ADVANCED DECISION SYSTEMS
1500 Plymouth Street, Mountain View, California 94043-1230 • (415) 960-7300 • FAX (415) 960-7500

ADS TR1159-01
December 1988

AN OPERATIONS MONITORING ASSISTANT (OMA) PHASE II FINAL REPORT

Daniel G. Shapiro
Richard F. Shu
Carl J. Tollander

DRAFT FINAL REPORT - PHASE II
SEPTEMBER 30, 1986 - NOVEMBER 30, 1988

Prepared as part of U.S. Government Contract #DAAB07-86-C0051
ADS Project Number 1159

Prepared for:

U. S. Army Communications-Electronics Command
Fort Monmouth, NJ 07703

REPORT DOCUMENTATION PAGE

ORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS N/A
URITY CLASSIFICATION AUTHORITY N/A	3 DISTRIBUTION/AVAILABILITY OF REPORT Only as directed by the Contracting Officer at U.S. Army Communications-Electronics Command, Fort Monmouth, NJ 07703
LASSIFICATION/DOWNGRADING SCHEDULE N/A	
ORMING ORGANIZATION REPORT NUMBER(S) 1159-01	5 MONITORING ORGANIZATION REPORT NUMBER(S) DAAB07-86-C0051

AE OF PERFORMING ORGANIZATION Advanced Decision Systems	6b OFFICE SYMBOL *(If applicable)* N/A	7a NAME OF MONITORING ORGANIZATION U.S. Army CECOM
RESS (City, State, and ZIP Code) 1500 Plymouth Street Mountain View, CA 94043-1230		7b ADDRESS (City, State, and ZIP Code) AMSEL-PC-5-A-C3-COMM SYS SUP ER RITA STONE FORT MONMOUTH, NJ 07703
AE OF FUNDING/SPONSORING ANIZATION U.S. Army CECOM	8b OFFICE SYMBOL *(If applicable)* N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
RESS (City, State, and ZIP Code) AMSEL-PC-5-A-C3-COMM SYS SUP ER RITA STONE FORT MONMOUTH, NJ 07703		10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

E *(Include Security Classification)*

AN OPERATIONS MONITORING ASSISTANT (OMA) PHASE II FINAL REPORT (UNCLASSIFIED)

SONAL AUTHOR(S)
DANIEL G. SHAPIRO, RICHARD F. SHU, CARL J. TOLLANDER

PE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
FINAL REPORT	FROM 9/30/88 TO 11/30/88	1988, December, 15	66

PLEMENTARY NOTATION

COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
LD	GROUP	SUB-GROUP	

TRACT The document is the final report for the Operations Monitoring Assistant (OMA) project, which has produced an implemented prototype that assesses the impact of current events on Corps and Division Level operations plans. This system takes inputs describing the operations plan, the orders of battle (own and enemy), the user's monitoring concerns, and battlefield reports, and produces a series of monitoring alerts identifying the problems and future opportunities created by each situation change. This feedback is at the mission level (for example, predicting enemy occupation of key terrain), and is tailorable to a particular commander's interests. Our technical approach is based on the concept of instrumenting a simulation with feature recognizers that extract monitoring data.

TRIBUTION/AVAILABILITY OF ABSTRACT NCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
AME OF RESPONSIBLE INDIVIDUAL Thomas Baldwin	22b TELEPHONE (*Include Area Code*) 201-532-3535 / 22c. OFFICE SYMBOL

RM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

Table of Contents

er phone conversation with Eli Bean
IC CECOM, Ft. Monmouth, NJ & Jeanette
ngery DTIC distribution should read
pproved for Public Relase.

List of Figures

List of Tables

1. Introduction

This document is the final report for the Operations Monitoring Assistant (OMA) project, which has been a two year, three man year effort (funded as a Phase II SBIR) to assess the impact of current events on Corps and Division Level operations plans.

Plan monitoring is performed at all echelons in the military, formally within the Current Operations branch of G3. The function of monitoring, however, lies at the juncture of several tasks (situation assessment, projection, and response generation) and involves the commander, the G2, G3 and G4. Situation assessment, performed by the G2, is the transformation of data reported from the field into a picture of the order of battle, including some assessment of what the enemy is intending to do *now*. Projection (our term), performed by the G2, G3 and G4, is the act of looking some distance into the future to see if a problem or opportunity is on its way. Response generation, directed by the commander and the G3, is the task of responding to current events by tasking maneuver units or allocating resources. This can occur at several levels of formality/severity; over the phone, via frag orders, or via complete reworking of the plan.

The key observation of this simple analysis is that projection is not automated in the operational military. As a result, important developments (good or bad) may go unrecognized for some time, decreasing the benefit of military planning and the effective application of military force. This is the motivation for our work on OMA; we wished to create an early warning tool for use in response generation. While the example is a bit extreme, OMA computes the equivalent of *for want of a nail ... the war was lost.*

1.1 System Capabilities

Our work has resulted in an implemented prototype, delivered to CECOM at Ft. Monmouth, NJ, with the following behavior. Given descriptions of:

- the operations plan,
- the orders of battle (own and enemy), and
- the user's monitoring concerns,

the system will take input describing;

- simulated battlefield reports, and
- changes to own force or expected enemy courses of action,

and produce a series of reports identifying the problems and future opportunities created by each situation change. This feedback is at the mission level (for example, predicting enemy occupation of key terrain), and is tailorable to a particular commander's interests. A method of implementing additional monitoring primitives is also provided.

Our technical approach is based on the use of simulation for projecting the consequences of current events, as shown in figure 1-1. In this view, future state is predicted by a coarse simulation, while significant events are *noticed* by a collection of monitoring functions which examine the situation data. In effect, we perform monitoring by instrumenting the simulation. This produces a collection of monitoring alerts which the system filters via the *user interest model.* In our implementation, alerts are phrased as analogues to SITREPS, COMSPOTS, and CLOSRPTS, which the commander might see from the battlefield if the predicted events came to pass. The user's interests are expressed in terms of report selection/matching functions. Impact assessment, meaning attribution of alerts to the original situation changes, is accomplished by a comparison function. I.e., we match the alert stream derived from the new situation to the alert stream produced *without* the new input reports (obtained from the previous simulation run). The differences are *attributed to* the situation updates.

This technical approach to monitoring has several advantages:

- it supports accurate prediction (as opposed to a heuristic, or simulation-free technique which is intuitively less reliable), and
- it provides a rich and expandable array of monitoring outputs (alerts are based on feature extraction primitives applied to a detailed world model; that model can support a wide range of feature detectors).

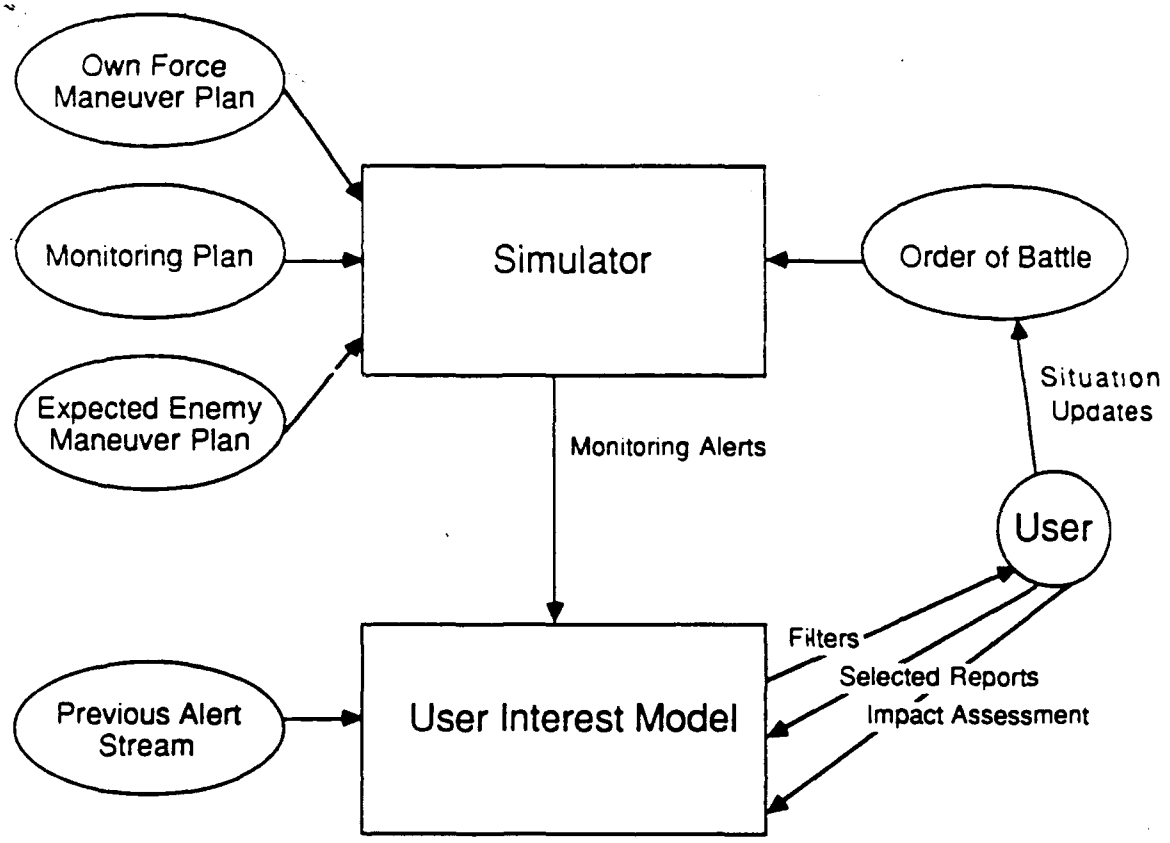
As we have implemented it, the principal disadvantages are:

- the monitor does not employ deep knowledge of the situation (it has no tactical understanding of the order of battle - it merely simulates it), and
- the execution speed is slower than desired (currently $\sim$10 minutes to project a Division level plan 24 hours into the future)

Within the context of monitoring by simulation, we have had to address several key technical problems. In outline form they are:

- representation of the maneuver plan,
- supporting reactive unit behavior so units can pursue their plan while responding to events at hand,
- representing the monitoring plan, and
- modelling causality for impact assessment (a somewhat syntactic solution for this is described above).

Figure 1-1: Conceptual Design of the Operations Monitor



These issues (and their solutions) are discussed in chapter 3.

1.2 Benefits to the Army

This monitoring prototype indicates that it is technically possible to build a monitoring system for use by Division and Corps level operations staff. Steps toward that automation have already appeared in experimental contexts (for example, machine readable message generation and routing systems), demonstrating the input link required by the OMA.

At the current time, Corps level monitoring involves a significant amount of manpower, and employs data that is acknowledged to be older than desired. Because of the quantity of information processed, clerical mistakes occur, and significant reports can on occasion be missed. In the absence of a simulation system, the implications of current events are never formally computed, allowing important problems and opportunities to go unseen, allowing preparation time to dwindle.

An operations monitoring system will significantly improve both the quality of information available to decision makers (by automating much of the situation update process), and support better decision making (by providing earlier warning of problems to correct and opportunities to exploit).

The benefits of such a system are quite large to the military.

1.3 Integration with ALBM

The OMA prototype has been implemented based on tools constructed for the Air Land Battle Management (ALBM) project, and is being incorporated into ALBM as a whole. Specifically, we employ the Heuristic Combat Evaluator (HCE) and Contingent Hierarchical Scripts (CHS) systems produced by BBN, a terrain reasoning abstraction built by Lockheed's Austin Division, and ALBM's copy of Intellicorp's KEE.

OMA will soon be able to accept ALBM plan objects as input, making it possible to monitor the Corps level plans generated by ALBM.

1.4 Guide to Reading

This final report provides discussions of the military plan monitoring domain (chapter 2), the technical issues it involves (chapter 3), the high level design of the OMA system (chapter 4), and our thoughts on future directions for this research (chapter 5). Appendix I describes several of the technical approaches we investigated but did not implement in our final product. These discussions bring out a number of interesting observations, and many of the ideas are worth pursuing in other contexts.

1.4.1 System Documentation

ADS is aware that CECOM intends to use, and build on the OMA code. In order to support that effort, we have written several additional pieces of documentation, and we have included code level information about the tools OMA employs.

The full list of documentation supplied to CECOM is as follows:

1. This final report (which provides the high level design and OMA system functionality information).

2. A user's manual [4], which explains how to load and operate the OMA prototype and demonstrations.

3. A programmer's manual [5], which discusses code architecture, file structure, the major data objects, and support utilities. We also discuss *hackers-only* operations for building OMA scenarios and for writing additional monitoring primitives.

4. The CHS user manual.

Note that the user's manual and the programmer's manual are meant as informal documents, whose purpose is to smooth the transition of our research software to CECOM. Product level documentation of our code is outside the scope of our Phase II SBIR contract.

2. The Problem Domain

As mentioned in the introduction, plan monitoring lies across the boundary of several military tasks; situation assessment, projection, and response generation. Since situation assessment involves all of information integration, and response generation can encompass all of planning, we have had to carefully scope our research effort. This chapter explains how we have restricted the monitoring problem for OMA. We begin with a discussion of plan monitoring from the military perspective, and then describe the subproblem we have addressed within OMA.

2.1 Plan Monitoring in the Military

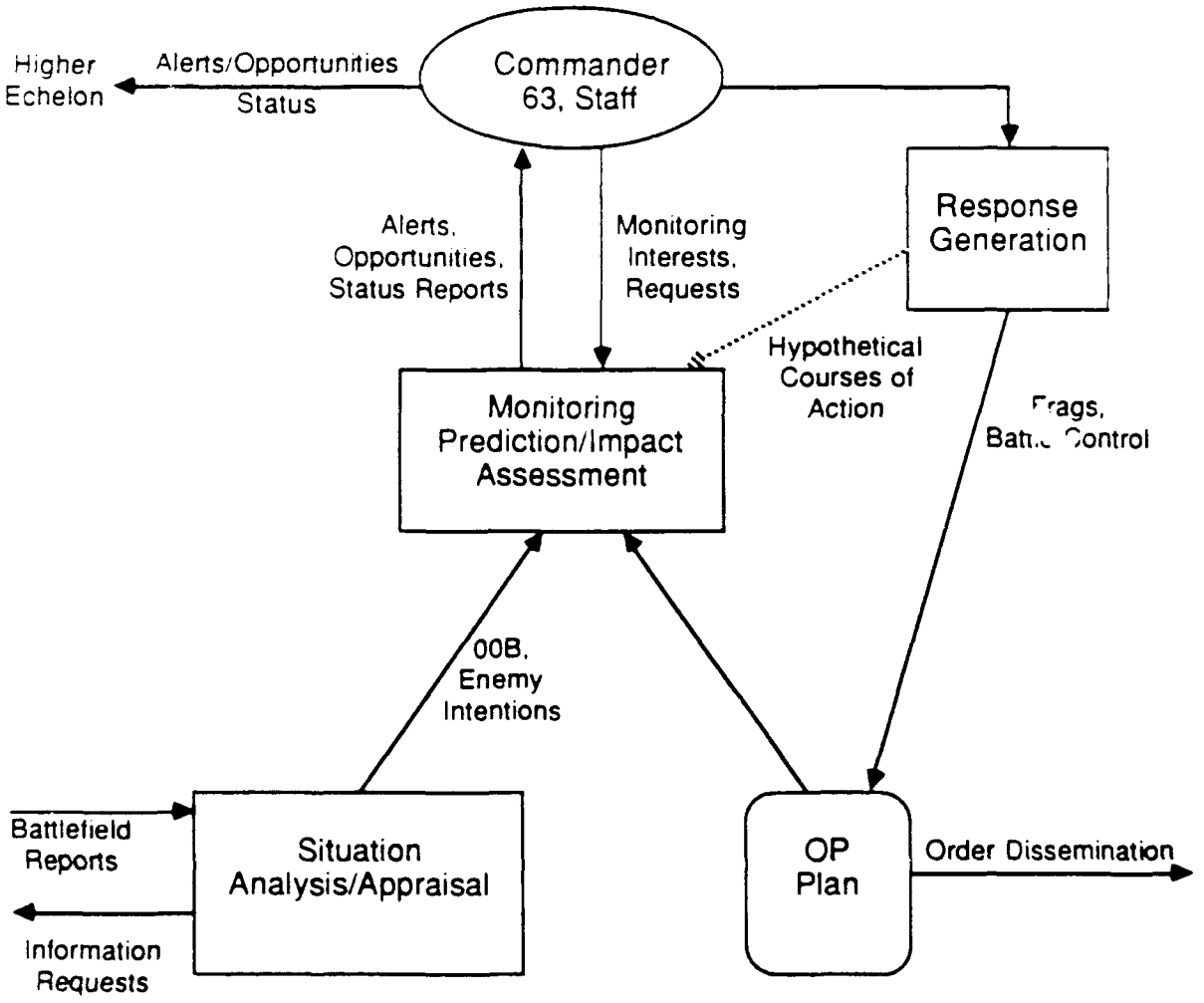
Figure 2-1 outlines the component tasks of military plan monitoring. Here, the principal input is a stream of reports from the battlefield, while the main outputs are task assignments, resource assignments and maneuver directives to subordinate commands. Internal to monitoring is the task of recognizing and anticipating problems or opportunities, so that the subordinate forces can be appropriately controlled. This task has three components; situation assessment (performed by the G2), projection (our term), performed by the G2, G3 and G4, and response generation, directed by the commander and supported by his staff, principally the G3. Situation assessment is the transformation of data reported from the field into a picture of the orders of battle, including some analysis of what the enemy is intending to do. Projection is the act of looking some distance into the future to see if a problem or opportunity is on its way. Response generation is the task of responding to current events, in order to avert problems or take advantage of growing opportunities. A good projection function will provide the military with highly desirable preparation time for response generation.

