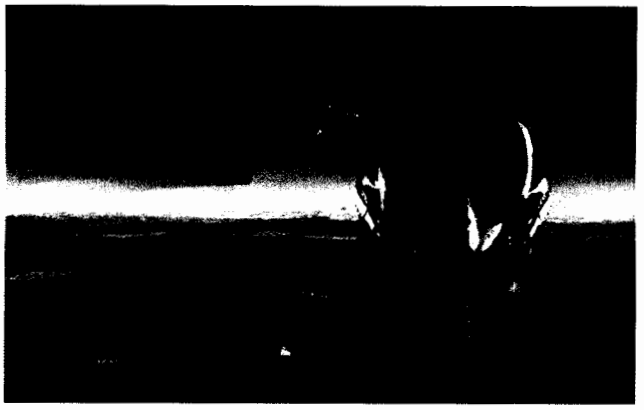
DCN: 12381

Theater Battle Management Core System Systems Engineering Case Study

By Josiah R. Collens, Jr.
The MITRE Corporation

And Bob Krause
Lockheed Martin Integrated
Systems and Solutions



Center for Systems Engineering
at the Air Force Institute of Technology (AFIT/SY)
2950 Hobson Way, Wright-Patterson AFB OH 45433-7765

PREFACE

In response to Air Force Secretary James G. Roche's charge to reinvigorate the systems engineering profession, the Air Force Institute of Technology (AFIT) undertook a broad spectrum of initiatives that included creating new and innovative instructional material. The Institute envisioned case studies on past programs as one of these new tools for teaching the principles of systems engineering.

Four case studies, the first set in a planned series, were developed with the oversight of the Subcommittee on Systems Engineering to the Air University Board of Visitors. The Subcommittee includes the following distinguished individuals:

Chairman

Dr. Alex Levis, AF/ST

Members

Brigadier General Tom Sheridan, AFSPC/DR
Dr. Daniel Stewart, AFMC/CD
Dr. George Friedman, University of Southern California
Dr. Andrew Sage, George Mason University
Dr. Elliot Axelband, University of Southern California
Dr. Dennis Buede, Innovative Decisions, Inc.
Dr. Dave Evans, Aerospace Institute

Dr. Levis and the Subcommittee on Systems Engineering crafted the idea of publishing these case studies, reviewed several proposals, selected four systems as the initial cases for study, and continued to provide guidance throughout their development. The Subcommittee's leading minds in systems engineering have been a guiding force to charter, review, and approve the work of the authors. The four case studies produced in this series are the C-5 Galaxy, the F-111, the Hubble Space Telescope, and the Theater Battle Management Core System.

Approved for Public Release; Distribution Unlimited

The views expressed in this Case Study are those of the author(s) and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

FOREWORD

At the direction of the Secretary of the Air Force, Dr. James G. Roche, the Air Force Institute of Technology (AFIT) established a Center for Systems Engineering (CSE) at its Wright-Patterson AFB, OH, campus in 2002. With academic oversight by a Subcommittee on Systems Engineering, chaired by Air Force Chief Scientist Dr. Alex Levis, the CSE was tasked to develop case studies focusing on the application of systems engineering principles within various aerospace programs. At a May 2003 meeting, the Subcommittee reviewed several proposals and selected the Hubble Telescope (space system), Theater Battle Management Core System (complex software development), F-111 fighter (joint program with significant involvement by the Office of the Secretary of Defense), and C-5 cargo airlifter (very large, complex aircraft). The committee drafted an initial case outline and learning objectives, and suggested the use of the Friedman-Sage Framework to guide overall analysis.

The CSE contracted for management support with Universal Technology Corporation (UTC) in July 2003. Principal investigators for the four cases included Mr. John Griffin for the C-5A, Dr. G. Keith Richey for the F-111, Mr. James Mattice for the Hubble Space Telescope, and Mr. Josh Collens from The MITRE Corporation for the Theater Battle Management Core System effort.

The Department of Defense continues to develop and acquire joint complex systems that deliver needed capabilities demanded by our warfighters. Systems engineering is the technical and technical management process that focuses explicitly on delivering and sustaining robust, high-quality, affordable products. The Air Force leadership, from the Secretary of the Air Force, to our Service Acquisition Executive, through the Commander of Air Force Materiel Command, has collectively stated the need to mature a sound systems engineering process throughout the Air Force.

These cases will support academic instruction on systems engineering within military service academies and at both civilian and military graduate schools. Plans exist for future case studies focusing on other areas. Suggestions have included various munitions programs, Joint service programs, logistics-led programs, science and technology/laboratory efforts, additional aircraft programs such as the B-2 bomber, and successful commercial systems.

As we uncovered historical facts and conducted key interviews with program managers and chief engineers, both within the government and those working for the various prime and subcontractors, we concluded that systems programs face similar challenges today. Applicable systems engineering principles and the effects of communication and the environment continue to challenge our ability to provide a balanced technical solution. We look forward to your comments on this case study and the others that follow.

MARK K. WILSON, SES

Director, Center for Systems Engineering
Air Force Institute of Technology
http://cse.afit.edu/

ACKNOWLEDGMENTS

The authors wish to acknowledge the following contributors: from Lockheed Martin: Pat Murphy, Renzo Bianchi, Greg Hinchman, Frank DeLalla, Sue Bergmeier, Larry Llewellyn, and Reese Delorey; from BAE Systems: Tom O'Lear; from The MITRE Corporation: Edwin Green, Roger Dumas, Robert Miller, Eric Estochen, Ioannis Kyratzoglou, Jerry Warner, Cheryl Dyer, and Margaret MacDonald; and from the United States Air Force: Major John Schafer, William Riley, and Jack Thiele.

At the Center for Systems Engineering (CSE), we would like to acknowledge the contributions of the following members: Lt. Col. John Colombi, who provided overall program management support and formulated a common outline with a consistent approach, and Mike Mueller and Major Tina DeAngelis for their significant contributions, which enabled us to meet CSE expectations and timelines. Finally, we thank the CSE support staff who enabled us to distribute a quality product in a timely manner.

Josiah R. Collens, Jr.
Bob Krause

EXECUTIVE SUMMARY

The Theater Battle Management Core System (TBMCS) is an integrated air command and control (C2) system that performs standardized, secure, automated air battle planning and execution management for Air Force, multi-service, and allied commanders in theaters of operation worldwide. TBMCS provides the means to plan, direct, and control all theater air operations and to coordinate with land, maritime, and special operations elements. It is deployed at C2 nodes at national, force and wing/unit-level elements. TBMCS operates in support of planners and decision makers at, and below, the level of Joint Force Air Component Commander. The system is modular and scalable for air, land, or sea transport and the deployed configurations can be tailored to meet a particular contingency.

This case study identifies and assesses the systems engineering process used by the Air Force and its prime contractor, Lockheed Martin. It describes the systems engineering process used from 1995 to 2000 to produce the first version of TBMCS (V1.0.1). The case study examines in detail five key systems engineering learning principles:

LP 1, Requirements Definition and Management. The government did not produce a Concept of Operations, key operational performance parameters, or a system specification for the contractor. The contractor was responsible for generating a system segment specification that had performance measures as goals, but not testable requirements. The government did produce a technical requirements document that defined a technical approach and levied certain standards on the contractor. There was no firm baseline for operational and system requirements from which the system could be built and tested. The requirements baseline was volatile up to system acceptance, which took place after TBMCS passed operational test and evaluation.

LP 2, System Architecture. The system architecture was defined at too high a level, which had a tremendous impact on system design and development. The government's mandates for software reuse and use of commercial software products were often contradictory and problematic for the system development. The layered system architecture did support system evolution and migration to modern technologies.

