

NAVAL POSTGRADUATE SCHOOL

Monterey, California



Developing Highly Predictable System Behavior in Real-Time Battle-Management Software

by

Dale Scott Caffall and James Bret Michael

29 September 2003

Approved for public release; distribution is unlimited.

Prepared for: Missile Defense Agency
7100 Defense Pentagon
Washington, D.C. 20301-7100

20031015 009

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RADM David R. Ellison, USN
Superintendent

Richard Elster
Provost

This report was prepared for Naval Postgraduate School as part of Dale Scott Caffall's dissertation research and James Bret Michael's sponsored research for the Missile Defense Agency.

Reproduction of all or part of this report is authorized.

This report was prepared by:

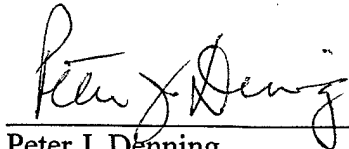


Dale Scott Caffall
C2BMC Chief Engineer
Missile Defense Agency



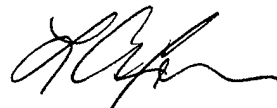
James Bret Michael
Associate Professor of Computer Science
Naval Postgraduate School

Reviewed by:



Peter J. Denning
Chairman, Department of Computer Science

Released by:



Leonard A. Ferrari
Associate Provost and
Dean of Research

REPORT DOCUMENTATION PAGE			Form approved OMB No 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 29 September 2003		3. REPORT TYPE AND DATES COVERED Technical Report
4. TITLE AND SUBTITLE Developing Highly Predictable System Behavior in Real-Time Battle-Management Software			5. FUNDING None	
6. AUTHOR(S) Dale Scott Caffall and James Bret Michael				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Department of Computer Science Naval Postgraduate School 833 Dyer Road, Code CS Monterey, CA 93943-5118			8. PERFORMING ORGANIZATION REPORT NUMBER NPS-CS-03-006	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words.) Given that battle-management solutions in system-of-systems environment are separately designed and developed on various operating platforms in different languages, predicting battle-management behavior of system-of-systems is not possible. As a rule, battle-management is executed at the system level rather than the desired system-of-systems level. Typically, C4 systems are non-real-time systems. Traditionally, weapon systems are real-time systems. If we are to match the performance of weapon systems and avoid the negative impact of forcing synchronization of battle manager software with weapon systems for messaging, then we must develop the battle manager as real-time software. We advocate the development of battle-management software as a real-time set of system functionality that addresses warfighter usage. To achieve the level of desired predictable battle-management behavior, we maintain that it is essential to develop a formal representation that captures the desired battle manager system behavior and test the formal representation against the expected battle-management properties. Furthermore, we assert that it is critical to develop the battle manager as a real-time software-intensive system to ensure the schedulability of battle-management tasks and provide for concurrent execution of such tasks where applicable.				
14. SUBJECT TERMS Battle-management, real-time, temporal logic, assertions, model-checking			15. NUMBER OF PAGES 49	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Developing Highly Predictable System Behavior in Real-Time Battle-Management Software

Dale Scott Caffall and James Bret Michael

Naval Postgraduate School

Abstract

Given that battle-management solutions in system-of-systems environment are separately designed and developed on various operating platforms in different languages, predicting battle-management behavior of system-of-systems is not possible. As a rule, battle management is executed at the system level rather than the desired system-of-systems level.

Typically, C4 systems are non-real-time systems. Traditionally, weapon systems are real-time systems. If we are to match the performance of weapon systems and avoid the negative impact of forcing synchronization of battle manager software with weapon systems for messaging, then we must develop the battle manager as real-time software.

We advocate the development of battle-management software as a real-time set of system functionality that addresses warfighter usage. To achieve the level of desired predictable battle-management behavior, we maintain that it is essential to develop a formal representation that captures the desired battle manager system behavior and test the formal representation against the expected battle-management properties.

Furthermore, we assert that it is critical to develop the battle manager as a real-time software-intensive system to ensure the schedulability of battle-management tasks and provide for concurrent execution of such tasks where applicable.

Introduction

The annals of human conflict are replete with the terrible results of the traditional war strategy of attrition in which opposing forces attempt to inflict more casualties on the enemy than the enemy can sustain and maintain a viable military force. This "mass-on-mass" strategy resulted in staggering losses of life in countless wars. For example, 623,026 soldiers lost their lives in the four years of the U.S. Civil War. At Antietam, the combined casualties of Union and Confederate forces totaled 26,134 soldiers on a single day of battle. [Leckie 90] The war of attrition concept was a costly strategy in terms of human life.

During the past decade, the Department of Defense (DoD) shifted military tactics from the traditional war of attrition to a transformational concept of full-spectrum dominance: the ability of US forces, operating unilaterally or in combination with multinational and interagency partners, to defeat any adversary and control any situation across the full range of military operations.

In Joint Vision 2020 (JV 2020), the Chairman, Joint Chiefs of Staff includes the following operational concepts that will support the achievement of full-spectrum dominance [Chairman 00]:

1. Dominant maneuver is the ability of joint forces to gain positional advantage with decisive speed and overwhelming operational tempo in the achievement of assigned military tasks.
2. Focused logistics is the ability to provide the joint force with the right personnel, equipment, and supplies in the right place, at the right time, and in the right quantity, across the full range of military operations.
3. Full dimensional protection is the ability of the joint force to protect its personnel and other assets required to decisively execute assigned tasks.
4. Precision engagement is the ability of joint forces to locate, surveil, discern, and track objectives or targets; select, organize, and use the correct systems; generate desired effects, assess results; and reengage with decisive speed and overwhelming operational tempo as required, throughout the full range of military operations.

We find the importance of battle management within the concept of precision engagement. For this technical report, we will define battle management as the decisions and actions executed in direct response to the activities of enemy forces in support of the Joint Chiefs of Staff's concept of precision engagement. [DA 96] Battle managers must rapidly make decisions to counter both enemy actions and force movements. Battle managers must correctly cope with the fog-of-war conditions that are ever-present during the prosecution of the war. The success or failure of the battle-management functions will determine the success or failure of joint forces with respect to the achievement of their assigned objectives. [Douglas 99]

Hypothesis

We believe that it is possible to develop a globally distributed, real-time software-intensive battle-management system that exhibits highly predictable system-software behavior, in which the system receives sensor information from land, sea, air, and space, and commits land-, sea-, air-, and space-based weapons to fire at identified targets. Furthermore, we believe that it is possible to employ linear temporal logic and model checking to a globally distributed, real-time battle-management system to develop desired system behavior to include the weapons-commit logic. We believe that the concepts in this technical report will extend the software engineering body of knowledge as follows:

1. Demonstrate that acquisition organizations can develop real-time software-intensive distributed systems that exhibit a high degree of predictability of system behavior.
2. Provide an engineering-based approach for developing a battle-management kernel (BMK) for missile-defense and other types of real-time systems used by combatant commands.

Battle-management software development concept

Ballistic missile defense (BMD) serves as the case study of the complex acquisition of battle-management systems because BMD encompasses an amalgamation of characteristics that are not found in single-system developments.¹ Some of the key characteristics of this system-of-systems are the following: (1) a globally-distributed network, (2) an operational battlespace that includes land, sea, air, and space, (3) capability to address multiple targets that can threaten a specific theater of operations or region of the world,² (4) management of concurrent battlespace activities, (5) some level of automated decision making regarding the release or hold of lethal weapons, and (6) stringent requirements for high levels of trustworthiness of the systems that provide BMD capabilities due to the fact that the threats to be encountered consist of weapons of mass destruction. Item number six makes unpredictable system behavior untenable from the public-policy, functional, and safety perspectives.

A BMK consists of the set of system components that are necessary to provide correct real-time execution of battle-management tasks in a system-of-systems context, both in nominal and degraded modes of system operation. To provide the reader with concrete examples of how we propose to design and test such a kernel, we describe some of the software-intensive aspects of battle management, including exercising rules of engagement, performing discrimination and correlation, conducting feature-aided tracking, and estimating the launch, impact, and intercept points of threat missiles.

A BMK is similar in purpose to an operating system (OS) kernel in that both kernels manage resources shared by competing entities. In the case of an OS kernel, the competing entities are computer processes vying for resources such as the CPU and memory. In the case of a BMK, the competing entities are all of the components of the system-of-systems that comprise the battle-management system, such as the C2 and weapon systems. The components in the kernel are expected to be stable compared to the other components in the system-of-systems. For instance, device drivers tend to be updated frequently and therefore in principle should not be included in the operating system kernel. If they are included, and even worse, tightly coupled to OS management functions, then it becomes challenging to make modifications to the kernel that do not affect other parts of the kernel. We would like to apply this same reasoning to BMK in order to simplify the design and maintenance of the kernels.

We also draw a parallel between BMK and safety kernels. The functions to be included in a safety kernel are those that must be performed to maintain a safe system state or bring a system back into a safe state after the occurrence of a safety-critical event. No other functions may be included in a safety kernel. An automated train protection (ATP) system is an example of a safety kernel. Such kernels are well documented, validated,

¹ A single-system development is the creation and maintenance of a system that is intended to be operated in isolation of other systems, or alternatively is intended to operate in a cooperative manner with other systems but the system is capable of operating in a standalone configuration; synonyms for this type of system include "stovepipe" or "standalone" system. A system-of-systems development is an antonym for single-system development.

² The primary mission of the U.S. Ballistic Missile Defense System is to protect the United States and its territories from exo-atmospheric threat missiles. However, the U.S. must also weigh the benefits and risks of engaging such threats within the airspace of other nations.

and verified before being considered for certification and accreditation. We view battle-management kernels in a similar light: they must work as advertised because the ability of the entire system-of-systems to be able to conduct warfare in the BMD battlespace is dependent on the BMK.

In our proposed approach, we envision software engineers developing the BMK as a real-time set of system functionality that addresses its use by warfighters, starting from a high-level statement of capabilities and refining these statements into successively lower levels of system artifacts. We define the BMK to be the software that contains the basic functions of battle management that will remain stable over time. Derived from the kill chain, these basic battle-management functions are called tasks, and will manage the use of the system's computing resources to ensure that all time-critical, battle-management events are processed as efficiently as possible.

In the context of DoD capability-based acquisition, the government specifies the capabilities for the system that are needed by the warfighter. The government contracts specify and refine the capabilities into system requirements, architectures, designs, and other system artifacts. In [Caffall 03], we demonstrate how the Unified Modeling Language (UML) can be used to refine a system-of-systems. In this report we extend our earlier investigation to include the explicit treatment of linear temporal logic for developing the BMK functional specifications and verifying the specifications using model checking.

Discussion

Battle management relies on two functions that influence the outcomes of battles: planning and command and control (C2). For this technical report, we define planning as that military planning that produces either an Operation Plan (OPLAN) or an Operations Order (OPORD) to employ military force against an adversary. We define C2 as the exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of the mission.

