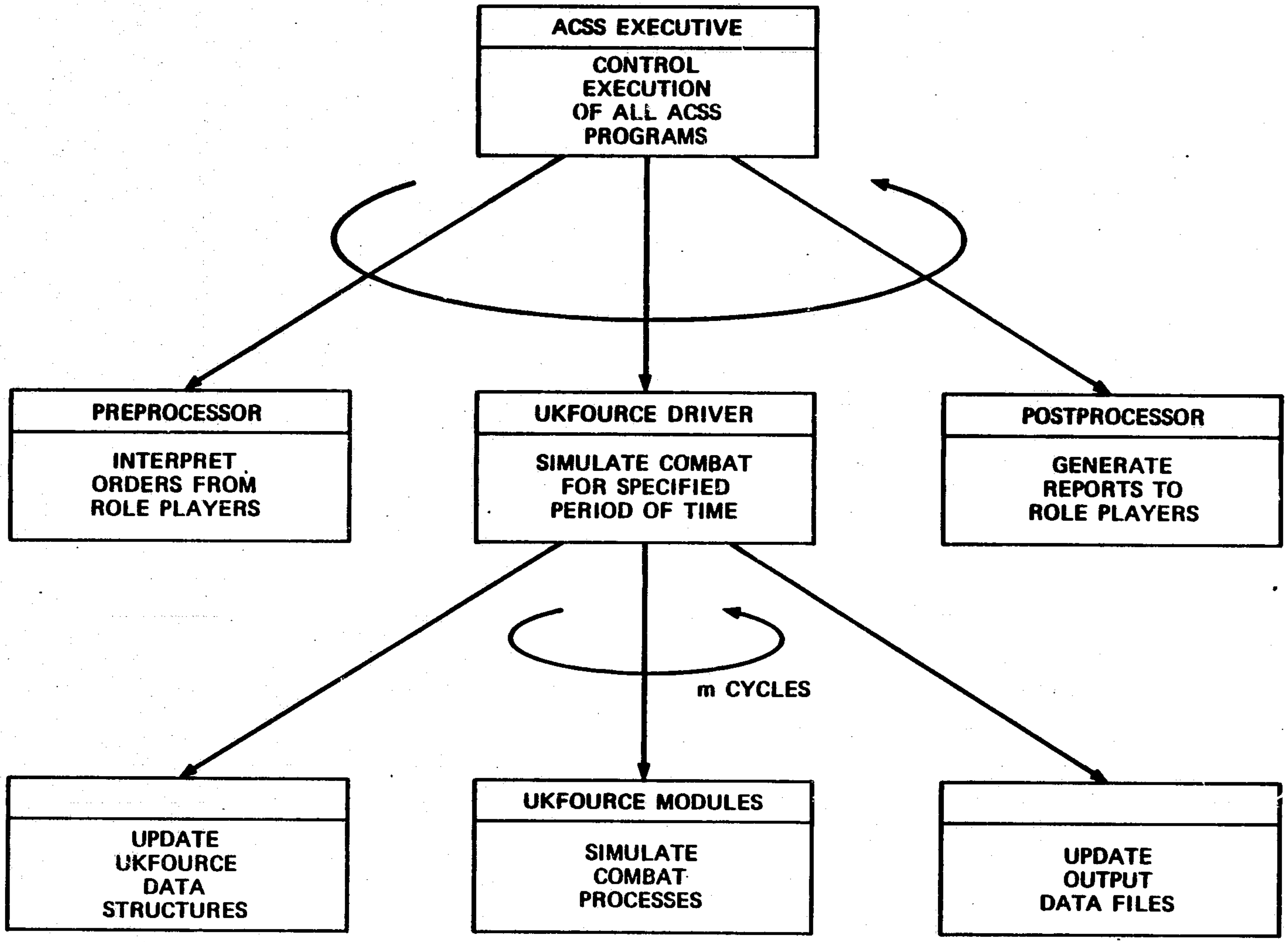

ACSS EXECUTIVE
CONTROL EXECUTION OF ALL ACSS PROGRAMS
PREPROCESSOR
INTERPRET ORDERS FROM ROLE PLAYERS
UKFOURCE DRIVER
SIMULATE COMBAT FOR SPECIFIED PERIOD OF TIME
POSTPROCESSOR
GENERATE REPORTS TO ROLE PLAYERS
m CYCLES
UPDATE UKFOURCE DATA STRUCTURES
UKFOURCE MODULES
SIMULATE COMBAT PROCESSES
UPDATE OUTPUT DATA FILES

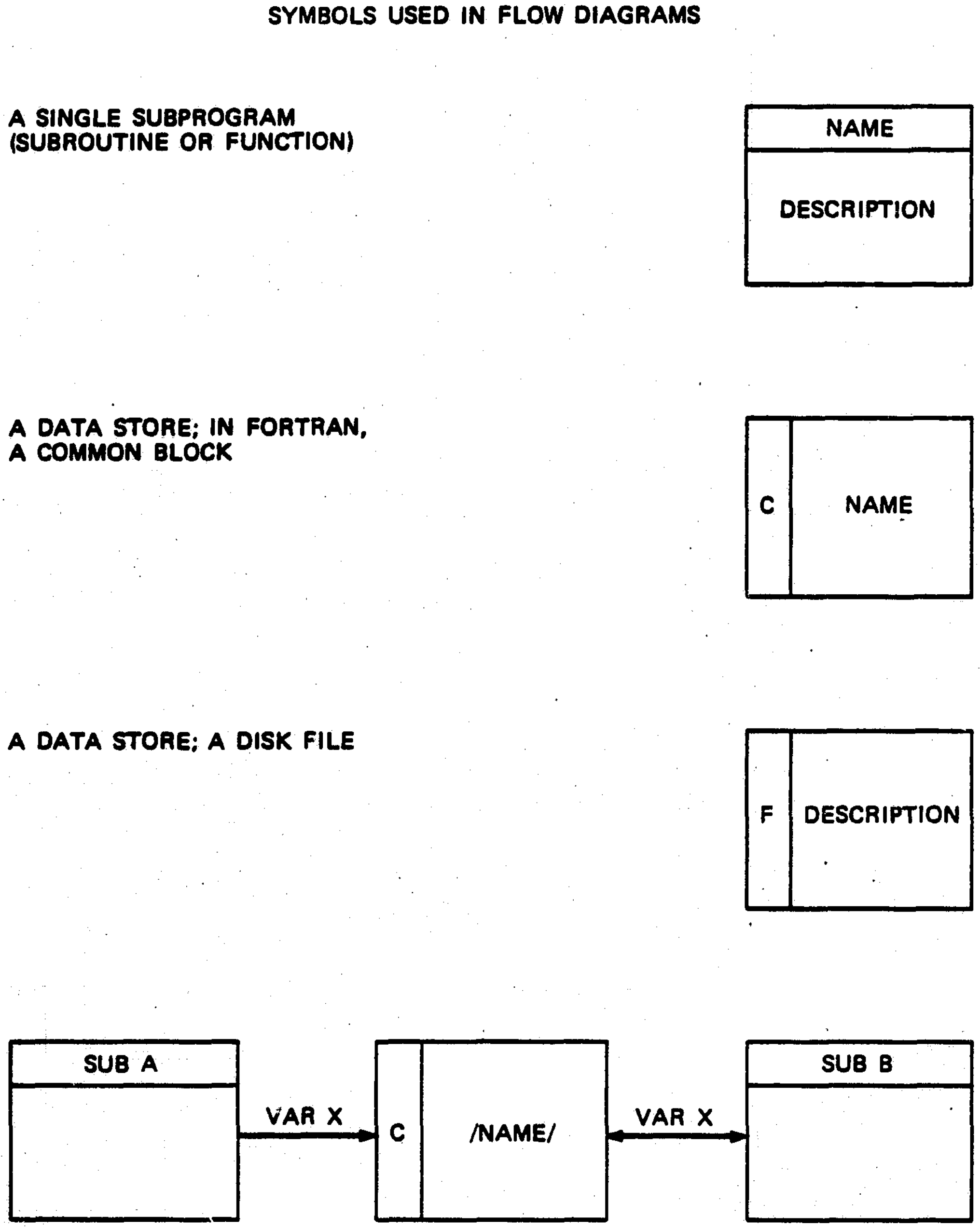
Figure 2-1. Preliminary design for adapting UKFOURCE to an interactive environment

the commands supplied by the role players and translates them into the formats needed by UKFOURCE. The second task controls the execution of the UKFOURCE simulation of ground combat. The first sub-task is to update the internal UKFOURCE data structures based on the new commands from the role players. The second sub-task is to model ground combat using the UKFOURCE modules. As indicated in Section 1, UKFOURCE is a time-step simulation that works in terms of major and minor cycles, so it would be straightforward to modify the UKFOURCE main driver so that it will execute only the number of major cycles specified by the executive. The final sub-task within the UKFOURCE task is to update data files with the results obtained during the simulated combat period. After suspending execution of UKFOURCE, the third task is then to translate the results from the simulation into reports for the relevant role players.

There are other candidate approaches for adapting UKFOURCE to an interactive environment. For example, the UKFOURCE executive could be expanded into the exercise executive, responding directly to input from role players on an interrupt basis. With this approach UKFOURCE itself would become an interactive program. One potential disadvantage of such an approach is that the frequency of interrupts is difficult to predict, particularly for a large number of role players. Processing the role player input in blocks is likely to be easier to manage and more efficient. The function of accepting the role player input, which does require immediate response, can be performed by a separate computer, for example a microcomputer at the role player's work station.

Separating the simulation process into three distinct steps (input, processing, output) as in Figure 2-1 has the advantages of simplifying clock control and facilitating simulation growth. The executive software can access the computer system clock to determine the current exercise time, compute the current combat time by using the total number of major cycles that have been processed, and determine whether it is time to initiate the next task. The simulation can be augmented by adding additional tasks. For example, a corps-level exercise system must support role players for multiple divisions. UKFOURCE treats only one division, and expanding it to directly treat multiple divisions would entail significant software changes. The organization in Figure 2-1 suggests a different approach, that of replicating UKFOURCE in multiple tasks to support the various role players. Furthermore, this design is compatible with a parallel processing approach that employs multiple computers, which may be necessary to preserve the desired ratio of exercise time to combat time.

Our approach for designing the input/output processors and for identifying the changes needed in UKFOURCE is to perform data flow analysis for all of the data needed by the role players. Figure 2-2 describes the symbols used in our data flow charts. A rectangle with a vertical line represents a data store, such as a disk file (denoted by 'F') or a Fortran COMMON block (denoted by 'C'). In high-level design charts the rectangle typically contains a general description of the data, while in more detailed design charts the rectangle contains the name of the store. A rectangle with a horizontal line represents a computer program, module, or subroutine, depending on the design level. For a low-level design the upper part of the rectangle contains the name of the subroutine. The lower part of the rectangle contains a description of the functions performed by this program unit on the data whose flow is described. Arrows indicate the direction of data flow; a subroutine connected to a data store by a double arrow reads and writes data to the store.


SYMBOLS USED IN FLOW DIAGRAMS
A SINGLE SUBPROGRAM (SUBROUTINE OR FUNCTION)
NAME
DESCRIPTION
A DATA STORE; IN FORTRAN, A COMMON BLOCK
C
NAME
A DATA STORE; A DISK FILE
F
DESCRIPTION
SUB A
VAR X
C
/NAME/
VAR X
SUB B

A) SUB A CREATES VAR X AND STORES IT IN COMMON /NAME/

B) SUB B READS VAR X FROM /NAME/ AND STORES IT WITH A NEW VALUE IN /NAME/

Figure 2-2. Symbols used in data flow diagrams

Figure 2-3 presents a high-level description of the data flow corresponding to the logic flow in Figure 2-1. The input side starts with a data file containing the orders collected from the role players. This file is read by a program that converts the orders into the variables used by UKFOURCE and stores them in Fortran COMMON blocks. These COMMON blocks are then accessed by the UKFOURCE subroutines while performing their calculations. The output side has two potential paths. In some cases UKFOURCE writes results to disk files, providing a well-defined interface to an output processor that uses the results to generate reports to the role players. The other path starts from COMMON blocks containing the simulation results and adds a subprogram to write the results to disk files.

Designing the input/output processors and the modifications for UKFOURCE entails identifying all of the data needed by the role players, identifying related variables in UKFOURCE, tracing the flow of each variable throughout the program, determining whether UKFOURCE usage of a variable is influenced by other variables, defining format transformations, and finally designing any new software using the standard methods of functional decomposition and step-wise refinement. This design task is difficult because UKFOURCE treats a large number of variables, sometimes with complex interactions. To illustrate the design process, the following section presents preliminary results for the case of movement orders for Blue maneuver units.

