Chapter 7

System Integration and Battle Management

CONTENTS

Boxes

Figure

Table

Chapter 7

System Integration and Battle Management

INTRODUCTION

Chapter 6 discusses developing, deploying, and maintaining a ballistic missile defense (BMD) system. Once deployed, BMD components would have to work together to form a fighting system. Maintaining such integration would require regular, routine support. This chapter looks at integrated operation of the system. Although some system capabilities could be used during peacetime, e.g., for surveillance, fully integrated use would only be required during battle.[1] Accordingly, most of this chapter is concerned with battle management, i.e., how the system would be managed to fight effectively.

[1]Peacetime simulations of battles would also require considerable integration, but would probably omit operations such as use of interactive discriminators and firing and controlling weapons.

A major assumption in the discussion that follows is that the system is sufficiently well-integrated during peacetime that it can be moved promptly to a full fighting status. As examples, the communications network that permits battle managers to exchange information would have to be working and the battle managers would need timely data on the number, kinds, and locations of resources available to them. Peacetime activities needed to keep the system integrated, such as sending updates of resource-availability data and new versions of software to battle managers, would have to be performed routinely. Operational readiness, testing and evaluation, and repair or replacement off ailing system components would also have to be routine.

BATTLE MANAGEMENT

The battle management portion of a BMD system would combine the data provided by sensors and the capabilities offered by weapons into a defensive system. The battle management computers would provide computational and decisionmaking capability. The battle management software would be the glue bonding the components together into a fighting system. Battle management includes strategy, tactics, resource allocation algorithms, and status reporting.

Battle management computing may be distributed among many different platforms or consolidated on just a few. In either case, the battle management functions would remain the same, although the capabilities needed in supporting functions, such as communications, might vary. This chapter describes the conduct of a battle from the battle management viewpoint, from alert through the actual battle sequence. *The scenarios only meant to be illustrative, not comprehensive. Its purpose is to convey a sense of the complexity of the battle management task, not to provide an actual battle management system design.*

In peacetime, the system might be in a quiescent mode, conserving fuel and other resources, with some components shielded from space. As the probability of a battle increased, the system might move through a series of alert levels, during which sensors such as the Boost Surveillance and ‘hacking System (BSTS) and Space Surveillance and Tracking System (SSTS) would be fully opened to space and weapons would be prepared for battle, including warm-up and status checks. At the highest alert level, the system would be fully prepared to fight a battle.

Note: Complete definitions of acronyms and intialisms are listed in Appendix B of this report.

The battle may be partitioned into different phases, each distinguished by a different set of offensive actions. The phases are boost, post-boost, mid-course, and terminal. For an individual reentry vehicle (RV) or decoy, the phases occur in the sequence given. Different RVs and decoys may be in different phases concurrently, requiring the defense to fight in different phases at the same time. In addition, the BMD system might have to defend itself against defense suppression attacks during any phase. For a description of the phases of ballistic missile flight, see chapter 3, table 3-6.

CONDUCT OF THE BATTLE

Our battle scenario assumes, for simplicity, a system with a second-phase architecture as described in chapter 3. We assume that it is in an alert stage from which it could be moved directly to fully automated battle management.[2] The battle would commence with the launch of Soviet intercontinental ballistic missiles (ICBMs). We assume a mix of ICBMs, some of which would burnout above the atmosphere, and some of which would bum out in the atmosphere. The ICBMs release post-boost vehicles (PBVs) carrying both RVs and decoys. The PBVs would maneuver to dispense their payloads, inserting each RV and decoy into a preplanned trajectory. RVs and decoys would then coast until they started to reenter the atmosphere. RVs would continue on to their targets, accompanied partway by those decoys designed to simulate reentry.

Besides launching ICBMs, the Soviets might employ a variety of defense-suppression measures. For example, they might launch direct-ascent anti-satellite weapons (ASATs) at BMD system satellites. Such a weapon might carry one or more warheads and decoys. The defense suppression attack might begin before an ICBM launch.

The following sections describe briefly the functions that a BMD system would have to perform during the battle. Requirements for recovering from damage and failures occurring during battle are given simplified treatment later. Table 7-1 gives a more detailed description, with examples, of the defensive functions, organized by function and by missile flight phase.

Battle Management Functions

In all phases of a battle the defensive system would have to track targets, assign weapons to destroy targets, aim, fire, and control those weapons, and assess the damage they do. It would also continually report on system status to human commanders, transmit information among computer battle managers within a battle phase, and from the battle managers in one phase (e.g., boost) to the battle managers in another phase (e.g., terminal). Additional functions, unique to each phase, are described in the following sections.

Each of these functions would involve making many decisions in short spaces of time using data obtained from a variety of sources. For example, aiming a weapon at a target might be based on tracking data obtained from a BSTS combined with data from a laser range finder located on satellite battle station, and would require the prediction of an intercept point for the target and weapon.

Boost Phase

The task of detecting booster launches would be unique to the boost phase as would be predicting the approximate trajectories of PBVs from those boosters penetrating the boost-phase defense. Trajectory prediction would needed so that space-based interceptors (SBIs)

[2]Although the assumption that there would be sufficient prior warning to an attack that a BMD system could be moved to an alert stage makes the scenario easier to describe, the system's designers could not depend on such an assumption to be true. There would have to be some provision to go from peace time to battle in seconds in the event that no warning is received or that such warnings are ignored.

Table 7-1 .—Ballistic Missile Defense Battle Management Functions

The components of any ballistic missile defense (BMD) system would have to be tied together in a battle management network. The table below lists the kinds of functions such a battle management system would have to perform, assuming a second- phase architecture of the type shown in Table 3-5. Computers would have to perform most of the functions. The BMD system architecture would specify locations and interrelationships among the computers. The system might be more or less centralized, more or less hierarchical. The elements of the system need to be tied together in a communications network. Chapter 8 of this report further discusses battle management communications and computation requirements.

Because different system components often perform the same functions in different ways, the table gives hypothetical examples of how the functions are accomplished in different battle phases. ***The "hypothetic/ examples' are just that: this table does not purport to outline a complete BMD architecture.***

The table is organized into 6 sections. The first 5 sections correspond to the boost, post-boost, mid-course, terminal, and self-defense parts of a BMD battle. The sixth outlines a battle of BMD system self-defense against anti-satellite weapons. The functions and their descriptions are the same for each section of the table; what varies is the way the functions are accomplished. Thus, to find out how objects might be tracked as part of the acquisition and discrimination function during the terminal phase, one reads the hypothetical example in the section of the table devoted to the terminal phase.