LP 3, System/Subsystem Design. The system and subsystem design was severely hampered by the complexity of legacy applications, misunderstanding of the maturity and complexity of commercial and third party software products, and a lack of understanding of how the system would be employed by the user.

LP 4, System Integration. Systems and interface integration was highly complex. The system integration was very difficult because of the lack of detail in the system architecture and the mandate to use government-furnished equipment that was not necessarily compatible with commercial off-the-shelf products. Integrating third party software products was an arduous process and required extensive oversight. The external system interfaces were not managed and were often impossible to test at the contractor's facility.

LP 5, Validation and Verification. The lack of a firm requirements baseline made validation and verification very difficult. The program was schedule driven and often ran parallel test processes without clear measures of success. Not being able to

replicate the operational environment prior to acceptance test created severe problems.

The lessons learned from TBMCS can be directly applied to other software-intensive programs that require the integration of vast numbers of third-party products with government-furnished equipment (GFE), such as hardware and communications. The key lesson is that there is no substitute for a well-defined systems engineering process. In the case of TBMCS, external influences drove a relaxation of discipline and rigor in the systems engineering process. In fact, the need for rigor and discipline in the process is even greater when a program lacks sufficient detail in the requirements, architecture, and system design, or when the contractor and government underestimate the complexity of software reuse and third-party integration. This was demonstrated by TBMCS in delivering the initial system (V1.0.1) in 2000, five years after contract award. The acquisition strategy of giving the contractor total system performance responsibility when over 90% of the program content is government-furnished equipment is fundamentally flawed. The contractor cannot be held accountable for performance if the contractor does not control all of the system components that affect performance. Perhaps the lack of formal requirements made the approach used – defining performance parameters as goals instead of requirements – the only possible approach in this particular case, but it should certainly not be adopted by other programs as a standard.

The lessons learned from the difficulty of fielding V1.0.1 had a very positive impact on the program's current systems engineering environment. TBMCS systems engineering processes have evolved to become mature and repeatable. The operational capability of TBMCS in Operations Enduring Freedom and Iraqi Freedom demonstrates the success of the current approach, as does the contractor's ability to field four subsequent releases in the short span of three years since the release of V1.0.1. Appendix 3 provides a detailed program history.

The foundation of the case study is the Friedman-Sage matrix [1], which shows the nine primary phases/functions of system development and the individual and shared responsibilities of the government and the contractor. This case study describes a matrix specific for TBMCS that was constructed on the basis of the published literature and the authors' interviews of the key participants supporting the program. The matrix illustrates the unique successes and failures in the application of the systems engineering process to TBMCS. It should be noted that all nine Friedman-Sage processes are now shared between the government and contractor; the degree of responsibility varies for each process, but the overall process is orchestrated as a team approach.

TABLE OF CONTENTS

PREFACE ii
FORWARD iii
ACKNOWLEDGEMENTS iv
EXECUTIVE SUMMARY v
1.0 SYSTEMS ENGINEERING PRINCIPLES 1
1.1 General Systems Engineering Process 1
1.2 TBMCS Major Learning Principles 5
2.0 SYSTEM DESCRIPTION 8
2.1 TBMCS Functional Overview 8
2.2 Air Battle Plan Rhythm 11
3.0 TBMCS SYSTEMS ENGINEERING LEARNING PRINCIPLES 13
3.1 Learning Principle 1 – Requirements Definition and Management 13
3.2 Learning Principle 2 – Systems Architecture 19
3.3 Learning Principle 3 – System/Subsystem Design 24
3.4 Learning Principle 4 – System Integration and Test 27
3.5 Learning Principle 5 – Validation and Verification 32
4.0 SUMMARY 38
5.0 REFERENCES 43
6.0 LIST OF APPENDICES 44
Appendix 1 - Completed Friedman Sage Matrix for TBMCS 45
Appendix 2 - Author Biography 47
Appendix 3 - Acronyms 48
Appendix 4 - Background and History of TBMCS 51
Appendix 5 - Risk Assessment and Management 61
Appendix 6 - System and Program Management 64

List of Figures

Figure 1-1 The Systems Engineering Process as Presented by the Defense Acquisition University 2
Figure 2-1 Notional Theater C4I 9
Figure 2-2 TBMCS Functional Description 9
Figure 2-3 TBMCS Interfaces for V1.1.3 11
Figure 2-4 Air Battle Plan Process 12
Figure 2-5 Today's "Battle Rhythm" = 72-Hour Cycle 12
Figure 3-1 Version Planning Process 14
Figure 3-2 TBMCS Participating Organizations (circa 1998) 15
Figure 3-3 QP Process Flow 19
Figure 3-4 N-Tiered Architecture 20
Figure 3-5 DII COE Architecture 21
Figure 3-6 Data Architecture 22
Figure 3-7 Physical/Hardware Architecture 23
Figure 3-8 Communications Architecture 23
Figure 3-9 Legacy Application Service Layers 25
Figure 3-10 Java Environment 26
Figure 3-11 Web Migration 27
Figure 3-12 Third Party Integration Process Flow 28
Figure 3-13 Product Timeline 30
Figure 3-14 Change Process for Current and Future TBMCS External Interfaces 31
Figure 3-15 TBMCS Test Relationships 32
Figure 3-16 Mission-Essential Task (MET) Decomposition 35
Figure A4-1 Initial Intent of Program 52
Figure A4-2 Al-Udeid, Qatar - Combined Air Operations Center (CAOC) 56
Figure A4-3 Problem Decision Flow 59
Figure A5-1 TBMCS Program Risk Management Process 61

List of Tables

Table 1-1 A Framework of Key Systems Engineering Concepts and Responsibilities 4
Table 1-2 A Framework for Systems Engineering Concept and Responsibility Domains [2] 7
Table 3-1 MOT&E Timeline 37
Table A1-1 The Friedman Sage Matrix for the TBMCS 45
Table A4-1 Operation Iraqi Freedom Sortie Count 56

1.0 SYSTEMS ENGINEERING PRINCIPLES

1.1 General Systems Engineering Process

1.1.1 Introduction

The Department of Defense continues to develop and acquire joint systems and to deliver needed capabilities to the warfighter. With a constant objective to improve and mature the acquisition process, it continues to pursue new and creative methodologies to purchase these technically complex systems. A sound systems engineering process, focused explicitly on delivering and sustaining robust, high-quality, affordable products that meet the needs of customers and stake holders must continue to evolve and mature. Systems engineering is the technical and technical management process that results in delivered products and systems that exhibit the best balance of cost and performance. The process must operate effectively with desired mission-level capabilities, establish system-level requirements, allocate these down to the lowest level of the design, and ensure validation and verification of performance, meeting cost and schedule constraints. The systems engineering process changes as the program progresses from one phase to the next, as do the tools and procedures. The process also changes over the decades, maturing, expanding, growing, and evolving from the base established during the conduct of past programs. Systems engineering has a long history. Examples can be found demonstrating a systemic application of effective engineering and engineering management, as well as poorly applied, but well defined processes. Throughout the many decades during which systems engineering has emerged as a discipline, many practices, processes, heuristics, and tools have been developed, documented, and applied.

Several core lifecycle stages have surfaced as consistently and continually challenging during any system program development. First, system development must proceed from a well-developed set of requirements. Regardless of overall waterfall or evolutionary acquisition approach, the system requirements must flow down to all subsystems and lower level components. System requirements need to be stable, balanced and must properly reflect all activities in all intended environments.