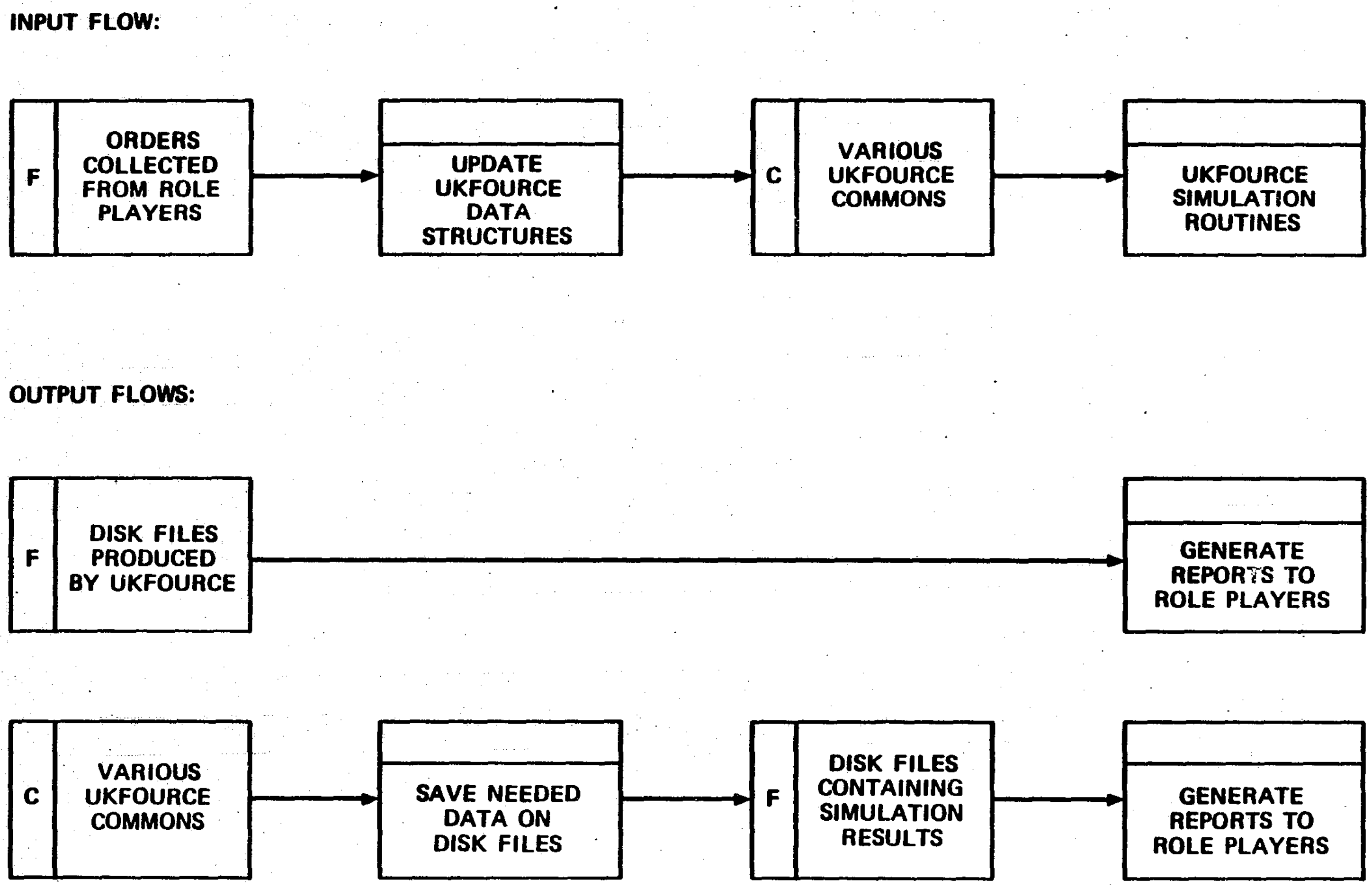

INPUT FLOW:
F
ORDERS COLLECTED FROM ROLE PLAYERS
UPDATE UKFOURCE DATA STRUCTURES
C
VARIOUS UKFOURCE COMMONS
UKFOURCE SIMULATION ROUTINES
OUTPUT FLOWS:
F
DISK FILES PRODUCED BY UKFOURCE
GENERATE REPORTS TO ROLE PLAYERS
C
VARIOUS UKFOURCE COMMONS
SAVE NEEDED DATA ON DISK FILES
F
DISK FILES CONTAINING SIMULATION RESULTS
GENERATE REPORTS TO ROLE PLAYERS

Figure 2-3. High-level data flows for modified UKFOURCE

3. TREATING MOVEMENT ORDERS FOR BLUE BATTALIONS

One of the fundamental tasks for the role players in the brigade cell is generating movement orders for the maneuver units being simulated. We have examined the UKFOURCE treatment of movement by battle groups (the U.K. equivalent of battalions) as an example of how the program might be modified to support interactive use.

UKFOURCE treats six aspects of movement. First, program logic based on decision tables determines whether units should move and selects their destinations. Second, the program determines whether units should move and selects their destinations. Third, the program determines whether a unit is able to move or is blocked, for example by obstacles or by enemy units. Fourth, the program computes the direction of movement (based on the current location and the immediate destination) and the speed of movement (taking into account factors such as adverse terrain and quality of roads). Fifth, the program updates the current unit location at the end of each minor cycle, based on the previous location, the direction and speed of movement, and the duration of a minor cycle. Sixth, the program automatically generates reports related to movement; for example, a unit under heavy attack may request permission to withdraw.

UKFOURCE software treats movement using different variables for modeling various classes of units. Therefore, a comprehensive design must address a number of different variables. The following discussion treats only those variables that would be relevant for controlling the movement of Blue battalions.

In the existing UKFOURCE, the user choreographs the potential movement of Blue battalions before running the program by selecting and entering a series of successive positions for each unit. For example, these positions might correspond to a set of prepared locations for a delaying defense; logic in the command/control module would then decide, based on the actual battle dynamics, when it would be appropriate for the unit to withdraw to the next defensive location. This software mechanism is completely general, however, and provides great flexibility for the user to guide the course of movement. In addition to this table of successive positions, the program carries for each battalion another variable defining its current destination. Under appropriate circumstances, the command/control logic may set the current destination to the next value in the table of successive positions.

UKFOURCE uses data files and named Fortran COMMON blocks, which are accessed by subroutines as needed. New variables generated in one part of the program (e.g., the command/control module) are placed in COMMON areas for use by other parts (e.g., the module that computes velocity). We have tracked the data flow for Blue movement logic by finding:

- Disk files and COMMON blocks containing relevant variables;
- All subroutines that access these COMMONs;
- Variables whose values affect the usage of the relevant variables.

Tracking the data flow in this manner is difficult because UKFOURCE uses a large number of data structures (more than 150 named COMMONs), the code that uses the variables is often unstructured, and sometimes variables are used for more than one function.

In the case of Blue battalions, the table of successive positions is stored in a multi-dimensioned array SPO and the list of current destinations is stored in an array DST.

Figure 3-1 describes the data flow as we have traced it. As far as UKFOURCE is concerned, the table of successive positions is obtained from a disk file that is created by a preprocessor run by the user. A subroutine in UKFOURCE copies this table into the COMMON area called /BLUBDE/ for use by other subroutines. Subroutine CMDBLU is a high-level driver that invokes routines that model the command/control process for Blue. The CMDBLU box includes the subset of these subroutines that deal with the movement variables. Some of these subroutines (NOAMMU, BTHREE, BFOUR, TFB, TFC, TFD, and DIV2) construct from SPO the new variable DST, which functions as the immediate destination. DST is stored in the COMMON area called /DESTS/, and from there it can be referenced by any of the subroutines shown. DST is used both as the current destination and as a control flag; under some circumstances DST is set to zero to indicate that the unit is not moving. Unfortunately, this usage interlocks with other control flags that indicate whether the unit is mounted or dismounted, whether the unit is supposed to be moving, and the unit's current mission (e.g., defend or delay). Within the command/control subroutines, all routines except DEFEND both use and update DST; DEFEND merely uses it. For example, DST is updated when motion is halted under circumstances such as:

- Current destination is reached;
- Progress is halted by an obstacle such as a river;
- Retreat is necessary;
- Retreating unit is overtaken by an enemy unit and must stand and fight;
- Maneuver unit is destroyed.

In addition to the command/control routines, three other subroutines employ but do not change DST. VECTOR uses the current location and destination to compute the direction of

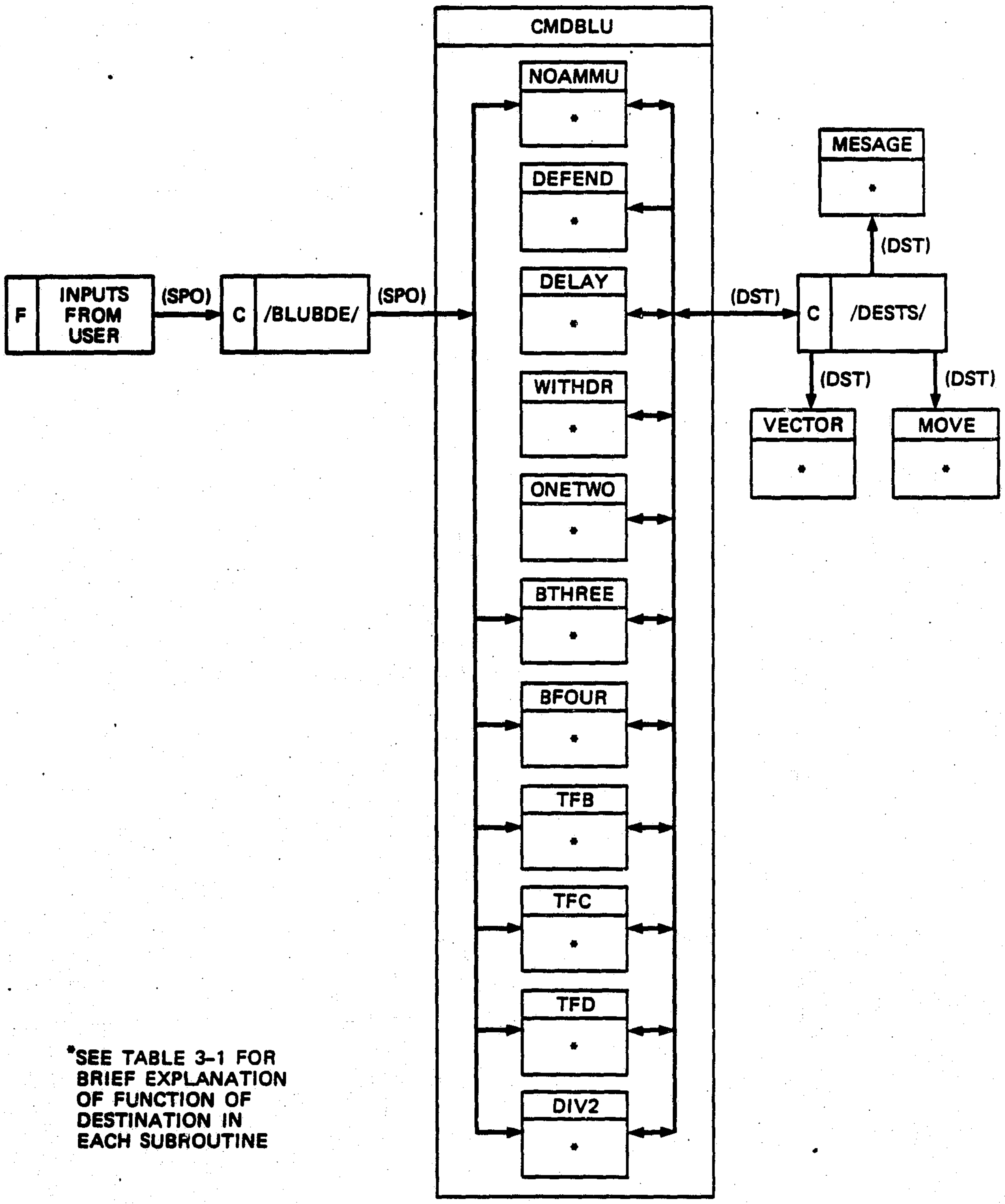


Figure 3-1. UKFOURCE data flow for destinations of Blue maneuver units

movement. MOVE computes the current speed of the moving units (skipping over units whose destination is set to zero) and updates their location based on one minor cycle of movement in the direction computed by VECTOR. MESSAGE is concerned with the dissemination of status reports when units arrive at their destinations. Table 3-1 describes the function of each subroutine with regard to its treatment of movement variables. For reference, Table 3-2 provides a more comprehensive description of the subroutine functions.

This treatment of movement variables for Blue battalions is very compatible with the general approach described in Section 2 for adapting UKFOURCE to an interactive environment. In the current batch-processing environment, the UKFOURCE user defines a single set of successive positions in advance of the run. In the proposed interactive environment, the role player supplies a new set of successive positions whenever he judges that the tactical situation calls for unit movement. As discussed in Ref. 2, the role player might use a cursor in a videographics display to designate the intermediate points in the planned route. System software would then determine the UTM coordinates of these points and translate them into the coordinate system used within UKFOURCE. The input preprocessor discussed in Section 2 would then write these data into the disk file used by UKFOURCE, which would have to be modified to read the file and load /BLUBDE/ with SPO each time the program is invoked by the system executive. The UKFOURCE subroutines that monitor the actual movement of the units, detect obstacles, halt the movement, and send reports to the brigade would all be extremely useful in the exercise context because they would free the role players of many time-consuming calculations that are performed manually in current exercises.

Figure 3-2 depicts the preliminary design of a data flow for Blue movement orders in an interactive environment. In

TABLE 3-1 SUBROUTINE FUNCTIONS RELATED TO MOVEMENT VARIABLES FOR BLUE BATTALIONS

Subroutine	Function of Destination in Subroutine
NOAMMU	If unit is out of ammo, destination is set equal to next sequential position.
DEFEND	Check if destination has been reached or the unit stopped for another reason.
DELAY	Checks for shared battalion destinations.
WITHDR	Checks for shared battalion destinations.
ONETWO	If destination is zero, retreat is directed to and over a river, for battle groups one and two.
BTHREE	If destination is zero, generates ambushes and directs retreat across river; sets destination to next sequential position; for battle group 3.
BFOUR	As in BTHREE, but for battle group 4.
TFB	For task force B, increments destination to next in sequence. If in front of closed bridge, sets destination to zero.
TFC	As in TFB, but for task force C.
TFD	As in TFB, but for task force D.
DIV2	Moves battle group 12 to committment position; updates destination using sequential positions.
VECTOR	Calculates change in position, from destination and current position.
MOVE	If destination has not been reached, the position is updated toward destination.
MESAGE	User destination is a control flag to determine which messages to send.