Boost Phase

Function	Description	Hypothetical Boost Phase Example
Acquire and discriminate objects	Sense objects of interest	Short and mid-wavelength infrared telescopes on BSTS detect hot exhaust plumes from launch of boosters
	Distinguish between targets to attack and decoys or debris	BSTS starts track files to distinguish moving ICBMs from stationary background and cloud clutter
	Track objects	SSTS sensors start to observe and record paths of identified objects
	Associate and Correlate objects sensed by different means or from different platforms	Battle management computers compare information gathered on two separate SSTS platforms and give same identification number to the same observed objects
Assess Situation	Estimate whether enemy is attacking, and if so with how many of what kinds of weapons with what battle tactics	BSTS detects ICBM launches, notes numbers and locations of launch sites, and determines types of missiles
	Assess which of own BMD forces are available for battle	Battle management computers determine which space based space-based interceptor (SBI) carrier vehicles (CVs) are in range of launched ICBM boosters, and will be in range of Post-Boost Vehicles (PBVs) when they are released from ICBMs.
Decide Course of Action	Authorize firing when ready, based on direction from higher authority if available, or as pre-authorized if not	If 3 or more ICBMs are launched within 1 minute, space battle management computers are programmed to command launch of space-based SBIs when the 4th ICBM is detected
	Determine strategy and battle plan	Determine plan for which kinds of ICBMs from which locations and which PBVs to attack first based on trajectories. CV positions. predicted RV impact points, and predicted times of PBV separation from missile
Select Targets & Direct Weapons	Choose which targets to strike	Select the booster or PBV whose trajectory will place it closest to the fly-out range of a particular CV
	Assign weapons	Battle management computer decides that SBIs no. 7888 and 7930 should attack target booster no. 754, and commands CVs to flight-check SBIs.
	Prepare engagement instructions	Battle management computers send flight plans and target track information to CVs.
	Assess kill: decide which targets have been destroyed	Remove a booster from the active target list
Employ Weapons	Control weapon	Feed target information to SBI guidance package
	Enable weapon	
	Prepare weapon	Conduct flight check of SBI

Table 7-1 .—Ballistic Missile Defense Battle Management Functions—continued

	Launch weapon	Open launch tube, eject and orient SBI, ignite SBI rocket motor
	Fly-out and kill	Battle manager transmits guidance update to SBI based on SSTS tracking data; SBI homes in on target booster or PBV

Post-Boost Phase

Function	Description	Hypothetical Post Boost Phase Example
Acquire and discriminate objects	Sense objects of interest	Infrared telescope on SSTS detects PBV after it separates from missile, and RVs and decoys after separation from PBV
	Distinguish between targets to attack and decoys or debris	From differences in IR signatures and other data, such as PBV recoils, sensor systems on SSTS distinguishes among PBVs, expended boosters, RVs, and decoys
	Track objects	SSTS sensors continue to observe and record paths of identified objects
	Associate and correlate objects sensed by different means or from different platforms	Computers on two separate SSTS platforms compare information gathered by each and give same identification number to the same observed objects
Assess Situation	Estimate whether enemy is attacking, and if so with how many of what kinds of weapons with what battle tactics	
	Assess which of own BMD forces are available for battle	Battle management computers determine whichs CVs are in range of targetable PBVs and RVs
Decide Course of Action	Authorize firing when ready	
	Determine Strategy and Battle Plan	Battle management computers determine plan for attacking targetable PBVs that have survived earlier SBI intercepts and when to start attacking RVs that have been deployed from PBVs
Select Targets & Direct Weapons	Choose which targets to strike	Battle management computers target the PBVs that will first be in range of SBIs
	Assign weapons	Battle management computer decides that space-based SBI no. 12,543 should attack PBV no. 328 and commands CVs to flight check SBIs
	Prepare engagement instructions	Battle management computers send flight plans and target track information to CVs
	Assess kill: decide which targets have been destroyed	Remove a PBV from the active target list
Employ Weapons	Control weapon	Feed target information to SBI guidance package
	Enable weapon	
	Prepare weapon	Conduct flight check of SBI
	Launch weapon	Open launch tube, eject and orient SBI, and fire SBI rocket motor
	Fly-out and kill	Battle manager transmits guidance update to SBI; SBI homes in on target PBV or RV

Mid-Course Phase

Function	Description	Hypothetical Mid-course Example
Acquire and discriminate objects	Sense objects of interest	Infrared telescope on SSTS detects RVs and decoys
	Distinguish between targets to attack and decoys or debris	From differences in motion after passage through dust cloud, laser range-finding radar on SSTS identifies target RVs v. decoys
	Track objects	SSTS sensors continue to observe and record paths of identified objects

Table 7-1.—Ballistic Missile Defense Battle Management Functions—continued

	Associate and correlate objects sensed by different means or from different platforms	Computers on two separate SSTS platforms compare information gathered by each and give same identification number to the same observed objects
Assess Situation	Estimate whether enemy is attacking, and if so with how many of what kinds of weapons with what battle tactics	
	Assess which of own BMD forces are available for battle	Computers determine which ERIS interceptors are in range of RVs
Decide Course of Action	Authorize firing when ready	
	Determine Strategy and Battle Plan	Determine plan for which RVs to attack first
Select Targets & Direct Weapons	Choose which targets to strike	Select the RVs within shortest flight time of a particular ERIS site
	Assign weapons	Battle management computer decides that ERIS interceptor no. 3001 should attack target RV no. 10,005 and commands fire control computer to flight check the interceptor
	Prepare engagement instructions	Battle management computer sends flight plan and target track information to ERIS fire control computer
	Assess kill: Decide which targets have been destroyed	Remove an RV in mid-course from the active target list
Employ Weapons	Control weapon	Feed target information to ERIS guidance package
	Enable weapon	Turn on ERIS warhead sensor
	Prepare weapon	Cool down ERIS homing sensor
	Launch weapon	Fire ERIS rocket motor
	Fly-out and kill	Battle manager transmits guidance updates to ERIS; ERIS homes in on target RV

Terminal Phase

Function	Description	Hypothetical Terminal Phase Example
Acquire and discriminate objects	Sense objects of interest	Infrared telescope on AOS detects RVs and decoys based on data received from SSTS; AOS 'passes data to TIR
	Distinguish between targets to attack and decoys or debris	From differences in motion after passage through the upper atmosphere, ground-based radar identifies target RVs v. decoys
	Track objects	Ground-based radars continue to observe and record paths of identified objects
	Associate and correlate objects sensed by different means or from different platforms	Ground-based battle management computer compares track information handed-off by space-based battle management computer with ground-based radar data and gives same identification number to the same observed objects
Assess Situation	Estimate whether enemy is attacking, and if so with how many of what kinds of weapons with what battle tactics	
	Assess which of own BMD forces are available for battle	Computers determine which HEDIs are in range of incoming RVs
Decide Course of Action	Authorize firing when ready	
	Determine Strategy and Battle Plan	Choose plan for which RVs to attack first
Select Targets & Direct Weapons	Choose which targets to strike	Select the RVs nearest to a target and that can be reached by a HEDI
	Assign weapons	Decide that HEDI no. 1897 should attack target RV no. 257
	Prepare engagement instructions	Ready flight plan and target tracking information for HEDI
	Assess kill: Decide which targets have been destroyed	Remove an RV in terminal from the active target list