Monitoring is a continuous process. Reports come in from the battlefield as events occur (though some reports have a fixed schedule), the G2 is constantly updating the order of battle, and the commander and G3 respond as the situation dictates. The generation of course of action modifications also occurs in parallel with immediate battle control.

The time scale and emphasis of monitoring also changes with echelon. At Corps, immediate intervention in the flow of battle is comparatively rare, while prediction and response generation must look far into the future. At Battalion, battle control decisions consume virtually all of the staff effort. Situation assessment is done by eye, and plans look ahead a matter of hours. The reasons for this variation are discussed further in section 2.1.5.

Military plans also have some surprising structure which affect the scoping of a monitoring aid. We discuss the content of military plans, the timescale of monitoring, and the major subtasks from Figure 2-1 in the following sections.

Figure 2-1: A Functional Breakdown of Monitoring in the Military



2.1.1 The Content of Military Plans

One might expect a military plan to be a completely enumerated web of maneuvers, with timetables and expected outcomes, that could be mechanically monitored on an action by action basis. In general, it turns out that this view is false, and that plans are surprisingly less detailed.

The issue is that situations change during execution of the plan. As a result, large units are directed with abstract commands in order to insure enough leeway for response. US doctrine also encourages commanders to tell subordinates *what* to accomplish, not *how*, in order to maximize their use of initiative. From a computational perspective, this means that mission statements are closer to problem specifications than action sequences. This is particularly true of defensive plans, which may consist of a terrain responsibility, coordination requirements, and the directive to *defend in sector*.

Offensive plans show similar effects. For example, the maneuver portion of a typical Division level mission statement might read:

- on order, attack forward in sector to secure objective alpha, defeating elements of the 1GTA,
- establish defensive positions on objective alpha,
- on order, conduct a fixing attack on the 55TR.

This statement specifies nothing about *how* to organize the forward attack (via hasty or deliberate attack, a frontal or flanking approach, envelopment or air support to cut enemy supplies, etc.). This information is generated by the Division commander while refining the course of action for *his* subordinates (which he will again communicate in terms of *what* to accomplish, not *how*). As the battle unfolds, the Division commander will give additional instructions to Brigade in response to the situation at hand.

This type of plan structure has several consequences for monitoring. First, the plan is obviously not constant across the duration of a battle; the monitor must accept inputs which alter the operations plan. Second, the monitor must produce, or have access to a refinement of the mission statement, so that it can know what tactics to monitor. We choose the latter, obtaining the plan refinement from the user (or alternatively, from the ALBM system). The level of resolution is also a question; Division commanders typically monitor battles in terms of battalion level forces, implying that OMA will need to know Battalion level maneuvers. Finally, in our approach of simulation based monitoring, the OMA must support a degree of autonomous response generation - so that simulated units can respond to the situation as it develops.

Another issue for automated monitoring is that the goal of a mission is not presented as

a state to be achieved, with clear success criteria. Any particular array of forces may or may not represent a successful *defend in sector* mission. Similarly, the success criteria for tactical actions (such as frontal attacks, flanking maneuvers, defense encircled, etc.) are a matter of interpretation. We have addressed this issue by predicting, and highlighting developments that are clearly important. However, we stop short of evaluating them in terms of success/failure.

2.1.2 Situation Assessment

Situation assessment is the task of understanding the order of battle. In Figure 2-1, the inputs are battlefield reports describing a wide range of data; unit location, strength and supply status, maneuver initiation and closing reports, enemy sightings, battle damage summaries, prisoner of war reports, and various intelligence and situation summaries. Approximately 50 of these report types are specified in Corps or Division tactical standard operating procedures. Some are normally provided on a regular basis, while others are issued as required. As a whole, this information filters up the command hierarchy, although in practice, the reporting structure does not strictly adhere to echelon (all Battalion reports to Brigade, etc.).

At each echelon, the G2 (or equivalent) uses this data to build a picture of the order of battle. Note that a good deal of this process is accomplished by inference, as when the G2 determines a red unit's location and identity because he received a message discussing a prisoner of war taken in some area. At a larger scale of inference, the G2 will hypothesize (perhaps privately) the enemy's objective from the situation data, or similarly (with the G3) the location of the enemy's main attack.

In more detail, situation assessment at Division and Corps level involves integration of information from organic and national resources such as satellite photography, air reconnaissance, signal intelligence, and direct battlefield observation. Regardless of the information sources, the net product is knowledge of the situation (unit IDs, location, their current behavior and status), with as much capability and intent information about the enemy as plausible. Situation assessment also concerns own force elements, as their location and status can be lost in the rush of events. Own force status reports flow through G3.

Given an accurate (or presumed accurate) picture of events, the commander can observe or predict problems which require his response. For our design purposes, we have separated this prediction problem from the task of deriving the current order of battle. In practice, the same people are involved, and the border between the tasks is somewhat obscured.

2.1.3 Projection

In computational terms, the output of situation assessment is a description of static state (unit location and supplies on hand), and unit control states (their current activities, and plan). Projection uses this information to identify problems or opportunities that are likely to develop in the future. The conclusions of this process are reported to the commander in the form of alerts if they are serious, or at regular staff briefings if the timeline for action is not as critical. The commander will always identify situations that particularly concern him, so the alert set must be tailored to his interests. However, a certain set of monitoring alerts are always relevant. For example, enemy occupation of key terrain, penetration of area boundaries, and any own force advances beyond designated sectors (especially if unopposed). The quantity, and frequency of monitoring alerts is also an issue; the commander should only be bothered with those reports that are clearly important.

One issue in projection is how far to look into the future. A maxim of any projection system is that the further one looks, the less reliable the predictions. In OMA, the error is introduced through assumptions about enemy actions, and by the simulation's own concept of battle outcomes and movement rates, etc. (We have made provision to override these). This implies the existence of a confidence horizon, or upper bound on projection, beyond which it makes no sense to look. It is difficult to determine this interval. The lower bound on look ahead is easier to identify. Given that the purpose of look ahead is to contemplate an intervention *now*, and that it takes time for a Corp or Division commander's directives to even begin execution, projection needs to look forward past that delay, and into the execution of the maneuver itself. For a Division, the delay is $\sim$6 hours minimum, making 12-24 hours of look ahead the right number. For Corps, 24-48 is an appropriate range.

2.1.4 Response Generation

Once problems or opportunities are identified, it is up to the commander, with consultation by his staff, to determine the appropriate response. Several levels of response are available here; the commander or G3 can pick up the comms line and discuss the situation with a subordinate commander, he can modify the course of action by issuing a frag order, or he can replan the course of action and initiate a major tactical alternative via a revised operations plan. This last alternative is developed by the staff in Combat Plans.

For monitoring, the issue is how much response generation to provide. As mentioned above, some capability must be present if projection is to function at all, but complete replanning is obviously beyond the scope of operations monitoring.

We have chosen to include exactly enough response generation to support projection under the assumption the plan as a whole is still correct (e.g., units will attempt to do

something reasonable in all cases). It is up to the user, given feedback in terms of monitoring alerts, to decide that the current course of action is unacceptable. (This is, after all, a major purpose of monitoring.)

At this point, the user can alter the course of action as desired.

2.1.5 Variations in Monitoring with Echelon

An interesting point is that the nature of monitoring changes with echelon owing to the physical scale of the units involved. In essence, the larger the unit, the larger the sector, and the longer it takes for both own force and enemy to alter the situation at that scale. As a result, battle control decisions at higher echelon become comparatively rare (there is no point in making them more frequently than the situation changes). Similarly, the operations plan concerns longer periods of time. This creates a need to look further into the future to evaluate courses of action. Projection, as we have defined it, becomes a major concern.

At higher echelon, there is more information to gather about the battlefield (and more methods for acquiring it), making situation assessment a major concern. Paradoxically, the information takes longer to collect, making it somewhat less valid. This problem is compounded by the fact that responses take longer to enact at higher echelon; the information used to support planning is even further out of date at the time the first Battalion begins to move.

On the whole then, situation assessment and monitoring must become the principal activities at high echelon, with battle control playing a secondary role.[1]

This situation is reversed at low echelon, such as Battalion, where the physical scale is small. Here, it takes very little time to alter the situation, making battle control decisions common, and forcing plans to encompass small periods of time. Projection is a minor concern. As the battlefield is small, there is much less information to gather, and situation assessment is accomplished primarily by eye.[2]

---

[1] If you will pardon the humor of abstracting this situation to the absurd, the commander of the Planetary Invasion Forces will be able to initiate a plan, but it will take so long to execute and modify that he will never be able to change it. This is reasonable, however, because the situation assessment task is so enormous that the data will never be collected before the campaign is at an end. The entire operation will have to be planned on the basis of simulation (which is guaranteed, by the way, to be inaccurate). It may be that there is a natural limit to the size of any organization.

[2] Again, moving to extremes, a single soldier should have a plan of almost no duration, and spend his energy assessing and reacting to the situation on the fly. He should have no need for monitoring (looking into the future) at all.

2.2 The Monitoring Problem Addressed by OMA

In response to the analysis above, we have scoped the monitoring problem as follows. The OMA system will:

- consider Division and Corps level plans,
- focus on projection, looking approximately 24 hours into the future,
- use the output of situation assessment as the input to monitoring (meaning the system expects message input describing the orders of battle, not data which has to be collated and examined to deduce the order of battle),
- consider the maneuver plan to be the plan *as refined by subordinate units* (avoiding the need for OMA to automatically refine the mission statement), and
- automate response generation to the degree necessary for projection.

Major plan refinements and course of action generation are explicitly not included.

Since we have been engaged in developing a research prototype, we have taken some additional liberty concerning the military users of the system; our implementation allows a single user to input situation updates, change the own force (or assumed enemy) maneuver plan, and examine the monitoring alerts.

Within this problem context, the OMA can still perform a range of possible functions. We discuss the options in the following section.

2.3 The Desired I/O of Monitoring

In order to define the capabilities of a monitoring aid, we developed a hypothetical man/machine dialogue for monitoring execution of the EAC scenario [3]. We determined that monitoring ideally involves the following capabilities:

1. maintaining a current picture of the order of battle (unit locations, strengths, etc.),
2. tracking execution of planned actions (e.g., checking the initiation and termination of maneuvers, or their failure to occur according to schedule),
3. time and distance calculations,
4. predicting future positions and unit strengths,

5. predicting the consequences of current events (identifying potential problems, and areas of exploitable success), especially with respect to the Corps and Division level missions as a whole,

6. providing the user with tools for filtering monitoring reports,

7. exploring hypothetical actions by own forces and enemy, and

8. providing explanations for projection reports.

The following functionality is also desirable;

- deduction of enemy intent (and as a corollary, the ability to flag intelligence reports that imply important enemy actions),

- response suggestion (automated generation/ranking of both own force and enemy options), and

- worst case analysis (monitoring based on worst case enemy tactics and pessimistic conflict results),

As an overview, our implementation demonstrates functions 1 through 5 above, stressing projection, and a version of 7 ("what if" exploration) which allows the user to experiment with unit parameters and own force/enemy goals. We have not implemented point 8 (explanation), or any of the items in the second list (response generation, worst case analysis, etc.), as they would require significant additional research to provide.

Chapter 3 discusses the key technical problems (and solution approaches) involved in building an OMA as described above, while chapter 4 discusses the monitoring system we have implemented and delivered to CECOM. The following sections describe the domain scenarios we have used to motivate our work.

2.4 The TRADOC Common Teaching Scenario

Our source of scenario data for driving OMA development has been the TRADOC Common Teaching Scenario (TCTS). The TCTS focuses on corps and divisional level planning, providing echelon above corps (EAC) intents as context for the corps level Operations Plan. Our first semiannual report [3] discussed the EAC scenario, including a hypothetical dialogue between the user and the monitoring tool, while our second semiannual report [6] described the 10th Corps component of that plan. Ultimately, we implemented two Division level portions of the 10th Corps plan within OMA. For context, we briefly describe the EAC scenario below. Sections 2.5.1 and 2.5.2 discuss the defensive and offensive scenarios examined within OMA.

2.4.1 The Situation

TCTS concerns planning in response to a Warsaw Pact invasion of West Germany. NATO forces are under the command of Allied Forces, Central Europe (AFCENT), which is organized into three army groups: NORTHAG, CENTAG and SOUTHAG.

AFCENT is opposed by the Western TVD. The Western TVD is organized into four fronts. They are:

- 1 Western Front opposing NORTHAG
- Northern Front (2nd echelon behind 1 Western Front)
- 2 Western Front opposing CENTAG
- Central Front opposing elements of CENTAG and SOUTHAG.

See Figure 2-2. The Northern Front is the main attack force. It is assembled in Poland to the east of the 1 Western Front. It will be deployed to take advantage of the greatest success attained in the initial attack. This is expected to take place in the 1 Western Front sector. The terrain in this sector is more conducive to armored movement.

2.4.2 The AFCENT Plan

The three major parts of the AFCENT plan to defeat an attack by Warsaw Pact forces are:

- Defend well forward in sector to stop the initial attack.
- Delay, gaining time to assemble forces and material for a counterattack.
- Counterattack to defeat the Western TVD.

We have focused on the beginning of the counterattack, where defense is giving way to offense. Assuming the Warsaw Pact attacks on August 19, a 2 day initial defense followed by a 9 day delay places the counterattack on Aug 28. Figure 2-2 shows the plan, with the positions as expected on this date.

Note that the red forces were most successful in the north, where the 1 Western Front created a salient near Hereford. Following doctrine, the Northern Front deployed into this salient in hope of penetrating the NORTHAG line. The Central Front made only limited gains but continues to attack. At the end of the delay phase, the 1 GTA is at approximately 70% strength, and CENTAG units are at ~90% strength.

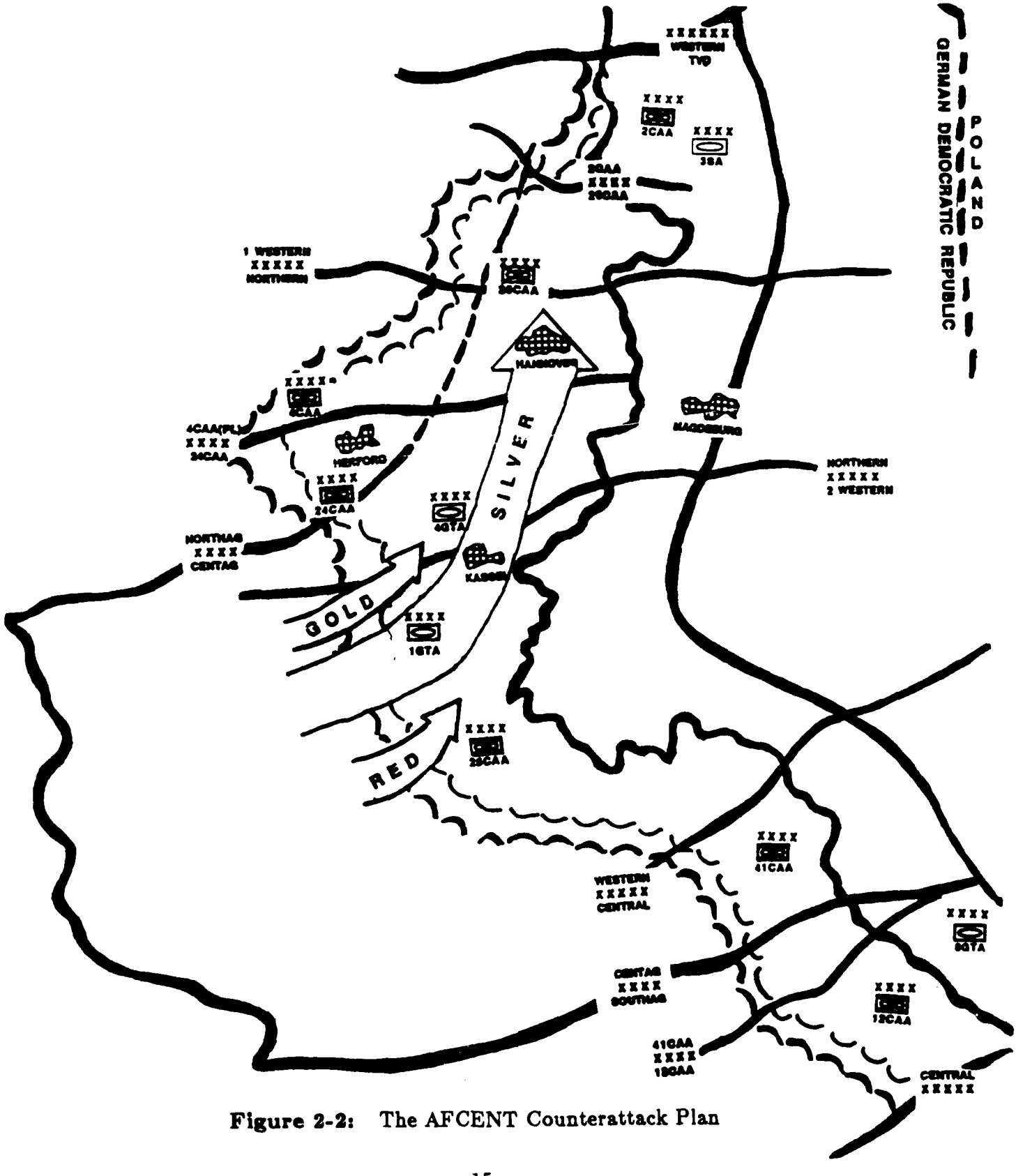


Figure 2-2: The AFCENT Counterattack Plan

AFCENT's plan is to defeat the Western TVD by destroying its main effort (Northern Front). To achieve this, the following objectives must be achieved:

- Penetrate the enemy LOC
- Exploit into the enemy rear
- Disrupt enemy's logistical support
- Block the strategic second-echelon from reinforcing the Northern Front
- Attack to defeat the first echelon of the Northern Front.