TABLE 3-2. GENERAL SUBROUTINE FUNCTIONS

SUBROUTINE	GENERAL SUBROUTINE FUNCTION
NOAMMU	Invoked when a battalion has become inactive, it checks whether the inactivity is due to the battalion having been destroyed or being merely out of ammunition. If the latter, the battalion's destination is set to the latest scheduled position, until the last position is reached.
DELAY	Delays Blue battalions, based on several criteria.
WITHDR	Withdraws Blue battalion, based on several criteria.
ONETWO	Controls Blue battalions one and two, directing retreat to and over a river.
BTHREE	Controls Blue battalion #3, generating two 3-minute ambushes and then directing retreat across a river.
BFOUR	Controls Blue battalion #4, generating three 3-minute ambushes and then directing retreat across a river.
TFB	Controls and directs task force B: It assigns POF, allocates fight ground attack sorties, withdraws the task force to corps reserve area, and coordinates movement to and through obstacles.
TFC	Like TFB, for task force C.
TFD	Like TFB, for task force D.
DIV2	Controls the second Blue division, allocating aircraft, helicopters and fighter ground attack sorties to the battalions. Commits the reserve battalion when the main

TABLE 3-2 CONCLUDED

	battle has been joined and the main thrust zone has been ascertained.
MOVE	Updates the positions of the Red and Blue maneuver units with the related sensors, command posts and forward observers.
MESAGE	Updates messages from all echelons of Blue and Red forces.
DEFEND	Resets the status and/or mission of the battalion using the relative combat effectiveness, the relative combat power and the range of the enemy as determining criteria.
VECTOR	Calculates the direction of travel and the movement statuses of all maneuver battalions and non-battalion command posts of both Red and Blue, and the artillery units of Blue (the Red artillery units are constrained to move in the x-direction only).

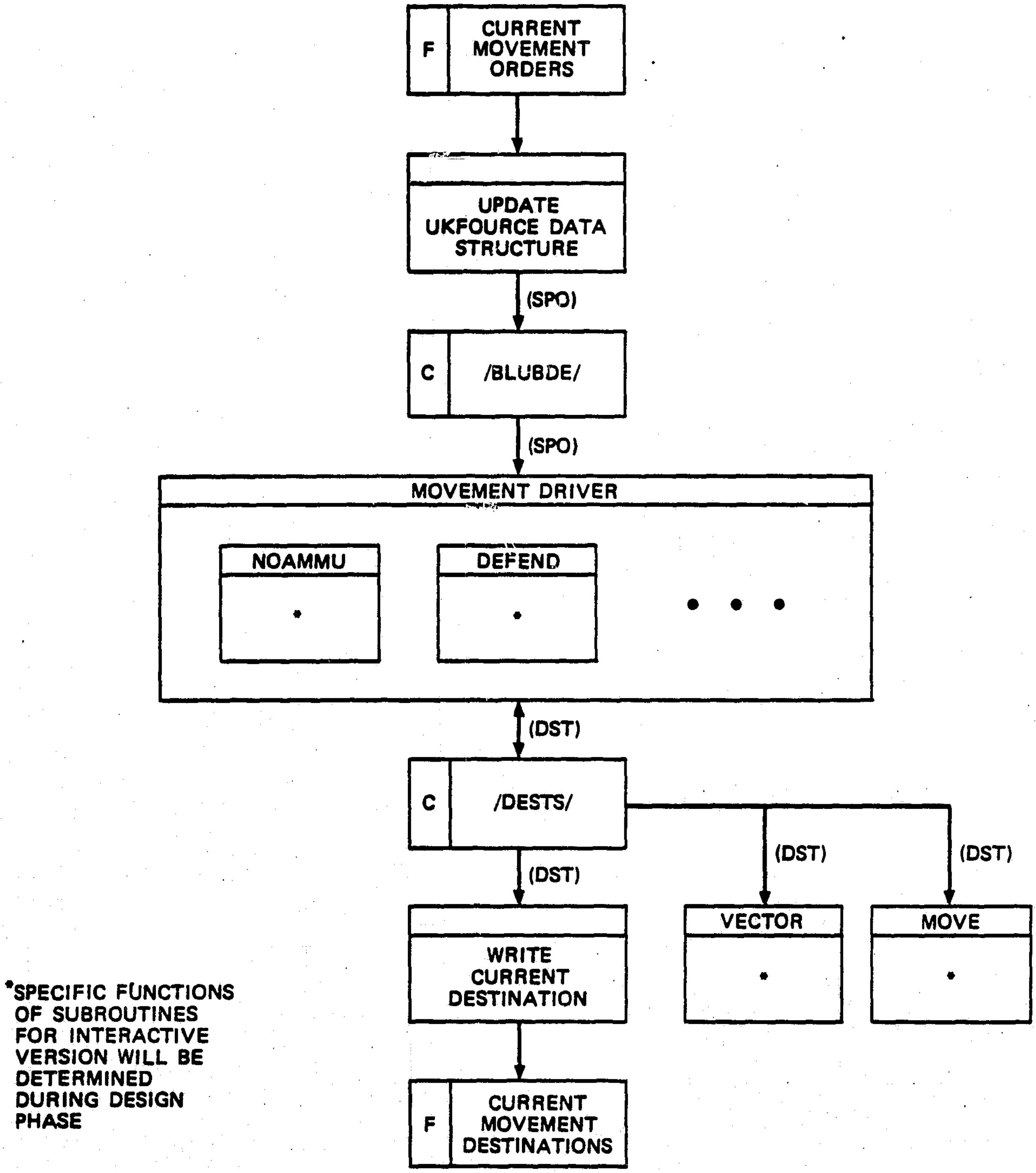

F
CURRENT MOVEMENT ORDERS
UPDATE UKFOURCE DATA STRUCTURE
(SPO)
C
/BLUBDE/
(SPO)
MOVEMENT DRIVER
NOAMMU
*
DEFEND
*
● ● ●
(DST)
C
/DESTS/
(DST)
(DST)
(DST)
WRITE CURRENT DESTINATION
VECTOR
*
MOVE
*
F
CURRENT MOVEMENT DESTINATIONS
*SPECIFIC FUNCTIONS OF SUBROUTINES FOR INTERACTIVE VERSION WILL BE DETERMINED DURING DESIGN PHASE

Figure 3-2. Preliminary data flow design for UKFOURCE kernel

many ways it is similar to the current data flow, consistent with our objective of minimizing software changes. A new subroutine (not labeled in the diagram) will be needed to refresh SPO each time UKFOURCE is executed, based on the movement orders that have been previously collected from the role players. The CMDBLU subroutine, which currently makes some of the tactical decisions that should be made by role players, would probably be replaced by a new driver to control the lower-level subroutines. Some of the lower-level routines might be deleted or modified, but many would be retained in their current form. As an extra feature, the flow diagram shows the current destination being written to a disk file, making it available for use by the exercise controllers in monitoring system behavior.

Even though the software changes implied by this preliminary design might be fairly minor, a significant amount of analysis and detailed design effort would be required. Each of the subroutines in Figure 3-1 would have to be evaluated more thoroughly, taking into account both the movement variables and the other related variables (such as unit mission) that would be defined by role players. Furthermore, this discussion has treated only the movement variables for Blue battalions; similar analysis would be required for Blue artillery units, sensors, and command posts as well as the various classes of OPFOR units.

4. DESCRIPTION OF UKFOURCE SOFTWARE

Various discussions of the UKFOURCE software exist already. (Refs. 3-9); each of the references contains some material on software structure. It has been necessary in our analyses, however, to rely primarily on the original Fortran code, supplemented by the descriptions in the TSI report (Ref. 10). Our descriptions of program modules are drawn from these sources and are abbreviated for ease in assimilation. Our presentation of logic flow comes from analyzing the original code, with occasional clarifications from each of the other references; it is intended to streamline for others the process of becoming familiar with UKFOURCE software.

UKFOURCE is written with a main program and calls to subroutines, which successively call further subroutines, to several tiers. At the top level, therefore, the program is in block or structured form. The presence of 301 subroutines renders desirable a classification system to guide programmers in pinpointing specific computations. This classification system could be based on either direct logic flow (subroutine calling trees) or on similarity in subroutine function. The UKFOURCE programmers chose the latter approach, and set up ten functional groups of subroutines called modules.

As the members of a single module perform related functions, the utility of the module classification system is enhanced by maximizing the degree of independence between modules. We have evaluated the coupling between modules in terms of subroutine calling relationships. Except for utility subroutines that can be called by any subroutine, and for one anomalistic subroutine called STAFF, the degree of independence is greater than 90 percent (more than 90 percent of the subroutines within a module call only subroutines within that same module). The ten modules with their constituent subroutines are listed in Table 4-1. Some of the subroutines have integers on the left and/or on the right, in the form:

n > SUBROUTINE NAME > m

In this format, "n" indicates the number of calls to SUBROUTINE NAME from outside the module, and "m" indicates the number of outside subroutines it calls. Brief definitions of the generic functions of the modules are given in Table 4-2; descriptions of the individual subroutines can be found (in a different order) in Appendix B of the TSI report (Ref. 10).

In the original FOURCE, the main driver contained a section of code (invoking subroutines) that was repeated. RARDE abstracted this section of code and packaged it as a separate subroutine, STAFF. STAFF is included in the executive module, but calls 29 subroutines from other modules, thereby functioning more like a main driver than like the other subroutines. Hence its contribution to modular coupling is anomalistic, and we conclude that UKFOURCE modules are substantially independent, which will facilitate modifying the program.

While UKFOURCE is in block form at the main program level, within each subroutine the logic flow generally bears little resemblance to contemporary ideals of structured programming. While this lack of structure is not of particular importance for the smaller, simpler routines, it is of paramount concern in interpreting large and complex ones like VECTOR and SPEED in which logic flow jumps back and forth, with at times up to ten streams of logic crossing a single line of code. This is easy for computers to manipulate, once written, but is very cumbersome for humans to write, to follow or to modify. Such a routine can, however, have these criss-crossing logic streams untangled in a straightforward way; the identical logic can be rewritten in completely structured form (block form at all levels), in about the same number of lines.

TABLE 4-1. UKFOURCE PROGRAM MODULES

ARTY STAFF PROCESSING MODULE	BACKUP AND RECOVERY MODULE	BATTLEFIELD REPRESENTATION MODULE	COMBAT SUPPORT MODULE	DIRECT FIRE MODULE
AOPSPR	1 > BACKUPERR	ASMBLE	ARTSUP	1 > ATCOEF
ASGAOPBTY	BRAINC	CELLS	BAREFX > 2	BLLOSS
ASGBTY	BRBINC	CLZIRK	BARRAR	CLENUP > 1
ASGFDC	BRCINC	1 > CPMOVE	BARRCP	DIFIRE > 2
BSROUT	BRDINC	1 > CPSPED	BARRMU	DSTROY
BTYAMO	BREINC	DEGFC1	BARRSN	ERASE > 1
BTYASN	BRFINC	DEGFC2	BLUARF	FRCTIR
BTYAVL > 1	BRGINC	DEGFC3	BLUCAS	GAPLOS
BTYLST	BRHINC	DEGFC4	BLUHEL	GBDACQ
DELETE	BRIINC	DEGFC5	1 > COMSUP	GEODET > 1
DEPLOY1	BRJINC	DIRECT	CPDMGE	GRDACQ
DEPLOY2	BRKINC	1 > GRNLTR	FASCAM > 1	HBKLRT
DEPLOY3	BRLINC	1 > INITIAL	INSECT	HRKLRT
FDCPR	BRMINC	MINEMO	MINFLD	INSERT > 1
FINDINDEX > 1	BROINC	1 > MOVE	RARBAR > 1	LOSEFF > 1
FMRGEN > 1	BRPINC	MOVERPVS > 1	RARBCP > 1	RDLOSS
FMRLST	BRRINC	1 > NEARX	RARBMU	SMKOUT > 1
FMROUT	BRSINC	RDRVRC	RARBSN	SMKPRG
FMRBPQ >1	BRTINC	REMOVE > 1	RAREFX > 1	SMOKFR
FMRRT	BRUINC	1 > ROLSRT	REDARF	SPSHUL
FMRTAM	BRWINC	SCONST		
MLRBCP > 1	1 > DUMPINIT > 1	SMKCOV		
MLRSBC	2 > DUMP_4C > 2	1 > SMKUPD		
MSGRCV	FLUSHOUT > 1	SPEED		
MSGSND > 2	GETDUMP > 1	UNFRC2		
1 > MSGSTO > 1	1 > LOG_4C > 2	UNFRC3		
OUTBSR	VARCOPY	UNFRC4		
SENRD	MOVEBYTES	UNFRC5		
SEQCLR	1 > RECOV_4C	UNFRC6		
SEQCPF	1 > RLOG_4C	UNFRC7		
SEQDT	XBREAD > 1	1 > VECTOR > 1		
SEQFNP > 1	XBWRIT > 1	VECTORRPVS		
SEQFQP > 1	XFERDA			
SEQPPR > 1				
SEQSPN				
UKAINT				
UKAOPS				
1 > UKARTY				
UKBC > 1				
UKBCP				
UKCBR				
UKFDC > 1				
UKFOO				
UKRPV				
UKSOP > 1				

TABLE 4-1. UKFOURCE PROGRAM MODULES (CONCLUDED)