Employ Weapons	Control weapon	Feed target information to HEDI guidance package
	Enable weapon	Turn on HEDI warhead sensor
	Prepare weapon	Cool down HEDI homing sensor
	Launch weapon	Fire HEDI rocket motor
	Fly-out and kill	Battle manager transmits guidance update to HEDI; HEDI homes in on target RV

BMD System Self-Defense

Function	Description	Hypothetical Self-Defense Example
Acquire and discriminate objects	Sense objects of interest	Infrared telescope on SSTS detects direct ascent ASAT
	Distinguish between targets to attack and decoys or debris	
	Track objects	SSTS sensors continue to observe and record paths of identified objects
	Associate and correlate objects sensed by different means or from different platforms	Computers on two separate SSTS platforms compare information gathered by each and give same identification number to the same observed objects
Assess Situation	Estimate whether enemy is attacking, and if so with how many of what kinds of weapons with what battle tactics	
	Assess which of own BMD forces are available for battle	Computers determine target of ASAT and which CVs may be used to defend against approaching ASAT
Decide Course of Action	Authorize firing when ready	
	Determine Strategy and Battle Plan	Choose plan for which ASATs to attack first
Select Targets & Direct Weapons	Choose which targets to strike	Select the ASAT nearest to a particular CV
	Assign weapons	Battle management computer decides that SBI no. 1024 should attack target ASAT no. 128, and commands CVs to flight check SBIs
	Prepare engagement instructions	Battle management computers send flight plans and target-track information to CVs
	Assess kill: decide which targets have been destroyed	Remove an ASAT from the active target list
Employ Weapons	Control weapon	Feed target information to SBI guidance package
	Enable weapon	
	Prepare weapon	Conduct flight check of SBI
	Launch weapon	Open launch tube, eject and orient SBI, and fire SBI rocket motor
	Fly-out and kill	Battle manager transmits guidance update to SBI; SBI homes in on target ASAT

NOTE: The first two columns of this table draw heavily from work of Albert W. Small and P. Kathleen Groveston, *Strategic Defense Battle Operations Framework,* (Bedford, MA: The MITRE Corp., July 1985). The hypothetical examples are supplied by OTA.

could be launched in time to intercept the PBVs before they dispensed their payloads.

Post-Boost Phase

Tasks unique to the post-boost phase would be noting the separation of PBV from missile and observing the PBV as it dispensed its payload. To have a chance to destroy most PBVs, the interceptors would have to have been launched during boost phase, perhaps before PBV separation. To intercept the PBVs, the system would have to guide the SBIs launched earlier. For PBVs that survived to dispense their payloads, the system might start discriminating between RVs and decoys, perhaps by trying to observe differences in PBV recoil during the dispensing process.

Mid-Course Phase

The primary problem for the defensive system during mid-course would be to discriminate real warheads from decoys. The number

of decoys maybe in the hundreds of thousands, or even greater. Decoys and warheads may appear very similar to optical, infrared, radar, and other sensors, both passive and active.[3]

Terminal Phase

During terminal phase, the defensive system would have to discriminate RVs from decoys using data handed off from earlier phases and using the atmosphere and radar signatures as discriminators.

Table 7-1 shows the different functions a BMD system would have to perform during battle, and how different components would participate in different phases of the battle. The table assumes a second-phase architecture, such as described in table 3-6. It also shows the functions that would serve to defend the system against defense suppression threats.

Interactions Among the Phases

A BMD system would not be a single, monolithic entity. Instead, it would comprise many different elements, some of which would participate in only one or two phases of the battle. In most system architectures, battle management would be conducted by different battle managers during the different phases. Furthermore, some battle managers might be fighting one phase of the battle while others are fighting a different phase. Boost, post-boost, and mid-course managers would be located in space, while terminal phase managers would likely be on the ground. For the system to function most effectively, information, such as tracks and status of RVs and decoys, would have to be communicated from battle managers in earlier phases to battle managers in later phases.

Interactions for Tracking Purposes

Establishing, distributing, and correlating track information is a good example of a problem involving interaction among different system elements and cooperation among battle managers. Detecting, identifying, and noting the current position of a target would not be sufficient for guiding a BMD weapon system. The target would move between the time that the sensor records its position and the time the weapon is fired. An SBI traveling at, say, 8 km/see, would take 250 seconds to reach an RV target if the SBI were fired at a range of 2,000 km from the impact point. During those 250 seconds, the RV would move 1750 km. Just as the hunter must lead the duck in flight as he fires his shotgun, the BMD system would have to aim its SBI well ahead of the speeding RV.[4]

The BMD system would therefore have to keep track of each target's motion and predict where the target would be at later times. The sensors would have to generate a‘ ‘track file, i.e., a history of each target's motion through space. Given the target's past history in terms of position, velocity, and acceleration in three *dimensions*, a computer could then predict its future position. This prediction could then be used to aim and fire the weapon system. After the SBI was fired, the sensors would have to continue to track the target (and possibly the SBI), the track files would have to be updated, and mid-course guidance corrections sent to keep the SBI on a collision course. Mid-course corrections would be mandatory if the target acceleration changed after the SBI was fired, as would occur with multi-stage missiles.

For directed-energy weapons or interactive sensor systems, the delay from the time that energy is emitted by the target until it reaches the sensor, and then from the firing of the weapon until the arrival of the kill energy traveling at the speed of light, would be very short, but not zero. At 2,000 km range, for example, 13 milliseconds would elapse from the last sensor reading until the time a laser beam could reach the target. The RV would move about 91 meters in this time, so some predictive ca-

[3]Chapters 4 and 10 discuss the issues of discrimination during mid-course in more detail.

[4]However, the RV would be moving on a ballistic or free-fall flight with no external acceleration other than the force ***of gra***vity. Therefore, predicting its future path or trajectory would be possible provided that the sensor generated two or more accurate three dimensional target positions. Predicting the future path of an accelerating, multi-stage missile would be much more difficult. RVs that could maneuver would worsen the difficulties.

pability would be required.[5] In addition, the observable characteristics of the target would change drastically during the tracking operation. The sensor systems would begin tracking the hot booster plume. For boost phase kills, the sensors would have to acquire and track the cold booster body, or rely on calculations of the missile body position relative to the booster plume for all booster/sensor orientations. After the last booster stage had burned out, the battle management computers would have to continue the track file on the surviving post-boost vehicles (PBV), even though tracking data might be derived from other types of sensors. Finally, as individual RV's and decoys were deployed, the track files would have to proliferate, taking the last PBV track projection as the recently deployed RV track file, until it could be verified and updated by long wavelength infrared sensors.