Planning includes the initial lay-down of joint and coalition forces, rules of engagement, provisioning, and re-supply. Planning "sets the table" for the military and establishes the initial ruleset that the warfighters will follow at the onset of the battle. Planning is a coordinated joint staff procedure used by a commander to determine the best method of accomplishing assigned tasks and to direct the action necessary to accomplish the mission. [JCS 03] Planning includes both the deliberate planning and crisis-action planning (CAP). Combatant commanders (COCOMs) conduct deliberate planning to develop a military response to a future hypothetical contingency while CAP takes place in response to a crisis in which the United States' national security interests are threatened and the President is considering a military response. [JFSC 00]

C2 functions are performed through an arrangement of personnel, equipment, communications, facilities, and procedures employed by a COCOM in planning, directing, coordinating, and controlling forces and operations in the accomplishment of the mission. [JCS 03] Through C2, the senior military leadership modifies and enhances the initial ruleset that governs the battlespace. (N.B.: Battlespace is defined as the environment, factors, and conditions that must be understood to successfully apply combat power, protect the force, or complete the mission. This includes the air, land, sea,

space, and the included enemy and friendly force; facilities; weather; terrain; electromagnetic spectrum; and the information environment within the operational areas and areas of interest. [JCS 03])

Recall from previous discussion that the Joint Staff defined Precision Engagement as follows:

...the ability of joint forces to locate, surveil, discern, and track objectives or targets; select, organize, and use the correct systems; generate desired effects, assess results; and reengage with decisive speed and overwhelming operational tempo as required, throughout the full range of military operations. [Chairman 00]

The basic construct of the definition is the identification of the functional flow of military activities that must occur to engage a threat object. This functional flow of military activities is colloquially known as the kill chain.

Rather than capriciously defining a kill chain for the battle-management function, we treat the functional flow of events that occur in the engagement of a military threat, starting with an examination of the original work of Colonel John Boyd (USAF, Ret.) and followed by the Navy's functional construct for missile defense, the Army's functional flow of events for deep operations, the Air Force's kill chain, and the Joint Chiefs of Staff's functional flow of events for theater ballistic missile defense (TBMD).

Observe-orient-decide-act

Colonel John Boyd was an avid student of military engagements. From his analysis of the engagement actions of commanders and famous battles, he formed a concept of what is known today as the Observe-Orient-Decide-Act (OODA) loop. He noted that in many of the engagements, one military force presented the other with a series of unexpected and threatening situations with which they had not been able to keep pace. The faster military force eventually defeated the slower military force. Boyd observed that military conflicts are time competitive.

In the OODA Loop, Boyd incorporated a temporal aspect in his analysis of military decision-making before and during battle. Decisions and actions that are delayed are often rendered ineffective because of the constantly changing circumstances. When a military adversary is involved, the operation is not only time-sensitive but also time-competitive. Time or opportunity neglected by one adversary can be exploited by the other. [Coram 02]

According to Boyd, military conflict can be seen as a series of time-competitive cycles through and OODA loop. Each military force in a conflict begins by observing themselves, the physical surroundings, and the adversary. Next, the military force orients itself; orientation refers to making a mental image or snapshot of the situation. Orientation is necessary because the fluid, chaotic nature of conflicts makes it impossible to process information as fast as military commanders can observe it. This necessitates applying a freeze-frame concept and provides a perspective or orientation.³ Once we have an orientation, military commanders must make a decision. The decision takes into

³ This is analogous to creating a materialized (i.e., stored) view of data by querying a database.

account all the factors present at the time of the orientation. Finally, the military commander must implement the decision. This requires action. One tactical adage states: "Decisions without actions are pointless and actions without decisions are reckless." Then the cycle begins anew as military commanders believe that their actions will have changed the situation. The cycle continues to repeat throughout a tactical operation. [Boyd 86]

The military force that can consistently go through the OODA loop faster than the other enemy force can, *ceteris paribus*, gains a tactical advantage. By the time the slower adversary reacts, the faster force is doing something different and the slower adversary's action may become ineffective. With each cycle, the action of the slower military force becomes increasingly ineffective by an increasingly larger margin. The aggregate resolution of these episodes will eventually determine the outcome of the conflict. For example, as long as the actions of the faster military force continue to prove successful, the slower military force will remain in a reactive posture while the commander of the faster military force maintains the freedom to act. No matter how desperately the slower military force strives to accomplish its military objectives, every action becomes less useful than the preceding one. As a result, the slower military force falls farther and farther behind. [Boyd 86] [Coram 02]

Detect, control, engage

At a Millennial Challenges Colloquium presentation in April 2000, Vice Admiral Rodney Rempt (then Rear Admiral and Deputy Assistant Secretary of the Navy for Theater Combat Systems) discussed Naval theater air and missile defense for the twenty-first century. He observed that some level of defense is the "price of admission" for carrying the battle to the shores of potential adversaries. He discussed the threat to the Fleet of cruise missiles, ballistic missiles, fighter-bombers, and unmanned aerial vehicles (UAVs); these threats are steadily increasing in lethality, accuracy, and range. Hence, Vice Admiral Rempt concluded that the Naval theater air and missile defense must formulate and apply a concept of Detect, Control, and Engage. [Rempt 01]

For the detect aspect of Naval theater air and missile defense, the concepts of multi-spectrum sensor netting and data fusion must be realized from a variety of active sensor arrays, passive staring infrared sensors, and bistatic sensors. The timely and accurate detection of current and future threats is absolutely essential in triggering military action to negate the threat.

For the control aspect, the Navy must realize a network of planning tools, automated decision aids, and the single integrated battle space. The Navy must develop solutions to potential threats before the threats are realized. Planning and identifying potential engagement zones, rules of engagement, and consequence management will become critical to the success of Naval theater air and missile defense.

For the engage aspect, the Navy must be able to deliver the appropriate force to negate current and future threats to the Fleet and its defended assets. The received information must be processed in a timely fashion so that Naval officers can make timely decisions for engaging potential threats. Indecision due to inconclusive or untimely information will have catastrophic consequences to Fleet assets.

Decide, detect, deliver, and assess

The Army defines targeting as the process of selecting targets and matching the appropriate response to them on the basis of operational requirements and capabilities. COCOMs use the functional construct of *decide, detect, deliver, and assess* to transform a COCOM's targeting intent into an engagement.

The emphasis of targeting is on identifying resources the enemy can least afford to lose or that provide him the opposing force with the greatest advantage, then further identifying the subset of those targets which must be acquired and attacked to achieve success. Denying these resources to the enemy make the enemy's military assets vulnerable to COCOMs' battle plans. These resources constitute critical enemy vulnerabilities. Successful targeting enables the COCOM to synchronize intelligence, maneuver, fire-support systems, and in addition to special operations forces, by attacking the right target with the best system and munitions at the right time.

The decide function, as the first step in the targeting process, provides the overall focus and sets priorities for collecting intelligence and planning attacks. Targeting priorities must be addressed for each phase or critical event of an operation.

Detect is the next critical function in the targeting process. The intelligence cell is the main figure in directing the effort to detect high-payoff targets identified in the decide function. This process determines accurate, identifiable, and timely requirements for collection systems.

The deliver function of the targeting process executes the target attack guidance and supports the COCOM's battle plan once the high-payoff targets have been located and identified. Some targets will not appear as anticipated. Target attack takes place only when the forecasted enemy activity occurs in the projected time or place. The detection and tracking of activities associated with the target becomes the trigger for target attack.

Combat assessment is the determination of the effectiveness of force employment during military operations. On the basis of battle damage assessment (BDA) reports, the COCOM continuously estimates the enemy's ability to make and sustain war and centers of gravity. During the review of the effects of the campaign, re-strike recommendations are proposed or executed. BDA is the timely and accurate estimate of damage resulting from the application of military force, either lethal or non-lethal, against a target. BDA in the targeting process pertains to the results of attacks on targets designated by the commander. [DA 96]

Find, fix, track, target, engage, assess

According to General John Jumper (Chief of Staff of the United States Air Force), today's Air Force is a "community of stovepipes." General Jumper wants to achieve horizontal integration that he defines as the "...ability to fuse data from every Air Force platform into a single repository of information, such as crews, planes, targets, and loads." His vision is to achieve horizontal integration is the accomplishment of the entire "kill chain" from a single source of information. General Jumper defines the kill chain as *find, fix, track, target, engage, and assess*. [Erwin 02]

As avowed by Lieutenant General Leslie Kenne (Air Force Deputy Chief of Staff for Warfighting Integration), the Air Force must "close the seams" in the kill chain by "integration of manned, unmanned, and space systems." Historically, technology limited the flow of information. Battlefield information delivery was limited to the speed of the horses and the ability of the commander to assess the battlefield information from afar. Execution was centralized as only the commander had the situational awareness of the entire battlefield. Consequently, reinforcement troops had no time to gain situational awareness. Thus, troops had to rely on their commander to direct their movements and placements, and hoped that the enemy had not conducted movements that countered the commander's situational awareness. [Kenne 03]

Today, technology provides the potential to maintain situational awareness for the entire military force. The military has developed an interconnected network of information with the objective of providing timely and accurate information to all points of the battlespace. The stovepipes discussed by General Jumper prevent the achievement of this objective and prevent effective battle-management in the battlespace.

Detect, identify, locate, track, destroy

In recent years, the threat of missile attack to American forces and allies in foreign lands has dramatically increased. Numerous nations own missiles that has forced the United States to address this potential threat. The proliferation of theater missiles, advances in missile technology, and the pursuit of weapons of mass destruction have provided potential adversaries with a lethal-attack capability against United States' interests.

As outlined by the Joint Chiefs of Staff, theater missile defense applies to the "...identification, integration, and employment of forces supported by other theater and national capabilities to *detect, identify, locate, track, minimize the effects of, and/or destroy* enemy [theater missiles]." Through this process, military commanders should be capable of countering threats from theater missiles and have the capability for rapid global deployment and theater mobility. [JCS 96]

For this technical report, we will employ a kill chain that is defined by the following five functions: ***Detect, Track, Assign Weapon, Engage, and Assess Kill***. These five functions address all the functions outlined in the definition of precision engagement to which the Joint Chiefs of Staff subscribe, in addition to all of the functions identified in the Boyd, Navy, Army, Air Force, and Joint Chiefs of Staff functional models.

Of the five kill chains described in the preceding paragraphs, only the Army and the Air Force identified an assess function that is required to determine whether the threat object is indeed negated. The assess function is essential to complete the engagement as defined by the Precision Engagement. The *fix* function of the Air Force kill chain is captured within the *track* function of our defined kill chain.

As can be observed in Table 1, the proposed kill chain proposed is complete with respect to addressing the major functions required to negate a threat object.