CENTAG will be responsible for defeating the 2 Western Front. It will then continue the offense to the northeast to sever the LOCs of the 2 Western and Northern Fronts. It will also be responsible for blocking the Soviet Group of Tank Armies (5 GTA and 7 GTA) east of Hanover. During the exploitation, NORTHAG will attack to defeat the first echelon of the Northern Front.

The desired result of this plan is shown in Figure 2-3. The essence of this situation is that the Northern and Western Fronts have deployed in the Hereford Salient, while 10th Corps has made sufficient speed in its northward movement (through the 1GTA) to flank all of the units in the salient, leaving them enveloped and without supplies. The speed of 10th Corps' movement is critical, as is the location and arrival time of the 5 GTA and 7 GTA. Other concerns are the potential use of the 4 GTA to reinforce the 1 GTA, and the deployment of uncommitted elements (i.e. 3 SA and 8 GTA). AFCENT uses a deception plan (see [3]) in an attempt to delay deployment of these uncommitted units.

Monitoring will be acutely concerned with 10th Corps progress, and with the consequences of enemy deployment options.

2.5 The OMA Scenarios

The following sections discuss two of the scenarios we have implemented within OMA (several more are available in the demonstration system, but these two are sufficient as examples). We consider a Brigade in defense, against the first wave of red forces as they cross the IGB, and a Division offensive which begins at the onset of the 10th Corps counterattack through the 1GTA. We have examined other scenarios for Brigade offense, and Division and Corps in defense. At Corps level, the scenarios become too large for our implementation to monitor efficiently.

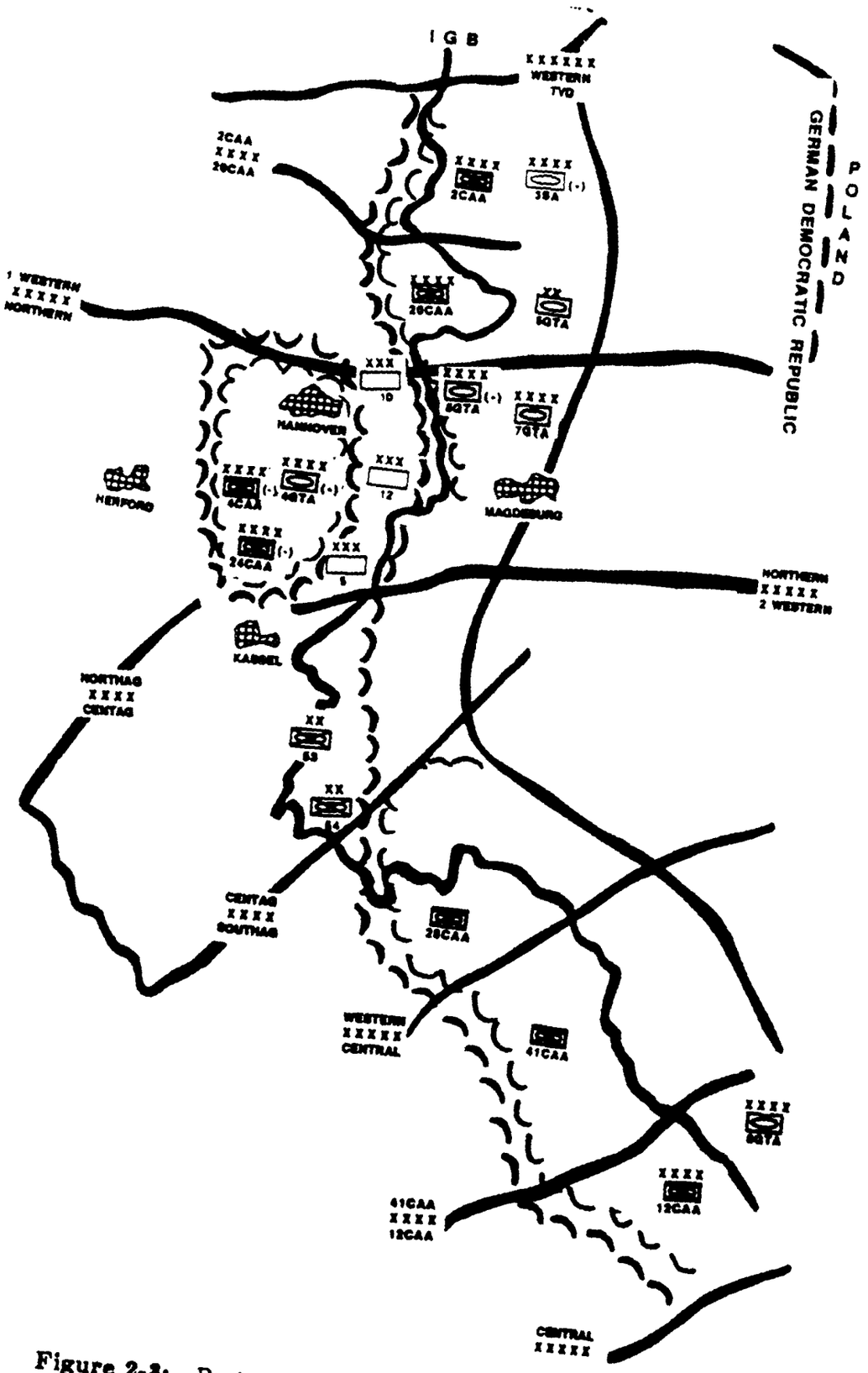


Figure 2-3: Projected Situation on D+12, September 9

2.5.1 Brigade in Defense

Figure 2-4 shows the force array for all of our defensive scenarios. This is a Corps level picture, showing the 10th Corps area, with its two Division sectors, for the 52nd Mechanized Division and the 23rd Armor Division. The Soviet forces have the better part of the 8th Combined Arms Army in the area, composed of the 79TD, 34MRD, 120MRD and 4MRD. Approximately 12 Regiments are shown. In this initial part of the battle, 10th Corps has not been reinforced with either the 54th or the 53rd Divisions, and must stop the threat advance alone.

The Brigade scenario concerns enemy units moving against elements of the 23rd Armor Division, in the security area of the defensive sector. The major monitoring concern is that NATO forces succeed in defending well forward. In the demonstration, the G2 receives input reports determining that the 1/91 Mechanized Battalion is at reduced operational strength, and that the 1/92 Mechanized Battalion has not arrived in its assembly area. The monitor determines how serious these problems are.

In the controlled experiment OMA performs, the simulation without these inputs shows US forces succeeding in the defense. With these inputs, however, enemy units succeed in passing the Division penetration boundary, indicating that a serious problem has developed.

2.5.2 Attack on the 1GTA

The counterattack scenario shown in figure 2-5 shows the 52nd Mechanized Division attack against 5 Mechanized Regiments of the Soviet 3GTD (arranged 3 forward, 2 back). Owing to the low initial strengths of the 52nd Mech Battalions (this counterattack begins after a prolonged delay operation), the attack has a force ration of between 1:1 and 1.5:1. In order to succeed, concentration of force is required. The offensive maneuver plan calls for a fixing attack by 2 Battalions on the 188MRR and a similar fixing attack on the 188TR. The remainder of the US forces (12 Battalions) are focused on the central unit, the 194TR. Note that this is the main US attack (thus the presence of 16 Battalions).

The goal of this maneuver is to break through the 1st echelon forces, *and* the second echelon units, since they form a thin wall in front of the open space required by the Corps level offensive plan. This plan (see figure 2-2) begins 12 hours earlier with a feint by the 23rd Armor Division, aimed at pulling the 16TR north, out of the path of the main attack.

The input report causing this run of the monitor is the notification that the 16TR is in fact moving north. The results of monitoring show US forces passing beyond their own area boundaries, as the attack succeeds (where formerly it did not). This indicates an opportunity to exploit, as well as successful execution of the 23rd Armor plan.

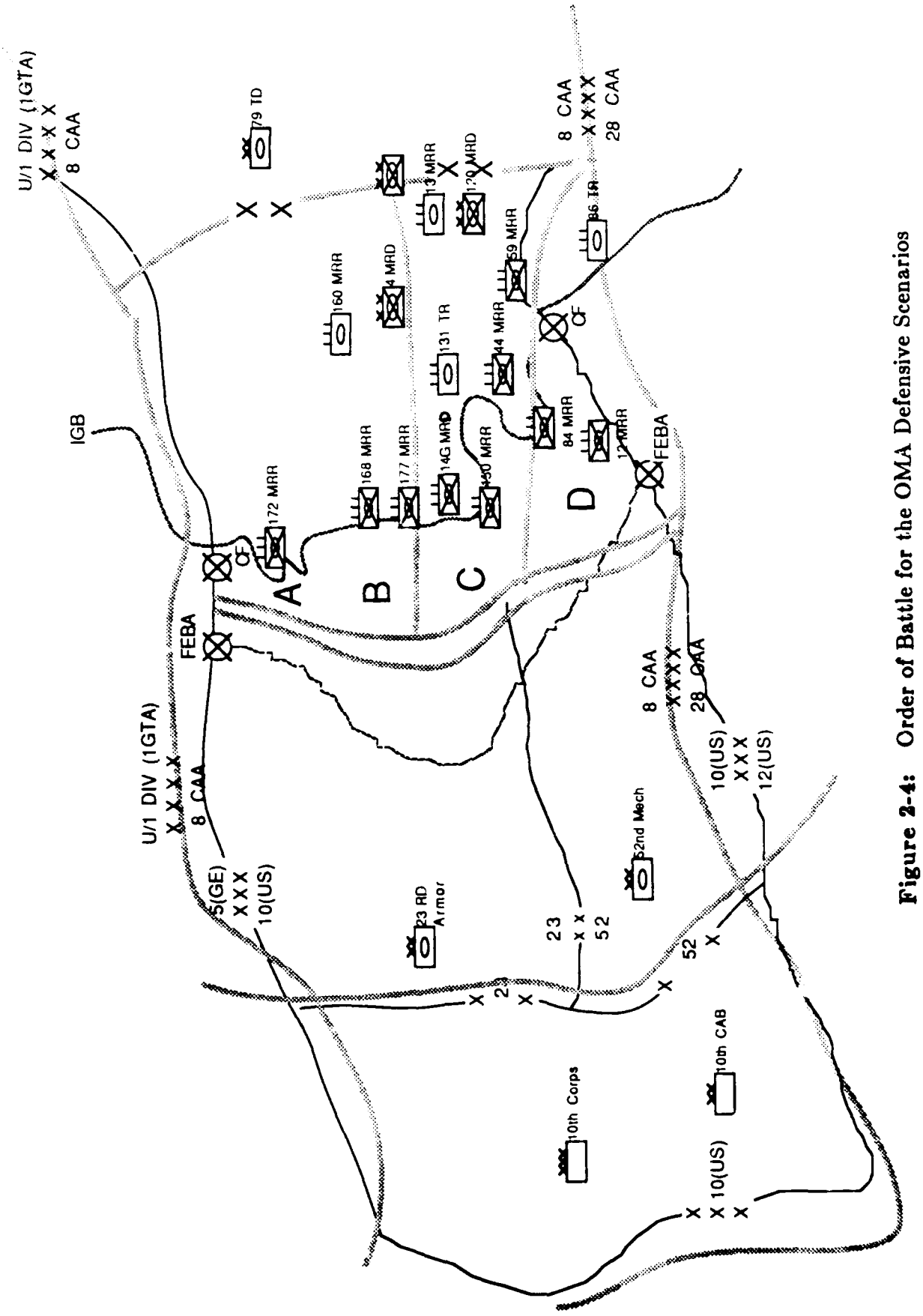


Figure 2-4: Order of Battle for the OMA Defensive Scenarios

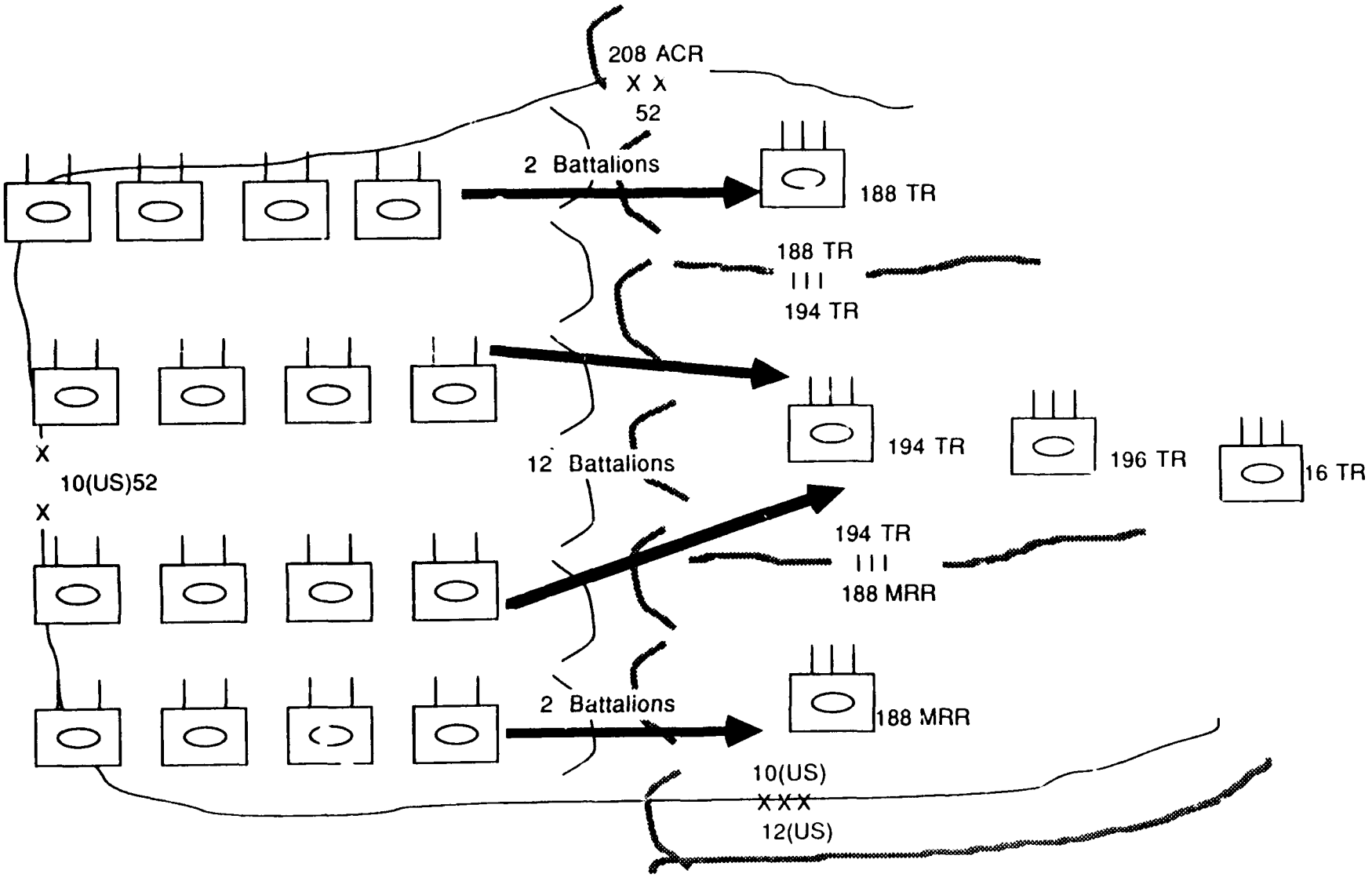


Figure 2-5: A Division Level Attack Scenario

3. Issues and Solutions

In the previous chapter, we defined monitoring for OMA to be the task of projecting the consequences of battlefield reports, taking mission statements which have been refined into maneuver plans as input, starting *after* situation assessment has been performed, and ending before any major replanning occurs. This functionality poses an easily identifiable set of technical problems:

- supporting accurate projection of future state,
- representing maneuver plans,
- representing the monitoring plan,
- allowing units to react to situations in the context of projection, and
- modelling causality for use in impact assessment (determining which input reports gave rise to the monitoring alerts).

We discuss each of these, and the key solution ideas below.

3.1 Accurate Prediction of Events

To provide accurate prediction of future state, we employ a simulation model. This model uses battalion level tokens with the attributes of strength and position, and includes combat resolution functions and movement calculations for engaged and nonengaged states. This simulation does not model many aspects of combat, for example, certain maneuver types (hasty attack/defense), the consumption of ammunition, and the effect of headquarters capability on battle. It represents a compromise between the desires for efficiency and prediction accuracy. We feel this particular compromise is appropriate for monitoring, since the goal is to make plausible predictions in support of monitoring alerts. We allow the user to override the simulation in situations where he disagrees with simulation results.

In the early phases of our work, we investigated several approaches to projection which did not employ simulation at all (projection through an influence net representing interactions on the battlefield, and deduction of consequences from current situation data alone). These techniques promised a different kind of monitoring feedback, expressed in terms of high level capability assessment, or tactical analyses of the battlefield. We found, however, that it became very difficult to reason about future consequences without knowledge of unit positions and intentions. As a result, we focused on a simulation based technique. It would be interesting to incorporate aspects of the other approaches in future work. They are discussed in appendix I.

3.2 Representing Maneuver Plans

As mentioned in section 2.2, the plan OMA monitors is expressed in terms of maneuvers to be executed by subordinate units. We use Battalion level tokens, and allow maneuver instructions such as *attack*, *defend*, *movement to contact*, *artillery attack*, *airborne attack*, etc. (There are approximately 50 of these.) This information is used to drive a simulation.

In order to execute maneuvers like the above, Battalions must have a degree of autonomy, both to decide *how* to execute a *movement to contact*, and how to react to situations that arise.