The data handling problem would be compounded by RV's that survived the boost and post-boost defensive attacks. The surviving RV's would usually pass out of the field of view of the initial sensor. The track file obtained from one sensor's data would then have to be correlated with the data from another sensor.[6] Track data would be passed to the appropriate weapons platform at each stage of the battle. Eventually the track files of surviving targets should be passed on to the ground-based terminal defensive systems to aid in the final kill attempts. Information on decoys and other rocket or killed target debris should also be transmitted, to avoid attacking too many false targets.

System Performance and Interaction

The ability to correlate data well from different sensors (required to get accurate three-dimensional track histories) could have a strong effect on system performance, as could the ability to correlate track files exchanged among battle managers both within and between phases. Poor performance in the early phases would mean many RVs leaking into later phases, with possible overload of resources assigned to mid-course and terminal phases.

Distinctions Among the Phases

In all phases the defensive system would have to perform many similar functions, such as tracking; weapons assignment; aiming, firing and controlling weapons; and reporting status. Sensors and weapons would vary considerably from one phase to another, however. For example, boost, post-boost, and mid-course tracking would be done primarily by space-based infrared sensors such as those incorporated in the BSTS and SSTS systems. Terminal phase tracking would be done by a combination of airborne infrared sensors and ground-based radars. The software and hardware used to perform the sensing, discrimination, and tracking functions indifferent phases would likely be quite different; aiming, firing, and controlling weapons might be similar.

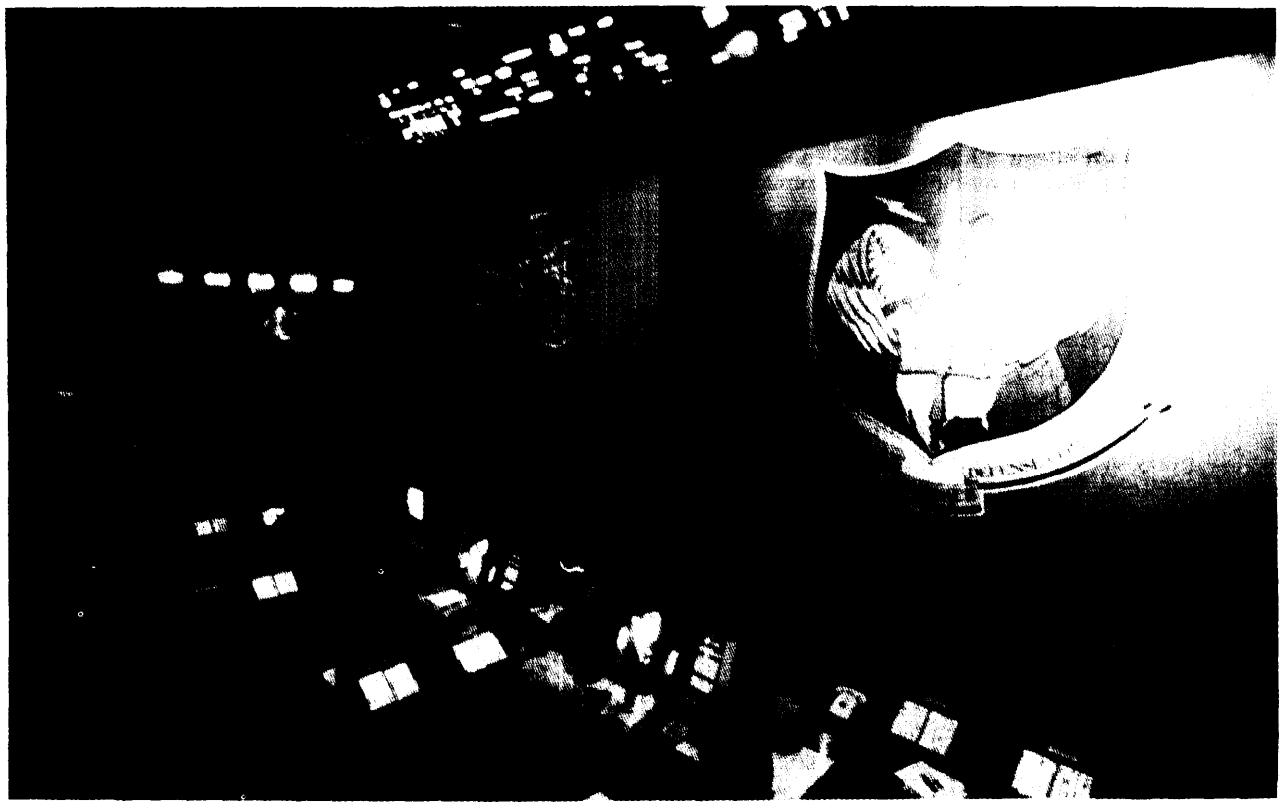
Some phases would require unique functions. A good example is interactive discrimination of RVs from decoys for mid-course defense. Candidates for such discrimination, such as neutral particle beam systems, would likely be controlled by unique computer software and hardware adapted to the task.

Reconfiguration

In addition to fighting the battle, the system would also have to be able to reconfigure itself during and after the battle, to compensate for damage done to it, in preparation for further or continued engagements. In the post-

[5] In general it would take additional time for the sensor signal processor to analyze the sensor data and for the track file to be updated after the last signal was received; the actual elapsed time between observation and the order to fire the weapon might be 5 to 10 seconds, so the target might move as much as 70 km even for a directed-energy weapon.

[6] If battle managers, sensors, and weapons were organized into autonomous battle groups, then each battle manager would have to hand off and receive track data as targets passed through the field of view of sensors in its group, or it would have to perform all of its own target acquisition and discrimination. If there were a single battle manager to handle all tracking and correlation, it would have to maintain track files on all targets. Such an organization would tie system survivability to survivability of the central battle manager. Finally, if battle groups were to use fixed battle management platforms, but different sensors and weapons as the battle continued, then the battle managers would have to correlate tracks from different sensors as the sensors moved in and out of its group.



N D NOR D M
m m g m m
m ry
m m
m

battle case, this might be done with human assistance.

Opportunities for Human Intervention

Tracking and discriminating objects, aiming and firing weapons, and managing the battle would require computers. During peacetime, humans could monitor surveillance data after it had been processed and displayed in a form suitable for human interpretation. Human decisionmakers could deduce from the events monitored, among other things, whether and when the defensive system should be placed in alert status, ready to cope with a battle. Once the battle started, however, the response time and data processing requirements would severely limit the opportunities for human intervention. There are four possible human intervention points under consideration in currently suggested BMD architectures:

- the decision to move the system from peacetime status to alert status,
- the decision to release weapons,
- the decision to switch to a back-up for one or more of the algorithms (see box 7-A) used by the battle management computers, and
- selection from a pre-specified set of tactics for terminal phase, made as a result of observations of earlier phases of the battle.