Table 1. Summary of Kill Chains

Boyd	Navy	Army	Air Force	JCS	Technical Report
Observe	Detect	Decide Detect	Find	Detect	Detect
Orient			Fix Track	Identify Locate Track	Track
Decide	Control		Target		Assign Weapon
Act	Engage	Deliver	Engage	Destroy	Engage
		Assess	Assess		Assess Kill

Statement of the problem

To appropriately frame the problem, we would once again recall the Joint Chiefs of Staff's definition for precision engagement:

...the ability of joint forces to locate, surveil, discern, and track objectives or targets; select, organize, and use the correct systems; generate desired effects, assess results; and reengage with decisive speed and overwhelming operational tempo as required, throughout the full range of military operations. [Chairman 00]

In March of 2003, Joint Forces Command identified eight key shortfalls in the desired achievement of effective Joint Task Force Command and Control. Those shortfalls include incomplete shared situational awareness, inadequate information superiority, and insufficient joint and coalition interoperability. [JFC 03] (N.B.: For this report, we define interoperability as the ability of systems, units, or forces to provide services to and accept services from other systems, units, or forces and to use the services so exchanged to enable them to operate effectively together.)

In a December 1997 report to Congress on the National Missile Defense (NMD) system, the General Accounting Office (GAO) identified technical issues in discrimination and data fusion. Additionally, the GAO reported the average time to develop major weapon systems is 9.9 years based on an analysis of fifty-nine acquisition programs. [GAO 97] This observation bolstered the claims of defense acquisition critics that development cycles are too long, too costly, and provide too little required functionality.

In his book "Software Fundamentals: Collected Papers by David L. Parnas," Dr. Parnas outlines six major characteristics of the battle-management software in the Strategic Defense Initiative (SDI) program (known today as the Ballistic Missile Defense System).

[Parnas 01] The below issues are as relevant today as during the time when Dr. Parnas published his observations:

The battle-management software must identify, track, and direct weapons towards targets whose characteristics may not be known with certainty until the moment of battle. The battle-management software must discriminate the threat objects from decoys and debris.

1. The battle-management computing will be accomplished through a network of computers that are connected to sensors and weapons as well as other battle-management computers. The behavior of the battle-management software cannot be predicted with confidence given the actual configuration of weapons, sensors, and battle managers at the moment of battle.

2. Developers cannot test the battle-management software under realistic conditions prior to actual use of the software.

3. The duration of the defense engagement will be short: it will not allow for either human intervention or debugging the software to overcome software faults at runtime.

4. The battle-management software will have absolute real-time deadlines for the computation that will consist of periodic processes to include detecting and identifying potential threat missiles, assigning a weapon to engage the threat missile, and providing an assessment of the interceptor-threat missile engagement. Because of the unpredictability of the computational requirements of each process, developers cannot predict the required resources for computation.

5. The missile defense system will include a large variety of sensors, weapons, and battle-management components for which all will be large, complex software systems. The suite of weapons and sensors will increase in number as the development progresses. The characteristics of these future weapons and sensors are not well defined and will likely remain fluid for many years. Additionally, all weapons and sensors will be subject to change independently of each other. As such, the battle-management software must integrate numerous dynamic software systems to the extent that has never before been achieved.

Given the above observations, we believe that the battle-management software must overcome the problems that are summarized in Table 2.

Table 2. Summary of Issues

Issue	Comment
Incomplete Shared Situational Awareness	Because of the numerous stovepipe developments, situational awareness is inconsistent among the networked platforms.
Inadequate Information Superiority	Because of the numerous stovepipe developments, information tends to remain within a single system platform.
Insufficient Joint/Coalition Interoperability	Because of our attempt to network our systems through interconnection rather than integration, joint and coalition remains limited in the operational battlespace.
Inadequate Discrimination	Because of differing algorithms in our systems as well as limitations in our interconnectivity solutions for networking, our systems cannot quickly and accurately discriminate decoys and debris from actual threat objects.
Unpredictable System Behavior	Because of the stovepipe developments of our systems and the interconnectivity solutions for our networking, the behavior of our systems are largely unknown.
Inadequate System Testing	Because of the shortened acquisition timelines as well as the increased complexity of our systems, system testing is primarily that of a test of selected functional threads.
Critical Software Faults at Runtime	Because of the short time duration of military engagements and the safety-critical nature of our systems, we must field our systems with undiscovered major critical software faults.
Real-Time Computational Deadlines	Because of the short duration of military engagements and the intensity of the battlespace, we must know that the most critical software tasks will be executed without fail.
Complex Integration	Because of the dynamic nature of potential threats and the requirement to enhance our systems as the threat increases in delivery and lethality, we must have the ability to quickly modify our systems and function as a single system-of-systems.
Prolonged Development Cycle	We must develop acquisition methodologies to reduce the acquisition cycles from nearly ten years to under a year to achieve essential military capabilities.

Significance of the problem

Given that the interconnected battle-management solutions in the system-of-systems environment are separately designed and developed on various operating platforms in different languages, predicting battle-management behavior of the system-of-systems is not possible. As a rule, battle management is still executed at the system level rather than the desired system-of-systems level. (N.B.: In this technical report, we define a system-of-systems as an amalgamation of legacy systems and developing systems that provide an enhanced military capability greater than that of any of the individual systems within the system-of-systems.)

Another factor that contributes to the challenge involved in predicting battle-management behavior is the acquisition practices currently employed in DoD. The increased pressure to rapidly move product into the operational battlespace tends to channel program managers into focusing on achieving functionality as quickly as possible. As such, the development community responds with a hurried and oftentimes inadequate design phase and follows with an intense period of coding. In the rush to rapidly develop a product, one can fall into the trap of exclusively seeking some level of achieved capability while ignoring the behavior of the software.

Because we cannot readily predict the system behavior of legacy battle-management systems, we tend to fulfill battle-management requirements as a new development. While the basic five functions do not change from system to system and from year to year, we choose to acquire a new battle-management system as a new development. What changes are the sensors used to collect information for the warfighters, the weapons used to engage threat targets, and the rules of engagement (ROEs) established in both the planning and the C2 functions.

Specific features within the battle-management software will change over time (e.g., discrimination algorithms, correlation algorithms, feature-aided tracking); however, we can isolate those features in components that can be interchanged at a time when developers are prepared to introduce new technology into the battle-management software.

Focus and scope

We recommend that the DoD consider adopting an architectural framework that treats the BMK as the meta component that binds the system-of-systems together. The BMK will transcend time in the sense that we envision it will be relatively stable and unchanged as compared to the components interfaced to the BMK.

We advocate the development of battle-management software as a real-time set of system functionality that addresses warfighter usage. To achieve the level of desired predictable system behavior, we maintain that it is essential to develop a formal representation that captures the desired system behavior of the battle manager and to test the formal representation against the expected battle-management properties, such as schedulability and concurrency.

From the table of identified battle-management issues (*vid.* Table 2), we will address those identified issues in the BMK.

Inadequate information superiority

We propose a software architecture that allows for different sensors to provide information to the BMK. In most weapon systems, the sensor is one of three major components: sensor component, weapon component, and a C2 component. (N.B.: In this technical report, we define a component as a software unit of composition with contractually specified interfaces and explicit context dependencies.)

Typically, the sensor information is processed locally within the weapon system. If information is shared with other weapons systems, the control component transmits processed information onto a network within the confines of the network protocol (e.g., Link 11, Link 16, Tactical Information Broadcast Service (TIBS), Tactical Related Applications (TRAP), Variable Message Format (VMF)). Information loss or misinterpretation of the information can result from the translation of one protocol to another, or between systems employing different implementations of the same protocol.

We propose to treat sensor as a separable component and connect it to the BMK as depicted in Figure 1. We will pull selected information from the sensor processor and transmit that information to the BMK. The BMK will process the information and provide correlated tracks to the connected C2 centers.

Inadequate discrimination

We will not attempt to explicitly address the discrimination problem in this technical report; however, we recognize discrimination as a feature that could change frequently as developers introduce new technology and new algorithms to battle-management systems. As such, we will use a software component to isolate discrimination as depicted in Figure 1 so that future upgrades such as advanced discrimination algorithms can be inserted in the discrimination component of the framework.

Incomplete shared situational awareness.

We propose a software framework that allows for a common scheme for correlation of track information from different sources and providing that correlated information to command and control centers. We propose that correlation software be developed and maintained as a software component that interfaces with the BMK.

Insufficient joint/coalition interoperability

The integration of legacy systems into a system-of-systems is a difficult task for acquisition organizations for many reasons to include coupling and cohesion that result in limited interoperability among systems. Our system-of-systems are interconnected systems that display a high degree of coupling and a low degree of cohesion. [Caffall 03] Legacy systems within the system-of-systems are based on different technologies and different implementations. As such, with the high degree of coupling, modifications to the software in the legacy systems could have a negative ripple effect in the behavior of the system-of-systems. Critics of system-of-systems acquisitions perceive a limited development process that does not consider the integration of timing predictability and fault-tolerant characteristics. [Meyers 01]

It is neither cost- nor time-effective to rewrite all the software in the BMD sensors, weapons, and C2 components. As previously noted, we developed each system independently of all the other systems. As such, each system's software is different with respect to architecture, design, missions and functions, language, operating systems, and persistent data storage schemes.

We believe that it is possible to develop an integrated BMK that significantly reduces the messaging among the systems in existing system-of-systems by using state changes in shared memory. This in turn, *ceteris paribus*, could increase the degree of interoperability between the battle-management function and the sensors, weapons, and C2 components. [Stewart 01] With the development of the BMK as real-time software, we can match the performance of the weapon systems and avoid the negative impact of forced synchronization of the battle manager with the sensors, weapons, and C2 components in the system-of-systems.

Rather than depending on a universal interface protocol such as military standard for Link 16 [DOD 02], we propose type interfaces for the BMK. We believe that we can achieve a higher degree of interoperability than what we currently experience by developing a number of smaller interfaces into the BMK rather than a single large interface as in the Link 16 military standard.

We recommend capturing desired system behavior in the interface definition rather than depending solely on messaging requirements as in the Link 16 military standard. Developers should maintain the BMK interfaces as separable, configurable items from the BMK. We propose constructing interfaces by type for BMD; that is, software engineers should consider constructing interfaces for ballistic missile defense elements such as an infrared sensor type, a radar type, a kinetic energy weapon, and a directed energy weapon. They must require that each instantiation of an interface type will include all the attributes and operations of its parent type similar to the concept of class inheritance in which a subclass inherits attributes and operations from its parent. [Booch 94] We propose that the definitions interfaces follow the conventions set forth in [Bachman 02]:

Interface Identity. Identify the name, component type, and version. Each interface will have a unique identify.

Interface Usage Definition. Specify the overall system behavior for which the interface will provide or receive services or data. Additionally, we will identify system-timing requirements for the interface.