We express these maneuver control functions as scripts, stated in terms of maneuver goals and procedures for accomplishing those goals. Figure 4-6 is an example. We work through the interpretation process for this script in section 4.4.6. The key observation here is that the script language provides a modular method for composing maneuvers. (E.g., the procedure entitled *maybe withdraw* can be shared.)

Our maneuver scripts, and the tools for composing and executing them were originally developed by BBN, through the Air Land Battle Management Project (which is also a consumer of OMA). BBN's Heuristic Combat Evaluator (HCE) and Contingent Hierarchical Script (CHS) systems are both incorporated into OMA.

3.3 Representing the Monitoring Plan

In our approach, monitoring functions examine the situation during projection and report important developments. The set of monitors active at any given time is the monitoring plan (this is set by the user).

We have augmented the HCE system with scripts for representing maneuver plans. These scripts are written in the same format as maneuver plans, meaning that they are goal oriented, hierarchical, and sharable. For example, *monitor operations* is can be executed by any blue unit during simulation. It is composed of goals for monitoring strength, phase lines, and key terrain, etc.

Since each unit can be executing many monitoring tasks, we require scripts with parallel control flow.

3.4 Providing Reactive Unit Behavior

In the process of executing maneuvers, units must react to situations as they develop. Existing military simulations appear to perform this function through one of three mechanisms:

1. relying on a person in the loop (to make decisions for the blue side for example),

2. including a large number of control bits that specify unit reactions in predefined situations, or by

3. writing complicated branching logic which enumerates responses to assumed enemy actions (all tailored to the specific situation).

Our approach is to work for a more general functionality by embedding situation tests within the scripts discussed above. This allows units to set up triggers for particular responses, all within the context of a maneuver script. As an example, the *move to contact* action is set up as a trigger within *attack towards objective*, keyed on a sighting of an enemy within the attack corridor. This removes what would otherwise be a good deal of hard coded behavior.

We examined the use an alternative to scripts, based on a commercial product called Statecharts [1], for the purpose of instilling this reactive control. Statecharts are hierarchical finite state machines that support parallel execution, and promise a clean theoretical foundation for implementing reactive behavior. However, time and funding limitations prevented us from pursuing this approach to conclusion. This approach is discussed in section I.3.

3.5 Modelling Causality for Impact Assessment

OMA is a decision support system aimed at giving a commander the information necessary for good battlefield decisions. As a result, the OMA not only has to recognize problems and opportunities as far in advance as possible, but it has to help the commander focus his corrective efforts. For this reason, we have included the task of impact assessment within OMA.

Impact assessment is the act of attributing monitoring alerts to the situation changes identified in the original battlefield reports. It measures the significance of current events. The issue is that projection and *attribution* are different problems. For example, if the G2 introduces a new enemy unit into the simulation, the monitor may produce 30 alerts, *many of which have nothing to do with that situation change.* The alerts would have been generated in any case.

There are several approaches for performing *attribution.* One is to record influences throughout the simulation run, creating a data flow diagram that links input reports to monitoring alerts. This is mechanically feasible, although it poses some questions about the relative strengths of battle interactions. "Causal" interactions could be lost among the set of all interactions known to occur. We explore this technique in section I.2.

A second approach is to build tactical descriptions of the battlefield (in terms of threats, counterthreats, and actions) and analyze the results of projection in tactical terms. This form of postmortem would demonstrate deep machine knowledge of the situation, although it has the disadvantage that knowledgeable analysts might disagree. The technology to support it is also non-trivial. We did not explore this concept, although we believe it is a logical expansion path for future work.

Our approach is both brute force and simple. We detect causal relations via a controlled experiment; we run the simulation forward twice, once *with* the input reports, and once *without*. Any differences between the alert streams is *caused by* the input reports. While this is a syntactic approach, it seems to provide a useful mechanism for focusing the commander's attention.

4. OMA System Design

Our technical approach to monitoring is based on the concept of instrumenting a simulation with monitoring concerns. This gives rise to the design shown in figure 4-1, which was discussed in the introduction to this report. There are two components; a simulation system which executes the maneuver plan, applying the monitoring tests, and a user interest model which filters monitoring alerts, attributing them to situation changes introduced by the user. We discussed the key issues and solutions involved with this approach in chapter 3. This chapter describes the capabilities and architecture of the implemented system.

4.1 Capabilities

The OMA system performs the following task: Given descriptions of;

- the operations plan,
- the order of battle, and
- the user's monitoring concerns,

the system will take input describing;

- simulated battlefield reports, and
- changes to own force or expected enemy course of action,

and produce a series of reports identifying the problems and future opportunities created by each situation change. This feedback is at the mission level (for example, predicting enemy occupation of key terrain), and is tailorable to a particular commander's interests. The user can extend the set of monitoring primitives.

The following example (abstracted from a Division in defense scenario) illustrates the capabilities of OMA. Here, an own force Division is defending against several enemy Divisions in the security area of the Corps level battle.

The user inputs the following reports (via a graphical interface):

```
The 1/91 Mech Battalion is at Strength .8 (Yellow)
The 1/92 Mech Battalion is at position NA315300
  (not in its assembly area)
```

The system projects this information forward through simulation, producing a stream of status reports and monitoring alerts:

```
00:40 Enemy units crossing FLOT
12G-MRR(BTR) at NB321273
146G-MRR within engagement range of 1/91-Mech-Bn
1/95-Mech-Bn force ratio .37 -> .25
1/91-Mech-Bn force-ratio .80 -> .31
1/91-Mech-Bn strength .8 (Yellow) -> .59 Black
  ...
```

The user then defines a collection of selectors for pruning this stream (which actually contains a large number of reports). Selectors are report matching functions, with adjustable fields:

```
      STRENGTH REPORT SELECTOR
sender: any
dtg-sent: any
strength: (< .7)
previous strength: any
strength category: any
previous strength category: any
```

This selector will match any strength report that shows a unit to be at strength 70% or less, regardless of the unit's ID, it's previous strength, or strength category, etc. Application of the complete set of the user's selectors produces the following display:

```
00:40 Enemy units crossing 23rd-Armor-Div FLOT
03:20 1/91-Mech-Bn strength Black
04:16 1/92-Mech-Bn strength Black
05:25 Enemy units in 23rd-Armor-Div Division Key Terrain
06:30 Enemy units penetrating 23rd-Armor-Div Rear boundary
```

Following this, the system performs impact assessment on the report data, identifying the subset of these filtered alerts which were *caused by* the initial situation changes. This is accomplished by comparing the filtered alert stream with a similar stream generated by running the simulation without the user supplied situation changes. This results in the following output:

```
Input reports:

The 1/91 Mech Battalion is at Strength .8 (Yellow)
The 1/92 Mech Battalion is at position NA315300

Unmatched Output Reports:

03:20 1/91-Mech-Bn strength Black
06:30 Enemy units penetrating 23rd-Armor-Div Rear boundary
```

These are the alerts which occur because of the input data (they do not take place in the scenario when the input reports are omitted). The results indicate that the 1/91-

Mech-Bn's loss of strength and the 1/92-Mech-Bn's inability to establish position are serious problems. The commander can then consider corrective actions.

At this point, the user can model receipt of additional battlefield reports, and generate a new set of alerts via OMA. These alerts will be compared against the situation described above when it comes time for impact assessment.

Note that the user can play "what if" games by applying different sets of updates to the same situation. It is also possible to move units and alter strength values in the middle of projection if the user disagrees with the system's opinion of battle outcomes. It turns out that the simulation engine we employ allows the user to travel forward and backwards through a simulation trace, and explore alternate outcomes of any battle. That mechanism is not employed within OMA, although it is accessible through future modifications.

4.2 System Architecture

At the abstract level, the OMA code architecture follows figure 4-1. The user interface is more prominent than implied by that diagram, but the major division between simulation, and the report examination function holds. As a result, we discuss the architecture in two parts; the user interface/interest model, and the simulation system, which supports construction, representation, and execution of both the maneuver and monitoring plans.

4.3 The User Interface and User Interest Model

A high level view of the user end of OMA is shown in figure 4-2. This diagram identifies the primary user interface modes, data abstractions, and abstract operations.

There are three interface modes; *situation update*, *projection*, and *impact assessment*. In situation update, the user is allowed to graphically modify the order of battle, and make changes to the maneuver and monitoring plans. In projection, he runs the simulation, and defines methods for pruning the monitoring and status reports the simulation returns. In impact assessment, the user compares the report streams associated with different simulation runs. These are normally applied in sequence. Each interface mode corresponds to a separate screen configuration in OMA, but all are implemented in terms of operations on the underlying data abstractions.

4.3.1 The Situation Abstraction

The *situation abstraction* represents a simulation of an order of battle. It contains the situation updates entered by the user (the input reports), the complete set of alerts generated by the monitor, the user-defined selectors which restrict that set, and the

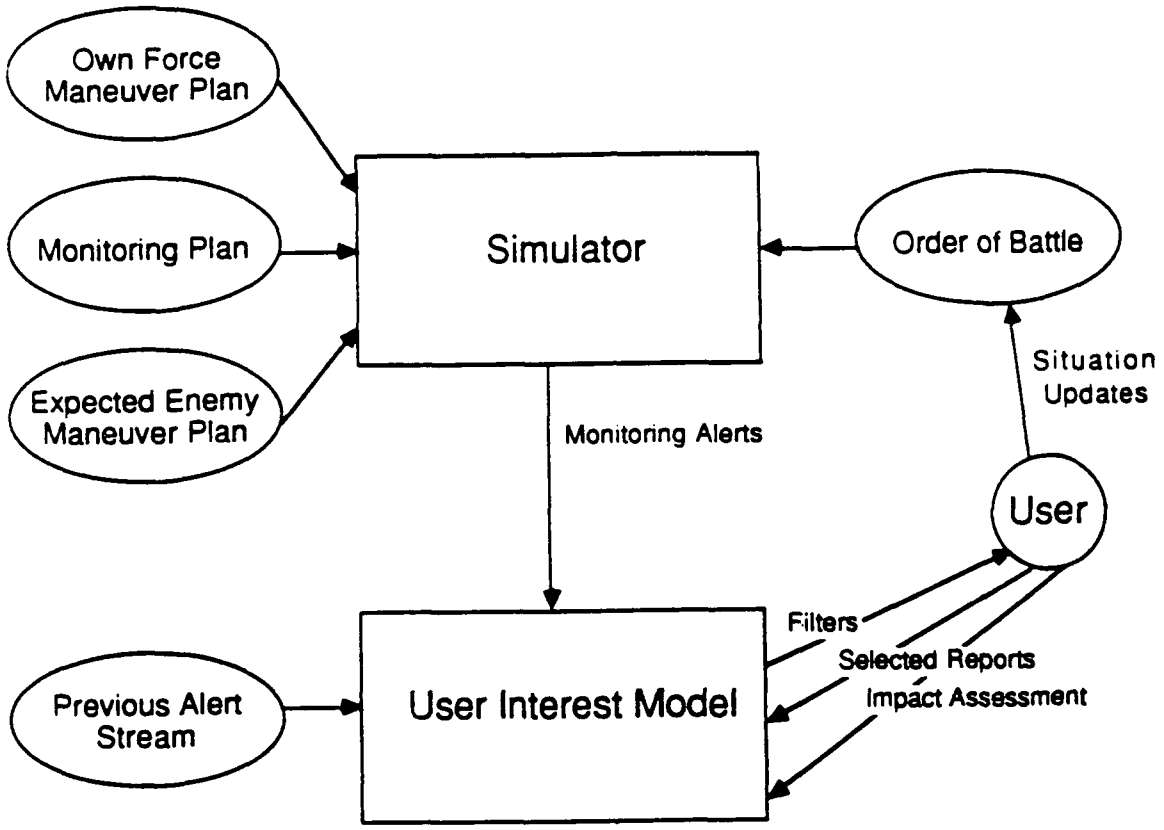


Figure 4-1: Conceptual Design of OMA

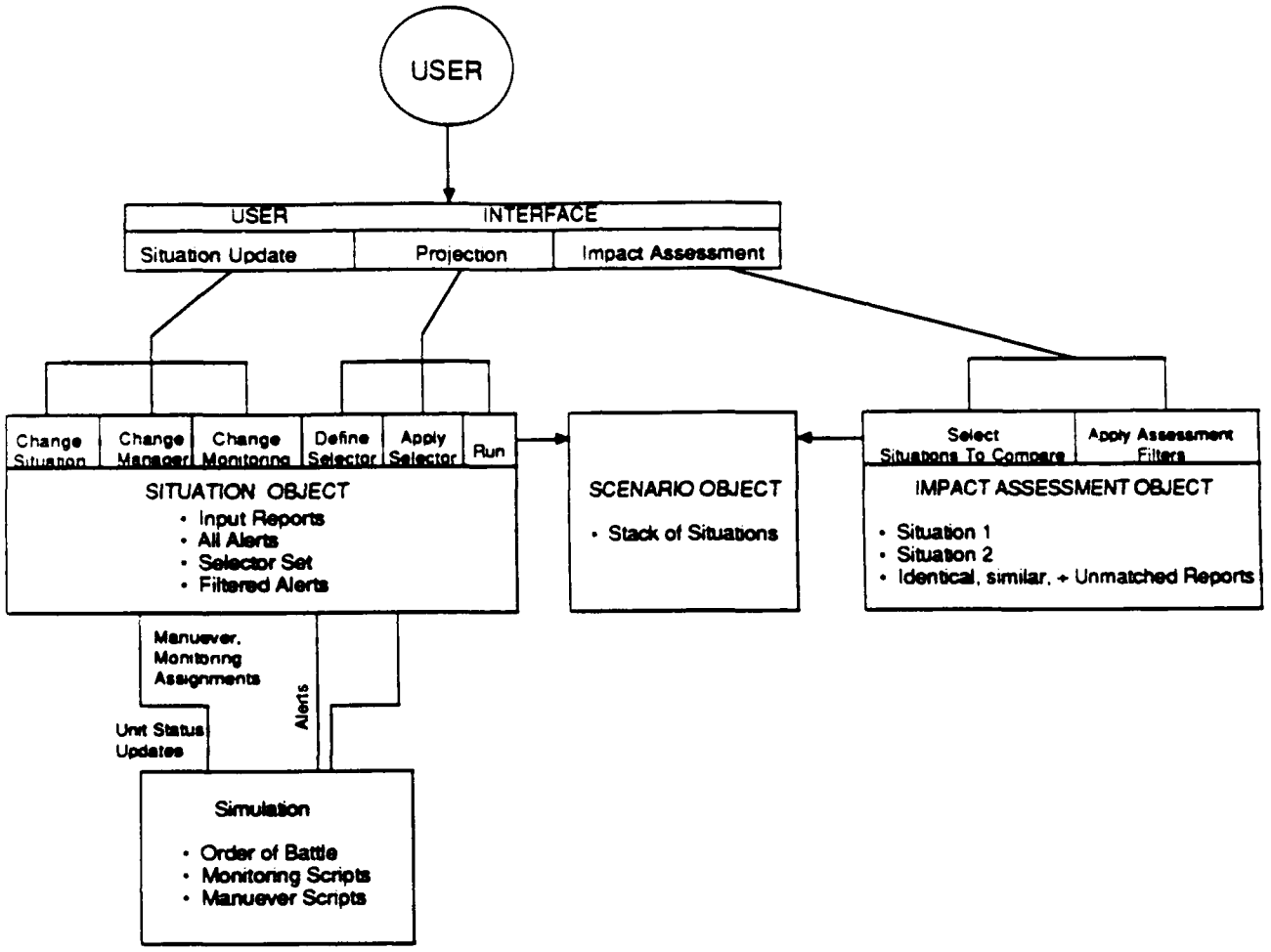


Figure 4-2: Architecture of the User Interface Component of OMA

restricted alert stream itself. The situation object has operations for modifying the order of battle, defining and applying selectors, and for invoking the simulator to generate monitoring reports.

Input reports modify data or select control structures used by the simulator. Owing to the simplicity of the simulation from a domain point of view, we support the following four input report types:

- unit location change,
- unit strength change,
- unit maneuver assignment (own force *and* enemy), and
- unit monitoring assignment.

The simulation does not represent consumable resources, the combat effect of headquarters functions, special weapons, or terrain (which is mutable at Corps and Division scale), so input reports for modifying this type of data are not required. Also note that the user is forced to make assumptions about enemy intentions so that the simulator can project the situation forward. These assumptions can be changed. The bulk of them are made at the time the overall scenario is defined, and only a few are generally modified during use of the monitor. The user enters reports via a graphical interface (clicking on units, dragging them, etc.), and the system automatically produces corresponding textual forms.

The user can also change the simulation's behavior by assigning predefined maneuver and monitoring scripts to named units. Since the simulation maintains the order of battle, the library of maneuver and monitoring scripts, and the goal stacks (script names) each unit has been directed to execute, the operations for manipulating them via the situation object are passed to functions within the simulator code. It is possible to change the actual maneuver and monitoring scripts, but not at run-time, since the task is a fairly intensive programming operation. It is accomplished through the use of lisp code, and a graphical editor built into the simulation system. The simulation system is discussed in section 4.4.6, while methods for changing scripts are discussed in the *hackers only* section of the Programmer's Manual [5].

The *run* operation on the situation abstraction invokes the simulator, which executes the maneuver and monitoring plans, producing status and alert messages which are kept in the situation object. As a second effect, the run operation places the new situation on the stack which defining the *scenario object*. After the simulation has been run twice, there are two fully instantiated situation objects (with monitoring alert streams). These can be compared via the impact assessment function.

The simulation currently produces the following types of alerts and status reports:

- strength reports,
- force ratio reports,
- closing reports,
- position reports,
- phase line reports, and
- key terrain reports.