EXECUTIVE MODULE	GENERAL ROUTINES MODULE	STAFF PROCESSING MODULE	TACTICAL DECISION MODULE	TARGET ACQUISITION MODULE
---------	--------	------	--------	-----------
CFCTRL	ARTPRT	1 > ADDREC	1 > BDEBNS	AQTPAM
CLOSEFS	6 > CLDPTH	AOI	BFOUR	1 > BLUACQ > 2
INIT_MI	COMPRT	BLOP	BSTR12	DIRECTCBRSRG
INITS > 1	45 > DERROR	BLUINT	BTHREE	PRTACQ
OPCON	EPOSIT	COMMO	1 > CMDBLU	1 > REDACQ > 2
OPENFILES	12 > ESLINT	DBD	CMDRDV	REDCOMMITMENT
PAMREAD	FIND	DBR	CMDREG	RETREIVEMSG > 1
PLCHECK > 2	GETOPTION	DBUP	1 > CMDRED	RPVINTEL
PLYRINIT	1 > IOCTRL	DBW	CRMAX	RPVREPORT > 1
ROPENFILES	MINPRT	DISTR	DEFEND	RPVSEARCH
STAFF > 29	PLTCON	EWIN	DELAY	SCANDATA
UKFOURCE > ALL	PLTP	GEN	DIRECTCPS	STOREMSG
	PRNTER	GEN2 > 1	DIRECTRPVS	TAANT1
	4 > QIOERR	GETAD > 2	DIV2	TAANT2
	SENPRT	INTELREPORT	1 > DSTSTR	TABCBR > 1
	STATS	INTFAC	FINDFLOTS	TABFOA > 3
	SUPWRT	JGREEN	FIRRED	TABFOB > 2
	TRAP	LNKAGE > 1	FIRRST	TABRPA
	WAIT	LOGG	JLDATA	TABRPV
	WPNPRT	MERGE	1 > MESAGE	TABSOP > 3
	ZAP1	ONOFF	MVRNF	TABSRG > 1
	ZAP8	OPSFAC > 1	NOAMMU	TAINT1
	19 > ZZAP	1 > PAM > 1	ONETWO	TAINT2
		PERINT	PRPTGT	TARCBR > 1
		PRODLY	PUTRS > 1	TARGSA > 2
		REDI	RCERCP	TARGSR > 1
		REDOB	REDBN	TARRCN > 2
		REDOP	REDEAD	TARRNA > 3
		REDUN	REDSM	TGTDT2 > 1
		RELAD	RESERV	TGTDT3
		1 > SCHEDT	ROUTERPV5	TPLSG1 > 1
		3 > SNRPT	RPVNEXTPOS	TRDET1 > 1
		1 > SNRR	RPVTASK1	TRDET2 > 1
		SPOT	RPVTASK2	TRDET5
		TRUINT	RPVTASKS456 > 1	VTRACE
		TSCHED	RPVZIGZAG	
		TYPSZE	TFB	
		ULOC	TFC	
		UNTECH	TFD	
			WITHDR	

TABLE 4-2. GENERIC FUNCTIONS OF UKFOURCE MODULES

MODULE	GENERIC FUNCTION
(1) ARTILLERY STAFF PROCESSING	Controls artillery staff processing functions. It receives requests for artillery and determines the allocations.
(2) BACKUP AND RECOVERY	Handles restart capability of UKFOURCE.
(3) BATTLEFIELD REPRESENTATION	Calculates layout of, and motion on, the battlefield.
(4) COMBAT SUPPORT	Controls support of direct combat (artillery, minefields, indirect fire, helicopters, etc.)
(5) DIRECT FIRE	Controls process and consequences of close combat on the battlefield.
(6) EXECUTIVE	Contains the main program and other executive-level routines; controls the running of UKFOURCE with its attendent files.
(7) STAFF PROCESSING	Controls processing of non-artillery staff functions like messages and scheduling.
(8) TACTICAL DECISION	Performs command/control functions for both Red and Blue forces.
(9) TARGET ACQUISITION	Controls the sensors and the accumulation of intelligence information they provide.
(10) UTILITIES (GENERAL ROUTINES)	Contains assorted routines to be used by many other subroutines.

Once the program is structured, it can be interpreted (and updated) much more readily by a programmer, and a parallel English description (called Program Design Language, or PDL) can be set up as an aid to the understanding. This parallel description substitutes an English descriptive phrase for one or several lines of code, within the actual logic structure. In this form a non-programmer can follow the program's logic flow. The PDL serves for structured programs as the older flow charts did for unstructured programs. We have developed descriptions of the UKFOURCE main program and some of the attendant subroutines in this PDL form. To accomplish this, the logic in each has been untangled (it has been rendered in structured form), with the actual sequence of operations remaining unchanged. Occasionally a "PROCEDURE" will be referenced; it functions as a subroutine call, and is a section of code that was multiply-traversed in the unstructured version. In the structured version it merely is called from several different locations in the code, to duplicate the original sequence.

The general format of the PDL is summarized in Table 4-3. The logical structures are IF-choices and loops (DO and WHILE). The simplest, the IF-choice, questions whether a condition (within parentheses following the IF) is true or false. If true, the indented statements are processed; if false, they are not processed.

Table 4-4 lists the subroutines for which PDL descriptions are provided in the rest of this section. These descriptions should assist an analyst in locating quickly specific functions in the code.

In preparing to adapt UKFOURCE to an interactive environment, we believe that it would be important (for speed and clarity) to structure the code in top-down order, in those areas where modications will need to be made, and that PDL

TABLE 4-3. PDL CONVENTIONS

- Each program begins with "PROGRAM NAME".
- Each subroutine begins with "SUBROUTINE NAME".
- Each procedure begins with "PROCEDURE NAME".
- An English phrase or sentence on one or more lines describes the function of a block of code.
- A comment is indicated via:

```
#
#  Comment line
#
```

- Logic structure is shown via the following forms (each successive logical level is indicated by an indentation):

```
IF (condition is true)
.  Perform this function
ENDIF

IF (condition is true)
.  Perform function 1
ELSE
.  Perform function 2
ENDIF

IF (condition 1 is true)
.  Perform function 1
ELSE IF (condition 2 is true)
.  Perform function 2
ENDIF

DO (the following for some given number of iterations)
.  Perform this function
ENDO
```

TABLE 4-3 CONCLUDED

```
WHILE (a given condition is true)
.  Perform this function
ENDWHILE
```

Notice that each such process is completely in block form, and begins with a key word (IF, WHILE, or DO) and finishes with END suffixed by the beginning key word (ENDIF, ENDWHILE, or ENDO). With just these it is possible to duplicate all the FORTRAN logic flows.

- PROCEDUREs, as defined above, are listed in subroutine format immediately following the subroutine that called them, separated by a line of the characters "******** . . .".
- Each program ends with ENDPROGRAM.
- Each subroutine ends with RETURN followed by END on the next line.
- Each prodecure ends with RETURN followed by END on the next line.

TABLE 4-4 UKFOURCE SUBROUTINE PDL LISTINGS

DIRECTORY

Example subroutine calling trees

```
UKFOURCE
    PLYRINIT
    DUMPINIT
    OPENFILES
    PAMREAD
        DERROR
    INITS
        ZZAP
            DERROR (as above)
            ZAP8
            ZAP1
        INIT_MI
    BDEBNS
    ROLSRT
    GRNLTR
        CELLS
        UNFRCn
```

Further example subroutines

```
INITIAL
IOCTRL
ROPENFILES
RECOV_4C
BACKUPERR
RLOG_4C
DUMP_4C
PLCHEK
COMSUP
CFCTRL
STAFF
SMKUPD
DSTSTR
CLOSEFS
```

descriptions be prepared for the involved subroutines. Thereafter, the detailed modifications to UKFOURCE can be made with much greater assurance.