Transition to Alert Status

Humans could decide to move from one level of alert to another in hours or minutes, as compared to fractions of seconds for computers

Box 7-A.—Algorithms

Methods for solving problems by the use of computers are often expressed as algorithms. As described by John Shore,

> An algorithm is a precise description of a method for solving a particular problem using operations or actions from a well-understood repertoire.[1]

More technical definitions often require that the description contain a finite number of steps, each of which can be performed in a finite amount of time, and that there be specific inputs and outputs. As explained by computer scientist Donald Knuth,

> The modern meaning of algorithm is quite similar to that of recipe, process, method, technique, procedure, routine, ...2

Carrying out the steps of an algorithm is known as "executing it." If one thinks of a recipe for baking a cake as an algorithm, then executing the algorithm consists of following the recipe to produce the cake. The following is a simplified example of an algorithm that might be used in the early stages of the design of a BMD system. The purpose of the algorithm is to detect the launch of boosters. We assume that the system uses a sensor on a satellite that can scan the Soviet Union. The sensor is composed of a number of different elements, each of which is sensitive to the radiation emitted by a booster. The Soviet Union is divided into regions, and each detector element periodically scans sequentially across a number of regions.

1. For each detector element, record all detected radiation sources greater than the threshold for a booster as the detector scans across the Soviet Union. Record the time of occurrence of each detection as well as the intensity of the source.
2. For each source recorded, identify its region of origin.
3. Compare the occurrences of sources in the current scan with occurrences from the previous two scans. Count all events consisting of occurrences of sources in the same region for three consecutive scans. Flag each such event as a launch.
4. If data have been saved from more than 2 consecutive scans, discard the data from the oldest scan and save the data from the current scan.
5. If no launches were observed, go back to step 1, otherwise continue with step 4.
6. If launches were observed, notify the system operator.

While this description is simplified, e.g., omitting consideration of booster movement across regions, it is an algorithm because the operations needed to perform each step could be completely specified; furthermore, it could be implemented as a computer program.

Although the number of steps used in describing an algorithm must be finite, the definition does not require that the algorithm terminate when executed. Many algorithms are designed to be non-terminating, such as the following simplified description of how a radar processing system might operate:

1. Send out radar pulse.
2. Wait a pre-calculated interval for a return pulse.
3. If there was a return pulse calculate the distance to the object.
4. Go back to step 1.

Despite not terminating, this algorithm still produces useful results. Some algorithms terminate under certain conditions, but do not terminate and produce no results under other conditions. Conditions under which algorithms do not produce the desired results are known as *error* or exception conditions and the occurrence of such conditions as *undesired events.* For the following simplified algorithm, which tracks a target based on radar returns, the failure of the radar pointing mechanism is an undesired event that causes the algorithm to continue endlessly, producing no useful result.

1. Retrieve the last known target location, velocity, and acceleration.
2. Calculate the estimated current target location.

[1] John Shore, *The Sachertorte Algorithm and Other Antidotes to Computer Anxiety (New* York, NY: Viking Press, 1985), p. 131.
[2] Donald E. Knuth, The Art *of* Computer *Programming, Vol. 1:* Fundamental Algorithms (Reading, MA: Addison Wesley, 1974), p. 4.

3. Point the radar at the estimated current target location and attempt object detection.
4. If the radar locates no object at the estimated range to the target, revise the estimated target location and return to step 3. Otherwise, continue with step 5.
5. If the radar locates an object at the estimated range to the target, update the target location, velocity, and acceleration.

A real version of such a tracking algorithm would have to take into account the possibility that the radar might fail in any of several different ways, or that earlier estimates of target position might be grossly wrong.

during battle. The humans' decisions could be based on data gathered from sources both within and outside the BMD system. The computerized battle-time decisions would be based on data acquired by the system's sensors.

Weapons Release

Once a human had permitted the transition to the highest level of alert, the system would function automatically, responding to threats as it perceived them.[7] It would be possible to build human intervention points for the release of weapons into the battle management process. In the first-phase and second-phase systems described in chapter 3, the first weapons to be released would be the SBIs. The period from the time that a missile launch was first perceived by BMD sensors until SBIs would have to be launched to intercept a missile still in its boost phase would be quite short—a few minutes at most. Accordingly, if humans were to control the release of weapons, they would have to monitor the defense system's operation continuously once *it* had moved to the highest alert status.

Since it may be necessary to release hundreds or thousands of SBIs within minutes, a human operator would not be able to authorize release of individual SBIs. Because of the rapid reaction times needed, continual human intervention in the weapons release process would likely degrade system effectiveness unacceptably. It might be feasible to intervene when previously unused weapon systems were brought into the battle. As an example, if neutral particle beam (NPB) weapons had not been used before enemy missiles reached midcourse, then a human might be called onto authorize their *use* during the mid-course part of the battle. Even such occasional intervention might degrade performance somewhat.

Switching to Back-ups

During the course of the battle it might be possible for a human observer to determine that a BMD system was malfunctioning. For a human to notice, the malfunction would probably have to be gross, such as a failure to fire interceptors or firing interceptors in obviously wrong directions. If the problem lay in the algorithms used by the battle management computers, and if the system were designed in such away that back-up algorithms were available to the computers, then the human might command the battle management computers to switch to a particular back-up algorithm.[8]

Human intervention of this type is rarely used in existing systems because the human cannot interpret the situation correctly in the available time, and because it is difficult to design the software to switch algorithms successfully in mid-computation. Inmost systems, the gain is not worth the added software complexity. The potential gain for BMD from such intervention would be that in the cases where

[7]The AEGIS ship defense system, often cited as performing ***many*** of the same functions as a BMD system, reacts completely automatically to incoming threats when in the highest level of alert mode. For some threats, AEGIS must react within 15 seconds from the time a threat is detected.

[8]Switching to a back-up algorithm should not be confused with situations where a computer uses built-in diagnostics to determine the occurrence of a hardware malfunction and then automatically switches operation to a redundant component. Such diagnostics and hardware redundancy for automatic switching are now used in some critical applications, such as airline transaction systems, telephone switching systems, and battle management systems such as AEGIS.

the system had been badly spoofed by the enemy, and the human operator quickly recognized the symptoms, cause, and needed corrective action, recovery might be possible in time to continue the battle. The risk would be that the operator would misjudge the situation, or that the complications involved in provialing the appropriate interface to the operator, both in additional software, hardware, and communications capability, would make the system less reliable.