Next, the system planning and analysis occur with important tradeoffs and a baseline architecture developed. These architectural artifacts can depict any legacy system modifications, introduction of new technologies and overall system-level behavior and performance. Modeling and simulation are generally employed to organize and assess alternatives at this introductory stage. System and subsystem design follows the functional architecture. Either newer object-oriented analysis and design or classic structured analysis using functional decomposition and information flows/ data modeling occurs. Design proceeds logically using key design reviews, tradeoff analysis, and prototyping to reduce any high-risk technology areas.

Important to the efficient decomposition and creation of the functional and physical architectural designs are the management of interfaces and integration of subsystems. This is applied to subsystems within a system, or across large, complex systems of systems. Once a solution is planned, analyzed, designed and constructed, validation and verification take place to ensure satisfaction of requirements. Definition of test criteria, measures of effectiveness (MOEs) and measures of performance (MOPs), established as part of the requirements process well before any component/ subsystem assembly, takes place.

There are several excellent representations of the systems engineering process presented in the literature. These depictions present the current state of the art in the maturity and evolution of the systems engineering process. One can find systems engineering process definitions, guides and handbooks from the International Council on Systems Engineering (INCOSE), European Industrial Association (EIA), Institute of Electrical and Electronics Engineers (IEEE), and various Department of Defense (DoD) agencies and organizations. They show the process as it should be applied by today's experienced practitioner. One of these processes, long used by the Defense Acquisition University (DAU), is depicted by Figure 1-1. It should be noted that this model is not accomplished in a single pass. Alternatively, it is an iterative and nested process that gets repeated at low and lower levels of definition and design.

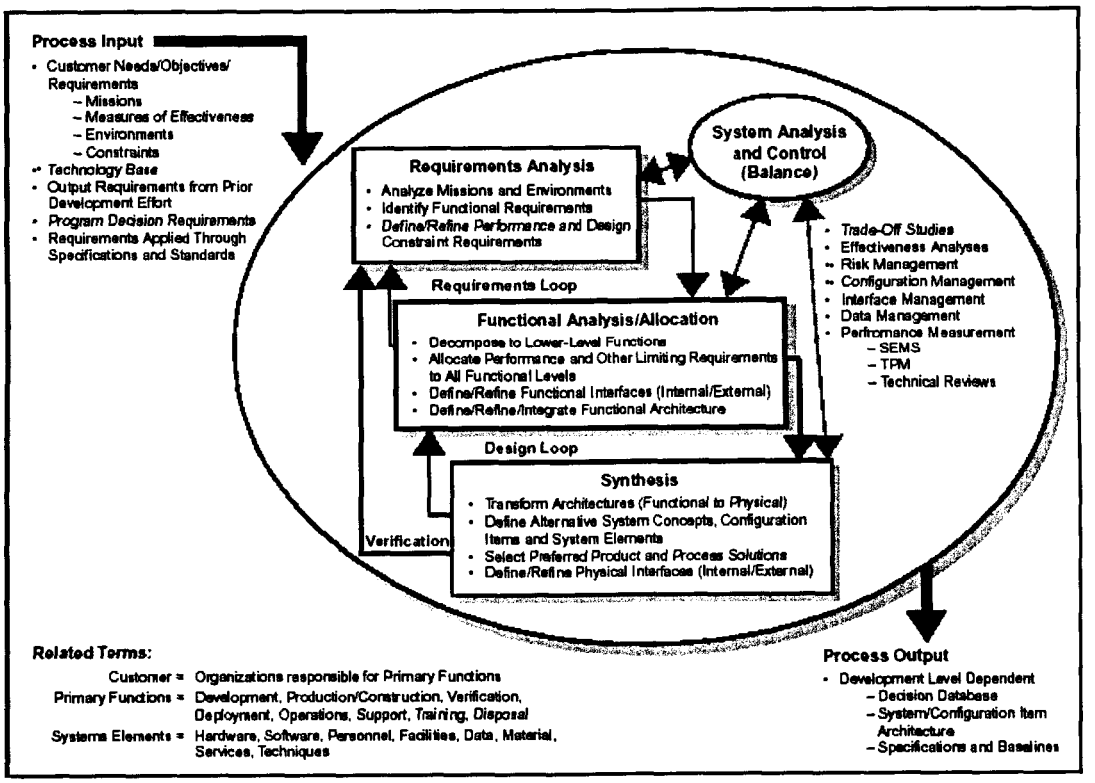


Figure 1-1. The Systems Engineering Process as Presented by the Defense Acquisition University

1.1.2 Evolving Systems Engineering Process

The DAU model, like all others, has been documented in the last two decades, and has expanded and developed to reflect a changing environment. Systems are becoming increasingly complex internally and more interconnected externally. The process used to develop the aircraft and systems of the past was a process effective at the time. It served the needs of the practitioners and resulted in many successful systems in our inventory. Notwithstanding, the cost and schedule performance of the past programs are fraught with examples of some well-managed programs and ones with less stellar execution. As the nation entered the 1980s and 1990s, large DoD and commercial acquisitions were overrunning costs and behind schedule. The aerospace industry and its organizations were becoming larger and were more

geographically and culturally distributed. The systems engineering process, as applied within the confines of a single system and a single company, is no longer the norm.

Today, many factors overshadow new acquisition, including system-of-systems (SoS) context, network centric warfare and operations, and the rapid growth in information technology. These factors have driven a new form of emergent systems engineering, which focuses on certain aspects of our current process. One of these increased areas of focus resides in the architectural definitions used during system analysis. This process will be differentiated by greater reliance on reusable, architectural views describing the system context and concept of operations, interoperability, information and data flows and network service-oriented characteristics. The DoD has recently made these architectural products, described in the DoD Architectural Framework (DoDAF), mandatory to enforce this new architecture-driven systems engineering process throughout the acquisition lifecycle.

1.1.3 Case Studies

The systems engineering process to be used in today's complex system-of-systems projects is a process matured and founded on the principles of systems developed in the past. The examples of systems engineering used on other programs, both past and present, provide a wealth of lessons to be used in applying and understanding today's process. It was this thinking that led to the construction of the four case studies released in this series.

The purpose of developing detailed case studies is to support the teaching of systems engineering principles. They will facilitate learning by emphasizing to the student the long-term consequences of the systems engineering and programmatic decisions on program success. The systems engineering case studies will assist in discussion of both successful and unsuccessful methodologies, processes, principles, tools, and decision material to assess the outcome of alternatives at the program/system level. In addition, the importance of using skills from multiple professions and engineering disciplines and collecting, assessing, and integrating varied functional data will be emphasized. When they are taken together, the student is provided real-world, detailed examples of how the process attempts to balance cost, schedule and performance.

The utilization and misutilization of systems engineering learning principles will be highlighted, with special emphasis on the conditions that foster and impede good systems engineering practice. Case studies should be used to illustrate both good and bad examples of acquisition management and learning principles, to include whether:

- Every system provides a balanced and optimized product to a customer
- Effective Requirements analysis was applied
- Consistent and rigorous application of systems engineering Management standards was applied
- Effective Test planning was accomplished
- There were effective major Technical program reviews
- Continuous Risk assessments and management was implemented
- There were reliable Cost estimates and policies
- They used disciplined application of Configuration Management
- A well defined System boundary was defined
- They used disciplined methodologies for complex systems
- Problem solving incorporated understanding of the System within bigger environment (customer's customer)

The systems engineering process transforms an operational need into a set of system elements. These system elements are allocated and translated by the systems engineering process into detailed requirements. The systems engineering process, from the identification of the need to the development and utilization of the product, must continuously integrate and balance the requirements, cost, and schedule to provide an operationally effective system throughout its life cycle. Case studies should also highlight the various interfaces and communications to achieve this optimization, which include:

- The program manager/systems engineering interface essential between the operational user and developer (acquirer) to translate the needs into the performance requirements for the system and subsystems.
- The government/contractor interface essential for the practice of systems engineering to translate and allocate the performance requirements into detailed requirements.
- The developer (acquirer)/User interface within the project, essential for the systems engineering practice of integration and balance.