Provided Resources. Identify the specific resources that the interface provides to the BMK. This will include the information that other software programs will require to invoke the interface as well as the result of invoking the interface. Additionally, we will provide interface usage restrictions that define under what conditions the interface may be invoked.

Defined Data Type. Identify the data types used in the interface to include the programming language, declaration of variables and constants, operations that may be performed on data types, and instructions on how to convert the values of the interface data type into other data types.

Error-Handling Capability. Provide the error conditions that might occur through the interface (e.g.; message translations from one protocol to another, out of bound values, illegal values). Additionally, provide the error-handling behavior of the interface.

Interface Characteristics. Identify the characteristics of the interface attributes to include data precision, data formats, reliability, timing, and constraints.

Interface Requirements. Identify the resources required from the software component that invokes the interfaces to include syntax, semantics, and usage restrictions.

Unpredictable system behavior

As previously discussed, software engineers cannot easily determine the system behavior of current battle-management systems. In the domain of the BMD, the BMK behavior must be predictable and must respond immediately to hostile actions of potential adversaries. The warfighters must have confidence that the battle-management software will perform its critical tasks as designed, and will not exhibit inappropriate system behavior such as reporting false ballistic-missile threats and issuing engagement commands for non-existent ballistic-missile threats.

In the operational battlespace, the BMK will control the behavior of various weapon systems over a global control network. Based on processed information within the BMK, it will report ballistic-missile threats to all layers of management up to the President of the United States. The BMK will assign a weapon to engage each detected ballistic-missile threat and order the engagement of each ballistic-missile threat.

It is neither feasible nor cost-effective to rewrite all the software in the sensors, weapons, and C2 components. As previously discussed, we developed each system independently of all the other systems. As such, each system's software is different with respect to architecture, design, missions and functions, language, operating systems, and persistent data storage schemes.

We believe that what is possible is to develop the BMK with predictable system behavior. Given that the BMK determines the existence of a ballistic-missile threat and orders the engagement of the ballistic-missile threat, we believe that predictable system behavior of the BMK will significantly improve the overall predictability of the BMD system-of-systems.

To achieve the level of desired predictable BMK behavior, software engineers could develop a formal representation that captures the desired system behavior of the BMK and verify the formal representation against the expected BMK properties.

We recommend that software engineers develop the formal representation of the BMK by using metric temporal logic (an extension of linear temporal logic [Drusinsky 02]) to describe the BMK functional specifications. To avoid the state-explosion problem, software engineers must carefully model the fundamental behavior of the BMK rather than a comprehensive specification of the BMK. [Gluch 99] [Lewis 01] Recall that the BMK will be real-time software. Therefore, the formal model of the BMK should be state-based as we desire to effect change in the sensors and weapons through state changes rather than messaging.

For example, consider the assigning of a weapon to an identified threat-ballistic missile. In typical C2 systems, the battle-management function sends a message to each weapon that a threat exists. After some time, the battle-management function sends a message to a specific weapon to engage the threat-ballistic missile and sends messages to the other weapons not to engage the threat-ballistic missile. In the BMK, we will use logic to determine the threat exists and change the state status of the ballistic-missile threat in shared data memory from INACTIVE to ACTIVE. After using logic to determine which weapon has the best shot opportunity, the BMK will assign that weapon to engage the active ballistic-missile threat by pairing the weapon to the ballistic-missile threat and changing its engagement status from INACTIVE to ENGAGE in the shared data memory.

Software engineers should consider the use of an automated model-checking tool to verify that the formal BMK model reaches each desired state as designed. The automated tool will determine whether the logic is appropriate to reach each state and whether the logic prevents reachable states inappropriately. With respect to the real-time aspects of the BMK software, we will use the model checker to: (1) determine whether a deadlock condition occurs, (2) ensure that the BMK executes the critical tasks under all battlespace conditions to include overload conditions, and (3) the system reaches each desired state within its time constraints. [Guaspari 00] The result of the state-based model verification will be the substantiation or repudiation of the desired BMK behavior. [Lewis 01]

Software engineers could address the so-called "state-explosion problem" by employing symbolic model checking. That is, they should abstractly represent a set of states by using a compact description rather than an explicit listing of all states. [Gallardo 03] The abstract model should be effective in uncovering BMK behavior errors with only a portion of the state space explored. [Chen 03]

Inadequate system testing

By incorporating assertions developed from the functional model and verified by the model-checking effort into the BMK, software engineers can develop embedded automatic test generation capabilities. Assertions have multiple benefits to include automated testing without pre-generation of expected results, debugging the BMK software, and reduced diagnostic time for identifying the subtle bugs within the BMK software. [Binder 01]

Critical software faults at runtime

Software engineers could incorporate assertions and error-handling schemes developed from the functional model and verified by the model-checking effort into the BMK. The error-handling schemes for breaks in logic will benefit the warfighters by either developing an automated logic-break recovery or notifying the warfighters of required manual actions.

Real-time computational deadlines

System-of-systems functional and performance expectations of the users continue to increase as the acquisition community continues to develop and field the products of C4 systems and weapon systems integration. The class of systems in which C2 and Battle-management are contained are called Command, Control, Communications, and Computers (C4) systems. Typically, C4 systems are non-real-time systems.

Traditionally, weapon systems are real-time systems. [Meyers 01] If we are to match the performance of the weapon systems and avoid the negative impact of forcing synchronization of the battle manager with the weapon system for messaging, then we must develop the battle manager as real-time software.

We advocate developing the BMK as real-time software that will be run on top of a hard-tasking, real-time operating system to ensure the schedulability of battle-management tasks and the concurrent nature of battle-management systems. A real-time defense system must exhibit the following behaviors [Douglas 99] [Sha 93]:

1. Predictable and immediate response to precarious battlespace conditions.
2. High degree of task schedulability. (N.B.: For this report, schedulability is defined as the degree of resource utilization for which the timing requirements of tasks can be assured.)
3. Stability under transient overload. If the real-time defense system is overloaded by multiple battlespace conditions and the system cannot meet all of its scheduled deadlines, then the real-time system must guarantee that it will meet the deadlines of the most critical tasks.

Frequently, non-real-time systems implement inputs and outputs as messages that works well in a non-real-time environment. Message passing in real-time systems does not work well for the following reasons [Stewart 01]:

1. Message passing requires synchronization between the message sender and the message receiver. This is a significant source of unpredictability in real-time scheduling of software tasks given that functional blocks of code execute synchronously to pass messages. Consequently, the analysis of the real-time system timing may prove to be impossible.
2. A significant opportunity for deadlock exists in real-time systems that attempt to incorporate either bi-directional communication between software processes or a messaging feedback loop. A better solution would be to use a state-based system. Software processes can bind to a single element in a state variable table. This would help to eliminate synchronization dependencies among software processes.
3. Messaging software schemes require significantly more overhead (e.g., error correction coding, interleaving methods, messaging protocol communications) than systems that use shared-memory techniques. Although messaging may be necessary for communications across networks, messaging is not efficient if random-access to the data is possible as is the situation for communications among software processes within a single processor.

Complex integration

We will identify the BMK functions that will experience frequent change during the operational phase of the acquisition lifecycle. We propose that these functions be transformed into software components (as defined in [Szyperski 02]) in order to reduce the complexity of software integration.

Prolonged development cycle

We believe that the BMK can serve as the basis for future battle-management systems. We believe that a developer could use the BMK as the core element to connect sensors, weapons, and user displays to provide the essential basic battle-management capabilities in a timeframe measured in months rather than years.

BMK development strategy

We believe that software engineers should consider developing the BMK as a real-time set of system functionality that addresses warfighter usage with respect to the kill chain. In this approach, they would develop a framework that contains the proposed BMK as well as contains battle-management software components that will experience the most change during the acquisition life cycle of a battle-management system.

As the initial step to the BMK development, we recommend performing a domain analysis of the battle-management functions. During this type of analysis, software engineers could derive warfighter usage requirements from battle-management use cases. They could refine the use cases as we develop sequence diagrams to depict the messaging requirements among the derived classes from the use cases. Software engineers could develop a state diagram for the BMK to identify the desired battle-management behavior. To conclude the domain analysis, they should identify and verify assumptions on battle-management operations to support the development of BMK specifications.

From the iterative review and refinement of these artifacts, software engineers could develop detailed specifications that focus on defining BMK behavior and achieving battle-management goals. They should consider the use of logic to describe the BMK specifications.

We recommend the verification of the functional specifications with the use of a model-checking tool to determine the degree of system behavior predictability with respect to state transitions and tolerance to battlespace environmental variables. The verification should focus on ensuring that the BMK can meet the specifications and exhibits the desired behavior. Software engineers might design test oracles that contain the full range of battle-management variables that are both within and outside the expected range of operational values for the ballistic missile defense.

Plan of execution for the BMK design and development

The BMK will act as "glueware" between software applications unique to each battle-management domain, and the sensors, C2 systems, and weapon systems in that battle-management domain. That is, the BMK will execute the five kill-chain functions by calling upon various components for computation.

We recommend the identification of the required interfaces into the BMK include sensors, weapons platforms, and C2 systems. Rather than point-to-point interfaces, we propose the development of type interfaces that define the behavior of each interface and the required specifications to realize each interface. We propose that the interfaces be developed and maintained as separate configurable items to preserve the identity of the interface and to minimize the opportunity for multiple versions of the interface. Additionally, it is important the interface identifies its operations and does not specify

implementations of its operations. [Crnkovic 02] This is frequently the case as interfaces are developed within the application software.

For ease of integration and maintainability, we propose the development of software components for the features that typically experience the majority of changes. Developers can realize a component-based framework with less effort and without unwieldy upgrade cycles as compared to fully integrated, monolithic software solutions. Additionally, a component-based framework allows for tailoring of the framework to address specific user needs. [Szyperski 02] For this report, we identify software components that include enforcing rules of engagement, conducting discrimination and correlation, performing feature-aided tracking, and estimating launch, impact, and intercept points.

Domain analysis

The first task is to construct a domain analysis of the BMD space to uncover the desired behavior of the BMK. Software engineers should base the domain analysis on the five functions of the kill chain identified for use in this technical report: Detect, Track, Assign Weapon, Engage, and Kill Assessment. The domain analysis should include use cases, sequence diagrams, and state diagrams. The goal should be to characterize the desired BMK behavior in the domain analysis. As a point of departure, software engineers can start with the conceptual framework described in [Caffall 03]. In the referenced work, we developed design artifacts for the ballistic missile defense system-of-systems. This work is directly applicable to developing the BMK given that the artifacts are modifiable to focus on the problem statement in this technical report.

Specifications

The second task will be to construct a set of specifications using temporal logic that will serve as a model of the BMK. The goal is to achieve a greater degree of clarity and focus in the specification of the desired BMK behavior as compared to traditional list of system requirements.