EXAMPLE SUBROUTINE CALLING TREES

```
     PROGRAM UKFOURCE
#
#  Main program of UKFOURCE: Performs initialization, calls
#  subroutines to carry our basis computations, and incre-
#  ments the simulation time.
#
     Initialize the data structures:  CALL PLYRINIT
     Set up dumping interval: CALL DUMPINIT
     IF (starting run)
     .  Open files: CALL OPENFILES
     .  Read staff function data: CALL PAMREAD
     .  Read user options; initialize database: CALL INITS
     .  Load Blue BG's into proper BDE's: CALL BDEBNS
     .  Sort the Red BG list: CALL ROLSRT
     .  Initialize cells for Blue and Red: CALL GRNLTR
     .  Establish preliminary artillery: CALL PRPTGT
     .  Initialize smoke cloud constants: CALL INITIAL
     .  Set time to zero
     .  Initialize periodic summaries (print and plot):
     .  CALL IOCTRL
     ELSE (restarting run)
     .  Open files for restart: CALL ROPENFILES
     .  IF (restarting from non-logging dump)
     .  .  Recover from dump: CALL RECOV_4C
     .  .  Report possible errors: CALL BACKUPERR
     .  ELSE (restarting from logging dump)
     .  .  Recover from logging dump: RLOG_4C
     .  .  Report any error: CALL BACKUPERR
     .  ENDIF
     ENDIF
#
#  Artillery preparation:
#
     IF (time < time for end of Red artillery preparation
     minus 25)
     .  IF (dump required at this time)
     .  .  Carry out a dump: CALL DUMP_4C
     .  ENDIF
     .  Check if FOURCE controller has requested action:
     .  CALL PLCHEK
     .  IF (error)
     .  .  Pause
     .  ENDIF
     .  Model combat support: CALL COMSUP
     .  Control covering force: CALL CFCTRL
     .  Perform staff functions: CALL STAFF
     .  Increment times
     .  Perform periodic summaries: CALL IOCTRL
     ENDIF (end of artillery preparation)
```

```
#
#   Begin major cycle:
#       WHILE (time remains in major cycle)
        .  IF (dump required at this time)
        .  .  Carry out a dump; CALL DUMP 4C
        .  ENDIF
        .  Check if FOURCE controller has requested action:
        .  CALL PLCHEK
        .  IF (error)
        .  . Pause
        .  ENDIF
        .  Set direction and movement status for all units;
        .  CALL VECTOR
        .  Update dimensions of each smoke cloud; CALL SMKUPD
#
#   Begin minor cycle:
#       .  WHILE (time remains in minor cycle)
        .  .  IF (new sorting needed of Red maneuver BG's)
        .  .  .  Sort Red BG list: CALL ROLSRT
        .  .  ENDIF
        .  .  Compute Red and Blue unit speeds: CALL SPEED
        .  .  Model direct fire: CALL DIFIRE
#
#   Assume Blue BG gets the first shot, and Red must delay
#   several major cycles before returning fire:
#
        .  .  WHILE (cycle over Red BG's)
        .  .  .  Check if Red can open fire; reset accordingly
        .  .  ENWHILE
        .  .  Move units: CALL MOVE
        .  .  Initialize cells for Blue and Red: CALL GRNLTR
        .  .  Increment minor time by one
        .  ENDWHILE
#
#   End minor cycle
#
        .  Compute speeds of Red and Blue artillery and non-
        .  BG command posts: CALL CPSPED
        .  Model combat support: CALL COMSUP
        .  Move units (artillery, non-BG command posts and
        .  sensors):
        .  CALL CPMOVE
        .  WHILE (cycle through BG's)
        .  .  IF (BG exists)
        .  .  .  Compute enemy strength and distance to near-
        .  .  .  est Red BG from Blue BG: CALL DSTSTR
        .  .  ENDIF
        .  ENDWHILE
        .  Control return of covering force: CALL CFCTRL
        .  Perform staff functions: CALL STAFF
        .  Increment times
        .  Output time
        .  Perform periodic summaries: CALL IOCTRL
        ENDWHILE
#
#   End major cycle
#
        Close files: CALL CLOSEFS
        END PROGRAM
```

```
      SUBROUTINE PLYRINIT
#
#   Initialize the event flag clusters and global data areas
#   used by UK4C and the controller process
#
      INCLUDE external files
      Associate 1st bank of event flags with UK4C
      IF (not normal)
         Write error message
      ELSE
      .  Associate 2nd bank of event flags with UK4C
      .  IF (not normal)
      .  .  Write error message
      .  ELSE
      .  .  DO FOR parts of event flag bank:
      .  .  .  Clear event flags of 1st bank
      .  .  .  IF (not normal)
      .  .  .  .  Write error message
      .  .  .  ELSE
      .  .  .  .  Clear event flags of 2nd bank
      .  .  .  .  IF (not normal)
      .  .  .  .  .  Write error message
      .  .  .  .  ELSE
      .  .  .  .  .  Clear elements of global section
      .  .  .  .  ENDIF
      .  .  .  ENDIF
      .  .  ENDO
      .  ENDIF
      END
      RETURN
      END
      SUBROUTINE DUMPINIT
#
#   Initialization routines for the backup and recovery module.
#   Opens the dump files and initializes all dump buffers.
#
      INCLUDE external files
      Initialize data
      Set error plays and open dump files
      Initialize dump buffers
      RETURN
      END
      SUBROUTINE OPENFILES
#
#   Opens files
#   (Important note on direct access files: they must not
#   increase in size during a run; otherwise, backup and
#   recovery will scramble them.  Use of MAXREC keyword is
#   supposed to trap this.)
#
    Initialize
#
#   Unit 1 is opened and closed by the data-reading subroutine
#   Unit 5 is opened by the operating system
#   Unit 19 is used for output of RED losses due to mines and arty
#
    Open files 3,6,8-12, 14-18, 20-22, 25-28, 33-40.
    IF (opening direct access files)
       Open files 2,7,23,24,29,45,47 (these are direct access
       files.)
    ENDIF
    RETURN
    END
    SUBROUTINE PAMREAD
#
#  Reads in staff function data
#
    INCLUDE external files
    Read staff function data
    Scan facility titles and command names to find last
    element used.
    IF (errors)
    .  Enter error routine: CALL DERROR
    ENDIF
    Initialize specific facility numbers
    RETURN
    END
```

```
      SUBROUTINE DERROR (error #)
#
#   Outputs error, and suspends execution with a PAUSE
#
    Print error #
    PAUSE
    RETURN
    END
```

```
          SUBROUTINE INITS
#
#    Reads in all user optics; initializes the data base;
#    Defines format of user option arrays
#         INCLUDE external files
          Initialize variables
          Load the RED battalion list
          Initialize blue perceived weapons(class A) and location
          Initialize RED perceived weapons and location
#
#    Load one basic load of ammunition:
          Load BLUE direct-fire
          Load RED direct-fire
          Load BLUE arty
          Initialize RED stop
#
#    Initialize starting strength (class A):
     .    BLUE starting strength
     .    Set distance to nearest blue for BN's in holding attack:
     .    RED starting strength
          Initialize units D,F,G,7: CALL ZZAP
          Read in options for this run
          IF (suppression)
          .   Write header on unit 12 for suppression and movement
          .   status
          ENDIF
          Close file 47
          Initialize MI variables and arrays: CALL INIT-MI
          RETURN
          END
```

```
        SUBROUTINE  ZZAP (array name, # elements, starting point)
#
#    Sets an array or part of an array to zeros.  The array
     can be of any type.
#
        Declare variables
        Initialize variables
        IF (array $> 2^{24}$)
        .  Note error: CALL DERROR
        ENDIF
        IF (array type and length are incommensurate with
        .  FORTRAN rules)
        .  Note error: CALL DERROR
        ENDIF
#       Calculate #8-byte units, and remaining #bytes, in array
        to be zeroed. (This allows the use of a pointer or an
        array reference)
        IF (#8-byte units $>$ 0)
        .  Zero the 8-byte units: CALL ZAP8
        ENDIF
        IF (# remaining bytes $>$ 0)
        .  Zero these remaining bytes: CALL ZAP1
        ENDIF
        RETURN
        END.
```

```
      SUBROUTINE ZAP8 (array name, length)
#
#     Zero 8-byte units from ZZAP
#
      Declare variables
      DO for all 8-byte units
      .  Zero each array element
      ENDO
      RETURN
      END
```

```
        SUBROUTINE ZAP1 (array name, length)
#
#   Zero remainder of bytes (after removing 8-bit bytes)
    from  ZZAP
#
    Declare variables
    DO for all remaining bytes
    .  Zero array element
    ENDO
    RETURN
    END
```

```
         SUBROUTINE INIT_MI
#
#   Initializes variables and arrays for the MI Module
    (connected with ending a run
#
         INCLUDE external file
         Set up minefield density array
         Establish initial values in array
         RETURN
         END
```

```
      SUBROUTINE BDEBNS
#
#   Loads BLUE BN's info BDE's
#

    INCLUDE external files
    Declarations
    Initialize variables
    DO for all BN's:
    .  IF (the BDE number for the BN is not zero)
    .  .  IF(  Brigade status   ≠6 )
    .  .  .  Load BN into BDE
    .  .  ENDIF
    .  ENDIF
    ENDO
    Store results on unit 6
    RETURN
    END
```

```
      SUBROUTINE ROLSRT
#
#  Provides a list of RED BN's sorted according to increasing
#  role
#
      INCLUDE external files
      Declare variables
      DO cycle through BN ID's
      .  DO cycle through all higher BN ID's than current one
      .  .  IF (higher-indexed BN has a larger role)
      .  .  .  Swap places in BN list, for BN ID's and asso-
      .  .  .  ciated variables
      .  .  ENDF
      .  ENDO
      ENDO
      RETURN
      END
      SUBROUTINE GRNLTR
#
#  Controls the determination of which cells are occupied
#  by each Red and Blue maneuver and artillery unit and what
#  fraction of that unit is contained in each cell.  This
#  information is used to calculate the terrain and vegeta-
#  tion types and the various obstacles that are encountered
#  by a unit when calculating the speed of the unit and the
#  possibility of firing in subsequent routines.  GRNLTR is
#  called at the end of each minor cycle.
#
      INCLUDE external files
      Initialize variables
      DO for all REd maneuver BG's
      .  IF (Red maneuver BG # > max Red maneuver BG #)
      .  .  DO for Red artillery BG's
      .  .  .  IF (Red artillery BG # > max Red artillery
      .  .  .  BG #)
      .  .  .  .  DO for Blue artillery batteries
      .  .  .  .  .  IF (Blue artillery battery # > max
      .  .  .  .  .  Blue artillery #)
      .  .  .  .  .  .  DO for Blue maneuver BG's
      .  .  .  .  .  .  .  IF (Blue maneuver BG # > max Blue
      .  .  .  .  .  .  .  maneuver BG #)
      .  .  .  .  .  .  .  .  RETURN
      .  .  .  .  .  .  .  ELSE
#
#  Granulate Blue maneuver BG's:
#
      .  .  .  .  .  .  .  .  Set unit as Blue maneuver BG
      .  .  .  .  .  .  .  .  Granulate the unit: PROCEDURE
      .  .  .  .  .  .  .  .  PROCGRN
      .  .  .  .  .  .  .  ENDIF
      .  .  .  .  .  .  ENDO
      .  .  .  .  .  ELSE
#
#  Granulate Blue artillery batteries
#     .  .  .  .  .  .  Set unit as Blue artillery battery
      .  .  .  .  .  .  Granulate the unit: PROCEDURE
      .  .  .  .  .  .  PROCGRN
      .  .  .  .  .  ENDIF
      .  .  .  .  ENDO
      .  .  .  ELSE
#
#  Granulate Red artillery BG's:
#
```

```
       .  .  .  .  Set unit as Red artillery BG
       .  .  .  .  Granulate the unit: PROCEDURE PROCGRN
       .  .  .  ENDIF
       .  .  ENDO
       .  ELSE
#
#  Granulate Red maneuver BG's
#
       .  .  Set unit as Red maneuver BG
       .  .  Granulate the unit: PROCEDURE PROCGRN
       .  ENDIF
       ENDO

       END

*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *

       PROCEDURE PROCGRN
#
#  Granulate units by cells
#
       Modify positions
       Determine cells occupied and the arrangement by type:
       CALL CELLS
       IF (in cell other than 1st row, 1st column)
       .  Find fraction of unit in each cell: PROCEDURE
       .  PROCFRA
       ELSE
       .  Set row and column #'s to minimum
       ENDIF
       RETURN
       ENDPROCEDURE

*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *

       PROCEFURE PROCFRA
#
#  Calls appropriate subroutines to find fraction of unit in
   each cell
#
       IF (1 row, > 1 column)
       .  Calculate cell occupations: CALL UNFRC2
       ELSE IF ( > 1 row, 1 column)
       .  Calculate cell occupations: CALL UNFRC3
       ELSE IF (2 rows, 2 columns)
       .  Calculate cell occupations: CALL UNFRC4
       ELSE IF ( > 2 rows, 2 columns)
       .  Calculate cell occupations: CALL UNFRC5
       ELSE IF (2 rows,  > 2 columns)
       .  Calculate cell occupations: CALL UNFRC6
       ELSE IF ( > 2 rows, > 2 columns)
```

```
.  Calculate Cell occupations: CALL UNFRC7
ENDIF
IF (unit is Blue maneuver BG)
.  Insert cell occupations
ELSE IF (unit is Blue artillery BG)
.  Insert call occupations
ELSE IF (unit is Red artillery BG)
.  Insert cell occupations
ELSE IF (unit is Red maneuver BG)
.  Insert cell occupations
ENDIF
RETURN
ENDPROCEDURE
```

```
      SUBROUTINE CELLS
#
#   Determine the cells occupied by a unit and the arrangement
#   type which governs calling of subroutines for fractional
#   cell computations
#
      INCLUDE external files
      Calculate limits and #'s of rows and columns
      Determine arrangement type, to be used in calling frac-
      tion-cell subroutines
      RETURN
      END
```

```
     SUBROUTINE UNFRC_(_ means 2,3,4,5,6 or 7)
#
#  Utility routine for GRNLTR, which calculates the fraction
#  of a unit within each occupied cell.  There are 6 of these
#  routines, with the "_" in the title being filled with
#  2,3,4,5,6,7; these #'s are the arrangement types:
#
#                 type       errors       cols
#
#                  2           1           >1
#                  3          >1            1
#                  4           2            2
#                  5          >2            2
#                  6           2           >2
#                  7          >2           >2
#
#
#  Let A,B stand for row or column (depends on type)
#
     INCLUDE external files
     Calculate area of unit
     Calculate y-dim for 1st A
     Initialize cell counter
     DO for middle A's
     .  Calculate x-dim for 1st B
     .  DO for middle B's
     .  .  Calculate x-dimension for B
     .  ENDO
     .  Calculate x-dim for last B
     .  IF (A being serviced is not the last A)
     .  .  Calculate y-dim for last A
     .  ENDIF
     ENDO
     RETURN
     END
```