The systems engineering process must manage risk, known and unknown, as well as internal and external. This objective will specifically capture those external factors and the impact of these uncontrollable influences, such as actions of Congress, changes in funding, new instructions/policies, changing stakeholders or user requirements or contractor and government staffing levels.

Lastly, the systems engineering process must respond to "Mega-Trends" in the systems engineering discipline itself, as the nature of systems engineering and related practices do vary with time.

1.1.4 Framework for Analysis

The case studies will be presented in a format that follows the learning principles specifically derived for the program, but will utilize the Friedman-Sage framework to organize the assessment of the application of the systems engineering process. The framework and the derived matrix can play an important role in developing case studies in systems engineering and systems management, especially case studies that involve systems acquisition. The framework presents a nine row by three column matrix shown in Table 1-1.

Table 1-1. A Framework of Key Systems Engineering Concepts and Responsibilities

Concept Domain	Responsibility Domain		
	1. Contractor Responsibility	2. Shared Responsibility	3. Government Responsibility
A. Requirements Definition and Management			
B. Systems Architecting and Conceptual Design			
C. System and Subsystem Detailed Design and Implementation			
D. Systems and Interface Integration			
E. Validation and Verification			
F. Deployment and Post Deployment			
G. Life Cycle Support			
H. Risk Assessment and Management			
I. System and Program Management			

Six of the nine concept domain areas in Table 1-1 represent phases in the systems engineering lifecycle:

A. Requirements Definition and Management

B. Systems Architecting and Conceptual Design

C. Detailed System and Subsystem Design and Implementation

D. Systems and Interface Integration

E. Validation and Verification

F. System Deployment and Post Deployment

Three of the nine concept areas represent necessary process and systems management support:

G. Life Cycle Support

H. Risk management

I. System and Program Management

While other concepts could be have been identified, the Framework suggests these nine are the most relevant to systems engineering in that they cover the essential life cycle processes in systems acquisition and the systems management support in the conduct of the process. Most other concept areas that were identified during the development of the matrix appear to be subsets of one of these. The three columns of this two-dimensional framework represent the responsibilities and perspectives of government and contractor, and the shared responsibilities between the government and the contractor.

The important feature of the Friedman-Sage framework is the matrix. The systems engineering case studies published by AFIT employ the Friedman-Sage construct and matrix as the baseline assessment tools to evaluate the conduct of the systems engineering process for the topic program. The Friedman Sage matrix is not a unique systems engineering applications tool per se, but rather a disciplined approach to evaluate the systems engineering process, tools, and procedures as applied to a program.

The Friedman-Sage matrix is based on two major premises as the founding objectives:

- In teaching systems engineering, case studies can be instructive in that they relate aspects of the real world to the student to provide valuable program experience and professional practice to academic theory.
- In teaching systems engineering in DoD, there has previously been a little distinction between duties and responsibilities of the government and industry activities. More often than not, the government role in systems engineering is the role as the requirements developer.

1.2 TBMCS Major Learning Principles

Table 1-2 depicts the Friedman-Sage matrix summarizing the Theater Battle Management Core System (TBMCS). The highlighted cells are the key processes this case study will address. This section will give a brief overview of the systems engineering processes that will serve as the key learning points.

LP 1, The requirements process for producing the first release of TBMCS was broken. The user and acquisition communities never were on the same page. The users of the system did not produce an Operational Requirements Document (ORD); rather, they told the acquisition program to use the existing requirements for the legacy system [3]. In turn, the acquisition community, knowing the legacy requirements were not sufficient, produced a Technical Requirements Document (TRD) that described the technical strategy for TBMCS and formed the basis for the contractor to develop the system-level specification. The three sets of documents did not align and as a result there were no performance requirements; instead, they were established as goals. The user also did not develop a concept of operations (CONOPS) describing how the system was to be used, and the contractor did not develop a concept of employment (CONEMP).

LP 2, The system architecture was initially defined at too high a level, thus impacting the design and development of the system. The contractor was constrained by a lack of requirements and by government mandates to use both government and commercial software and hardware products. The contractor had defined a layered approach and adopted the Common Object Request Broker (CORBA) as middleware, but limited understanding of the technology and insufficiently detailed definition of the interfaces had tremendous impact on the development and integration schedule [4].

LP 3, The system and subsystem design was severely hampered by the complexity of legacy applications, misunderstanding of the maturity and complexity of commercial and third party software products, and the lack of understanding of how the system would be used and employed by the user. In addition the lack of detail and documentation had significant impacts on system design and test. The major results were schedule slippages and a reformed acquisition process. The impacts on the program affected the schedule, cost, and performance.

LP 4, Integration for a system of this complexity was very difficult. Integration was required in three areas: applications, external interfaces, and databases. Integrating applications proved very expensive in terms of cost and schedule. A constraint was the directed reuse of software at both the application and infrastructure layers, as defined in the TRD [5]. The intent was to build a common software infrastructure with open interfaces that would allow third party applications to plug in and play. For such an implementation to work the interfaces must define the inputs and outputs in sufficient detail, and the third party must be willing to modify its product. This approach is very difficult to implement, especially when the contractor does not own or control the products. Another influence was that the user dictated functionality on the basis of concept demonstrations. The products were never mature and cost millions of dollars to fix and fit into the baseline.

The contractor's software development kit was immature and difficult for subcontractors to use and implement. TBMCS has over 64 external interfaces. The contractor was not able to simulate and or exercise those interfaces until system test. The other major issue was configuration control of the interfaces: formal agreements were not always in place and changes were prevalent. Integration with the intelligence database was very difficult. The baseline continued to move and the interfaces to the applications were troublesome. The major constraint was the impossibility of locking down the databases and interfaces, because once TBMCS became operational in the field it had be interoperable with other systems. Making changes at the last minute prior to an operational test affected other parts of the system and forced a great deal of regression testing, resulting in cost increases and schedule slippages.

LP 5, Testing on TBMCS was problematic. There was tremendous pressure to field TBMCS as the system of record for the year 2000, but the system was not ready for test and all the test planning was bypassed. Also, without firm requirements, it was difficult to ascertain what the pass/fail criterion was. The testing process was parallel and tests overlapped without sufficient time to fix the problems identified. A major constraint was the inability to test in an environment that represents the operational environment. Also, not having a CONEMP drove test planning to define tests that did not reflect the operational use. As a result, a latent design flaw was not discovered until operational test, causing the program to slip six months.

Table 1-2. A Framework for Systems Engineering Concept and Responsibility Domains [2]

Concept Domain	Responsibility Domain		
	1. SE Contractor Responsibility	2. Shared Responsibility	3. Government Responsibility
A. Requirements Definition and Management			LP 1, Requirements Definition and Management
B. Systems Architecting and Conceptual Design		LP 2, System Architecture	
C. System and Subsystem Detailed Design and Implementation		LP 3, System/Subsystem Design	
D. Systems and Interface Integration	LP 4, System Integration		
E. Validation and Verification		LP 5, Validation and Verification	
F. Deployment and Post Deployment			
G. Life Cycle Support			
H. Risk Assessment and Management			
I. System and Program Management			

2.0 SYSTEM DESCRIPTION

The Theater Battle Management Core System (TBMCS) is an integrated air command and control (C2) system that performs standardized, secure, automated air battle planning and execution management for Air Force, multi-service, and allied commanders in theaters of operation worldwide. It is deployed at C2 nodes at national-, force-, and wing-/unit-level elements in support of planners and decision makers at and below the Joint Force Air Component Commander (JFACC) level. TBMCS encompasses hardware, software, communications links, spares, personnel, training, and other resources to ensure robust and sustainable theater air operations. The system is modular and scalable for air, land, or sea transport, and its deployed configuration can be tailored to meet the requirements of the theater situation.