We recommend that software engineers develop a sufficient amount of information to automatically produce test cases for the implementation. Otherwise, they run the risk of developing so-called "cartoon models" that are only useful for drafting and refining potential solutions. Software engineers need to develop test-ready models of the BMK. In order to be testable, a model should contain all the features of the BMK, preserve sufficient detail that is critical for discovering faults, and faithfully represents the essential states, actions, and transitions in the state diagram. [Binder 01]

If the BMK model is to be useful for this effort and in future acquisition efforts, it must exhibit the following properties outlined in [Selic 03]:

Appropriate level of abstraction. A model is a representation of some entity. In the development of the representation, modelers abstract away details that is not necessary for others to gain an understanding of the fundamental nature of the represented entity. As modelers abstract away details, they must ensure that the core capabilities of the represented entity are captured in the model.

For the BMK model, we want to capture the essential functionality of the battle-management function and we want to capture the functionality defined in the interfaces to the BMK. It is not important to model the actual processing involved with each of the five functional areas; however, we must ensure that we model the generation of triggers for state transitions.

High degree of understandability. A model must describe the abstracted system behavior in a clear and logical manner to both the software engineer and the software maintainer. If either party cannot understand the model, then the model holds limited utility in the software lifecycle acquisition.

For the BMK model, we want to depict the battle-manager behavior in a logical fashion so that the designer can faithfully realize the BMK specifications in accordance to the artifacts developed in the domain analysis. Additionally, an understandable model supports software integration and software maintenance efforts.

High measure of accuracy. Although we desire to hide the unimportant details of the BMK, it is important to accurately specify the details and the associated parameters to ensure that the model will provide utility to the software engineer. Additionally, the model must yield outputs that are within defined error bounds to ensure the model faithfully represents the desired system.

For the BMK model, software engineers should capture the desired parameters in the logic statements. We must accurately capture the BMK response requirements such as the maximum allotted time from track identification to weapon assignment on that track. We must accurately capture the BMK required calculation requirements such as the location ellipse of a tracked object. We must accurately capture BMK limits such as the maximum number of concurrently tracked objects.

High level of predictiveness. The model must correctly and consistently mirror the behavior of the desired system. For example, given that the system is in a known state and given the known inputs, the model should transition to the appropriate state without fail.

For the BMK model, we must ensure that the model faithfully represents the behavior of battle-management operations. We must ensure that the BMK states are reached appropriately and the transition triggers are reflective of the projected BMD battlespace. For example, the model must transition from the state in which tracking occurs to the state in which a weapon assignment occurs each and every time the model is presented with the appropriate transition events. Just as important, the model must not transition for events other than what was designed for the BMK.

Software engineers should consider using temporal logic to define assertions for the BMK specifications. We believe that the use of assertions through temporal logic will yield specifications that are verifiably consistent and accurate. We believe that the use of assertions through temporal logic will result in verifiably predictable BMK behavior.

It is our experience that the vast majority of engineers involved with acquisition of software-intensive systems are not familiar with software formalisms. Additionally, we assert that few of the many system engineers in acquisition could follow temporal logic without some level of instruction. As such, software engineers may choose to minimize

the use of typical temporal logic symbols and attempt to develop the specifications in as close to natural language as possible while still maintaining the degree of rigor that temporal logic lends to specification development.

This approach is necessary to gain buy-in from system engineers and engineering managers. Acquisition efforts require significant commitments of human and financial capital. Introducing new acquisition methods to replace that which is familiar and comfortable is generally viewed as risky and foolish. Proposed changes must be readily evident to system engineers and engineering managers, or the proposed changes will not be adopted. As an example of this approach, we offer the following example of assigning a weapon to a tracked object:

User Goal: Assign a Weapon to a Tracked Object

Narrative: The BMK must assign a weapon to engage a threat object before it impacts or detonates over pre-designated defended area. The BMK must determine whether the tracked object is a ballistic-missile threat. The BMK must determine whether the predicted impact point is within the defended area as defined by military planners. The BMK must determine which weapons are available. The BMK must determine which weapon(s) can engage the tracked object. The BMK must assign the appropriate weapon to prosecute the engagement of the tracked object.

The logic to assign a weapon to a tracked object is as follows:

Weapon assigned to track object is true iff:

*(Tracked object is a ballistic-missile threat) &
(Predicted impact point is within defended area) &
(Weapon is available) &
(Weapon interceptor capability is adequate)*

We would outline the specification as follows:

Variables:

Boolean: Weapon_Assigned
// Weapon assigned to tracked object is true

Boolean: Ballistic_Threat
// Tracked object is ballistic-missile threat

Boolean: IPP_Within_Defended_Area
// This statement is true if the predicted impact point lies inside the physical dimensions of the defended area.

Boolean: Weapon_Available
// True if one or weapons are capable of immediately launching an interceptor.

Boolean: Intercept_Point_Min_Within_Intercept_Range
// True if the minimum intercept point lies within the interceptor range volume.

Boolean: Unknown_Track
 // True if track object has yet to be identified as a ballistic-missile threat

Set: Tracked_Object
 // Contains detected characteristics of a ballistic-missile threat

Multiset: Threat_Profile
 // Contains sets of characteristics for known ballistic-missile threats

String: Tracked_Object_Status
 // Identifies status of Tracked_Object. Will be Active, Killed, Hit, or Dropped

Integer: Unknown_Track_Life
 // Time duration from detection to present time – expressed in seconds.

Boolean: IPP_Within_Defended_Area
 // True if IPP of ballistic-missile threat lies within defended area

Real: IPP_Latitude
 // Latitude of IPP

Real: IPP_Longitude
 // Longitude of IPP

Real: Defended_Area_Max_Latitude
 // Maximum latitude value of defended area

Real: Defended_Area_Min_Latitude
 // Minimum latitude value of defended area

Real: Defended_Area_Max_Longitude
 // Maximum longitude value of defended area

Real: Defended_Area_Min_Longitude
 // Minimum longitude value of defended area

Boolean: Weapon_Status_Operational
 // True if weapon is operationally available to fight

Boolean: Weapon_Launcher_Armed
 // True if weapon launcher is armed and ready to fire

Set: Min_Intercept_Point
 // Minimum intercept point at which an intercept at points closer to defended area would result in negative consequences to the defended area. Expressed in longitude and latitude. Typed as a set.

Multiset: `Interceptor_Range_Volume`

// All points within the range of the interceptor. Expressed in longitude and latitude.

Assertions:

Always `Weapon_Assigned <=> ((Ballistic_Threat) & (IPP_Within_Defended_Area) & (Weapon_Available) & (Intercept_Point_Min_Within_Intercept_Range))`

Always `Ballistic_Threat <=> (Unknown_Track) Until (Tracked_Object  $\cap$  Threat_Profile) & (Tracked_Object_Status = Active) & (Unknown_Track_Life) < 60`

Always `IPP_Within_Defended_Area <=> (IPP_Latitude  $\leq$  Defended_Area_Max_Latitude) & (IPP_Latitude  $\geq$  Defended_Area_Min_Latitude) & (IPP_Longitude  $\leq$  Defended_Area_Max_Longitude) & (IPP_Longitude  $\geq$  Defended_Area_Min_Longitude)`

Always `Weapon_Available <=> (Weapon_Health_Operational) & (Weapon_Launcher_Armed)`

Always `Intercept_Point_Min_Within_Intercept_Range <=> (Min_Intercept_Point  $\cap$  Interceptor_Range_Volume)`

Model checking

Software engineers should verify the functional specifications by employing the techniques of model checking. For this report, we will define model checking as the systematic approach for testing functional assertions and substantiating the desired system behavior in the model. Model checking is not a proof of correctness; however, model checking involves creating functional models of a system and analyzing the model against the formal representations of the desired behavior. [Lewis 01]

For the BMK, software engineers should verify the functional specifications using an automated model-checking tool that can accept the developed specifications and exercise the assertions over a number of time cycles. They should identify any inconsistencies and breaks in logic through the use of the model-checking tool. From the results of the model checking, software engineers can correct our specifications and the artifacts from the domain analysis as required.

Software engineers must be cognizant of the state-explosion problem in model checking. For this report, we will define state explosion as the size of the state space exceeds the memory capacity of the automated tool to check every trace in the model. [Gallardo 03] Through abstraction of the BMK functions in our specifications, software engineers can employ the concept of symbolic model checking in which Boolean functions are employed to represent transition relations as well as sets of states. Specifically, they

should adopt a compact representation of the state space, such as that provided by binary decision diagrams (BDDs) to simplify the BMK states by removing sub-trees and redundant edges on the BMK's Boolean decision tree. [Clarke 01] In other words, they can modify the complex logic decisions at the bottom of the tree to simple Boolean statements so that we can capture the essence of the system behavior in the upper portions of the decision tree. By reducing the high number of lower-level logic statements that develop very specific solutions and have limited impact on the overall system behavior, software engineers should be able to manage the state-explosion problem.

An example of the state-explosion problem in the BMK, consider the following assertion:

Always Intercept_Point_Min_Within_Intercept_Range $\Leftrightarrow$
(Min_Intercept_Point $\cap$ Interceptor_Range_Volume)

Note that the number of points in Interceptor_Range_Volume could be large and that we are seeking to ensure that one specific point (Min_Intercept_Point) is within the set of points that define Interceptor_Range_Volume. Rather than use model checking to ensure that this condition is true, we could abstract the assertion to either a True or False for Intercept_Point_Min_Within_Intercept_Range. This will reduce the number of traces through the model to verify this assertion.