Selection of Tactics

Because the boost, post-boost, and midcourse phases of a BMD battle would last 20-30 minutes, a human commander might be able to evaluate the results of those phases in time to affect the tactics used during the terminal phase. To do so, he would have to be presented with status reports during the battle. Based on his analysis of the battle, and on choices of previously-determined tactics presented to him by an automated battle manager, he could choose the terminal phase tactics to be used. Again, because of the time-scales and data volumes involved, he would probably not be able to alter his choice once the terminal phase began.

Increasing degrees of human intervention would require increasing complexity in the interface between humans and the battle management system. A sophisticated interface between human and computer would be needed, allowing the human to observe status and issue commands, and, when appropriate, receive acknowledgements. Such an interface would add complexity to the software. Furthermore, the human operator(s) would probably have to have authority to release weapons, as there might not be time to consult with higher authorities.

Common to all BMD system designs that require human intervention at any stage is the need to provide secure, rapid communications between the human and the battle management computers. If part of the system were in space, then most likely there would be a need for space-to-ground communications.[9]

For all of the preceding reasons, it seems likely that a BMD system would operate almost completely automatically once moved to an alert status in preparation for battle.

The preceding analysis illustrates the difficult trade-offs involved in designing a battle management system. In considering the interface between humans and the system, the designer must trade off communications security against the need for human intervention against system structure against complication of the computing tasks. He must also balance system performance against all other considerations, deciding whether the system could perform better and more dependably with the aid of a human than without, and whether any extra complication in the human-computer interfaces would be worth whatever capability and trustworthiness might be added by the human.

[9]Even if a human operator were space-based, he might need to communicate with higher authority on the ground. Such communications would probably not require as rapid data communication rates as battle-management-to-operator communications.

SUPPORTING FUNCTIONS

Table 7-1 shows the primary battle management functions, but does not include several supporting functions. Most important of these are communications and recovery from damage and from failures. Both of these functions are needed in all phases, with communications playing its traditionally crucial role in battle management and with recovery invoked as needed. Both communications and recovery procedures would be completely automated during a battle. Because of the short decision times involved during boost, post-boost, and terminal phases, recovery would have to be extremely rapid; delays would result in RVs moving on to the next phase or reaching their targets. Long delays in recovery could also reduce

the opportunities for a battle manager fighting in one phase to pass information along to battle managers fighting in the next phase.

Communications

Automated communications links between sensors and battle management computers, between different battle management computers, between battle management computers and weapons, and between battle management computers and humans, would all be needed for effective battle management. Data would be continually transmitted over a battle management communications network during all battle phases.

Recovery From Damage and Failures

Present in all phases would be the need to recognize and recover from system failures and from damage to system resources. Individual system components would have to monitor and report their own status continually. They would have to try to recover from local failures, whether internally generated—perhaps by a software error-or externally generated—perhaps by a detonation of a nearby nuclear antisatellite weapon causing radiation damage in a computer chip.

Some instances of system damage and failure, such as destruction of several adjacent battle management platforms or communications controllers, would require recovery based on "global information," i.e., information about the status of the entire BMD system and the entire battle. Examples are knowing how to reroute communications around damaged nodes in the communications network, or knowing which battle management computers were in position, both physically and in terms of resources available, to take over the functions of a disabled battle manager.

COMPLEXITY OF BATTLE MANAGEMENT

Conduct of a successful BMD battle would be similar to the conduct of a large conventional battle in that it would require the orchestration of many different kinds of components under precise timing constraints. The problem may be ameliorated somewhat by preplanning some of the orchestration. The difference is that in a BMD battle the time constraints would be tighter, the battle space would be larger, the fighting would largely be automated, the components would be previously untested in battle, and there would be little chance to employ human ingenuity to counter unanticipated threats or strategies.

The only kind of BMD system for which the U.S. has battle management software development experience and an understanding of the attendant problems is a terminal defense system, such as SAFEGUARD. Some consider even this experience as suspect, since SAFEGUARD and other systems like it were never used in a real battle. Adding a boost-phase defense would add complexity to the system and require the inclusion of technologies hitherto untried in battle. It would also be the first time that software was used to control highly automated space-based weapons.

Adding amid-course defense would probably increase the software complexity past that of any existing systems. The burden of effectively integrating information from different sensors, controlling different weapons, coordinating interactive discrimination to distinguish among hundreds of thousands of potential targets, and selecting effective strategy and tactics—all while trying to defend against active countermeasures-would fall on the software. (Software issues are discussed in detail in ch. 9.)

Approaches to reducing complexity center on "divide and conquer" strategies applied to architecture definition, aided by simulations of the effectiveness of different battle management architectures. Those who favor such approaches believe that the system could be designed and built in small, relatively inde-

pendent pieces that could each be adequately tested, and that could be jointly subjected to peacetime tests. As an example, each interceptor carrier vehicle might contain a battle manager, designed to fight independently of other battle managers. They argue that the system could be easily expanded by adding more pieces, e.g., more CVs each with its own battle manager. Since the pieces would tend to be independent, the reliability of the system would be more strongly related to the reliability of an individual piece, rather than to the joint reliability of all pieces, i.e., knowing that individual pieces were reliable would suggest that the whole system was reliable. Also, the failure of a single piece might not be as catastrophic as in an architecture where the pieces were highly interdependent.

Those who doubt the effectiveness of such a strategy in the face of the complexity induced by BMD requirements argue that making the pieces independent would require making them very complex. They further note that historically no approach has led to the development of a weapon system whose software worked correctly the first time it was used in battle. The greater complexity of BMD software over existing weapon systems leads them to believe that a BMD system would have little chance of doing so.

BATTLE MANAGEMENT ARCHITECTURE

A *battle management architecture* is a specification of the battle management functions to be performed by different system components and the relationships among those components. Components may be software, such as a set of computer programs that would allocate weapons to targets, or hardware, such as the computer(s) used to execute those programs. (See also ch. 3.)

A significant architectural tradeoff concerns the degree of coupling among battle managers (see box 7-B). Some proposed architectures use a very loosely coupled system with little communication among battle managers, similar to the "almost perfect" architecture described in the Fletcher Report.[10] Such architectures tend to locate battle managers on board carrier vehicles. Others use a more tightly coupled system with track and other data exchanged among battle managers for coordination purposes. They often locate battle managers on separate platforms.