TBMCS provides the means to plan, direct, and control all theater air operations and to coordinate with land, maritime, and special operations elements. The system fully supports peacetime training and daily operations, as well as timely reaction to contingencies. TBMCS implements interoperable functionality with other command, control, communications, computers, and intelligence (C4I) systems in theater air warfare.

TBMCS has several core components, including migrating stovepipe or legacy systems such as the Joint Maritime Command Information System and the Contingency Theater Automated Planning System (CTAPS). TBMCS complies with the Defense Information Infrastructure Common Operating Environment (DII COE), and includes a common operational picture.

2.1 TBMCS Functional Overview

As shown in Figure 2-1, TBMCS spans three major C4I facilities – the Air Operations Center (AOC), the Air Support Operations Center (ASOC), and the Unit-Level Operations Centers – and connects to many external theater C4I systems. The following paragraphs describe the scope of automation in each of these operations centers [4].

Figure 2-2 depicts the functional breakdown for the AOC, ASOC, and unit-level operations. The intelligence, surveillance, and reconnaissance (ISR) and system support are the centerpieces for all three theater elements.

2.1.1 Air Operations Center

The AOC, which houses the JFACC, is the top-level C4I element in TBMCS. The AOC is responsible for intelligence development and theater targeting, air operations planning, airspace planning and control, tasking development and distribution, mission execution monitoring and re-planning, and force integration. TBMCS provides data communications, system administration and services, and mission applications for the AOC mission.

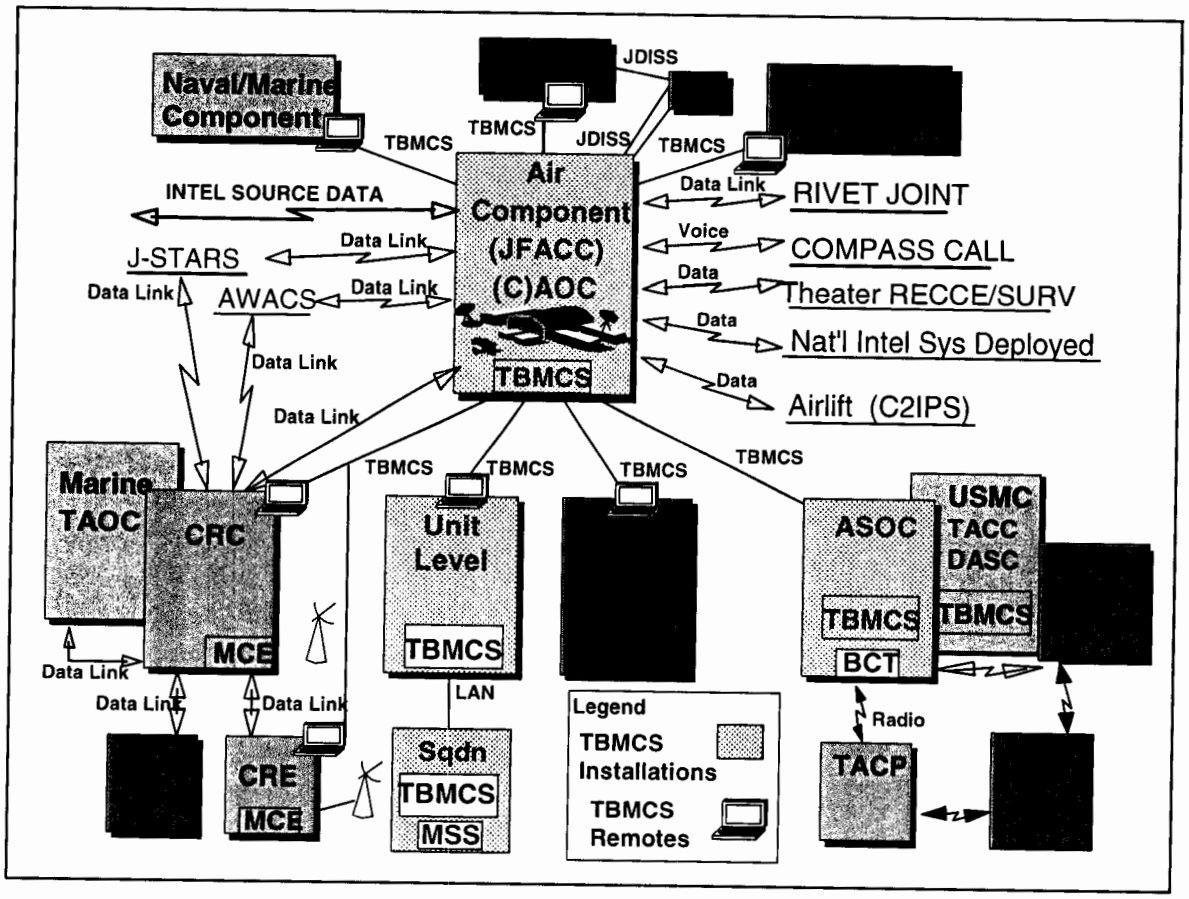


Figure 2-1. Notional Theater C4I

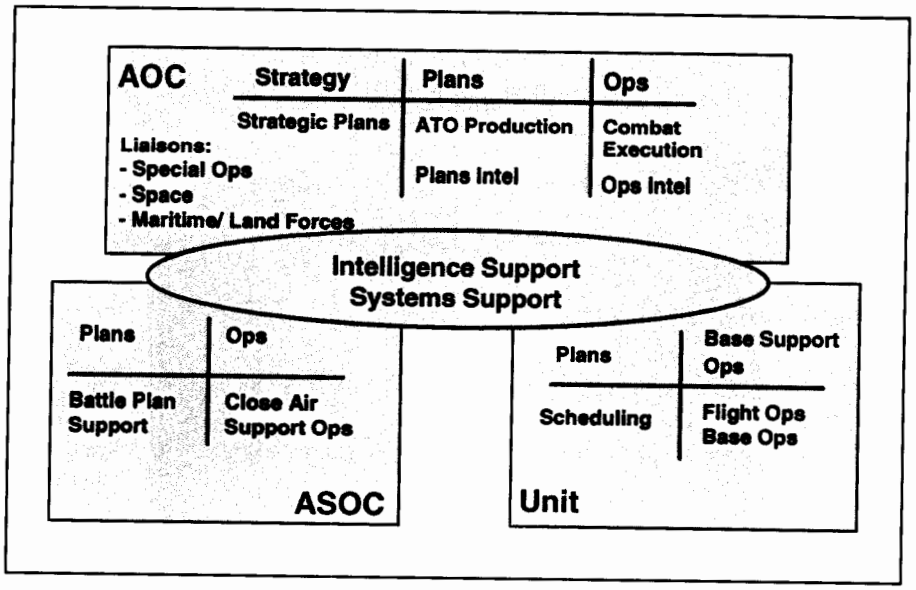


Figure 2-2. TBMCS Functional Description

Although Figure 2-1 appears to depict the JFACC as the Air Force theater commander, the JFACC can come from the other military services. In these cases, the AOC may be located at that service's C2 facility. The TBMCS functions that are designated as joint modules support the JFACC in joint operational environments. Specifically, when TBMCS supports the operational concept of "JFACC Afloat" it supports the JFACC aboard a Navy command ship. Regardless of the JFACC's parent service, TBMCS provides remote terminal capabilities at a significant number of other C4I facilities to communicate C2 information. These remote terminals are notional, as also indicated in Figure 2-1, and can receive, filter, and sort tasking as well as monitor and update mission status.