FURTHER SUBROUTINE EXAMPLES

```
         SUBROUTINE INITIAL
#
#   Initialize the smoke cloud constants that are used in
#   calculating the movement of a cloud
#
         INCLUDE external files
         Calculate some of the smoke cloud constants: CALL SCONST
         Set up smoke cloud constants array
         RETURN
         END
```

```
      SUBROUTINE IOCTRL
#
#  Controls the calling of print and plot routine for the
#  periodic summaries
#
      INCLUDE external files
      Calculate and output staff processing statistics: CALL
      STATS
      Calculate and output list of communication nets, with
      cumulative activity: CALL COMPRT
      IF (internal has passed for unit position output)
         Output unit positions: CALL EPOSIT
      ENDIF
      IF (interval has passed for BG, ARTY, and minefield
         effects output)
         Output BG, ARTY, minefield effects
      ENDIF
      RETURN
      END
      SUBROUTINE ROPENFILES
#
#   Called instead of OPENFILES for restart.  Ensures that
#   the direct access files are compatible with the run
#   situation.
#
      INCLUDE exterior file
      Open units 2,3,6-12, 14-18, 20-29, 33-35, 38-40
#
#     Units 1, 4, 5, 19 are opened elsewhere
#
      RETURN
      END
      SUBROUTINE RECOV_4C
#
#  Main interface to the outside world for recovering from
#  a dump.
#  Looks at all dump headers to find the most recent dump to
#  use.
#  Recovers the 4C "DATABASE" from the chosen dump file.
#
      INCLUDE external files
      Declaration of variables
      Set error flag
      Force an initial dump file read
      If (not a log dump)
      .  Write to unit 6: "Recovery of data base starts"
      .  Find most recent dump file: CALL GETDUMP
      ENDIF
      IF dump file exists
      .  Skip header blocks
      .  Recover all data:
      .  CALL BRAINC
      .  CALL BRBINC
      .  CALL BRCINC
      .  CALL BRDINC
      .  CALL BREINC
      .  CALL BRFINC
      .  CALL BRGINC
      .  CALL BRHINC
      .  CALL BRIINC
      .  CALL BRJINC
      .  CALL BRKINC
      .  CALL BRLINC
      .  CALL BRMINC
      .  CALL BRNINC
      .  CALL BROINC
      .  CALL BRPINC
      .  CALL BRRINC
      .  CALL BRSINC
      .  CALL BRTINC
      .  CALL BRUINC
      .  CALL BRWINC
      .  CALL KFERDA
      .  IF (this is a log dump)
      .  .  Set dump index to 1
      .  ENDIF
      .  IF (not a log dump)
      .  .  Write: "Recovery complete"
      .  ENDIF
      .  Reset error index
      ENDIF
      RETURN
      END
      SUBROUTINE BACKUPERR
#
#   Handles the error reporting for the backup and recovery
#   module
#
      IF (error # is< o, or > 5)
      .  Pause
      ENDIF
      IF (error # =o)
      .  Write "Total elements of array   buffer size"
      ELSE IF (error # =1)
      .  Write "QIO failure in backup-recovery update"
      ELSE IF (error # =2)
      .  Write "Backup-recovery module failure"
      ENDIF
      Pause
      RETURN
      END
      SUBROUTINE RLOG_4C
#
#  Recovers 4C "DATABASE" from a specified logging dump
#
      INCLUDE exterior files
      Declare variables
      Prompt for name and directory of logging file
      Open QIO channel; CALL QIOPEN
      IF (error on opening)
      .  Write "Error opening"
      ELSE
      .  Recover from dump:  CALL RECOV-4C
      .  Close    QIO channel
      .  IF (error)
      .  .  Write "Error closing QIO channel"
      .  ENDIF
      ENDIF
      RETURN
      END
      SUBROUTINE DUMP_4C
#
#  Main interface to outside world for dumping.  Creates
#  dump header, copies all common block elements and other
#  relevant data to dump file.
#
      INCLUDE external files
      Set error flag
      IF (not a log dump)
      .  Work at which dump file to use
      .  Set up dump header for this dump; CALL IDATE
      ENDIF
      IF (no errors)
      .  Write header
      .  IF (input/output status block not normal)
         .  Output IO status block; CALL QIOERR
      .  ENDIF
      .  Reset buffer
      .  Handle rest of data:
      .  CALL BRAINC
      .  CALL BRBINC
                   :
      .  CALL BRMINC
      .  CALL BROINC
      .  CALL BRPINC
      .  CALL BRRINC
      .  CALL BRSINC
      .  CALL BRTINC
      .  CALL BRUINC
      .  CALL BRWINC
      .  CALL FLUSHOUT
      .  IF (error)
      .  .  Report error: CALL PACKUPERR
      .  ENDIF
      .  Transfer contents of direct access files to dump file:
      .  CALL XFERDA
      .  Recall header block and update it
      .  IF (no errors)
      .  .  Write header
      .  .  IF (IO status bllock out normal)
      .  .  .  Output IO status block: CALL QIOERR
      .  .  ENDIF
      .  .  IF (no errors)
      .  .  .  Write header again
      .  .  .  IF (IO status block out normal)
      .  .  .  .  Output IO status block: CALL QIOERR
      .  .  .  ENDIF
      .  .  .  Report errors:  CALL BACKUPERR
      .  .  .  IF (not log dump)
.   .   .   .   Update last-dump index
.   .   .   .   Output message of creation of dump file
.   .   .   ENDIF
.   .   ENDIF
.   ENDIF
ENDIF
RETURN
END
```

```
      SUBROUTINE PLCHEK
#  Called every major cycle to see if the controller process
#  has requested action via his event flags
#
      INCLUDE external files
      Declare variables
#
#  Loop over all event flags, taking action for each one that
#  has been set:
#
      DO for each event flag:
      .  Set one event flag
      .  IF (no error)
      .  .  Lock out controller process to prevent use of the
      .  .  same option while we are processing the last re-
      .  .  quest.
      .  .  IF (option pointer is 1)
      .  .  .   Make backup dump at specified time, if reached.
      .  .  .   Clear lock on this option
      .  .  ELSE IF (option pointer is 2)
      .  .  .   Make logging dump at specified time, if reached,
      .  .  .   to specified file
      .  .  .   Clear lock on this option
      .  .  ELSE IF (option pointer is 3)
      .  .  .   IF (controller's requested stop time has been
      .  .  .   .  reached)
      .  .  .   .  STOP program
      .  .  .   ENDIF
      .  .  ELSE IF (time for new set of output files)
      .  .  .   Output files
      .  .  ENDIF
      .  ENDIF
      ENDO
      RETURN
      END
```

```
        SUBROUTINE COMSUP
#
#   Controls the calculation of the efforts of combat support
#   for both Blue and Red
#
#
        INCLUDE external files
        IF (game simulation time < artillery preparation time)
        .  DO up to max
        .  .  IF (time > initial firing time)
        .  .  .  IF (time < firing control time)
        .  .  .  .  Set target unit ID
        .  .  .  .  IF (target is a maneuver unit)
        .  .  .  .  .  Set x,y coordinates
        .  .  .  .  ELSE (target is an artillery unit)
        .  .  .  .  .  Set s,y coordinates
        .  .  .  .  ENDIF
        .  .  .  .  Locate the aim points into arrays
        .  .  .  ENDIF
        .  .  ENDIF
        .  ENDO
        .  With the preparation over, erase arrays and reset
           firing rates
        ENDIF
        Maintain suppression flags and duration counters: CALL
        ARTSUP
        Consider Red artillery firing (by NN): CALL REDARF
        IF (time ≥ (artillery preparation time -25))
        .  Calculate the minefield casualties if unit is moving
        .  through a minefield: CALL MINFLD
        .  Consider Blue artillery firing by battery: CALL BLUARF
        .  Represent the allocation of helicopters to unit sectors:
        .  CALL BLUHEL
        .  Represent the allocation of close air support to unit
        .  sectors: CALL BLUCAS

        ENDIF
        RETURN
        END
```

```
         SUBROUTINE CFCTRL
#
#   Controls the covering force, ensuring it returns under
#   the operational control of battle groups 5 and 6.
#
         INCLUDE external file
         DO for battalions
         .  IF (BN not destroyed)
         .  .  IF (BN location equals location of battle group 5)
         .  .  .  Merge battle groups: CALL OPCON
         .  .  ENDIF
         .  .  IF (BN location equals location of battle group 6)
         .  .  .  Merge battle groups: CALL OPCON
         .  .  ENDIF
         .  ENDIF
         ENDO
         RETURN
         END
```

```
     SUBROUTINE STAFF
#
#   Controls execution of staff processing functions
#
    Initialize
    Turn time checks on and off
    Control sensors: CALL ONOFF
    Direct movement of counter Battery Radars and Sound Ranging
    Units: CALL DIRECT CBRSRG
    Control Blue sensors: CALL BLUACQ
    Control Red sensors: CALL REDACQ
    Utility functions CALL BLOP
    Utility functions CALL REDOP
    Utility functions CALL EWIN
    Control staff messages: CALL PAM
    Control ARTY Staff processing: CALL UKARTY
    Update messages from all echelons of Red and Blue:
    CALL MESAGE
    Direct the tactical decision making for RED: CALL CMDRED
    Direct the tactical decision making for RED: CALL CMDBLU
    RETURN
```

```
     SUBROUTINE SMKUPD
#
#   Updates the length, breadth and height of each smoke cloud.
#
     IF (smoke clouds exist)
     .  DO for smoke clouds
     .  .  Determine type, mass, time of impact, time since impact
     .  .  burn time of smoke round (lifetime).
     .  .  Calculate height, length and breadth of cloud:
     .  .  CALL CLZIRK
     .  .  Calculate head and tail parameters of cloud
     .  .  Calculate cloud transparency
     .  .  IF (transparency sufficient for vision)
     .  .  .  Remove cloud from consideration:  CALL REMOVE
     .  .  .  Set height, width, upwind and downwind edges
     .  .  .  of cloud
     .  .  .  IF (desire printout of smoke cloud parameters)
     .  .  .  .  Print smoke cloud parameters
     .  .  .  ENDIF
     .  .  ENDIF
     .  ENDO
     ENDIF
     RETURN
     END
     SUBROUTINE DSTSTR
#
#  Computes enemy strength to 3000 m and distance to nearest
#  Red battalion, from Blue battalion.
#
     DO for all Red battalions
     .  Determine Red SW and NE Y-coord's
     .  IF (Blue NE Y-coord > Red NE Y-coord, and Blue SW
     .  .  Y-coord  Red SW-Y coord)
     .  .  IF (BN's not destroyed)
     .  .  .  Determine x-distance between Red and Blue
     .  .  .  IF (this x-distance ≤ 3000 m)
     .  .  .  .  IF (using perceived instead of true strengths)
     .  .  .  .  .  Compute Red strengths to 3000 m
     .  .  .  .  .  Save smallest distance
     .  .  .  .  ENDIF
     .  .  .  .  IF (vector magnitude distance between Blue
     .  .  .  .  .  and Red > x-distance between Blue or Red)
     .  .  .  .  .  Set vector magnitude distance = x-distance
     .  .  .  .  ENDIF
     .  .  .  ENDIF
     .  .  ENDIF
     .  ENDIF
     ENDO
     Compute current strength of Blue battalion
     IF (Blue BN strength < 1)
     .  Set BN strength = 1
     ENDIF
     RETURN
     END
```

```
      SUBROUTINE CLOSEFS
#
#   Close files at end of run, or if controller requests the
#   switching of output files.  In the latter case, dump files
#   are left alone.
#
      IF (closing all channels)
      .  Close units 2,7,23,24,29
      ELSE (close all channels except those assigned to direct
      .  access files)
      .  Close units, 3,5,6,8,9,11-18,21,22,25-28,33-41.
      ENDIF
#
#   Units 10,19,20 are not used
#   Units 1,4 closed by READDATA
#   Units 30-32 are I/O backup channels, and are closed else-
#   where.
#   Unit 45 closed by COMPRT
#   Unit 46 closed by STATS
#   Unit 47 closed by INITS
#
      RETURN
      END
```

APPENDIX A. REFERENCES

1. Bordeaux, T.A., User Requirements for an Automated CPX Support System, RDA-TR-185500-001, October 1983.

2. Bordeaux, T.A., Carson, E.T., Shinnick, F.M., Preliminary Design for an Automated CPX Support System, RDA-TR-185500-002, October 1983.

3. Parish, R.M., Modeling Command, Control, and Communications, U.S. Army TRADOC Systems Analysis Activity, May 1979.

4. Command, Control, Communications, and Combat Effectiveness Model Documentation, Volume I: Design Report, TRASANA TM 3-78, October 1978.

5. Command, Control, Communications, and Combat Effectiveness Model Documentation, Volume III: Implementation Report, TRASANA TM 3-78, October 1978.

6. Command, Control, Communications and Combat Effectiveness Model User's Manual, TRASANA TR XX-82, 1982.

7. U.K. FOURCE 82, Volume 1: Outline Definition, Royal Armament Research and Development Establishment, Joint TRASANA/RARDE Technical Report 1/82, May 1982.

8. U.K. FOURCE 82, Volume 3: Algorithms, Calculations and Processes, Royal Armament Research and Development Establishment, Joint TRASANA/RARDE Technical Report 3/82, May 1982.

9. U.K. FOURCE 82, Volume 4: Data Output and Reduction, Royal Armament Research and Development Establishment, Joint TRASAND/RARDE Technical Report 4/82, May 1982.

10. Rinkle, R., Champion, J., Gilbert, A, Evaluation of UKFOURCE for CPX Driver, Technical Solutions, Inc., September 1983.

APPENDIX B. SUMMARY OF TRASANA FOURCE SOFTWARE

TRASANA provided a complete set of FOURCE listings in March, 1983. The description of the FOURCE software in this section is based primarily upon the listings, with supplemental information drawn from the TRASANA documents referenced in Appendix A.

Figure B-1 and Table B-1 summarize the software organization of FOURCE. Figure B-1 partitions the software into 12 separate modules that interact hierarchically. Each module consists of a set of Fortran program units that model a specified function. Each module (except the utilities module) includes a single top-level driver that controls the execution of all of the program units. The absence of control coupling between modules would simplify using a module as a separate program, but the modules are coupled through access to common data structures, particularly Fortran named COMMON blocks. Table B-1 gives the name of the top-level unit in each module.

The executive module comprises 32 program units. Subroutines in this module model the grid representation of the battlefield, compute the effects of mines and smoke, determine movement direction and speed for all units, and perform housekeeping functions. This module also contains the main driver that controls the FOURCE simulation. Figure B-2 is a simplified representation of the FOURCE control logic. FOURCE is a time-step simulation that uses both major and minor cycles. The major cycle, whose nominal value is one minute, contains six minor cycles. At the start of each major cycle, the program performs six minor cycles, alternately updating unit locations and modeling direct fire. The program then accounts for the effects of combat support (e.g., artillery and helicopters) on conflict resolution. The remaining functions then model sensor acquisition of targets and generation of spot reports,

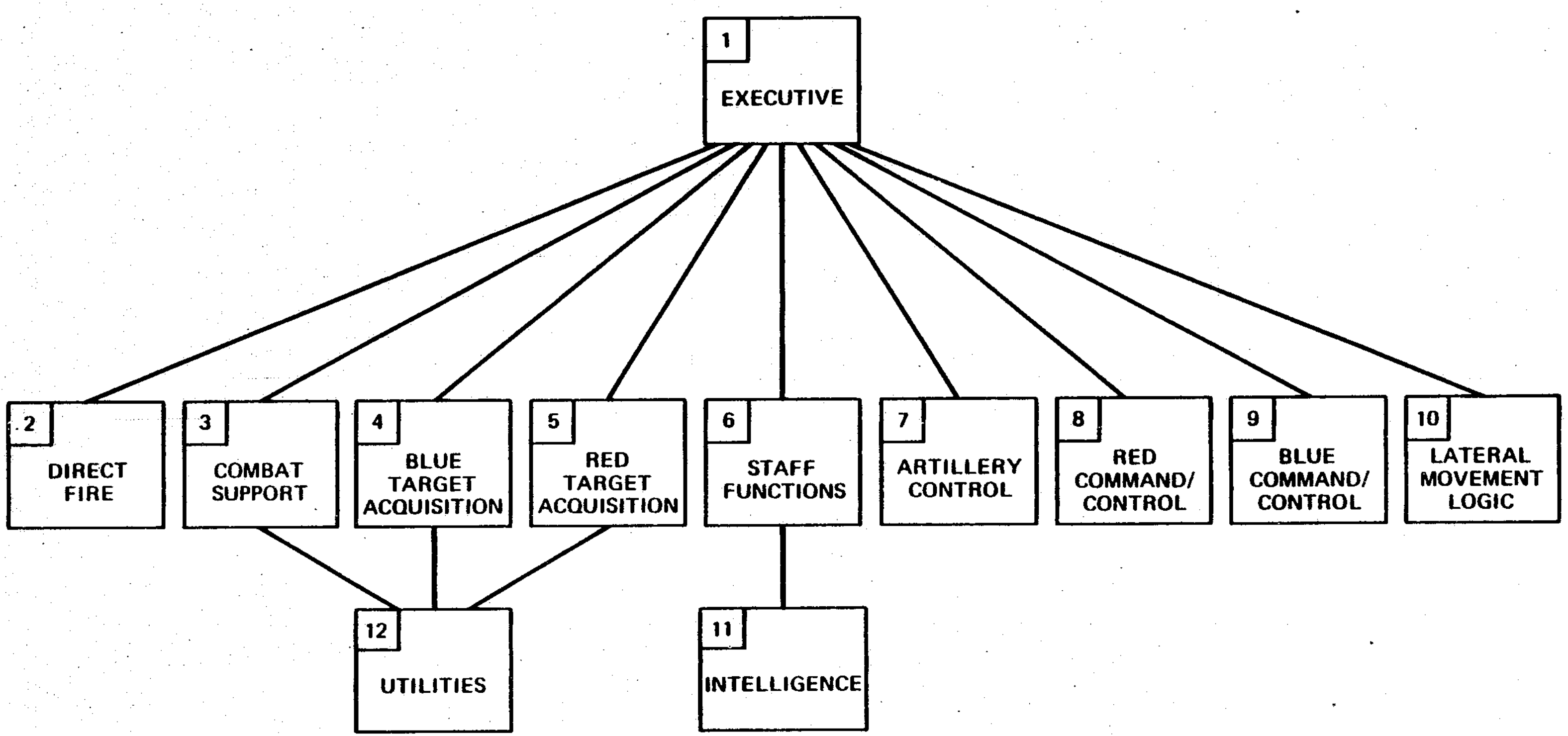


Figure B-1. Software configuration for Trasana FOURCE

TABLE B-1. SUMMARY OF TRASANA FOURCE MODULES

Module	Top-level subroutine	Number of subroutines
1) Executive	MAIN	32
2) Direct Fire	DIFIRE	19
3) Combat Support	COMSUP	28
4) Blue Target Acquisition	BLUACQ	25
5) Red Target Acquisition	REDACQ	9
6) Staff Functions	PAM	48
7) Artillery Control	USARTY	38
8) Red Command/Control	CMDRED	10
9) Blue Command/Control	CMDBLU	18
10) Move Laterally	CMDMOV	9
11) Intelligence	BLUINT	13
12) Acquisition Utilities	-	6
	Total	255