Framework design

Software engineers should develop a framework in which the BMK connects to software components used for calculations in battle-management as well as the interfaces to external components of systems such as sensors, C2, and weapons. The objective of this framework is to show a design of a battle manager as an integration of various components rather than a single software application. In this approach, we consider weapon systems to be comprised of components rather than a single entity. [Caffall 03]

By treating the each software application and each software interface as components, we believe that acquisition organizations can develop battle managers with more efficiency, reduced development times, and higher quality than current state-of-the-practice methods. [Crnkovic 02] The high-level architectural view for the BMK is depicted in Figure 1:

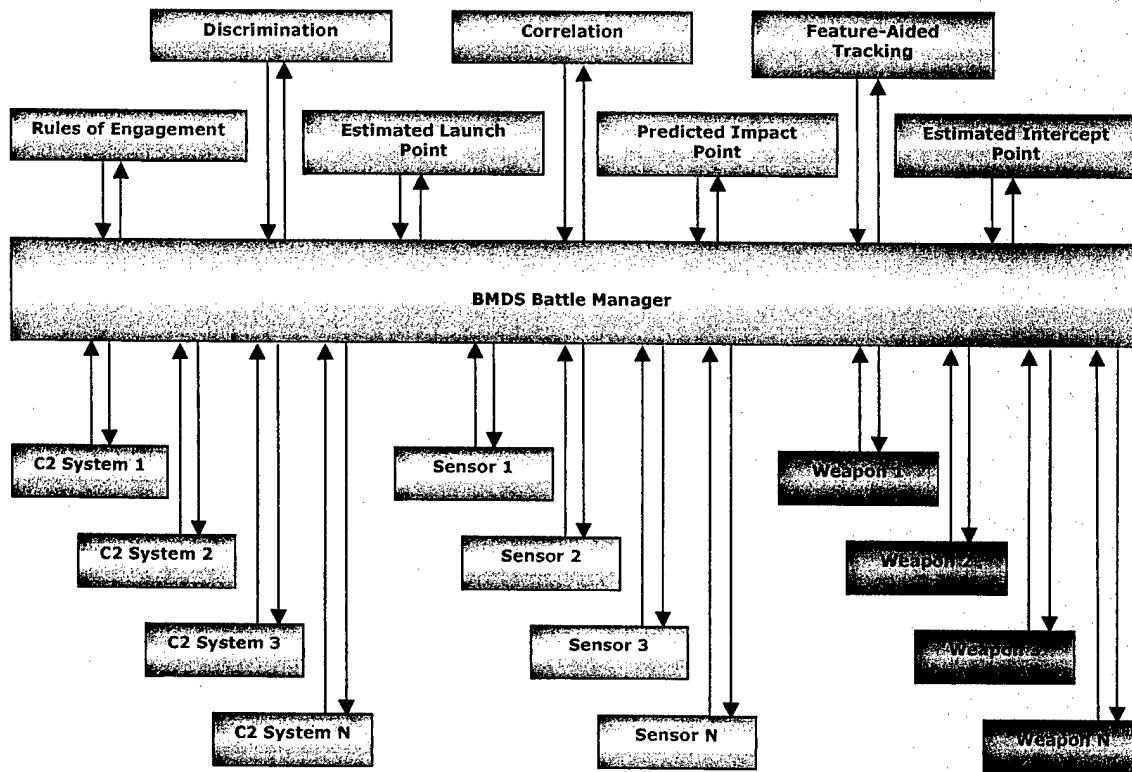


Figure 1. BMK and Components

Demonstration

Software engineers should consider developing a prototype of the BMK framework and demonstrating its behavior, capabilities, and limitations. They should test this prototype to determine the degree of system behavior predictability with respect to state transitions and tolerance to battlespace environmental variables.

While the demonstration is not intended to be an exhaustive test, it will offer a degree of robustness to accompany the capabilities of the BMK prototype. (N.B.: Robustness is defined as the characteristic of a system that is failure and fault tolerant. Such a system handles unexpected states in a manner that minimizes performance degradation, data corruption, and incorrect output.)

We propose the following partial list of metrics be used as part of the BMK demonstration:

1. Maximum number of concurrent tracks
2. Percentage of processed tracks (birth to death) to total received tracks
3. Percentage of correlated tracks to total correlation opportunities

4. Percentage of discriminated tracks to total discrimination opportunities
5. Percentage of weapon/target assignments to total weapon/target pairing opportunities
6. Percentage of received weapon assignments to total weapon assignment opportunities
7. Percentage of launch authorizations to total weapon assignment opportunities
8. Percentage of re-engaged tracks to total re-engagement opportunities
9. Percentage of undesired state changes to total illegal and out-of-bounds inputs
10. Percentage of system crashes and system lockups to total illegal and out-of-bound inputs

Summary of recommendations

1. We envision software engineers realizing the basic functions of battle management as a kernel that will remain stable over time. Derived from the kill chain, the BMK will manage the use of the computing resources to ensure that all time-critical, battle-management events are processed as efficiently as possible.
2. We envision software engineers developing the BMK as a real-time set of system functionality that addresses its use by warfighters, starting from a high-level statement of capabilities and refining these statements into successively lower levels of system artifacts. The system artifacts should be refined from the perspective of developing test- and verification-ready models (i.e., representations of the system-of-systems that are amendable to automated testing and verification).
3. Software engineers should capture the desired system behavior in the interface definition rather than depending solely on messaging requirements.
4. Software engineers should construct interfaces for BMD elements such as an infrared sensor type, a radar type, a kinetic energy weapon, and a directed energy weapon.
5. Software engineers should develop a formal representation that captures the desired system behavior of the BMK and verify the formal representation against the expected BMK properties to achieve the level of desired predictable BMK behavior.
6. Software engineers should consider developing the formal representation of the BMK by using temporal logic to describe the functional specifications of the BMK.
7. Software engineers should verify the functional specifications with the use of a model-checking tool to determine the degree of system behavior predictability with respect to state transitions and tolerance to battlespace environmental variables.

8. To avoid the state-explosion problem, software engineers should carefully model the fundamental behavior of the BMK rather than a comprehensive specification of the BMK.
9. By incorporating assertions developed from the functional model and verified by the model-checking effort into the BMK, software engineers can develop embedded automatic test generation capabilities.
10. Software engineers should incorporate assertions and error-handling schemes developed from the functional model and verified by the model-checking effort into the BMK.
11. Software engineers should develop the required BMK interfaces as type interfaces that define the behavior of each interface and the required specifications to realize each interface.
12. For ease of integration and maintainability, we propose the development of software components for the features that typically experience the majority of changes.

Proposed advances

It is our belief that software engineers can develop a BMK that addresses the five basic functions and fulfills basic warfighter usage requirements for a battle-management capability. We believe that acquisition agencies within DoD can use the proposed BMK framework as a point of departure in the development of such systems with the potential benefits of acquiring systems on time, within budget, and with the desired level of capability as defined by the warfighters.

In addition, we believe that this approach will extend the software engineering body of knowledge as follows:

1. Demonstrate that acquisition organizations can develop real-time systems that exhibit a high degree of system behavior predictability in a large distributed system.
2. Provide a battle-management kernel that acquisition organizations can base future battle-management system developments on the battle-management kernel.

Glossary

Acquisition: The process in which the Department of Defense obtains materiel solutions to identified problems in mission need statements.

Assertion: A predicate expression whose value is either true or false.

Algorithm: A set of logical and mathematical processes to accomplish a given function with a processor or computer.

Ballistic missile: A rocket-propelled vehicle moving under its own momentum and the force of gravity that does not rely upon aerodynamic surfaces to produce lift and consequently follows a ballistic trajectory when thrust is terminated.

Ballistic missile defense: All active and passive measures designed to detect, identify, track, and defeat attacking ballistic missiles (and entities), in both strategic and theater tactical roles, during any portion of their flight trajectory (boost, ascent, midcourse, or terminal) or to nullify or reduce the effectiveness of such an attack.

Battle management: The decisions and actions executed in direct response to the activities of enemy forces in support of the Joint Chiefs of Staff's precision engagement concept.

Battle-management kernel: The software that contains the basic functions of battle management that will remain stable over time. Derived from the kill chain, these basic battle-management functions are called tasks, and will manage the use of the system's computing resources to ensure that all time-critical, battle-management events are processed as efficiently as possible.

Battlespace: All aspects of air, surface, subsurface, land, space, and the electromagnetic spectrum that encompass the area of influence and area of interest.

Central processing unit: A section of a computer responsible for execution of programs. This section manipulates the data, generates control signals, and stores results.

Chain of command: The succession of commanding officers from a superior to a subordinate through which command is exercised.

Classification: The process of establishing the type of an object being tracked. The object type out of the classification process may be high level (e.g., an air vehicle, an ASM, a TBM object, an interceptor missile, or unknown type) or very specific (e.g., SCUD B, SM-2, etc).

Coalition: An ad hoc arrangement between two or more nations for common action.

Combatant command: One of the unified or specified combatant commands established by the President.

Combatant command (command authority): Non-transferable command authority established by Title 10, United States Code, section 164, exercised only by commanders

of unified or specified combatant commands unless otherwise directed by the President or the Secretary of Defense. Combatant command (command authority) is the authority of a combatant commander to perform those functions of command over assigned forces involving organizing and employing commands and forces, assigning tasks, designating objectives, and giving authoritative direction over all aspects of military operations, joint training, and logistics necessary to accomplish the missions assigned to the command. Also called **COCOM**.

Combatant commander: A commander in chief of one of the unified or specified combatant commands established by the President.

Combat information: Unevaluated data gathered by or provided directly to the tactical commander that, due to its highly perishable nature or the criticality of the situation, cannot be processed into tactical intelligence in time to satisfy the users' tactical intelligence requirements.

Command: 1. The authority that a commander in the Armed Forces lawfully exercises over subordinates by virtue of rank or assignment. Command includes the authority and responsibility for effectively using available resources and for planning the employment of, organizing, directing, coordinating, and controlling military forces for the accomplishment of assigned missions. It also includes responsibility for health, welfare, morale, and discipline of assigned personnel. 2. An order given by a commander; that is, the will of the commander expressed for the purpose of bringing about a particular action. 3. A unit or units, an organization, or an area under the command of one individual.

Command and control: The exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of the mission. Command and control functions are performed through an arrangement of personnel, equipment, communications, facilities, and procedures employed by a commander in planning, directing, coordinating, and controlling forces and operations in the accomplishment of the mission.

Command and control system: The facilities, equipment, communications, procedures, and personnel essential to a commander for planning, directing, and controlling operations of assigned forces pursuant to the missions assigned.

Command, Control, Communications, and Computer Systems (C4 Systems). Integrated systems of doctrine, procedures, organizational structures, personnel, equipment, facilities, and communications designed to support a commander's exercise of command and control through all phases of the operational continuum.

Command and control warfare: The integrated use of operations security (OPSEC), military deception, psychological operations (PSYOP), electronic warfare (EW), and physical destruction, mutually supported by intelligence, to deny information to, influence, degrade, or destroy adversary command and control capabilities, while protecting friendly command and control capabilities against such actions. Command and control warfare applies across the operational continuum and at all levels of conflict. Also called **C2W**. C2W is both offensive and defensive: a. counter-C2-To prevent effective C2

of adversary forces by denying information to, influencing, degrading, or destroying the adversary C2 system. b. C2-protection-To maintain effective command and control of own forces by turning to friendly advantage or negating adversary efforts to deny information to, influence, degrade, or destroy the friendly C2 system.

Component: A software unit of composition with contractually specified interfaces and explicit context dependencies.

Control: Authority which may be less than full command exercised by a commander over part of the activities of subordinate or other organizations.

Correlation: The process of assigning or computing weights to determine that two or more sensed tracks are for the same object.

Crisis action planning: The time-sensitive planning for the deployment, employment, and sustainment of assigned and allocated forces and resources that occurs in response to a situation that may result in actual military operations. Crisis action planners base their plan on the circumstances that exist at the time planning occurs. Also called CAP

Data: A representation of individual facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means.

Deliberate planning: A planning process for the deployment and employment of apportioned forces and resources that occurs in response to a hypothetical situation. Deliberate planners rely heavily on assumptions regarding the circumstances that will exist when the plan is executed.

Detection: Discrimination of an object from its background and its assignment to the class of potentially interesting objects.