Box 7-B.—Centralization, Distribution, and Coupling

A centralized system concentrates computing resources in one location and may consist of several processors that share the same memory and are housed together physically. Such a system is known as a multiprocessor. The processors are able to communicate with each other at very high data rates, and are said to be ightly *coupled.* As the processors are physically moved apart, acquire their own, separate memories, and as the data communication rates among them decrease, they acquire the characteristics of a distributed system, also called decentralized, and are said to be *loosely coupled.*

An important factor in understanding the degree of coupling is the criterion used to partition the battle space into segments so that each battle management computer has responsibility for a segment. Indeed, criteria for segmentation are one way of distinguishing among architectures. Segments might be geographically determined and of fixed location, or might be determined by the clustering of targets as they move through space, or might be determined by the location of battle

[10] James C. Fletcher, Study Chairman and B. McMillan, Panel Chairman, Report *of* the Study on *Eliminating the Threat Posed by Nuclear Ballistic Missiles: Volume V, Battle Management, Communications, and Data Processing,* (Washington, DC: Department of Defense, Defensive Technologies Study Team, Oct. 1983), p. 19.

managers, CVs sensors, and other system resources during a battle.

Although the Eastport Group" recommended that BMD battle managers be hierarchically structured, the Fletcher Report[12] suggested that a logical battle management structure that was almost perfect would not be hierarchical, but would consist of a single battle manager replicated a number of different times, with each copy physically located on a different platform. The Fletcher report also noted that such an architecture might be very costly, that there might be equally effective and cheaper alternatives, and that it was important to look at technical issues that distinguished among those alternatives. An example given was the effectiveness of algorithms that allocated weapons to targets based only on local data. As yet, few detailed studies of such technical issues appear to have been made.

The Eastport Group recommended the development of a decentralized, hierarchical battle management architecture. '3 Architectures currently under consideration for BMD systems are consistent with that recommendation. In a typical such architecture, each battle manager would report as necessary to battle managers at higher levels, and would receive commands from them. There might be 3 layers of battle managers; the lowest layer would be local battle managers, which perform the fighting functions. The next layer would be regional

[11] Eastport Study Group Report, "Report to the Director, Strategic Defense Initiative Organization, 1985. "

[12] Fletcher Report, op. cit., footnote 10, pp. 9-21.

[13] The Eastport Study Group is the name used to refer to the SDIO Panel on Computing in Support of Battle Management. It was appointed "to devise an appropriate computational/communication response to the SDI battle management computing problem and make recommendations for a research and technology development program to implement the response. " Its report was issued in December 1985.

Box 7-C.—Hierarchies and System Design

Designers of systems find it useful to impose a structure on the design. For complex systems, several different structures may be used, each allowing the designer to concentrate on a different concern. In systems where many components are involved, such as complex software systems, large communications systems, and complex weapons systems, the structures used are often hierarchical. Each hierarchy may be specified by identifying the participating components and a relationship among them. The military command structure is an example of a hierarchy. The components are individuals of different rank, and the relationship is "obeys the commands of, " e.g., a lieutenant obeys the commands of a captain.

Many proposed SDI battle management architectures use some variation of the relationship "resource contentions are resolved by. " Thus, local battle managers' resource contentions are resolved by regional battle managers. However, another important battle management hierarchy is "communicates track data to. " This latter hierarchy is important in determining communications needs for the BMD system, and is sometimes confused with the former.

A *tree* is a special form of hierarchy in which a component at one level is only related to one component at the next higher level. The military chain of command is an example. A lieutenant is only commanded by one captain, although a captain may have several lieutenants under his command. The Eastport Group considered battle management architectures that were structured as trees to be the most promising for SDI.'

Structures may describe relationships among entities in a design, independent of physical relationships among system components. Such structures are sometimes known as *logical structures.*[2]

'The Eastport Study Group is the name used to refer to the SDIO Panel on Computing in Support of Battle Management. It was appointed "to devise an appropriate computational/communication response to the SDI battle management computing problem and make recommendations for a research and technology development program to implement the response. " Its report was *issued* in December 1985.

[2] See, for example, ***Report*** *of the* ***Study on Eliminating the*** Threat Posed by ***Nuclear Ballistic Missiles,*** James C. Fletcher, Study Chairman, Volume V, ***Battle Management, Communications,*** *and* ***Data*** *Processing,* (Washington, DC: Department of Defense, Defensive Technologies Study Team, October 1983), p. 18.

battle managers, which would resolve contentions for resources among local battle managers, as battle managers, sensors, and weapons moved, and which would assign responsibility for targets passing between battle spaces. At the top would be a global battle manager that would establish strategy for the regional and local battle managers and that which would provide the interface between humans and the system. The battle space would be partitioned into segments such that each battle manager in the lowest layer of the hierarchy had responsibility for a segment. As battle managers and targets moved through the battle space, information concerning them, such as type of target, location of target, and trajectory of target, would have to be moved from one computer to another.

Some recent proposals have suggested fewer layers in the battle management hierarchy. In such architectures, the hierarchy of automated battle managers is flat, i.e., there are no regional battle managers, and the top layer is a human commander. Such organizations have been designed so that battle managers may act almost independently of each other.

The volume of data to be communicated among the battle management computers would depend on the degree of coordination among the battle managers required by the battle management architecture. (See chapter 8 for estimates of communication requirements.) The determining factor is the amount of target tracking information that would have to be exchanged. Since there might be hundreds of thousands of objects to be tracked during mid-course, architectures that required tracks of all objects to be exchanged among battle managers would place a heavier load on communications than those that required no object tracks to be exchanged. The price paid for exchanging less information, however, would be the traditional one: the ability to coordinate the actions of different battle managers would be hampered and the overall efficiency of battle management might be decreased.

The efficiency-volume trade-off may be seen by considering the transition from one phase of the battle to the next. As an example, the terminal-phase battle managers would have the best chance to destroy targets if they received target-tracking information from the mid-course battle managers. Without such information they would have to acquire, track, and discriminate among targets before pointing and firing weapons. With such information they would only have to continue tracking targets and point and fire weapons. In such a situation, one might suggest combining the mid-course and terminal-phase battle managers into one set of programs on one computer as opposed to a more distributed system within-information transmitted among battle managers. Unfortunately, this organization would probably complicate the battle management task, since there would be somewhat different functions to be performed in the different phases, and since the way functions would be implemented in different phases would be different.[14]

The Eastport Group believed that a hierarchical battle management organization would simplify the computing job of each battle manager and would allow the battle managers to act without frequent interchange of information.[15] The concerns of each battle manager could then be simplified more than in a non-hierarchical organization, battle managers could be added to the system as needed, and the system would still survive if a few lower level battle managers were lost.

[14]Since different weapons would be used in the terminal phase as compared to the mid-course, pointing and controlling the weapons would be done differently. Similarly, different sensors may be used to discriminate between targets and decoys, requiring the allocation of resources with different characteristics and therefore a different resource allocation algorithm.

[15]Some earlier proposed architectures required the battle managers to be tightly coupled, exchanging considerable information with each other frequently. The Eastport Study Group rejected such an architecture because of the computing and communications burden it would place on the battle managers, and because of the complexity it would induce in the battle management software.