2.1.2 Air Support Operations Center (ASOC)

The ASOC is a specialized operations center responsible for detailed planning, preparation, execution, direction, and control of the air effort supporting the ground force commander's maneuver objectives. In addition to "pre-planned" air support, the ASOC also provides fast response to requests for immediate close air support or reconnaissance. The ASOC receives and validates requests, coordinates with the approving authority, and tasks available air resources to meet the land component commander's requirements. TBMCS provides a mobile computer hardware configuration, data communications, and mission application software to assist the ASOC mission.

2.1.3 Unit-Level Operations

The unit level is the execution arm of TBMCS. This level is responsible for receiving tasking from higher headquarters, translating the tasking into a unit flying schedule, managing unit-level resources to fulfill the flying schedule, executing the flying schedule, and reporting the results. TBMCS provides data communications, core support, and mission application software for unit-level C2 and resource management missions.

2.1.4 Joint Intelligence Center (JIC)

The JIC is a high-level joint intelligence organization responsible for maintaining and disseminating information on enemy forces. It serves as the distribution node for intelligence information. TBMCS provides application functionality for the data communications, core support, and intelligence missions to the JIC.

2.1.5 External Interfaces

TBMCS interoperates with a number of other C4I and management information systems in the evolving theater battle management arena. Communication with airborne platforms such as the Airborne Warning and Control System (AWACS) and Joint Surveillance Target Attack Radar System (JSTARS) takes place through the Tactical Data Information Link (TADIL) and Joint Tactical Information Distribution System (JTIDS) networks and a common processor. TBMCS receives intelligence inputs at the Secret level, and produces and disseminates intelligence products (orders of battle and target information) at the collateral (Secret) and/or Secret Releasable levels for a variety of force- and unit-level users. This broader set of interfaces is depicted in Figure 2-3 [6].

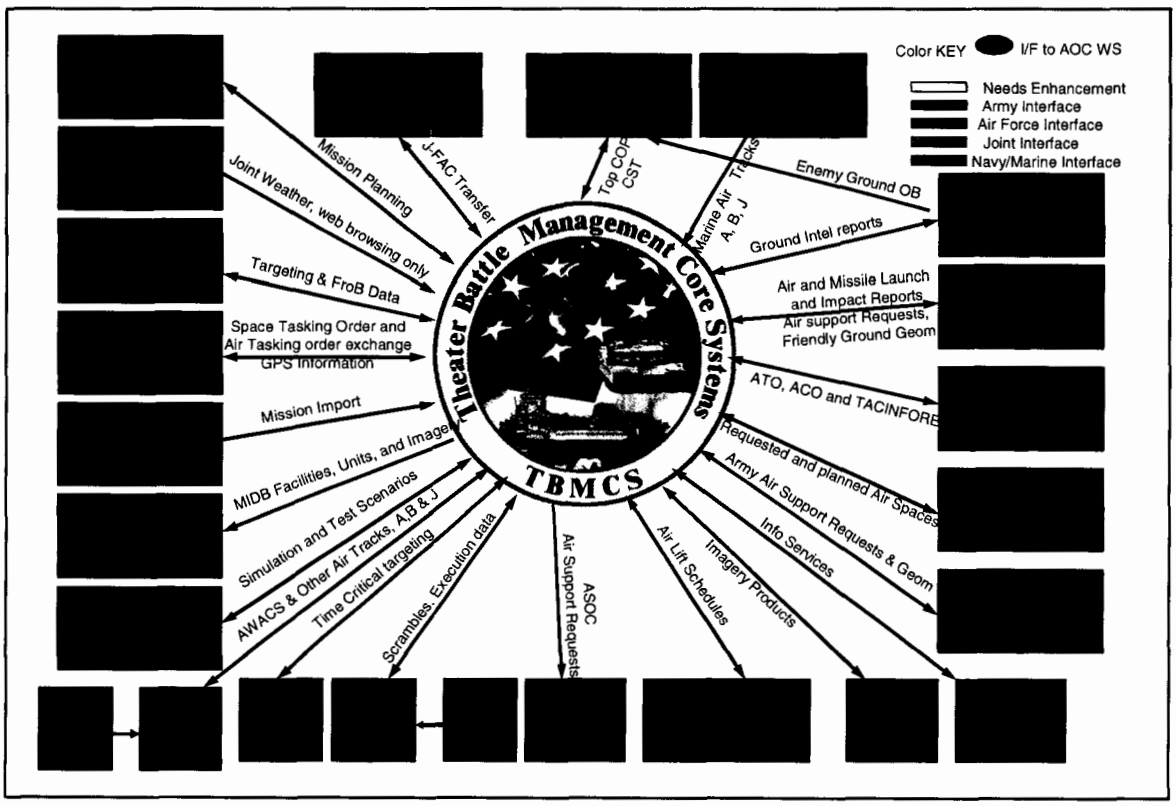


Figure 2-3. TBMCS Interfaces for V1.1.3

2.2 Air Battle Plan Rhythm

Figure 2-4 depicts the creation and execution of the air battle plan (ABP). It is typically a six-step process that takes place over a 72-hour window [7]. The steps are serial in nature but overlap during the 72-hour cycle. It starts with the JFACC guidance defining the objectives and strategy-to-tasks for the ABP. Intelligence analysts then generate a joint target list. Once the JFACC approves the list, the AOC develops the master air attack plan, taking the available resources into account. The AOC next produces an ABP from which the air tasking order (ATO) is created and disseminates it to the joint organizations. Units then plan their missions to include all mission parameters (weapon types, assigned pilots, takeoff and landing times, routes, and targets). The units execute the ABP and feed the results back into the process. The cycle repeats until the JFACC objectives have been met.

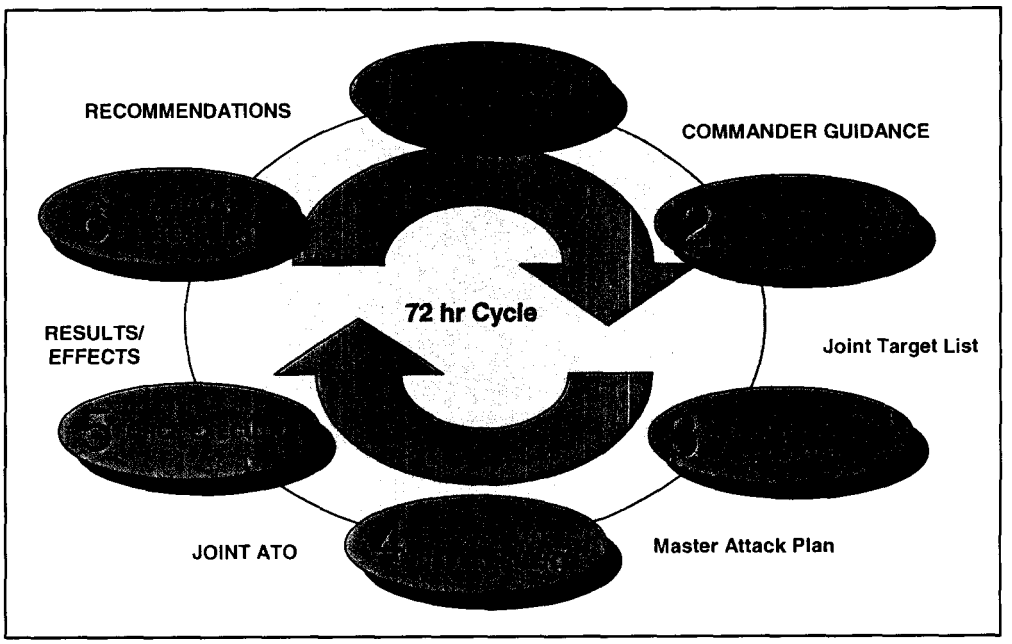


Figure 2-4. Air Battle Plan Process

Three ABP cycles in various stages can execute simultaneously. For example, the Combined Air Operations Center (CAOC) might be planning ABP C while developing ABP B and executing ABP A. The actual implementation is much more dynamic and fluid, but the basic structure remains the same. Figure 2-5 depicts the process and the overlaps.