The situation abstraction also provides operations for defining and applying *selectors*, which prune this alert stream to the most important messages. Selectors function by matching templates to reports. There is one template for each report type. An example of a strength selector was shown in section 4.1. An example force ratio selector is shown below:

```
sender: any
dtg-sent: any
force-ratio: (< .33)
```

Each of the fields has to match for a report to pass the selector. In this case, the reports must indicate a force ratio of 3:1 against the sender. The user can place any lisp form in the slots.

In OMA, all selectors are positive, meaning that they include matches in the final product. A simple modification would allow negative selectors, which would restrict the output of the positive selection functions. An example negative selector might say, "don't show me any Battalion force ratio reports, regardless of the positive selectors I have defined". (We omitted negative selectors because of the potential logical confusions.)

4.3.2 The Impact Assessment Abstraction

The impact assessment abstraction holds a comparison between two situations, *old* and *new*. This comparison is computed via a report by report matching function. It produces four categories of outputs:

- reports in the old stream but not in new,
- reports in new but not in old,
- reports in old which are similar to reports in new, and
- reports which are identical between old and new.

The issue behind these comparisons is that a situation update can both cause and prevent events from occurring. For example, moving a unit may cause a new engagement to take place, and simultaneously prevent an encounter which formerly occurred. The same reasoning applies at the level of Division consequences.

The impact assessment object provides predefined filters which allow the user to focus in on similarity, difference, or equality detections. The criteria which define these comparisons are defined in methods associated with each report type. Currently, strength reports are distinct if they come from different units, and are at least similar if the units are from the same. Strength reports are identical if they come from the same unit and are within 5 percentage points of eachother (time of receipt is not a factor). Reports that denote phase line crossings are identical if they refer to the same phase line and happen within 15 minutes of one another. They are similar if the same phase line is involved. The other report types have analogous comparison functions, which can be easily redefined.

Impact assessment is normally applied between a parent situation, and its immediate descendant, in order to analyze the effect of a single set of battlefield reports. It is worth noting that situations which are somewhat distant in time can also be compared, as long as the battlefield changes between them can be identified. For example, if reports at 0800 and 0900 hours distinguish situations A (in effect at 0700), B, and C, we can compare A and C by using *both* sets of reports as the input change. The result is a less specific attribution of cause and effect, but useful just the same. The deltas between remote situations can be computed by scanning the situations in the scenario object.

4.4 Script-based Simulation of Combat Operations

The simulation component of OMA is based on two tools implemented by BBN in support of the Air Land Battle Management Project; the Heuristic Combat Evaluator (HCE), and the Contingent Hierarchichal Script (CHS) language. CHS implements what we would call reactive scripts for directing agents, while HCE specializes those scripts to military maneuvers for Battalion level forces. This section discusses both systems, and provides examples of maneuver scripts.

Because maneuver directives for simulated units still specify tasks at a very high level (e.g. attack or defend), the combat simulator must have the ability to translate these descriptions into more detailed prescriptions for action. HCE accomplishes this by combining the use of CHS scripts with a knowledge base of Blue and Red doctrines.

Contingent Hierarchical Scripts are a knowledge representation which model behavior by capturing domain specific knowledge about goals and procedures. We will use the term *scripts* to denote collections of inter-related goals and procedures.

The need to make scripts contingent arises from the inability to determine *a priori* the situations which an agent will encounter. The agent's scripts must be able to react to different features of the current situation to determine the appropriate action.

One form of contingency in CHS scripts is the use of declarative knowledge to determine whether a script is appropriate in the current situation. The appropriateness of a script may be determined as a result of previous actions or the occurrence of events outside the domain of the script's control. This declarative knowledge is represented as a set of conditions which indicate that the goal has already been achieved or that the goal cannot be achieved in the current environment. In these cases, execution of the goal script will not occur.

Another form of contingency in CHS scripts is explicit representation of control flow. Execution follows different paths in the script depending on the outcomes of control conditions (contingencies). In addition, if a goal can be achieved in more than one way, the script can specify whether all alternatives are to be explored sequentially or in parallel.

CHS scripts are hierarchical in that a script for achieving a low-level goal can also be a sub-goal in a script for a higher-level goal. CHS provides a capability for representing goals, procedures that achieve goals, and the order in which the goals and procedures are to be achieved/executed.

Goals define an agent's objectives or desired states that can come into being through the execution of situation dependent procedures. Procedures prescribe actions and situation dependent contingencies intended to take an agent from an undesired state to a goal or desired state. Goals, in other words, indicate what to do and procedures describe how to do it.

4.4.1 The CHS language

Scripts are specified in the CHS language which is a high-level language for modelling behaviors and decision procedures. There is a textual form of the language which is used to store scripts on disk. This textual form is implemented via LISP macros and allows the user to create and modify scripts using a text editor. Programming in this textual form can be difficult because several lines of text are required to specify each node in a procedure. Thus, specifying a procedure may take as much as a page of text. Specifying all the subgoals and procedures related to a single top-level goal may therefore take several pages. It can be difficult to visualize the relationships of the various procedures and subgoals in a non-trivial network.

To alleviate the difficulties associated with the textual form, a visual form of the language is provided. The user can use the Procedure Editor to create and modify

scripts by means of interactive graphics. The Procedure Editor can load and save scripts in the textual form. Thus, the textual and visual forms of the language are completely compatible.

The following sections describe the CHS language and its interpretation in high level terms. We have found that the task of modifying CHS scripts requires a fairly detailed understanding which is beyond the scope of this document. We provide an additional layer of explanation detail in [5].

4.4.2 CHS Definitions

A CHS script consists of *goals* and *procedures*. Goals define an agent's objectives or desired states that can come into being through the execution of situation dependent procedures. Procedures prescribe actions and situation dependent contingencies intended to take an agent from an undesired state to a goal or desired state.

Goals are organized into hierarchies; top-level goals (i.e. goals which are not subgoals of a higher level goal) are the root nodes of the goal hierarchy. Goal nodes have children nodes which specify how the goal may be achieved. The children nodes can be other goal nodes, AND/OR nodes, or action nodes. Action nodes may invoke CHS procedures, LISP code or initiate activities. Action nodes are leaf nodes in the goal hierarchy; they have no children nodes.

The achievement of a goal is defined in terms of the achievement of its subgoals. The relationship between a goal and its subgoals is expressed by AND/OR links.

Procedures are represented as networks of nodes. The nodes which can be used to construct a procedure are: start nodes, control nodes, wait nodes, outcomes nodes, action nodes, spawn nodes and end nodes.

- **Start nodes** are place-holders for indexing the beginning of scripts.

- **Control nodes** have two or more branches leading to different execution paths. There is a condition predicate for each branch that, if true at the time the control node is reached, indicates that that branch should be followed.

- **Wait nodes** allow cause execution of the procedure to be suspended either for a specified amount of time or until a condition is true. An outcomes node causes control to pass to different execution paths, depending on the outcome of an event.

- **Action nodes** There are four kinds of action nodes: procedure nodes, immediate action nodes, movement nodes and duration nodes. Procedure

nodes invoke procedures. Immediate action nodes invoke LISP functions. Movement nodes initiate activities (actions with duration) that move simulation objects. Duration nodes initiate activities (actions with duration) which do not move simulation objects.

- **Spawn nodes** create a new top-level goal. The new goal can be spawned in *serial* or in *parallel*. If it is spawned in serial, the spawning goal is suspended until the spawned goal terminates. If the new goal is spawned in parallel, the spawning goal continues execution without waiting for the spawned goal to terminate.

- **End nodes** signal the end of a script. There are three kinds of end nodes: exit nodes, termination nodes, and suspend nodes. Termination nodes and exit nodes end the execution of a script, signalling respectively failure and success. Suspend nodes suspend execution of the parent goal until a condition is met.

4.4.3 Control of Simulation in CHS

HCE is an application of CHS, an object-oriented, event-driven simulation framework. The central organizing principle of CHS is that of agents which perform activities in pursuit of a set of goals. These activities are modelled via an event-based simulation.

A simulation consists of a set of state variables which take on different values over time. In tick-based simulations, the state variables are updated at regular time intervals (called *ticks*). In event-based simulations, the state variables are updated at irregular time intervals (called *events*).

The event-driven temporal paradigm speeds up the simulation by not spending time on uninteresting portions of the battle. For example, the interesting portion of a road march is when it ends (either by reaching the objective or by being engaged enroute). The interesting portions of a combat engagement are when it terminates and/or when other forces join in.

The object-oriented representational paradigm is well-suited for the ALBM domain. Each military unit corresponds to an object. Each object invokes its methods (functions) on its local knowledge in response to stimuli (messages). Benefits of the object-oriented approach to simulation include ease of understanding (e.g. simplicity, clean mappings of real-world objects to programming constructs), modularity (including inheritance), flexibility, and ease of capturing behavior causality.

In an object-oriented simulation, the state variables are represented as slots of objects. Because the majority of code in an object-oriented system consists of procedural

attachments to data structures, it is not appropriate to present a traditional architecture diagram of showing control and data flow between code modules. For instance, the top-level loop of the CHS simulator simply tells each agent to process its unaccomplished goals. Each agent, in turn, loops through its unaccomplished goals and tells each goal to execute the next node. Each goal executes nodes until it is *blocked.* This process continues until one of the termination conditions becomes true or until all agents have accomplished their goals.

Figure 4-3 shows the three queues used by the CHS simulation engine. The *Active Agents Queue* is a list of agents who have unaccomplished goals. The *Scheduled Activities Queue* is a list of all current activities. The *Scheduled Events Queue* is a list of all outstanding events.

As Figure 4-3 shows, the *Active Agents Queue* hold units of different echelons (e.g., corps, division, brigade). Each agent manages three queues. The *Current Goals* queue is a list of goals which the agent is currently trying to accomplish. The *Current Activities Queue* is a list of activities that the agent is currently engaged in. The *Blocked Events Queue* is a list of *events* that are *blocked.* Each blocked event has a *blocking condition* associated with it. When the blocking condition becomes true, the event is no longer blocked.

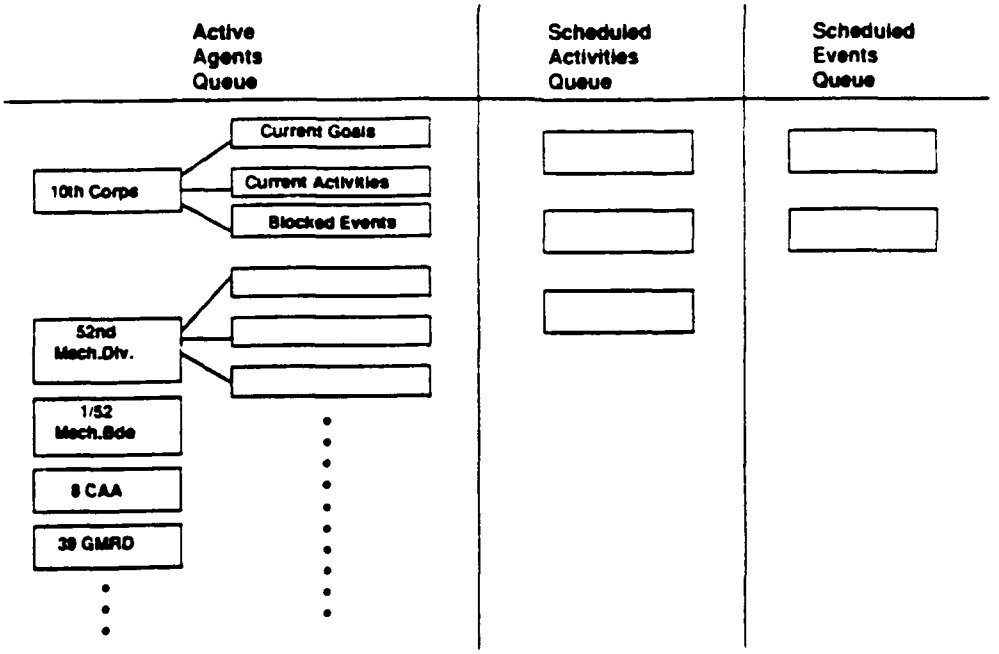


Figure 4-3: Simulation Queues

4.4.4 Example Scripts for Maneuver Plans

In this section, we present examples of HCE tactical scripts. Figure 4-4 and Figure 4-5 show scripts for Defense and Attack, respectively. These scripts are executed in the course of simulating an engagement between two opposing units. In this example, there is a Blue battalion with a goal of Defense Task, and a Red regiment with a goal of Attack Task. In the OMA defensive scenario, this situation arises repeatedly as various regiments of the 8th CAA attack battalions of 10th (US) Corps.

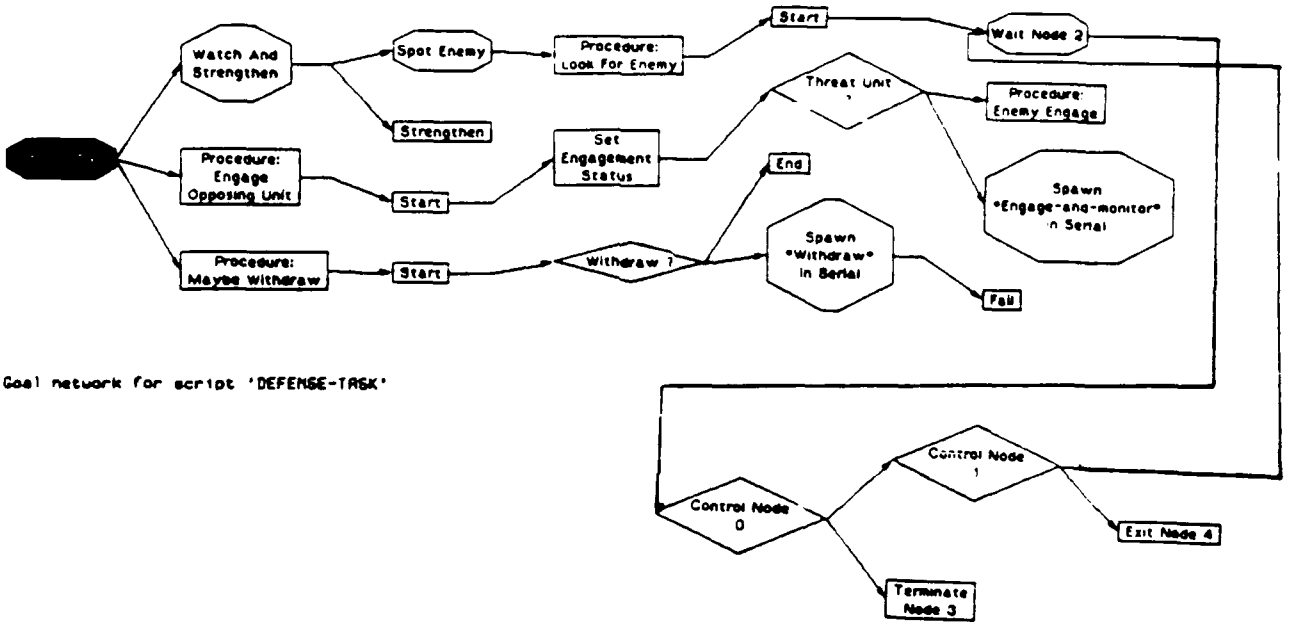
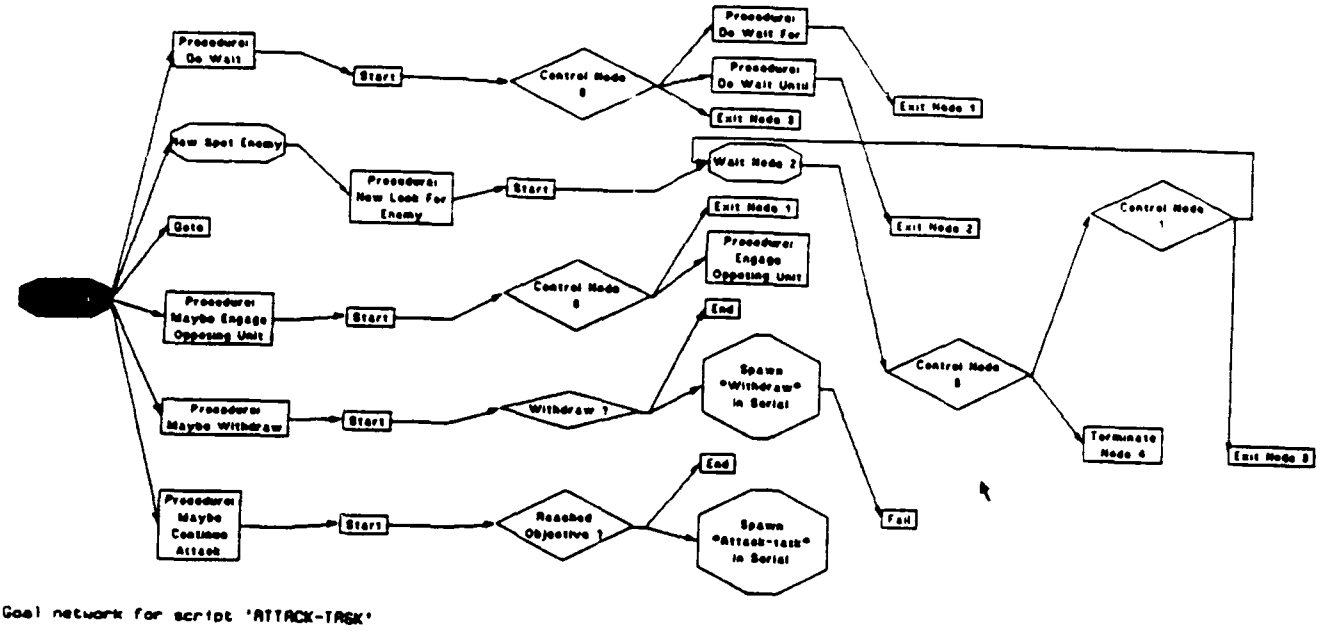


Figure 4-4: CHS Script for Defense

What does a defend mission mean to a unit? Basically, it means that the unit should watch for any approaching enemies, while fortifying its position. If an enemy unit enters the defensive perimeter, the defending unit should engage it. Depending on the outcome of the battle, either the defending unit has completed a successful defense by repelling the attackers, or it will have to withdraw after suffering heavy losses. This knowledge is represented in HCE in the tactical script *Defense Task* (Figure 4-4).