Decentralizing battle management means that the battle management task would be physically distributed among different computers. Decentralization would permit other battle managers to continue fighting even if a local battle manager were disabled. However, if the degree of coupling were high, the loss of data from the disabled battle manager might result in reduced effectiveness of the others. Without a specific design and a way of effectively testing architectures, it is difficult to verify claims about their merits and deficiencies. Such tests would have to be based on simulations and on whatever peacetime tests could be conducted.[16] However, the apparent disadvantages of a decentralized, hierarchical system would be:

- contentions for resources would have to be resolved at upper levels of the hierarchy, possibly adding complexity to the computational problem as a whole,
- the actions of battle managers would be based mainly on local data, perhaps resulting in inefficiencies, e.g., adjacent battle managers might both shoot at some of the same targets, thereby wasting shots, unless sophisticated battle management algorithms to compensate for the information loss could be developed,
- if strategic and tactical decisionmaking were concentrated at one level in the hierarchy, disabling some or all of that level could greatly reduce system performance. Such damage would be easier to accomplish if there were relatively few battle managers at that level, as might be true at the higher levels of the hierarchy.

The Eastport group believed that the advantages of a hierarchical, decentralized system far outweighed the disadvantages. Evaluation of advantages and disadvantages must await a design specific enough to be tested, and an effective test method.

No matter the choice of structure for battle management, some technology would be strained and software dependability would be a key issue. Centralization would appear particularly to stress computational performance and survivability. Decentralization would appear to require more sophisticated software at the local battle manager level and would increase the weapons supply needed. All architectures would require secure communication, whether to exchange track data, or to receive sensor data, or to communicate with the ground.

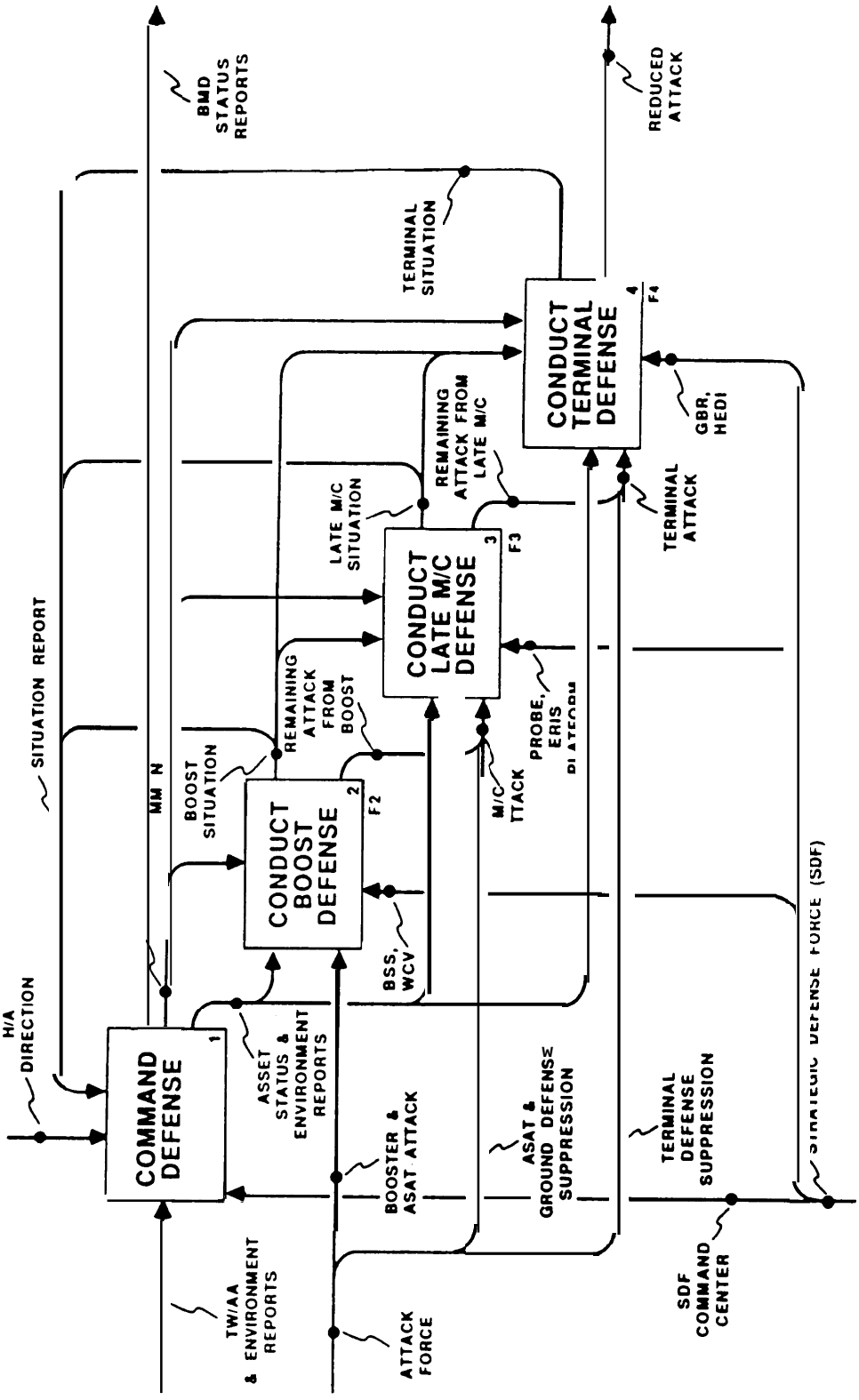
[16] The proposed National Test Bed, might provide some of the simulation capabilities needed for architecture evaluation.

CONCLUSIONS

Ballistic missile defense battle management would be an extremely complex process. The number of objects, volume of space, and speed at which decisions would have to be made during a battle preclude most human participation. Aside from authorizing the system to move to alert status, prepared to fight automatically, at best the human's role would be to authorize the initial release of weapons and to change to back-up, previously-prepared, strategy or tactics. Decisions about which weapons to use, when to use them, and against which targets to use them would all be automated. Inclusion of human intervention points would likely add complexity to an already complex system and to compromise system performance in some situations. On the other hand, if an attacker had successfully foiled the primary defensive strategy, human intervention might allow recovery from defeat.

Battle management architectures as yet proposed are not specific enough for their claimed advantages and disadvantages to be effectively evaluated. Such evaluation must await both better architecture specifications and the development of an effective evaluation technique, perhaps based on simulation.

Figure 7-1.—Analytic Chart of Battle Management Functions in a Phase-one BMD System



Designing the battle management sub-system (including command, control, and communications) for a BMD system will require precise specification of all the necessary channels of communication and points of decision. "Pipeline" charts like this help analysts conceptualize battle management complexities. Later, computer programs must be engineered to carry out the functions identified by the analysts.

SOURCE: Strategic Defense Initiative Organization, 1987.