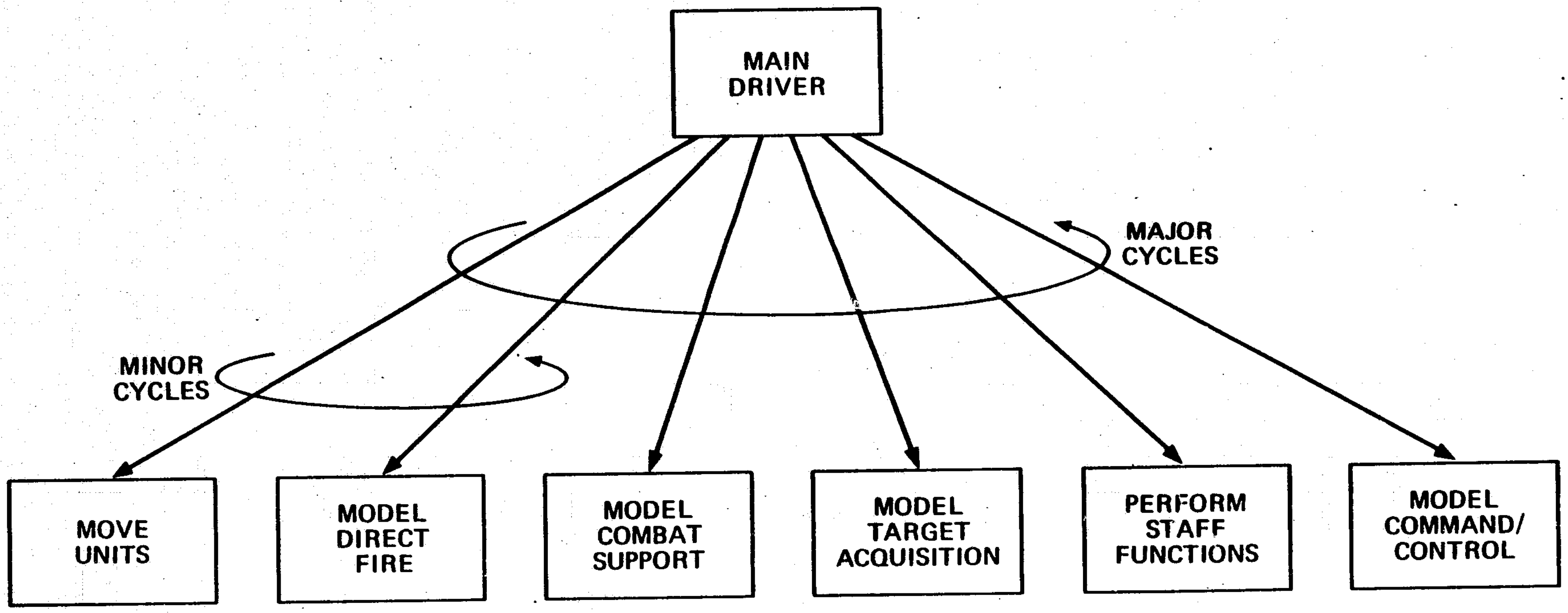


Figure B-2. Functional summary of FOURCE staff processing of the sensor reports, and command/control of the maneuver units.

Figure B-3 is a more detailed (but still simplified) representation of the FOURCE main driver using the procedure description language (PDL) notation defined in Section 4. The outer WHILE loop controls the time stepping by major cycles, and the inner WHILE loop corresponds to the iteration over minor cycles. The first operation within each major cycle uses the commands from the previous cycle to update the movement status and direction for all maneuver units. Computing the speed of movement, modeling direct fire, and moving the units are then all performed on a minor cycle basis. Separate modules perform the target acquisition and command functions for the opposing sides (Blue and Red).

Tables B-2 through B-13 define the software modules in Figure B-1 by listing all of the program units in each module. For each subroutine, the table gives the number of Fortran statements and the total number of lines in the source file. The total number includes comment lines and compiler instructions, so the difference between the two table entries provides an estimate of the extent to which the subroutine is commented. The 255 program units contain a total of 31,040 lines, 19,267 of which are Fortran statements. These totals do not count the COMMON blocks, which are stored in separate files. The average subroutine contains 122 lines, 38% of which are comments. These statistics indicate that FOURCE is a modular program with extensive comments. Some of the subroutines, however, are much larger than average. The largest subroutines are CMDBDE (1198 lines) and DIFIRE (734 lines); these important routines are more difficult to understand because of their size.

The software makes extensive use of Univac packing routines (particularly the function BITS) and some use of Univac assembly language for dynamic memory allocation.

```
MAIN
#
#
Initialize data structures.
WHILE (time remains in simulation)
$ (
     Set direction and movement status
     for all units:  CALL VECTOR.
     WHILE (time remains in major cycle)
     $ (
          Compute unit speeds:  CALL SPEED.
          Model direct fire:  CALL DIFIRE.
          Move units:  CALL MOVE.
          Advance time by one minor cycle.
     $ )
     Model combat support:  CALL COMSUP.
     Model target acquisition by Blue:  CALL BLUACQ.
     Model target acquisition by Red:  CALL REDACQ.
     Perform staff functions:  CALL PAM.
     Perform artillery functions:  CALL USARTY.
     Model Blue command/control:  CALL CMDBLU.
     Model Red command/control:  CALL CMDRED.
     Advance time by one major cycle.
     IF (user requested)
          Write out current battle status.
$ )
END
```

Figure B-3. PDL description of FOURCE main driver.

TABLE B-2. MODULE 1: EXECUTIVE

Top-Level Unit: MAIN

Program unit	Total lines	Fortran statements
ASMBLE	67	22
CELLS	64	28
CLZIRK	33	31
CPMOVE	310	191
CPSPED	348	181
DEGFC1	205	76
DEGFC2	117	43
DEGFC3	107	40
DEGFC4	122	66
DEGFC5	102	33
DIRECT	66	27
GRNLTR	187	105
MAIN	204	144
MINEMO	86	27
MOVE	428	214
NEARX	76	23
ONOFF	83	67
RDRVRC	154	58
REDJAM	46	41
REMOVE	19	18
ROLSRT	70	22
SMKCOV	81	58
SMKUPD	61	52
SPEED	311	163
SUPWRT	64	41
UNFRC2	66	28
UNFRC3	66	28
UNFRC4	66	28
UNFRC5	74	33
UNFRC6	74	33
UNFRC7	75	37
VECTOR	374	183
32	4206	2141
Average:	131	67

TABLE B-3. MODULE 2: DIRECT FIRE

Top-Level Unit: DIFIRE

Program Unit	Total lines	Fortran statements
ATCOEF	480	223
BLLOSS	152	53
CLENUP	129	60
DIFIRE	734	356
DSTROY	88	35
ERASE	61	22
FRCTIR	294	99
GAPLOS	95	60
GBDACQ	89	52
GOEDET	144	53
GRDACQ	89	53
HBKLRT	69	36
HRKLRT	62	35
LOSEFF	161	110
POSTUR	115	66
RDLOSS	153	53
SMKOUT	37	34
SMOKFR	83	52
SPSHUL	146	57
19	3181	1509
Average:	167	168

TABLE B-4. MODULE 3: COMBAT SUPPORT

Top-Level Unit: COMSUP

Program unit	Total lines	Fortran statements
ARTSUP	133	44
BAREFX	113	47
BARRAR	175	75
BARRCP	98	37
BARTMU	254	99
BARRSN	112	39
BLUARF	201	85
BLUCAS	109	41
BLUHEL	139	53
CLDPTH	179	133
COMSUP	102	32
CPDMGE	45	36
DERROR	11	9
ESLINT	32	15
EXSCOR	43	39
FASCAM	67	20
INSECT	57	16
INSERT	22	21
MINFLD	224	97
PRPTGT	48	39
RARBAR	204	92
RARBCP	104	38
RARBMU	260	102
RARBSN	107	36
RAREFX	105	38
REDARF	234	91
SMASSP	16	15
SMKMIS	62	48
28	3256	1437
Average:	116	51

TABLE B-5. MODULE 4: BLUE TARGET ACQUISITION

Top-Level Unit: BLUACQ

Program unit	Total lines	Fortran statements
BLUACQ	140	113
MOV	119	98
OV1CON	25	16
RPV	230	178
SOTCON	14	11
TAANT1	54	26
TAANT2	90	72
TABCBR	142	31
TABFOA	115	98
TABFOB	22	93
TABGSA	41	39
TABGSR	32	29
TABOVA	72	54
TABOV1	59	44
TABRPA	58	51
TABRPV	55	43
TABSOT	65	48
TABSTA	77	60
TABUGA	56	43
TABUGS	89	43
TGTDT1	66	36
TGTDT2	72	47
TGTDT3	41	27
TGTDT4	28	21
TGTDT5	64	30
25	1826	1351
Average:	73	54

TABLE B-6. MODULE 2: RED TARGET ACQUISITION

Top-Level Unit: REDACQ

Program unit	Total lines	Fortran statements
REDACQ	109	101
TARCBR	41	34
TARGSA	41	38
TARGSR	34	27
TARRCN	110	85
TARRNA	117	95
TRDET1	50	34
TRDET2	74	48
TRDET4	22	14
9	598	476
Average:	66	53

TABLE B-7. MODULE 6: STAFF FUNCTIONS

Top-Level Unit: PAM

Program unit	Total lines	Fortran statements
AOI	354	249
ASAC	191	116
BLOP	291	259
CFOLD	54	31
CFREP	75	44
CNET	6	86
COMPRT	51	38
COMQIN	88	68
COMQUE	68	38
COURER	39	31
CPNR	34	26
CPTYPE	124	114
CSMSG	84	55
CSNET	87	65
DBLOSS	62	47
DBUP	60	37
DISTR	60	47
EWIN	92	78
FLGSTS	48	43
FMRPBQ	89	38
GEN	66	46
GETAD	100	59
HFSIG	54	34
HTMOD	58	35
JGREEN	58	54
LNKAGE	74	59
LOGG	97	79
MSGTO	25	20
NOISY	26	18
OPSFAC	254	202
OTTO	106	54
PAM	181	117
PRODLY	40	83
QCHECK	70	45
REDI	34	26
REDOB	65	49
REDOP	91	66
SIGCON	254	166
SITREP	12	10
SNRPT	312	254
SNRR	119	93
SPOT	100	93
STATS	105	78

TABLE B-7. CONCLUDED

Program unit	Total lines	Fortran statements
TNPINT	267	178
TSCHED	213	133
UHFREL	35	31
ULOC	78	55
VHFSIG	36	21
ZSNITF	23	22
48	4952	3602
Average:	103	75

TABLE B-8. MODULE 7: ARTILLERY CONTROL

Top-Level Unit: USARTY

Program unit	Total lines	Fortran statements
BADUMP	26	20
BAMOVE	241	124
BARECV	92	60
BASEND	49	39
BAWOF	122	88
BNFDC	256	156
BSRCLR	17	17
BSRLST	97	49
BSROUT	16	12
BTYAMO	26	20
BTYAUL	141	77
BTYFDC	464	304
DIVFDC	323	204
FMRFIN	57	40
FMRGEN	71	52
FMRLST	82	42
FMROUT	32	16
FMRPBQ	89	38
FMRPOF	58	31
FMRTAM	103	38
MSGRCV	52	45
MSGSND	143	102
ORGLST	91	49
ORGOUT	16	12
PRBN	70	36
PRBTY	66	38
PRDIV	20	16
SENRD	39	35
SEQCLR	83	30
SEQDT	37	20
SEQFNP	74	46
SEQFQP	27	17
SEQPRT	20	15
SEQSPN	88	49
SOTAS	107	64
USARTY	115	75
USCBR	133	81
USFIST	139	84
38	3682	2241

Average: 102

TABLE B-9. MODULE 8: RED COMMAND/CONTROL

Top-Level Unit: CMDRED

Program unit	Total lines	Fortran statements
CMDRDV	384	301
CMDRED	97	66
CMDREG	314	243
CRMAX	26	18
FIRRED	243	175
FIRRST	144	115
MESAGE	485	325
PUTRS	67	54
REDBN	70	47
REDEAD	232	166
10	2062	1510
Average:	206	151