Scripts are displayed as a network of nodes. Hexagonal nodes indicate goals and wait nodes. Diamond-shaped nodes indicate control and outcome nodes. Rectangular nodes indicate procedure nodes, start nodes, action nodes, exit nodes, and terminate nodes. Control flows in the direction of the arrows, generally left to right. When a control node is reached, a LISP predicate function decides which path to follow next. The encapsulating procedure returns when an end node is reached.

The *Defense Task* goal (root goal of the *Defense Task* script) is achieved by achieving



Goal network for script 'ATTACK-TASK'

Figure 4-5: CHS Script for Attack

three subgoals: the goal *Watch And Strengthen*, the procedure *Engage Opposing Unit*, and the procedure *Maybe Withdraw*. In other words, for the *Defense Task* script, the root goal and its first level expansion consist of the four leftmost nodes of Figure 4-4. The CHS Procedure Editor shows additional expansion levels as a convenience to the user.

The root goal succeeds only when all of its subgoals have been achieved. CHS allows the user to specify ordering constraints and AND/OR links among subgoals to ensure that a goal is achieved under the correct conditions. The CHS Procedure Editor can display these constraints as the Precedence Graph for a script. Figures 4-6 and 4-7 show the precedence views of the *Defense Task* and Attack Task goal scripts, respectively. In these views, dashed lines represent AND/OR links, while solid lines indicate temporal precedence. Thus, Figure 4-6 shows that a successful *Defense Task* would consist of first successfully completing *Watch And Strengthen*, then Engage Opposing Unit, and finally *Maybe Withdraw*. The information contained in the Precedence Graph for a script is crucial to a meaningful simulation.

It should be noted that many different units may have a *Defense Task* goal simultaneously. CHS avoids redundant code and excessive storage requirements by having each unit carry a binding environment for the script parameters, and also keep track internally of where it is in the script. For instance, in this *Defense Task* example, the Who parameter would be bound to the name of the Blue battalion (e.g. :FRIENDLY-GENERIC-TF-BATTALION833).

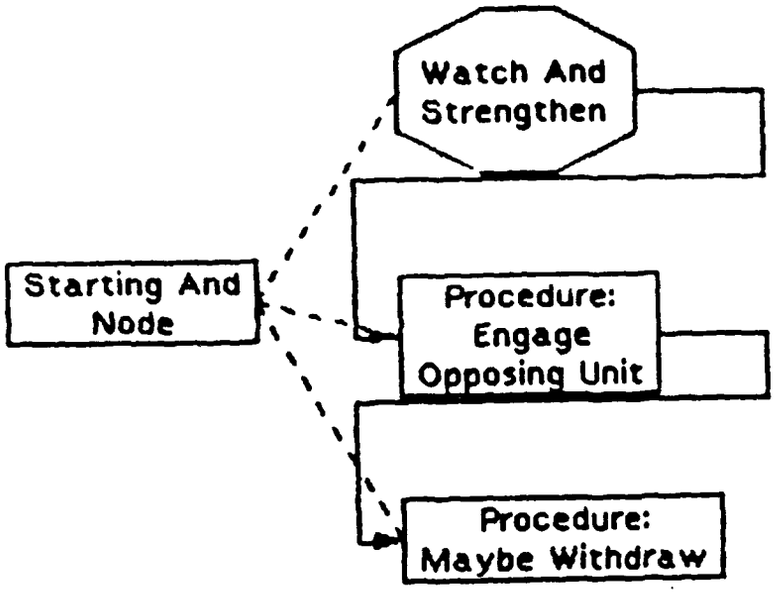


Figure 4-6: CHS Precedence Graph for Defense

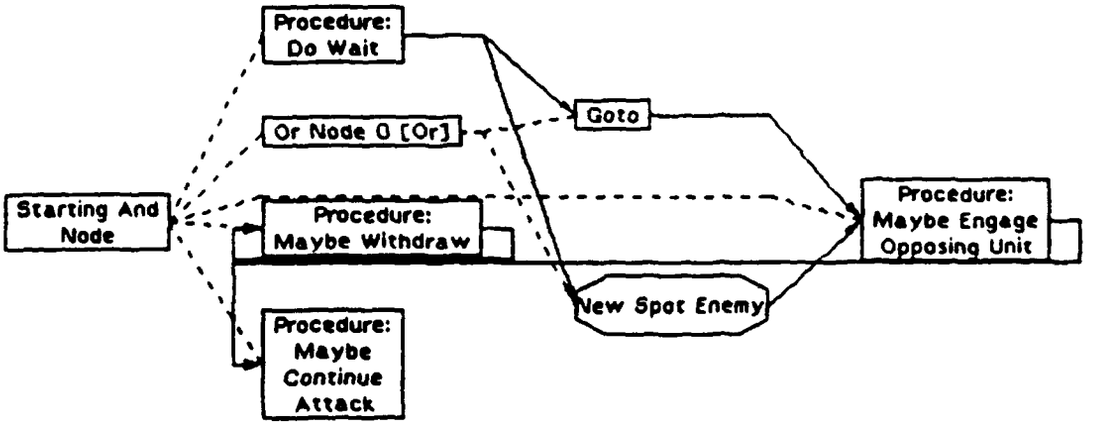


Figure 4-7: CHS Precedence Graph for Attack

4.4.5 A List of HCE Scripts

The previous section has shown how military domain knowledge may be implemented in Contingent Hierarchical Scripts. In actuality, HCE contains approximately 50 distinct scripts, which we list by name in table 4-1.

Table 4-1: HCE Maneuver Scripts

- 1ST-ECHELON-ATTACK
- 2ND-ECHELON-ATTACK
- AIRBORNE-ATTACK-TASK
- APPROACH-TO-CONTACT-TASK
- ARTY-ATTACK
- ARTY-ATTACK-TASK
- ARTY-MOVE-AND-FIRE
- ARTY-MOVE-AND-FIRE-ORDERS
- ARTY-MOVE-AND-FIRE-TASK
- ASSIGN-TASK
- ASSIGN-UNIT-TO
- ATTACH-TASK
- ATTACK-SEQUENCE
- ATTACK-TASK
- ATTACK-W-ARTY-TASK
- ATTACK-WITH-ARTY
- ATTACK-WITH-ARTY-SEQUENCE
- COUNTER-ATTACK
- COUNTER-BATTERY-TASK
- COUNTERATTACK-TASK
- DEASSIGN-TASK
- DEASSIGN-UNIT-FROM
- DEFEND-ORDERS
- DEFENSE-SEQUENCE
- DEFENSE-TASK
- DELAY-G
- DELAY-TASK
- DETACH-TASK
- DIRECT-SUPPORT-TASK
- END-ARTY
- ENGAGE-AND-MONITOR
- GO-TO-OBJECTIVE
- GO-TO-WITH-SPOT
- GUARD-TASK
- INDIRECT-FIRES-TASK
- LEAD-ATTACK
- LEAD-DEFENSE
- MBA-FORCE
- MOVEMENT-TO-CONTACT-TASK
- NEW-SPOT-ENEMY
- NON-TASK
- ROAD-MARCH-TASK
- SCREEN-TASK
- SECURITY-FORCE
- SECURITY-FORCE-ORDERS
- SPAWN-ARTY-ATTACK
- SPOT-ENEMY
- START-ARTY
- SUPPORTING-TASK
- SURVIVE-ARTY-ATTACK
- TASK
- WATCH-AND-STRENGTHEN
- WITHDRAW-TASK

4.4.6 A Simulation Example

This section discusses the flow of control within HCE at the simulation level - how HCE synchronizes the interactions of multiple agents and scripts. We use the same scenario referenced in section 4.4.4.

Figure 4-8 shows the internal state of the simulator at the start of the scenario (0200

HRS). The Blue battalion has *Defense Task* as its top-level goal while the Blue battalion has *Attack Task* as its top-level goal. There are no scheduled activities or events.

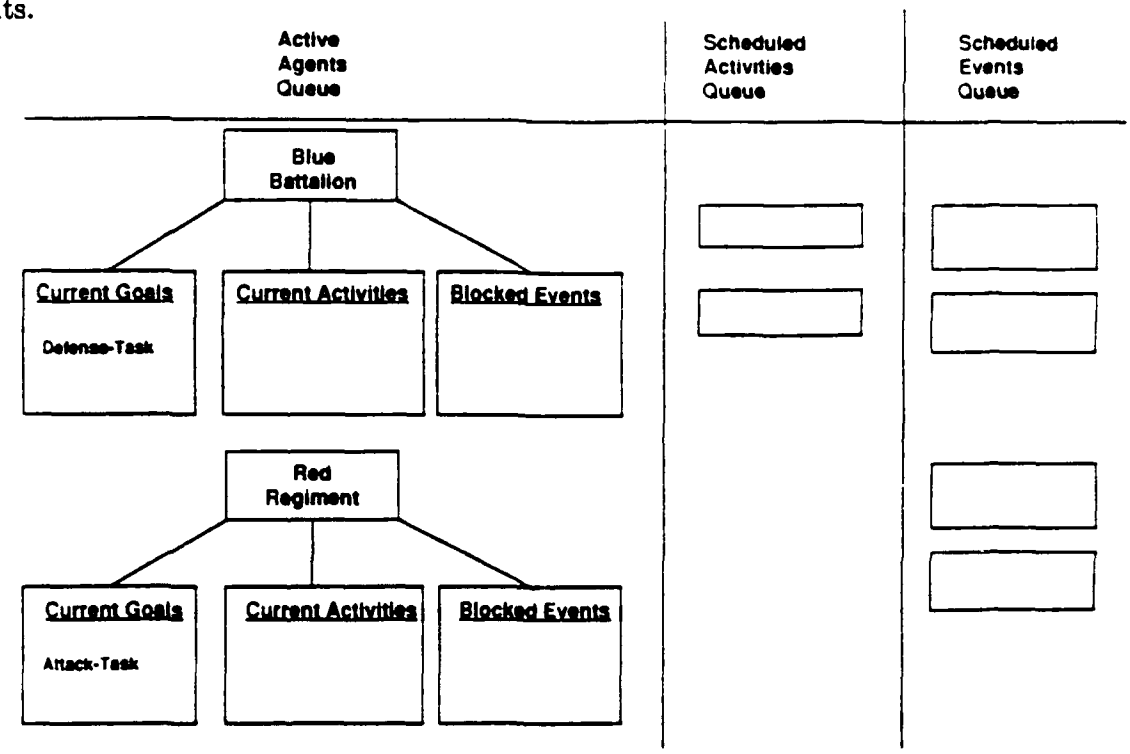


Figure 4-8: Simulation Example (1) 0200 Hrs.

Figure 4-9 shows the internal state of the simulator after the top-level goals have been expanded. Note that the simulator clock does not advance. Scripts model the cognitive process of the unit commander and are assumed to take an insignificant amount of time. The Blue battalion's *Defense Task* has been expanded into three subgoals: *Watch And Strengthen*, *Engage Opposing Unit* and *Maybe Withdraw*. *Engage Opposing Unit* and *Maybe Withdraw* are *blocked* because they cannot start until *Watch And Strengthen* has completed. *Watch And Strengthen* in turn has been expanded into *Spot Enemy* and *Look for Enemy*.

Figure 4-10 shows the internal state of the simulator after the units engage each other. The Blue battalion's subgoal of *Watch And Strengthen* has completed. It is currently executing *Engage Opposing Unit*. The goal *Maybe Withdraw* remains blocked because it cannot start until *Engage Opposing Unit* has completed. The Red battalion's goals are in a similar state. Note that both units share the same activity **Defend513**. This occurs because there is a protocol that, when units engage in combat, there is only one combat activity. Also note that the end of activity **Defend513** has been estimated to be at 1100 Hrs. This *event* has been placed on the *Scheduled Events Queue*.

Figure 4-11 shows the internal state of the simulator after the end of the engagement.

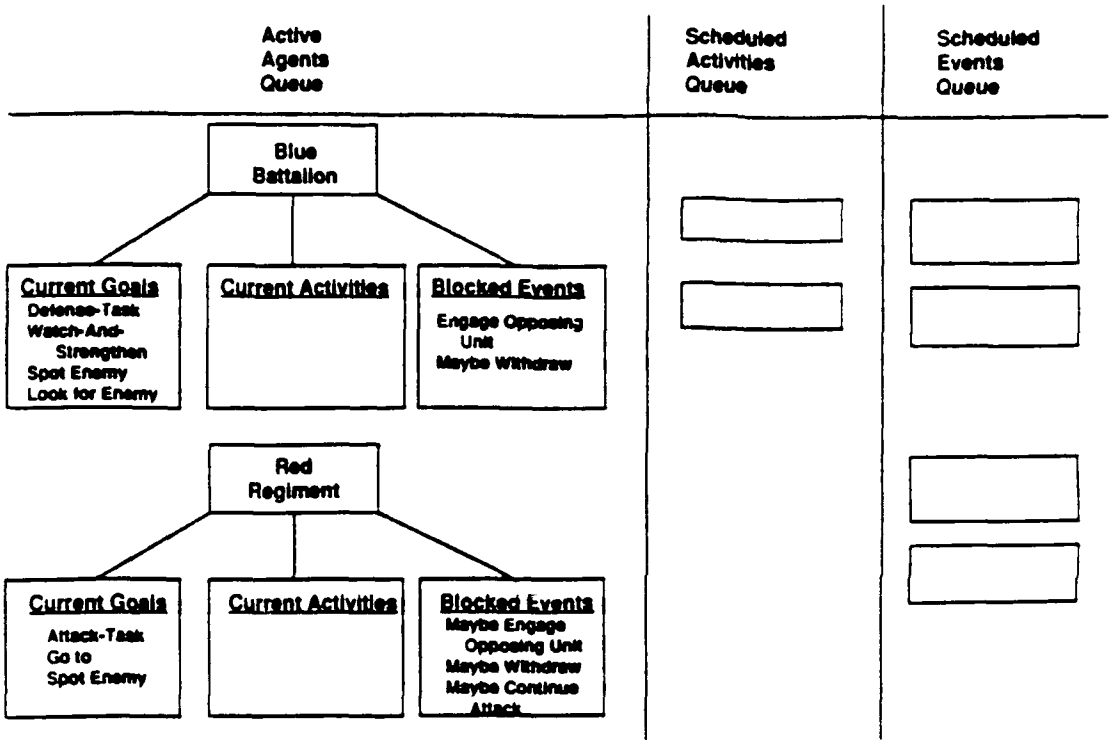


Figure 4-9: Simulation Example (2) 0200 Hrs.

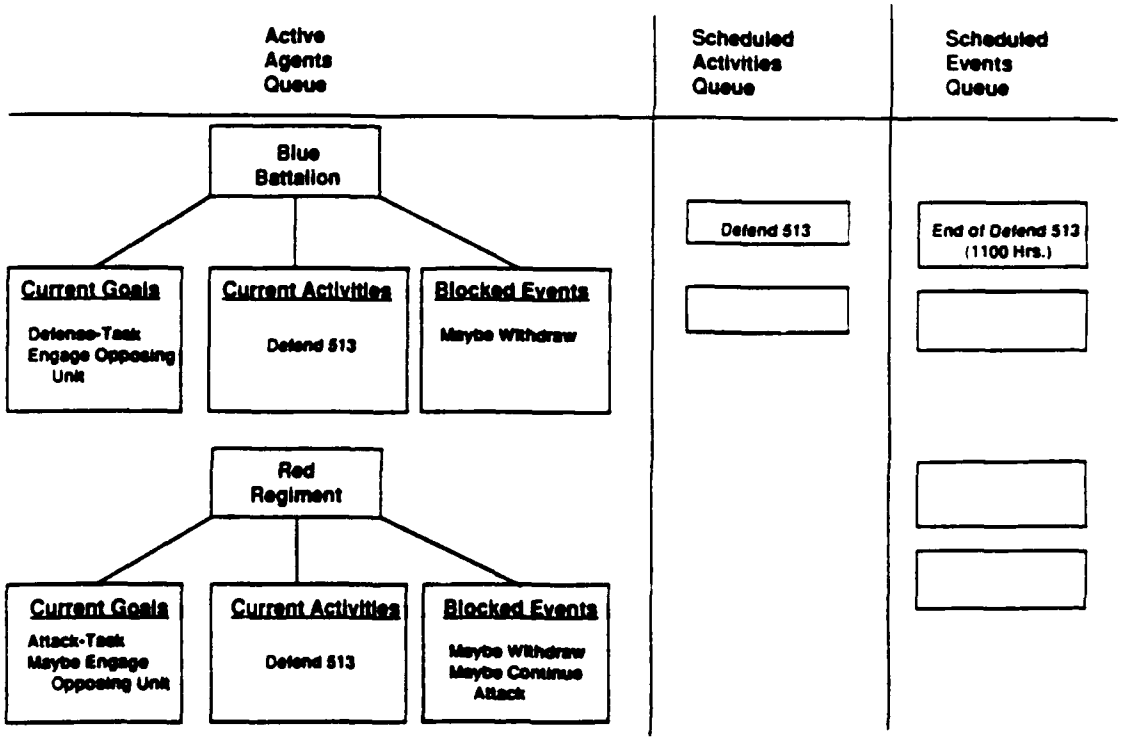


Figure 4-10: Simulation Example (3) 0500 Hrs.