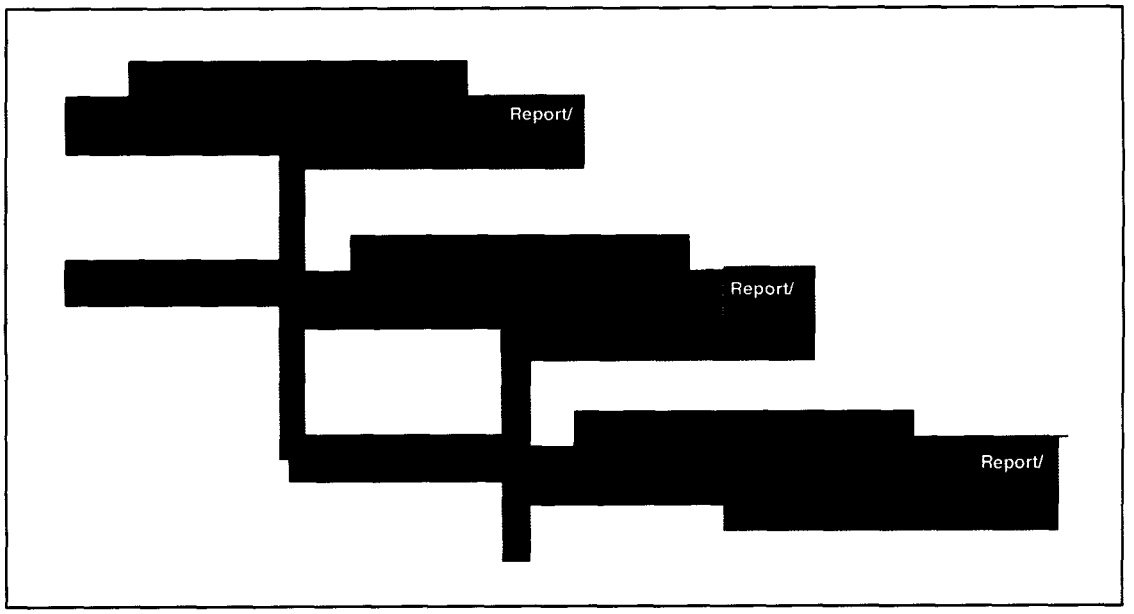


Figure 2-5. Today's "Battle Rhythm" = 72-Hour Cycle

3.0 TBMCS SYSTEMS ENGINEERING LEARNING PRINCIPLES

As the TBMCS program developed (1995–2000), roles and responsibilities shifted between the prime contractor, Lockheed Martin Integrated Systems and Solutions (LM-IS&S; hereafter referred to as LM), and the government. They became predominantly shared functions after the core baseline, Version 1.0.1 (V1.0.1), passed operational test and was approved for system fielding in October 2000. Table 1-2 highlights the responsibilities for the nine core processes.

TBMCS was a classic example of a system-of-systems integration program subject to extraordinary external influences. The original implementation strategy called for the government to hire a contractor to integrate legacy systems using modern commercial off-the-shelf (COTS) information technologies. The intent was to provide a consistent user interface, independent of the application, riding on a common software backplane. It was also believed that the system would evolve over time and that seams between legacy systems would eventually disappear as the individual components merged into one integrated system. In addition, as system integrator, the contractor would have total responsibility for the system and would use its own development processes with minimal government oversight. Basically, this meant that the government identified an objective and then removed itself from active participation in achieving it. This approach sounded deceptively simple and straightforward, but it proved very difficult to implement because of the huge number of organizational interactions, external influences, and constraints on the program.

This section will analyze four systems engineering learning principles that have had profound impacts on the program: requirements, system architecture/design, system integration, and verification and validation. The subsection for each process will describe the specific roles, influences, impacts, and lessons learned.

3.1 Learning Principle 1 – Requirements Definition and Management

> The government did not produce a Concept of Operations, key operational performance parameters, or a system specification for the contractor. The contractor was responsible for generating a system segment specification that had performance measures as goals and not testable requirements. The government did produce a technical requirements document that defined a technical approach and levied certain standards on the contractor. There was no firm baseline for operational and system requirements from which the system could be built and tested. The requirements baseline was volatile up to system acceptance, which took place after operational test and evaluation.

The requirements process for TBMCS V1.0.1 was profoundly flawed from the start. The user and acquisition communities were never really in sync. The acquisition community had a utopian vision of a single modern, integrated, joint C2 system, but had no operator requirements to support it; instead, the requirements were legacies from the existing systems being integrated into TBMCS. TBMCS itself had no requirements and no CONOPS that described how the system would work as single integrated capability. The test community and other services found this a major problem. What capabilities was TBMCS supposed to provide, and how was the system to be used? As a result, TBMCS lacked a system specification, and system performance measures were merely goals rather than hard requirements. The criteria for assessing system

performance became somewhat subjective and left room for interpretation. In fact, the formal, documented performance was not agreed to until the operational test plan was approved. The testing process was long and arduous. In addition, the requirements were derived from legacy functionality and continually changed depending on which software application the government wanted LM to incorporate into the baseline – a critical problem in itself.

The requirements process reflected a shared effort between the government and the contractor. The chart shown in Figure 3-1 was used during the early phases of TBMCS to illustrate the sequence of reviews, with the notes showing government vs. contractor tasks for each review [4].