Discrimination: Process that allows selecting lethal from non-lethal targets in same threat complex. The process usually involves sensors, signal/data processors, feature extraction algorithms, and decision architectures.

Domain analysis: The process of identifying and formalizing constraints on input, state, and output values.

Dominant maneuver: The ability of joint forces to gain positional advantage with decisive speed and overwhelming operational tempo in the achievement of assigned military tasks.

Endo-atmospheric: Within the earth's atmosphere. The altitude commonly used to separate the endo- and exo-atmospheric regimes varies from 100 km to 120 km.

Engage: A fire control order used to direct or authorize units and/or weapon systems to fire on a designated target.

Engagement: A tactical conflict, usually between opposing lower echelons maneuver forces.

Exo-atmospheric: Above the atmosphere where the drag is negligible. The altitude commonly used to separate the endo- and exo-atmospheric regimes varies from 100 km to 120 km.

Failure: The inability of a system or component to perform a required function within specified limits.

Fault: An incorrect statement, step, process, or data definition in a software program.

Focused logistics: The ability to provide the joint force the right personnel, equipment, and supplies in the right place, at the right time, and in the right quantity, across the full range of military operations.

Full dimensional protection: The ability of the joint force to protect its personnel and other assets required to decisively execute assigned tasks.

Glueware: The software application that integrates a number of components through interfaces to the software application for the purpose of achieving a broader capability than any of the individual components.

Identification: The process of determining that a tracked object is a friendly, neutral, hostile, or unknown object, or the result of that process.

Information: The meaning that a human assigns to data by means of the known conventions used in their representation.

Intelligence: The product resulting from the collection, processing, integration, analysis, evaluation, and interpretation of available information concerning foreign countries or areas.

Interface: Software that enables an application to work with user, another application, operating system, or computer hardware.

Interoperability: The ability of systems, units, or forces to provide services to and accept services from other systems, units, or forces and to use the services so exchanged to enable them to operate effectively together.

Joint: Connotes activities, operations, organizations, etc., in which elements of two or more Military Departments participate.

Joint force: A general term applied to a force composed of significant elements, assigned or attached, of two or more Military Departments, operating under a single joint force commander.

Joint task force: A joint force that is constituted and so designated by the Secretary of Defense, a combatant commander, a sub-unified commander, or an existing joint task force commander.

Kernel (real-time): A real-time kernel is software that manages the use of the CPU and memory to ensure that all time-critical events are processed as efficiently as possible. A real-time kernel can help simplify a software design because it allows a project to be divided into multiple independent elements called tasks.

Kernel (battle management): The part of a system, including software, that when all functions not essential to battle management are taken away, remains, and that functions even when one or more non-essential functions are disabled.

Kill assessment: A process, based on sensor data, that examines in real time the results of an engagement and determines whether the warhead was broken open or not. Based on the outcome the battle manager would decide to or not to fire again at that target.

Kill chain: The sequence of events that must occur for a threat to successfully engage and kill its target. For this dissertation, the elements of the kill chain are: Detect, Track, Assign Weapon, Engage, and Kill Assessment.

Link 16 (formerly TADIL-J): A secure, high capacity, jam-resistant, node-less data link which uses the Joint Tactical Information Distribution System (JTIDS) transmission characteristics and the protocols, conventions, and fixed-length message formats defined by the JTIDS Technical Interface Design Plan (TIDP).

Mission: The task, together with the purpose, that clearly indicates the action to be taken and the reason therefore.

Mission type order: Order to a unit to perform a mission without specifying how it is to be accomplished.

Model checking: The systematic approach for testing functional assertions and substantiating the desired system behavior in the model. Model checking is not a proof of correctness; however, model checking involves creating functional models of a system and analyzing the model against the formal representations of the desired behavior.

Operational control: Transferable command authority that may be exercised by commanders at any echelon at or below the level of combatant command. Operational control is inherent in Combatant Command (command authority) and is the authority to perform those functions of command over subordinate forces involving organizing and employing commands and forces, assigning tasks, designating objectives, and giving authoritative direction necessary to accomplish the mission. Also called **OPCON**.

Planning: That military planning that produces either an Operation Plan (OPLAN) or an Operations Order (OPORD) to employ military force against an adversary.

Precision engagement: The ability of joint forces to locate, surveil, discern, and track objectives or targets; select, organize, and use the correct systems; generate desired effects, assess results; and reengage with decisive speed and overwhelming operational tempo as required, throughout the full range of military operations.

Predicate: A function that represents the truth or falsehood of some condition.

Real-time: A problem, system, or application that is concurrent in nature and has timing constraints whereby incoming events must be processed within a given timeframe.

Robustness: A characteristic of a system that is failure and fault tolerant. Such a system handles unexpected states in a manner that minimizes performance degradation, data corruption, and incorrect output.

Rules of engagement: Directives issued by competent military authority that delineate the circumstances and limitations under which United States forces will initiate and/or continue combat engagement with other forces encountered. Also called ROE.

Schedulability: The determination of whether a group of tasks, whose individual CPU utilization is known, will meet their deadlines.

Sensor: A device that responds to a physical stimulus (as heat, light, sound, pressure, magnetism, or a particular motion) and transmits a resulting impulse for measurement or operating a control.

Sensor netting: Process of sharing information about targets of interest collected by two or more sensors with the objective of improving defense's knowledge of targets. Objective of sensor netting is to improve accuracy of sensor data by correlating, fusing, integrating, weighting, or associating sensed information at one or more locations in netted community. Process can be centralized, distributed, or hierarchical.

Situational awareness: Perception of available facts, comprehension of the facts in relation to the individual's expert knowledge, and projecting how the situation is likely to develop in the future.

Specified command: A command that has broad continuing missions and that is established by the President through the Secretary of Defense with the advice and assistance of the Chairman of the Joint Chiefs of Staff. It normally is composed of forces from a single Military Department. Also called specified combatant command.

State: A recognizable situation that exists over an interval of time.

State explosion: The condition in which the size of the state space grows exponentially.

State transition: A change in state that is caused by an input event.

Surveillance: The systematic observation of aerospace, surface or subsurface areas, places, persons, or things, by visual, aural, electronic, photographic, or other means.

System-of-systems: An amalgamation of legacy systems and developing systems that provide an enhanced military capability greater than that of any of the individual systems within the system-of-systems.

Tactical control: The detailed and, usually, local direction and control of movements or maneuvers necessary to accomplish missions or tasks assigned. Also called **TACON**.

Targeting: 1. The process of selecting targets and matching the appropriate response to them taking account of operational requirements and capabilities. 2. The analysis of enemy situations relative to the commander's mission, objectives, and capabilities at the commander's disposal, to identify and nominate specific vulnerabilities that, if exploited, will accomplish the commander's purpose through delaying, disrupting, disabling, or destroying enemy forces or resources critical to the enemy.

Task: A task is a program that competes for CPU time and is generally written as an infinite loop

Temporal logic: An extension of propositional logic that incorporates special operators that cater for time. With temporal logic one can specify how components, protocols, objects, modules, procedures and functions behave as time progresses. The specification is done with temporal logic statements that make assertions about properties and relationships in the past, present, and the future.

Test-ready model: A model that contains sufficient information to automatically produce test cases for its implementation.

Time-critical task: A task for which there is a deadline for which the task must usually (soft) or must always (hard) meet.

Time-critical targets: Those targets requiring immediate response because they pose (or will soon pose) a clear and present danger to friendly forces, or are highly lucrative, fleeting targets of opportunity.

Track: 1. Estimated position/velocity states and a representation of the uncertainty of the estimate for an object or unresolved cluster of objects based on filtered observations from one or more sensors. 2. Estimated trajectory of an apparent object or group of objects. 3. Sequence of observations judged to be from the same object or group of objects

Unified command: A command with broad continuing missions under a single commander and composed of forces from two or more Military Departments, and which is established by the President, through the Secretary of Defense with the advice and assistance of the Chairman of the Joint Chiefs of Staff. Also called unified combatant command.

Validation: Confirmation by examination and provisions of objective evidence that the particular requirements for a specific intended use are fulfilled.

Verification: Confirmation by examination and provisions of objective evidence that specified requirements have been fulfilled.

Weapon tasking: Message sent to weapon by battle manager that contains information such as target-weapon pairing, launch time, etc.

Acronyms

AADC Area air defense commander

ABL Airborne laser

ABM Anti-ballistic missile

ACTD Advanced concept technology demonstration

AD Air defense

ADA Air defense artillery

ADCP Air defense communications platform

ADG Active defense group

ALERT Attack and launch early report to theater

AO Area of Operations

AOA Amphibious objective area

AOC Air Operations Center

AOR Area of responsibility

ATACMS Army tactical missile system

ATO Air tasking order

AWACS Airborne warning and control system

BDA Battle damage assessment

BMC4I Battle-management command, control, communications, computers, and intelligence

BMD Ballistic missile defense

BMDS Ballistic missile defense system

BMK Battle-management kernel

BPI Boost-phase intercept

CAP Crisis action planning

C2 Command and control CAP Combat air patrol

C3I Command, Control, Communications, and intelligence

CEC Cooperative engagement capability
CENTCOM United States Central Command
CEP Circular error probable
CIC Combat information center
CJCS Chairman, Joint Chiefs of Staff
CM Configuration management
CO Commanding officer
COA Course of action
COCOM Combatant Commander
COEA Cost and operational effectiveness analysis
CONOPS Concept of operations
CONPLAN Operations plan in concept format
CONUS Continental United States (excluding Alaska and Hawaii)
COP Common operational picture
COTS Commercial off the shelf
CPU Central processing unit
CRC Control and reporting center
DAL Defended asset list
DE Directed energy
DIA Defense Intelligence Agency
DII COE Defense information infrastructure common operating environment
DISA Defense Information Systems Agency
DoD Department of Defense
DSP Defense Support Program
EO Electrical-optical
EUCOM United States European Command
EW Early warning

EXORD Execute order

GAO General Accounting Office

GBI Ground-based interceptor

GBR [THAAD] Ground-based radar

GCCS Global command and control system

GEM Guidance enhanced missile (PATRIOT)

GGIG Global information grid

GMD Ground-based Missile Defense

GPS Global Positioning System

HQ Headquarters

IA Information assurance

ICBM Intercontinental ballistic missile

ICC Information Coordination Central (PATRIOT)