The Blue battalion's subgoal of *Engage Opposing Unit* has completed. It is currently executing *Maybe Withdraw.* The Red battalion's goals are in a similar state. It is now executing *Maybe Continue Attack.*

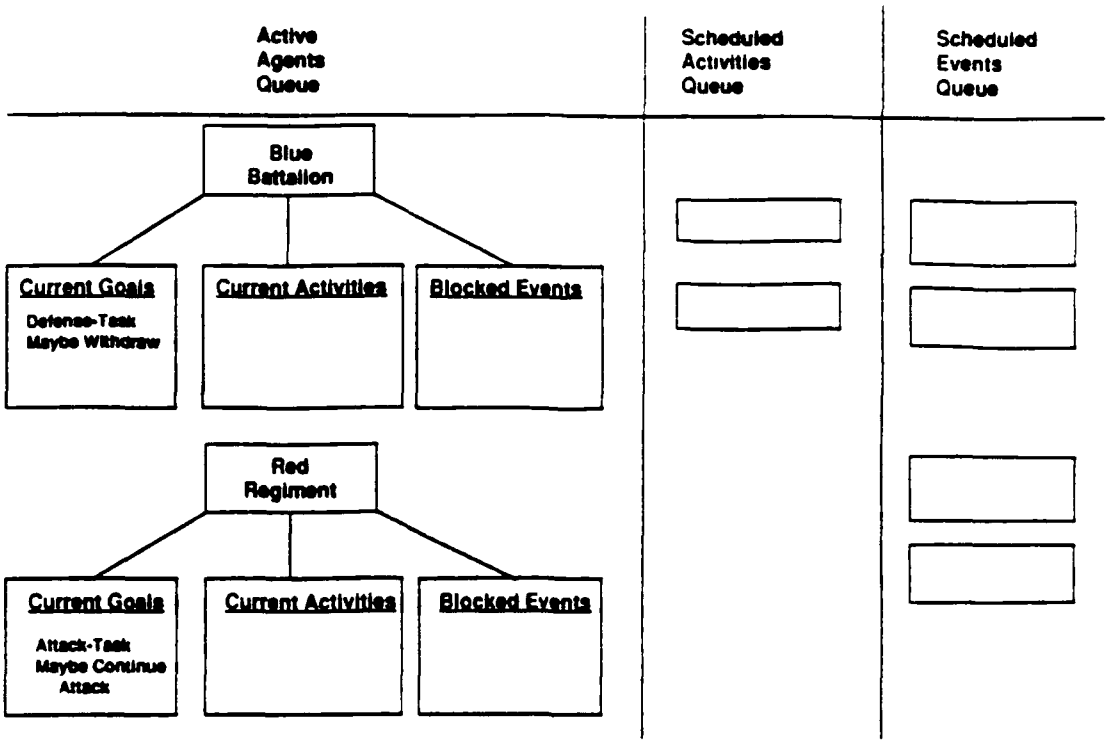


Figure 4-11: Simulation Example (4) 1100 Hrs.

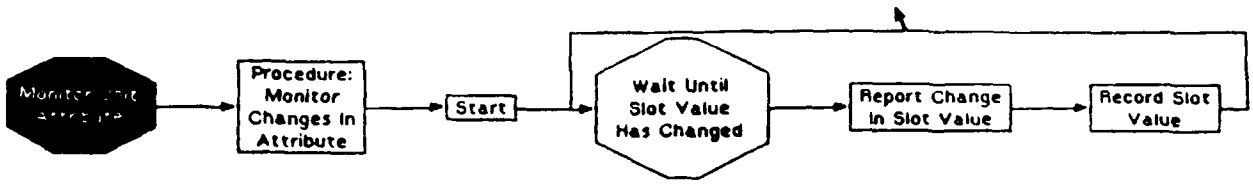
4.5 Representation of The Monitoring Plan

The monitoring plan is represented with *HCE* scripts, similar to the maneuver plans discussed above. At the current time, OMA contains the following monitoring scripts:

- monitor unit attribute,
- monitor engagement,
- monitor phase line, and
- monitor key terrain.

The attribute monitor is defined to sense changes in any unit attribute (currently, the simulation only knows about location and strength). It operates by repeatedly testing the attribute against a predefined value. The script for monitor attribute is shown in figure 4-12.

Evaluation of this procedure is straightforward, with the change detection and alert generation procedures being executed in a continuous loop. The lisp code which actually performs the test and generates the report is called as each box is entered. The name of the attribute to monitor, and the criteria for sending an alert are both passed in as parameters when this plan is invoked.



Monitoring scripts can be aggregated in the same way as maneuver plans. So, for example, the script for *monitor operations* has monitor strength, monitor phase lines, and monitor engagement as subgoals. All of these are executed in parallel.

Figure 4-12: The Monitor Attribute Procedure

5. Future Work

We see several directions for future work on OMA:

- build a more robust version of the system and evaluate it in the military,
- augment OMA with an explanation facility (at the prototype level), and
- incorporate OMA into a plan generation and evaluation system (ALBM).

In order to build a more robust version of OMA, many of the features of the prototype implementation will have to be refined. Obvious tasks are:

- build more monitoring scripts,
- enhance the simulator's speed,
- incorporate better decision functions for Brigade and Division level units (for *example, to allocate reserves), and*
- expand the simulation to include terrain, maneuver modes (hasty/deliberate attack), consumable resources, combat headquarters functions, etc.

It is important to remember that the OMA simulation is intentionally coarse, as a compromise between the need for accuracy and execution speed. The simulation should not be enhanced to the point where it becomes non-functional for OMA.

With these enhancements, it would make sense to take OMA and evaluate it in a military command, such as Ft. Lewis or Ft. Hood. We envision a year long development period followed by a 6 month cooperative effort with military users. The purpose would be to gain feedback for use in building an operational system.

Our experience with influence nets suggests they can be used to add an explanation facility to OMA. As discussed in section I.2, an influence net can be constructed as a byproduct of simulation. Once built, it provides a cogent analysis of the relationships between events the simulation believes will occur. Our approach to influence nets also supports a limited form of projection. This might be used to identify best and worst case intervention by the enemy.

The option to employ OMA within ALBM is already being explored, although in a limited fashion. OMA is already compatible with the ALBM code; it is built on the same support tools (CHS, HCE, and terrain), and it accepts the plan representation generated by ALBM as input. One obvious task is to employ the monitor to evaluate plans, adding functionality for evaluating/ranking the severity of the monitoring alerts

so that ALBM can compare courses of action. ALBM also has a directive to perform execution monitoring, which is exactly the task OMA solves. There is clearly a great deal to be gained by transfer of the technology.

I. Approaches NOT Taken

We explored several approaches in the course of the project which were ultimately not a part of our final solution. We document these below.

I.1 Monitoring as an Inference Process

The first approach we considered viewed monitoring as an inference process, which input a situation description, a plan, and a new event, and inferred the expected results, without simulation. See our Phase I final report for details [2].

Figure I-1 shows the scenario we examined in our early work. Here, the corps commander has chosen to organize his defense around an air terminal and a railhead. While holding a line with airborne infantry, the commander has begun a counteroffensive using air power to interdict enemy supplies, and a reserve armored infantry division in counterattack, to flank and ultimately envelop enemy positions.

The hypothetical input report is that the attack helicopter battalion countering a red tank regiment threat to the railhead incurs heavy losses. We proposed the following deduction sequence:

- The attack helicopter battalion will soon be incapable of stopping the enemy advance.
- The red tank regiment will have an uncountered threat to the railhead.
- If the railhead is threatened, the supplies for the armored infantry division are threatened.
- If the supplies are cut, the division's objective of isolating and enveloping the enemy is in doubt.
- If the division cannot isolate and envelope the enemy, the corps plan as a whole might not be achieved.

This inference process relied on a tactical understanding of the battlefield in terms of threats, counter-threats, and their effects, and on a view of the plan as a sequence of actions (with dependencies) designed to produce a named situation. Given this analysis, the absence of a counterthreat was presumed to allow the threat to complete, its effects to be recorded (e.g., destruction of the railhead), and the inference chain to continue, as above.

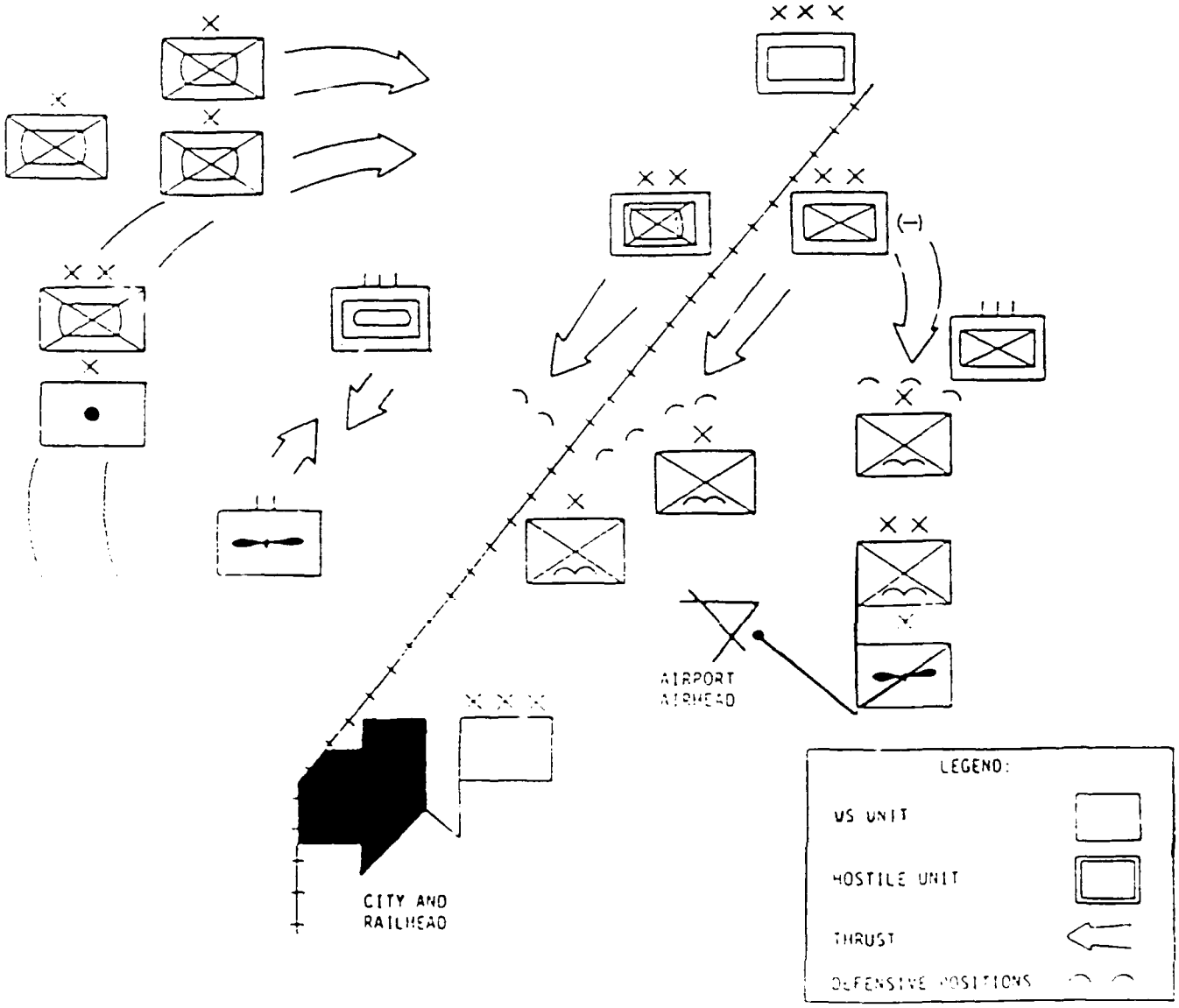


Figure I-1: Scenario from the Phase II Proposal

I.1.1 Analysis

Monitoring by inference has several desired properties, specifically that it is (presumably) faster than simulation, and provides a direct route to high level feedback about the plan. It has the undesirable property, however, that its results are probably less believable. In the process as we described it, time and distance were only poorly modelled, and there were no formal combat models (effecting believability). We also required a known sequence of actions, which may be reasonable for offensive plans, but is inadequate in the case of defensive situations, where the enemy threat itself is not known.

Our net feeling is that inference and simulation are dual methods of predicting the future, and that successful monitoring functions can be cast in either view. The appropriate technique depends upon the questions asked - abstract impact assessment (at the level of net plan feasibility) may be approachable via inference, but combat maneuvers are a more fine grained phenomena; simulation methods may be required for accurate prediction.

The use of tactical analyses to support monitoring, and/or explanation of monitoring results is a worthwhile future direction.

I.2 Projection Through a Network Describing the Maneuver Plan

In our second semi-annual report [6], we discussed a design for a projection system that would operate by propagating influences through a network of actions describing the plan.

Figure I-2 shows an example counterattack plan, here for the 10th CAB to neutralize expected attacks by 3 red regiments on elements of the US 52nd, 53rd and 54th Mechanized Divisions. This plan depends critically on the stamina of the 10th CAB, and will be effected adversely by any decrease in their capability. We would like to project the consequences of any such change.

Figure I-3 provides a influence net for the counterattack plan shown in figure I-2, with the addition of a Spetznaz attack on 10 CAB fuel, after the 10th CAB has attacked the 55th TR. The monitor employs this net by propagating values for unit *strength*, *fuel*, and action *degree of performance* across links which represent the *effects* and *dependencies* between actions. For example, the consequences of the Spetznaz attack are as follows:

- 10 CAB fuel supplies reduced to low level
- 10th CAB attack on 115 TR will be weak