TABLE B-10. MODULE 9: BLUE COMMAND/CONTROL

Top-Level Unit: CMDBLU

Program unit	Total lines	Fortran statements
BDEBNS	166	130
BDEINT	176	142
CNMAX	25	17
CMDBDE	1198	809
CMDBDV	250	170
CMDBLU	199	107
DEFEND	312	240
DELAY	225	173
DSTSTR	47	34
JLDATA	105	77
MVRNF	112	70
RELOAD	128	79
RESERV	37	28
RESREQ	229	133
SMKMIS	62	48
SUPSIT	32	21
UPBDBN	67	58
WITHDR	157	119
18	3527	2455
Average:	196	136

TABLE B-11. MODULE 10: MOVE LATERALLY

Top-Level Unit: CMDMOV

Program unit	Total Lines	Fortran statements
ADJUST	264	176
CHKPOS	30	21
CMDMOV	13	9
CMMAX	25	17
FDNCEL	43	32
MSN4	134	88
MVLAT	547	360
REVIVE	296	242
9	164	1031
Average:	167	115

TABLE B-12: MODULE 11: INTELLIGENCE

Top-Level Unit: BLUINT

Program unit	Total lines	Fortran statements
ADDREC	61	46
BD2TO3	85	67
BLYINT	39	26
DBD	77	42
DBR	51	30
ITALK2	131	83
INTFAC	212	169
MERGE	91	59
MTHRST	296	243
REDUN	266	203
TRUINT	89	66
TYPSZE	119	81
UNTECH	34	27
13	1551	1142
Average:	119	88

TABLE B-13. MODULE 12: ACQUISITION UTILITIES

Top-Level Unit: NONE

Program unit	Total lines	Fortran statements
AQTPAM	67	26
PRTACQ	240	105
TAINT1	101	47
TAINT2	152	101
TPLSG1	58	32
VTRACE	105	61
6	723	372
Average:	121	62

RDA-TR-185500-004

PRELIMINARY PROGRAM PLAN FOR DEVELOPMENT OF AN AUTOMATED CPX SUPPORT SYSTEM

OCTOBER 1983

By:
T. A. BORDEAUX
E. T. CARSON
F. M. SHINNICK

Submitted To:
JET PROPULSION LABORATORY of the
CALIFORNIA INSTITUTE OF TECHNOLOGY
4800 Oak Grove Drive
Pasadena, CA 91103

R & D ASSOCIATES
Post Office Box 9695
Marina del Rey,
California, 90291

4640 ADMIRALTY WAY • MARINA DEL REY • TELEPHONE: (213) 822-1715

PREFACE

This is the fourth of a series of reports on a two-month effort to conduct a preliminary design of an automated CPX support system. The effort was performed for the Jet Propulsion Laboratory of the California Institute of Technology under Contract No. 956622.

CONTENTS

ILLUSTRATIONS

1. INTRODUCTION

RDA has performed a two-month effort to develop a preliminary design for an automated system to supports corps-level command post exercises. As the first task in this effort, we identified and documented user requirements for such a system (Ref. 1). The second task was to perform a functional analysis based on the requirements and to develop a preliminary functional design (Ref. 2). Another task, reported on in Reference 3, was to examine the suitability of the TRASANA FOURCE computer program for inclusion in the automated CPX support system. As the final task, we have prepared a preliminary program plan for development of the system which is documented in this report.

1.1 SYSTEM DEVELOPMENT APPROACH

We describe herein an approach to development of an automated CPX support system that emphasizes close interaction with the ultimate user of the system. It is based on fielding successively larger and more complex increments of the system so that user experience can be fed back into the design. We believe that an incremental approach is prudent because of the uncertainties associated with building such a large system; each phase provides results that can be used to improve the estimates of computer hardware/software requirements and to refine the design of the application programs. From the user's perspective, this approach has the advantage of providing an initial capability earlier than would otherwise be possible; experience in using the system should provide him with insights on the implications of an automated system for his exercise plans.

The first and smallest increment to be fielded would provide for limited play of one brigade. It would involve very limited representations of combat support and combat service support. This phase could use leased equipment or a computer service bureau, thereby deferring hardware purchases until the estimates of processing requirements can be confirmed experimentally. Capabilities to include play at division and finally corps level would be added in steps. Likewise, more detailed representations of artillery and air along with logistics also would be included in steps.

The set of computer programs that the role-players (or player-controllers) interact with and that calculate the outcomes of combat, combat support and combat service support processes are of course central to an automated exercise support system. The performance requirements for programs which represent models of maneuver unit behavior can be defined to a reasonable degree of specificity based on descriptions of the opposing units' organizations, their weapon systems and the terrain on which they will operate. On the other hand, the amount of detail that must be accommodated for combat support and combat service support depends upon the degree to which the user wishes those units to participate in the exercise as players, i.e., which portions of the units are to be trained. For example, if air units are to be trained in planning individual missions, the models representing air and air defense operations must be sensitive to mission profiles; hence, highly detailed and complex. If on the other hand, air units (and air defense) are directed by role-players, their representation can be more highly aggregated and simple, requiring much less computational power.

For purposes of development of an preliminary program plan, it has been assumed herein that the FOURCE simulation model is used as the kernel of the automated CPX support system. This assumption was made in order to have a basis for identifying tasks that would have to be accomplished and for estimating schedules.

It also was assumed that additions would be made to the FOURCE kernel to represent combat support and combat service support. Those additions are assumed to be the minimum essential needed to satisfy the user requirement that logistics and personnel decisions made by the players have the proper impact on the tactical situation. This requirement does not necessarily imply that combat service or combat service support units participate as players in the exercise.

One version of the FOURCE model runs on a VAX 11/780 computer; therefore, for planning purposes we have elected to use that equipment as representative for costing purposes. That machine is made by the Digital Equipment Corporation, which also makes the IVIS microcomputer-based videographics system. Therefore, for convenience we also have used that equipment as representative for costing purposes. The preliminary design report (Ref. 2) describes the selected computer hardware and vendor software, and presents a cost breakdown.

The following sections of the report describe the suggested increments of capability, the rationale for their selection, major development tasks that would have to be accomplished, and estimated schedules and costs for a development program.

2. DEVELOPMENT PLAN

The preliminary development program consists of four phases, all of which overlap in time. Each phase leads to the fielding of an exercise support capability that is useful in its own right and can be used beneficially for exercise and evaluation purposes. The use of each provides a mechanism for feeding back user experience to improve subsequent capabilities. Each phase includes a task to develop a more detailed development plan and system design for the succeeding phase.

2.1 PHASE 1 - BRIGADE EXERCISE SUPPORT

The primary objective of the first phase of the development program is to provide to the user an initial exercise support system capability. The initial capability would support a brigade-level exercise. The command control and message flow structure of FOURCE would be exploited to substitute a live brigade headquarters and up to five subordinate maneuver battalion headquarters for their counterpart computer-played headquarters now in FOURCE. Only the minimal necessary changes would be made to the model to support the one brigade and its battalions on an interactive basis. The auxillary objectives are to demonstrate the utility of FOURCE for interactive applications, to obtain more accurate estimates of computer resource requirements for a complete system, and to develop a more detailed design for modifying FOURCE. In accord with these objectives, the existing software would be modified in the most expedient manner to exploit the model's existing capabilities.

PRECEDING PAGE BLANK NOT FILMED

The existing doctrinal attack in echelon by the opposing force would be retained. The opposing force command structure would continue to be non-interactive; i.e., its tactical decisions would be made by the computer based on existing decision rules. Firefights of approximately thirty minutes or so would be played. No videographics would be employed in this phase.

One emphasis in this phase is on developing the message flow from subordinate units represented in the computer to battalion headquarters. It is envisioned at this time that computer-generated messages on enemy contact, unit position location and status and so forth would be printed out for transmission by role-players via telephone or radio to brigade. Orders from brigade also would be put into the computer by role players. The second emphasis is on validating the combat resolution models of the combat processes now in FOURCE. The third, and most important, thrust of this phase is to get a preliminary support capability into the hands of the users so that they have an opportunity to contribute their operational knowledge and experience to the design and development of later versions and the target system.

2.2 PHASE 2 - LIMITED DIVISION EXERCISE SUPPORT

The primary objective of this phase of the program is to field the capability to support division-level exercises involving all four maneuver brigades. Accomplishing this requires obtaining and installing both the central system and enough satellite systems to support the exercise director's staff, the brigade role players, and OPFOR (although the OPFOR satellite need not have the full complement of hardware). Videographics would be made available for interactive use by the role-players and the exercise director's staff. The opposing force also would be made interactive so that their use of tactics other than those currently programmed into FOURCE could be accommodated if desired. We anticipate that the capability to revert to the stylized attack on a non-interactive basis will be retained. This approach would have the advantage of allowing the exercise staff to conduct limited evaluations without having to staff the OPFOR unit.

A major change would be made to the software during this phase to extend the model down to company-level resolution for Blue. Only limited representation of combat support and combat service support elements would be included during this phase. During this phase, all software would be brought under formal configuration control. The development of all application software (including modifications to FOURCE) would be accomplished using modern methods of top-down design, step-wise refinement, and structured programming.

The detailed design and development of the satellite systems to be used in the various role-player cells will be accomplished during this phase. This will provide the opportunity to validate the preliminary hardware selection. In addition, more substantial changes will have been made in the software which will provide the additional opportunity to have benchmarks against which to assess the capacity of the central computer system to handle a corps-level system.

2.3 PHASE 3 - FULL DIVISION CPX SUPPORT

The objective of this phase is to develop and field the increment of capability required to support a CPX for a full division. Role-player hardware and software will be included now for the combat brigade (air assault), for fixed-wing air support, division artillery and logistics including resupply, maintenance and personnel as well as for other user-selected combat support and combat service support elements.

Additional exercise management capabilities to be provided are (1) the ability to record and replay in accelerated time the status of forces and (2) the capability to back the exercise up to some previous point to allow the exercise to proceed from different tactical decisions.

Formal documentation of the application software will be continued through this phase. Final decisions on the target system hardware configuration will be made based on the adequacy of the capacity and responsiveness of the then-current hardware and software combination.

2.4 PHASE 4 - CORPS AUTOMATED CPX SUPPORT SYSTEM

The objective of this phase is to install the final increment of exercise support capability. The capability to be provided will support a corps consisting of three divisions and a separate brigade/regiment plus corps organic combat support and combat service support. The effort involved in this phase primarily involves adding the hardware and integrating the software for the additional units. The final formal documentation on the system would be developed and delivered during this phase.

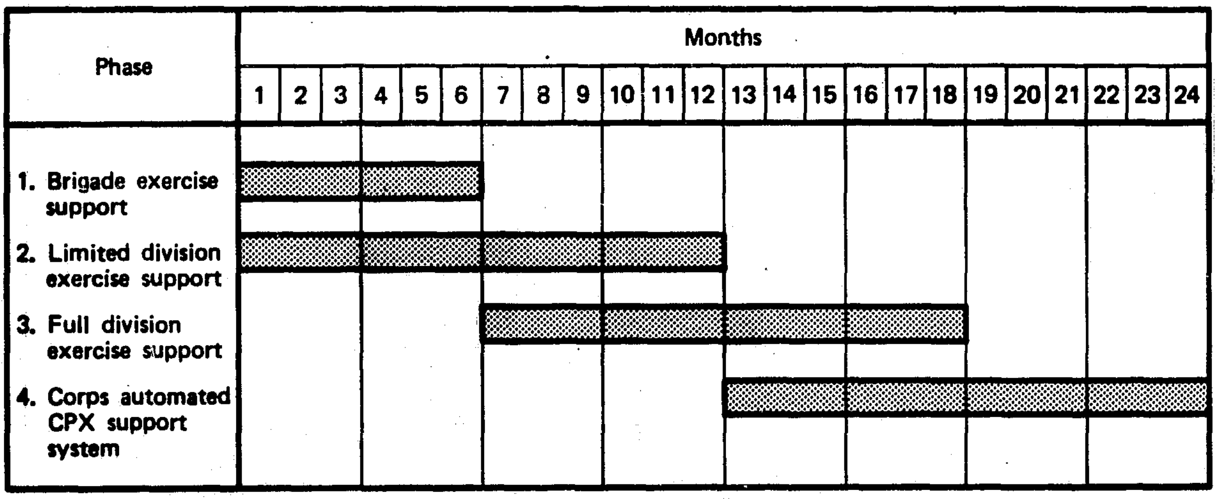
2.5 PROGRAM SCHEDULE

It is estimated that the four phases of the development program could be accomplished within a period of 24 months, thereby allowing time for user evaluation and feedback. The schedule could be compressed by combining some of the tasks. The duration of and relationship between the phases are shown in Figure 1.

2.6 PRELIMINARY PROGRAM COST ESTIMATE

It is estimated that the total program will cost approximately $4,991,000. This is itemized by phases in Figure 2.

Labor cost is estimated at a wraparound rate of $110,000 per man-year. Material cost is manufacturer's list price, including an allowance for taxes, spares, and miscellaneous items.



Phase	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	Months																							
1. Brigade exercise support	▨	▨	▨	▨	▨	▨																		
2. Limited division exercise support	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
3. Full division exercise support							▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨						
4. Corps automated CPX support system													▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Figure 1. Preliminary program schedule.

Phase	Labor	Material	Total
1. Brigade exercise support	$ 400,000	$ 153,000	$ 553,000
2. Limited division exercise support	900,000	343,000	1,243,000
3. Full division CPX support	900,000	230,000	1,130,000
4. Corps automated CPX support system	900,000	572,000	1,472,000
5. Central computer system		545,000	545,000
6. Central computer maintenance (24 mo @ $2000/mo)		48,000	48,000
Totals	$3,100,000	$1,891,000	$4,991,000

Figure 2. Planning purpose program cost estimate.