IER Information exchange requirement

IOC Initial operational capability

IPB Intelligence preparation of the battle space

IR Infrared

IRBM Intermediate-range ballistic missile

IRST Infrared search and track

ITW/AA Integrated tactical warning/attack assessment

JCS Joint Chiefs of Staff

JCTN Joint composite tracking network

JDN Joint data network

JEZ Joint engagement zone

JFACC Joint force air component commander

JFC Joint force commander

JFCOM Joint Forces Command

JFMCC Joint force maritime component commander
JIC Joint intelligence center
JMCIS Joint maritime command information system
JP Joint publication
JPN Joint planning network
JS Joint staff
JSOC Joint Special Operations Command
JSTARS Joint surveillance and target attack radar system
JTA Joint technical architecture
JTAGS Joint tactical ground station
JTF Joint task force
JTIDS Joint tactical information distribution system
JTMD Joint theater missile defense
KE Kinetic energy
KV Kill vehicle
KW Kinetic warhead
MDA Missile Defense Agency
MEADS Medium extended air defense system
MEZ Missile engagement zone
MNS Mission need statement
MLRS Multiple launch rocket system
MRBM Medium-range ballistic missile
NATO North Atlantic Treaty Organization
NBC nuclear, biological, and chemical
NCA National Command Authority
NMCC National Military Command Center
NMD National missile defense

NORTHCOM United States Northern Command
OCONUS Outside the continental United States
OOAD Object-oriented analysis and design
OOB Operational order of battle
OODA Observe, orient, decide, act
OPLAN Operations plan
OPORD Operations order
OSD Office of the Secretary of Defense
PAC Patriot advanced capability
PACOM Pacific Command
PATRIOT phased array tracking radar intercept on target
PDAL Prioritized defended asset list
 P_k Probability of kill
POM Program objective memorandum
R&D Research and development
RAS Replenishment at sea
RCS Radar cross-section
R&D Research and development
RDT&E Research, development, test, and evaluation
RF Radio frequency
ROE Rules of engagement
RV Reentry vehicle
SAM Surface-to-air missile
SATCOM Satellite communications
SBIRS-LOW Space-based infrared system-low earth orbit
SBWS Space-based warning system (DSP + TES)
SDI Strategic Defense Initiative

SMTS Space and missile tracking system
SOCOM United States Special Operations Command
SOF Special operations forces
STRATCOM United States Strategic Command
SRBM Short-range ballistic missile
TACON Tactical control
TAOC Tactical air operations center
TBM Theater ballistic missile
TBM-WMD Theater ballistic missile—weapons of mass destruction
TBMD Theater ballistic missile defense
TCT Time critical target
TDDS Tactical data distribution system
TEL Transporter-erector-launcher (for TBM)
THAAD Theater high-altitude area defense
TIBS Tactical information broadcast service
TLAM Tomahawk land attack missile
TM Theater missile
TMD Theater missile defense
TOC Tactical operations center
TPFDD Time-phased force and deployment data
TPFDL Time-phased force and deployment list
TRAP TRE and related applications (now TDDS)
TRE Tactical receive equipment
UAV Unmanned aerial vehicle
UCP Unified Command Plan
UML Unified Modeling Language
UOES User operational evaluation system

USA United States Army

USAF United States Air Force

USMC United States Marine Corps

USN United States Navy

VCJCS Vice-Chairman, Joint Chiefs of Staff

WMD Weapons of mass destruction

Glossary of Logic Symbols

$\&$ and

$|$ or

$\Leftrightarrow$ if and only if

$\Rightarrow$ implies

$>$ greater than

$<$ less than

$\geq$ greater than or equal to

$\leq$ less than or equal to

$\cap$ intersect

$\cup$ union

$=$ equals

$\neq$ does not equal

$\forall$ for all ($\forall x$ means that for all $x \dots$)

$\exists$ there exists ($\exists x$ means that there exists an x such that $\dots$)

$\neg$ not

$//$ comment

References

- [Bachman 02] Bachman, F., Bass, L., Clements, P., Garlan, D., Ivers, J., Little, R., Nord, R., and Stafford, J. Documenting software architecture: Documenting interfaces. Technical Note CMU/SEI-2002-TN-015, Software Engineering Institute, Pittsburgh, Penn., June 2002.
- [Binder 01] Binder, R. V. *Testing Object-Oriented Systems: Models, Patterns, and Tools*, Reading, Mass.: Addison-Wesley, June 2001.
- [Booch 94] Booch, G. *Object-Oriented Analysis and Design with Applications*. Reading, Mass.: Addison-Wesley, 2nd ed., 1994.
- [Boyd 86] Boyd, J. R. "A discourse on winning and losing: Patterns of conflict." Lecture notes, Dec. 1986. (Typewritten)
- [Caffall 03] Caffall, D. S. Conceptual Framework Approach for System-of-Systems Software Developments, Master's Thesis, Naval Postgraduate School, Monterey, Calif., Mar. 2003.
- [Chairman 00] US Department of Defense. *Joint Vision 2020*. Washington, D.C.: US Government Printing Office, June 2000.
- [Chen 03] Chen, X. and Hsieh, H. Case studies of model checking for embedded system designs. In *Proc. Third Int. Conf. on Application of Concurrency to System Design*, IEEE (Guimarães, Portugal, June 2003), pp. 20-28.
- [Clarke 01] Clarke, E., Grumberg, O., Jha, S., Lu, Y., and Veith, H. Progress on the state explosion problem in model checking. In Wilhelm, R., ed., *Lecture Notes in Computer Science: Informatics - 10 Years Back. 10 Years Ahead*, Vol. 2000, Heidelberg, Ger.: Springer-Verlag, 2001, pp. 176-194.
- [Coram 02] Coram, R. *Boyd: The Fighter Pilot Who Changed the Art of War*. New York: Little Brown and Co., 2002.
- [Crnkovic 02] Crnkovic, I. and Larsson, M., eds. *Building Reliable Component-Based Software Systems*. Norwood, Mass.: Artech House, 2002.
- [Gallardo 03] del Mar Gallardo, M., Martínez, J., Merino, P., and Pimentel, E. Abstract model checking and refinement of temporal logic in α SPIN. In *Proc. Third Int. Conf. on Application of Concurrency to System Design*, IEEE (Guimarães, Port., June 2003), pp. 245-246.

- [DA 96] US Department of the Army. *Tactics, Techniques, and Procedures for the Targeting Process*. Field Manual 6-20-10, May 1996.
- [DOD 02] US Department of Defense. Tactical Digital Information Link (TADIL) J Message Standard. MIL-STD-6016A, Joint Technical Architecture - Version 4.0, July 2002.
- [Douglas 99] Douglass, B. P. *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns*. Reading, Mass.: Addison-Wesley, 1999.
- [Drusinsky 02] Drusinsky, D. ESC: Formal spec languages ensure design code quality, *EE Times*, Mar. 7, 2002.
- [Erwin 02] Erwin, S. I. General Jumper: Time to change traditional program advocacy, *National Defense*, July 2002, pp. 14-15.
- [GAO 97] National Missile Defense: Schedule and Technical Risks Represent Significant Development Challenges. Report GAO/NSIAD-98-28, US General Accounting Office, Washington, D.C., Dec. 1997.
- [Gluch 99] Gluch, D. P. and Brockway, J. An introduction to software engineering practices using model-based verification. Technical Report CMU/SEI-99-TR-005, Software Engineering Institute, Pittsburgh, Penn., Apr. 1999.
- [Guaspari 00] Guaspari, D. and Naydich, D. Analysis of Real-Time Code by Model Checking," in *Proc. Nineteenth Digital Avionics Systems Conf.*, IEEE (Philadelphia., Penn., Oct. 2000), Vol. 1, pp. 1D5.1-1D5.8.
- [JCS 03] US Department of Defense. *Department of Defense Dictionary of Military and Associated Terms*. Joint Pub. 1-02, Apr. 12, 2001 (as amended through May 23, 2003).
- [JCS 96] US Department of Defense. *Doctrine for Joint Theater Missile Defense*. Joint Pub. 3-01.5, Joint Chiefs of Staff, Feb. 1996.
- [JFC 03] US Department of Defense. *Joint Force Command and Control Concept to Guide Standing Joint Force Headquarters Development by 2005*. Joint Forces Command, Mar. 5, 2003.
- [JFSC 00] US Department of Defense. *The Joint Staff Officer's Guide 2000*. JFSC Pub 1, National Defense University, Joint Forces Staff College, Norfolk, Va., 2000.

- [Kenne 03] Kenne, L. F. Tightening the kill chain: Broadening information access, *Intercom: J. Air Force C4I Community* 44, 1 (Jan. 2003): 6-9.
- [Leckie 90] Leckie, R. *None Died In Vain*. New York: Harpers Collins, 1990.
- [Lewis 01] Lewis, G. A., Comella-Dorda, S., Gluch, D. P., Hudak, J., and Weinstock, C. Model-based verification: Analysis guidelines. Technical Note CMU/SEI-2001-TN-028, Software Engineering Institute, Pittsburgh, Penn., Dec. 2001.
- [Meyers 01] Meyers, C. B., Feiler, P. H., Marz, T. Proc. Real-Time Systems Engineering Workshop. Special Report CMU/SEI-2001-SR-022, Software Engineering Institute, Pittsburgh, Penn., Aug. 2001.
- [Parnas 01] Parnas, David L. *Software Fundamentals: Collected Papers by David L. Parnas*. Reading, Mass.: Addison-Wesley, 2001.
- [Rempt 01] Rempt, R. P. The Navy in the twenty-first century, Part II: Theater Air and Missile Defense, *Johns Hopkins APL Technical Digest* 22, 1 (2001): 21-28.
- [Selic 03] Selic, B. The pragmatics of model-driven development, *IEEE Software*, Sept./Oct. 2003, pp. 19-25.
- [Sha 93] Sha, L. and Sathaye, S. Distributed real-time system design: Theoretical concepts and applications. Technical Report CMU/SEI-93-TR-002, Software Engineering Institute, Pittsburgh, Penn., Mar. 1993.
- [Stewart 01] Stewart, D. B. Twenty-five most common mistakes with real-time software development. In *Proc. Embedded Systems Conf.* (San Francisco, Calif., Apr. 2001), Gilroy, Calif.: CMP Media.
- [Szyperski 02] Szyperski, C. *Component Software: Beyond Object-Oriented Programming*. Reading, Mass.: Addison-Wesley, 2nd ed., 2002.

Initial Distribution List

1. Defense Technical Information Center
8725 John J. Kingman Rd., STE 0944
Ft. Belvoir, VA 22060-6218
2. Dudley Knox Library, Code 013
Naval Postgraduate School
Monterey, CA 93943-5100
3. Technical report Office, Code 09
Naval Postgraduate School
Monterey, CA 93943-5138
4. James Bret Michael, Code CS/Mj
Associate Professor
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943-5118
bmichael@nps.navy.mil
5. COL Kevin Greaney USA
Director of Modeling and Simulation
Deputate for System Engineering
Missile Defense Agency
7100 Defense Pentagon
Washington, D.C. 20301-7100
Kevin.Greaney@mda.osd.mil
6. Dale Scott Caffall
C2BMC Chief Engineer
Missile Defense Agency
7100 Defense Pentagon
Washington, D.C. 20301-7100
Butch.Caffall@mda.osd.mil