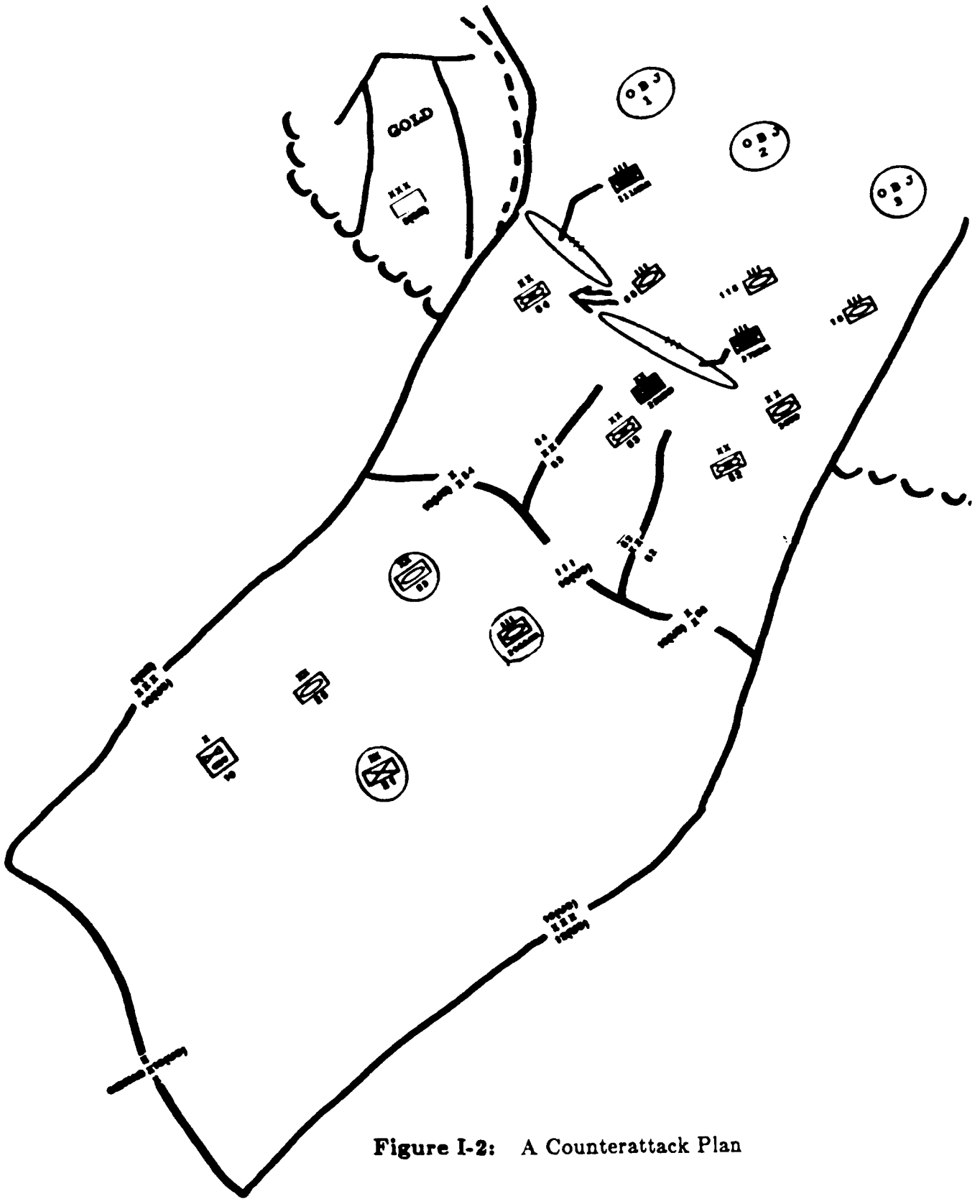


Figure I-2: A Counterattack Plan

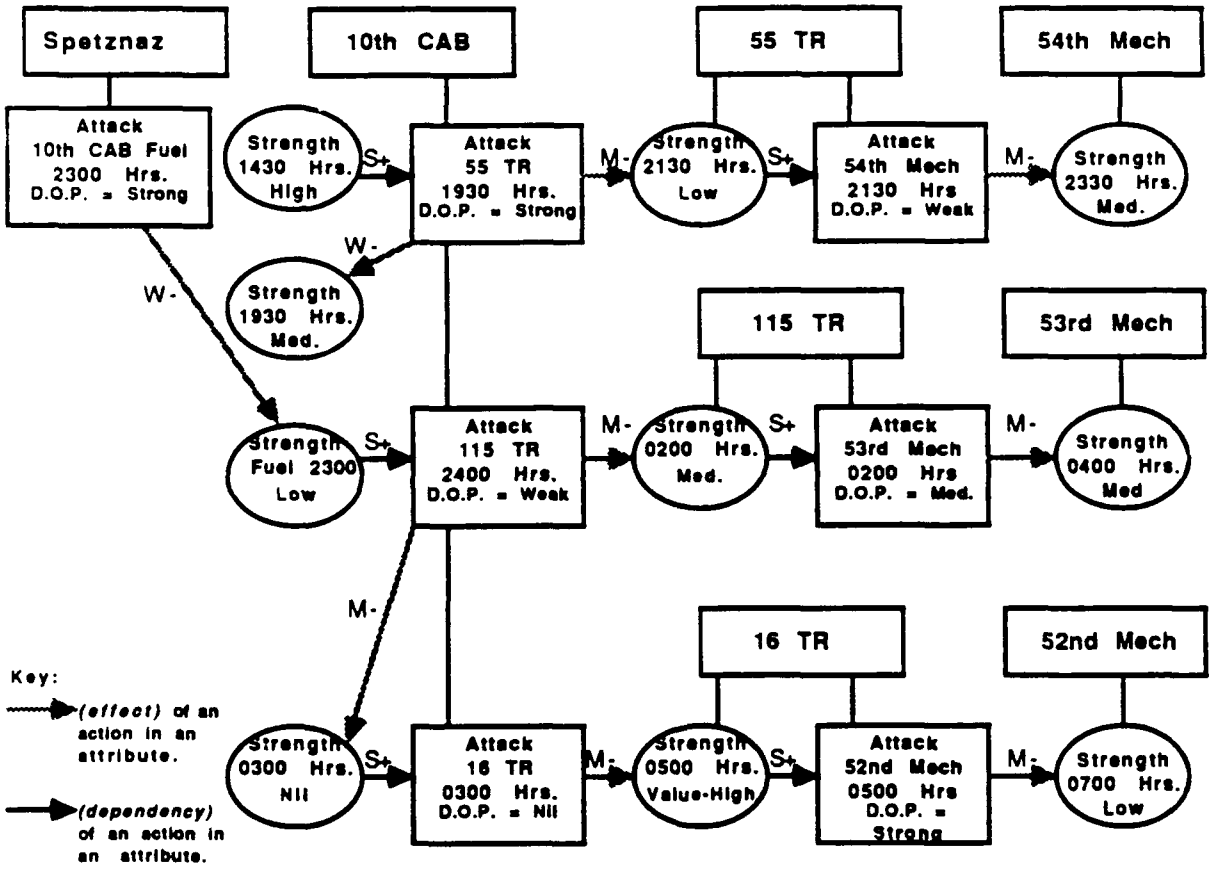


Figure I-3: An Influence Net for a Spetznaz Attack and the Counterattack Plan

- 10th CAB attack on 16 TR will be very weak (nil)
- Strength of 115 TR after 10 CAB attack will be medium
- Strength of 16 TR after 10 CAB attack will be high
- Degree of performance of 115 TR attack on 53rd Mech will be medium
- Strength of 53rd Mech will be medium.
- 16 TR attack on 52nd Mech will be strong.
- Strength of 52 Mech will be low.

We defined calculi for managing these propagation steps in terms of simple array manipulations.

For example, the propagation table for computing *action degree of performance* from *dependency strength* and *attribute value* is as follows:

Value of Attribute \ Strength of Dependency	Weak	Moderate	Strong
Low	Not at all	Not at all	Weak
Medium	Not at all	Weak	Moderate
High	Weak	Moderate	Strong

This table assumes that there is such a thing as an inherent dependency of an action on a parameter, such as strength (it seems reasonable that the efficacy of an attack is strongly dependent on unit strength, but not quite so clear in the case of fuel). An example use of this calculus is shown below:

The Level of Performance of the Movement action
depends on the Fuel attribute *strongly*.
Value of the Fuel attribute is *medium*.
Implied level of performance is *medium*.

A similar table was defined for propagating action degree of performance across *effect* links onto new values for unit attributes:

Performance Level of Action \ Inherent Ability to Achieve Effect	Weak	Moderate	Strong

Weak	Not at all	Not at all	Weak
Moderate	Not at all	Weak	Moderate
Strong	Weak	Moderate	Strong

The entries in this table should be interpreted as deltas to the previous parameter value. For example,

Example:

The Inherent Ability of a Brigade Attack
to reduce a Division's strength is *weak*.
If the attack level of performance is *moderate*.
Then the actual effect will be *none*.
The Division strength will not change.

The notion of an inherent effect is meant to encompass a variety of factors, such as relative combat power, terrain, and weather effects. This quantity had to be set manually for each action in the plan. Mechanisms for resolving multiple effects and dependencies were discussed in [6].

With extension of these ideas we hoped to produce mission level feedback from the operation's monitor. For example, we expected output like the following:

Initial Spot Report:

10th CAB reports:

Spetznaz unit has attacked fuel supplies
Fuel supplies reduced to low level
Attack capability reduced to low

... as above ...

53rd Mech attack on 57 MRR will be weak
52nd Mech attack will be weak

Strength of 57th MRR will be strong

Strength of 7 TD will be strong
Strength of 1 GTA will be moderate

```
Likelihood of the 53rd Mech successfully
completing its mission will be weak

Likelihood of the 52nd Mech successfully
completing its mission will be weak

Likelihood of the 10th Corps successfully
completing its mission will be weak
```

Output of this form is clearly interesting to the 10th Corp G3.

I.2.1 Analysis

Oddly enough, our analysis of the network propagation model is that it is useful, but for explanation as opposed to projection. The difficulty is that the composition of the network has to be determined in advance of propagation, since the network itself has no representation of the battlefield, of unit locations, or of a unit's intent which can be resolved into action selections. I.e., each network describes the *results*, or *output* of a battle simulation, and cannot substitute for the simulation itself.

In its role as an explanation facility, a network like the above can easily be used to record data dependencies during simulation, and therefore provide a model of causal relationships between new events and their consequences (e.g., a spetznaz attack an a corps level mission failure). Analysis of the data flow, coupled with propagation functions like the above might be used to identify critical actions which would produce the largest benefit by application of additional resources. Symmetrically, one could plausibly identify actions which provide the enemy with the best opportunity for interference.

We have suggested exploration of these avenues in the section on future work.

I.3 Providing Reactive Behavior via Higraphs

Within the solution approach that implements monitoring by instrumenting a simulation, several technical problems have to be solved.

- Mission statements (which are typically abstract) have to be refined into executable plans.

- Units must be given the intelligence to react to situations as they develop during simulation.

- Since nothing ever works according to a plan, units must be able to react at

variance with the plan (for example, when they are driven into defense when the mission says, "attack").

These issues are addressed by many combat simulation systems, generally via the following techniques:

1. relying on a person in the loop (to make decisions for the blue side for example),

2. including a large number of control bits that specify unit reactions in an expected set of predefined situations, or by

3. defining scripts which (without fail) will be executed in the current context.

Since the task of monitoring seems to involve exploration of many non-planned scenarios, we felt it was inappropriate to pursue approaches which were substantially hard wired, or which required significant user interaction. As a result, we investigated a mechanism for providing what we termed "reactive" unit behavior.

An article by Harel [1] was influential in our thinking. In it, plans are expressed as higraphs, which can be viewed as hierarchical finite state machines with parallel states, where the state machine can employ process control primitives (to start, stop, resume or suspend an action). We applied this concept in OMA by creating a finite state machine structure that sequences the activities in the military plan. These activities are represented by similar structures in turn, which encode the reactions necessary to execute that plan step in the given context.

Figure I-4 is an example of a portion of the 54th Mechanized Division's plan. The important features to notice are that the Division as a whole has sequential as well as parallel tasks (moving towards the objective, attacking 1GTA elements in sector, fixing the 55TR), and that the transition criteria can be either explicit or implicit. An explicit criteria is "on order, fix the 55TR", while an implicit criteria is termination of the "move to objective one" task. Each of these tasks, with associated triggers and maneuver models, become elements of the current state/program as discussed above.

This plan supports hierarchical refinement in reaction to current events. For example, the task called " attack elements of the 1GTA" is unexecutable as stated since the relevant enemy units have not been identified. The solution is to associate a decision function with the "attack elements" state who's purpose is to expand the plan. That decision function is executed by the simulator when the attack state is engaged (exactly as maneuver models are executed), and it chooses among available options based on the information at hand. This results in the plan refinement shown in figure I-4b.

Given plans expressed in this format, we added a layer of system architecture which was

Figure I-4: A Portion of the Plan for the 54th Mechanized Division

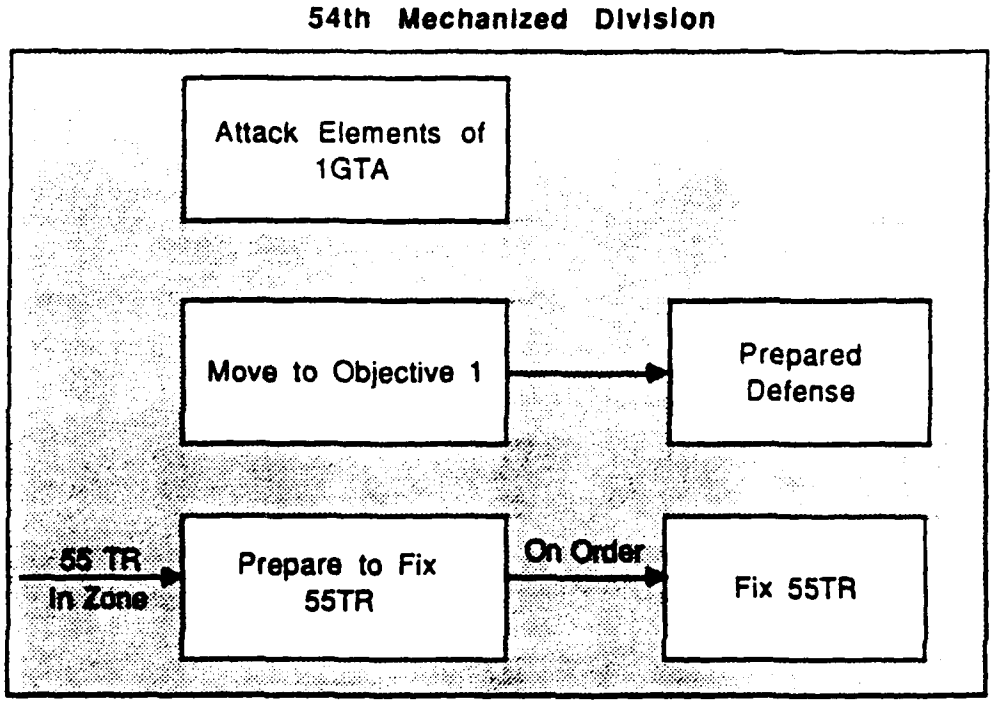


Figure 1-2a: Elements of the 54th Mechanized Division's Plan

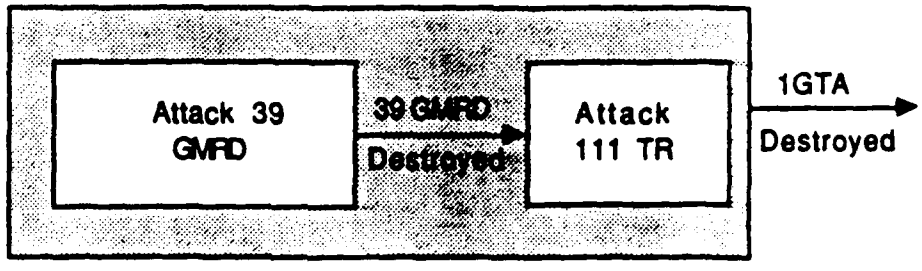


Figure I-4: A Portion of the Plan for the 54th Mechanized Division

responsible for triggering the relevant plan fragments, and for maintaining reactions to events not contained in the specific plan, but relevant in the context of the larger mission (for example, a reaction to attack a new enemy unit which appears in sector if you are not already engaged).

This architecture is shown in figure I-5. Here, there are two main data abstractions; one representing the overall plan (with operations for determining the next state and for augmenting the plan), and one representing the portion of the plan (the current state/program) in effect at a given instant. This abstraction has operations for adding and deleting states (modifying the program which governs action), for running that program (via the simulator), and for expanding a state into a more detailed plan for action in response to detected events. Communication between these abstractions are as follows:

- Since termination of a maneuver model may indicate completion (or failure) of a plan step, the event is communicated to the next state operation on the current plan.

- When state transition triggers associated with a current state fire (for example, when then 54th reaches objective 1), that fact is communicated to the next state operation on the plan.

- The next state operation calls the add and delete operations on the current state abstraction to update the triggers and programs it contains.

- The execute and add functions may call the expand operation to obtain a more detailed plan for action by consulting the reactive plan library. (For example, to resolve "attack forward in sector towards objective 1" into a sequence of attacks on the specific enemy units in the area of operations).

- The expand state operation communicates the expanded plan to the current plan by the "add plan" operation.

- The expand state operation updates the current state by selecting the appropriate maneuver model from the activity model library if one is available.

A projection engine built in this fashion would have a considerable capacity to react to unexpected situations.

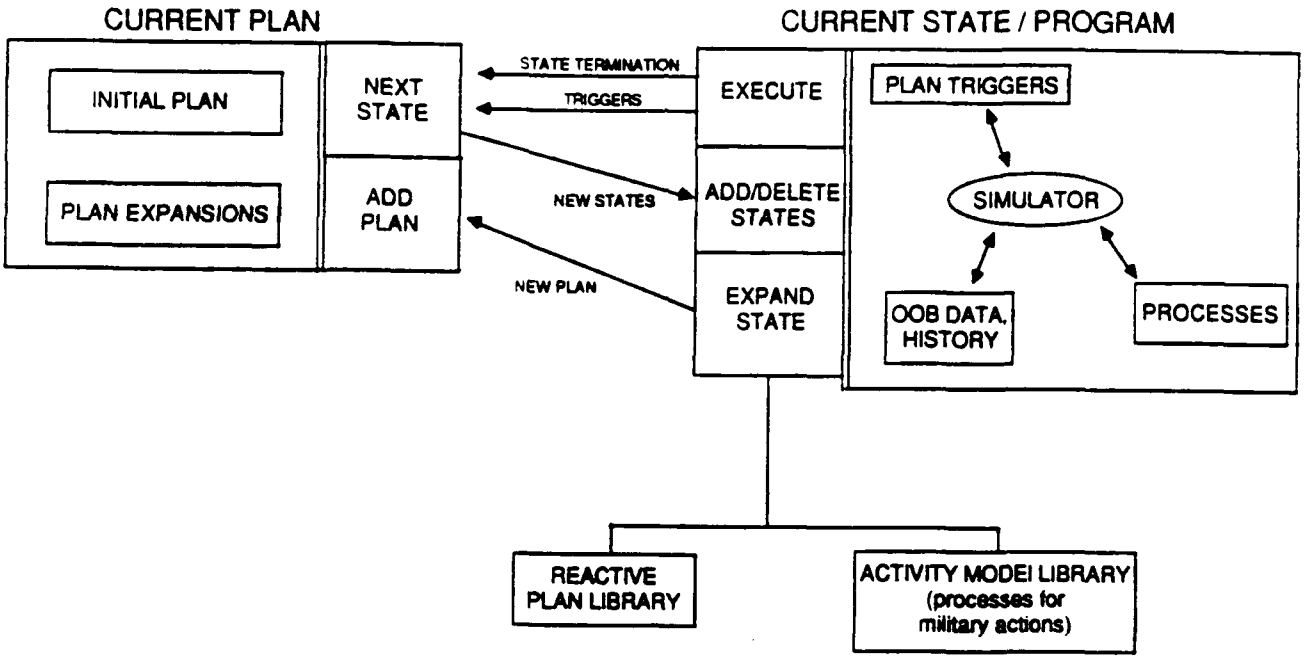


Figure I-5: An Implementation View of a Reactive Execution System

I.3.1 Analysis

We feel that the approach of implementing reactive behavior via higraphs and the above control architecture appears sound, but found that is was not doable within the time and dollar limitations of the project. In particular, substantial effort was involved in implementing the higraph representation, its edit and execution functions, and in capturing enough domain knowledge via higraphs to produce reasonable/accurate military scenarios.

Our solution was to adapt an existing simulation system to our needs.

References

[1] Harel, D.
Statecharts: A Visual Formalism for Complex Systems.
Sci. Comput. Prog. 8:231-274, 1987.

[2] Payne, J.R., Stachnick, G.L., Rosenschein, J., and Shapiro, D.
Operations Monitoring Assistant System Design.
Final report TR-1113-01, Advanced Decision Systems, July, 1986.

[3] Shapiro, D., Shu, R., and Tollander, C.
The Operations Monitoring Problem.
Semi-annual report TR-1159-01, Advanced Decision Systems, July, 1987.

[4] Shapiro, D., Shu, R., and Tollander, C.
OMA User's Manual.
Technical Report TR-1159-04, Advanced Decision Systems, December, 1988.

[5] Shapiro, D., Shu, R., and Tollander, C.
OMA Programmer's Manual.
Technical Report TR-1159-05, Advanced Decision Systems, December, 1988.

[6] Shapiro, D., Shu, R., and Tollander, C.
Projection of Future State Through an Operations Plan.
Semi-annual report TR-1159-02, Advanced Decision Systems, January, 1988.