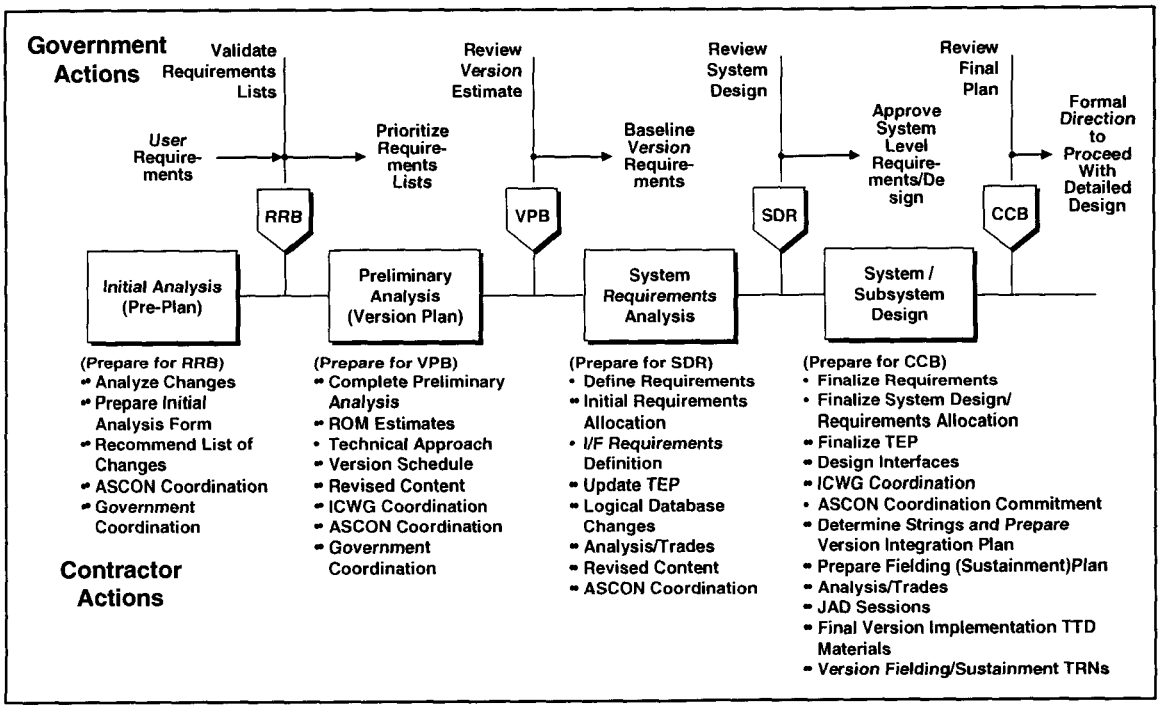


Figure 3-1. Version Planning Process

Operational Requirements Process

The operational requirements definition for TBMCS was ambiguous at best. TBMCS is an Air Force-led program with significant joint service participation; thus, the requirements came from many sources and evolved over time (see Figure 3-2).

Initially (1995), TBMCS followed a MIL STD-2167A type of process, starting with a Requirement Review Board about six weeks after contract award. As a process improvement after the TBMCS core release (V1.0.1, October 2000), the Air Force established an Operational Requirements Working Group (ORWG) chaired by the Air Force Command, Control, Intelligence, Surveillance, and Reconnaissance Center (AFC2ISRC; the lead agency for the government's requirements process), with representatives from the services, the system program office (SPO), the contractor, and the Joint Chiefs of Staff (JCS). While this group helped to prioritize requirements, its main focus was on the "little r's" for fixing legacy system problem reports and not on the big "R's" for defining TBMCS requirements.

The operational requirements baseline did not stabilize until September 1999, four years after contract award. The foundation of the operational requirements came from three legacy systems: those for CTAPS, the Wing Command and Control System (WCCS), and the Combat Intelligence System (CIS) [3]. The user, AFC2ISRC, believed that these requirements remained valid and that there was no need to establish a new operational requirements document (ORD). The user also believed the CONOPS for CTAPS was sufficient to serve as the TBMCS CONOPS. The acquisition program office disagreed: CTAPS requirements were not representative of what that acquisition office had in mind for TBMCS. For example, the requirements described neither the deployment of the system nor the interaction among the AOC, ASOC, and the Unit-Level Operation Centers.

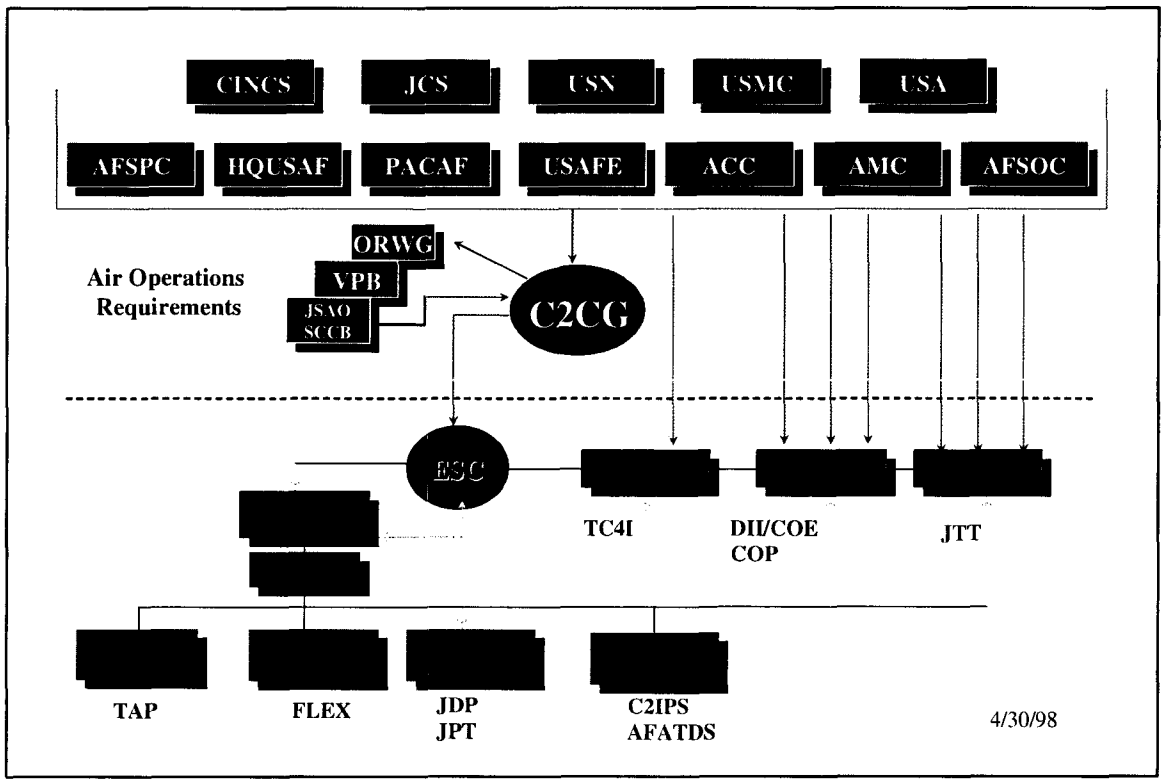


Figure 3-2. TBMCS Participating Organizations (circa 1998)

Figure 3-2 shows the number of organizations involved in the TBMCS requirements process in the year before TBMCS underwent its first operational test and evaluation (OT&E). This chart illustrates how the operational user community was feeding requirements into TBMCS from the top down, while various functional components that were directed into the system simultaneously drove requirements from their existing implementations back into the system. Responsibility for defining the requirements fluctuated among the Air Force major commands (MAJCOMs) as well as the numbered Air Forces. There was no formal concept of employment. The Tactics, Techniques, and Procedures (TTPs) were inconsistent and varied from theater to theater. The joint user community, contractor, and test community never agreed on and formalized the measures of effectiveness (MOEs) and performance (MOPs); instead, all participants had their own "pet rocks."

To gain some sort of consensus among the stakeholders, AFC2ISRC compiled the requirements from the three legacy ORDs into a System Version Requirements Document (SVRD) in January 1998. The joint operational test community did not consider the requirements in the SVRD testable, because they were too detailed and it was difficult to identify the critical operational issues. The test community also complained that the SVRD was not an ORD and there was no criterion for operational test acceptability [8].

In January 1999 the ORWG produced another document, called Mission Critical Functions (MCFs) that defined each service's legacy MCFs [4] The test community wanted a still higher level of abstraction, so AFC2ISRC mapped the 45 MCFs originally identified into five Key Legacy Functions (KLFs). In September 1999, six months after TBMCS failed its first operational test, the user community redefined the KLFs to include only 19 MCFs; the other 26 MCFs were deemed important but not critical. The re-scoping was intended to minimize risk and improve the probability of receiving a favorable fielding decision. The fielding decision for TBMCS V1.0.1 was to be based on the effectiveness and suitability of the five KLFs:

1. The capability to nominate and prioritize targets,
2. The capability to plan and disseminate the daily ABP,
3. The capability to receive and parse the ABP,
4. The capability to plan a detailed flying schedule within four hours (for Air Force units), and
5. The capability to monitor and control execution of the ABP.

TBMCS demonstrated that it satisfied these KLFs at the multiservice OT&E (MOT&E) in July 2000 and received a favorable fielding decision.

System Requirements Process

TBMCS was envisioned as the C2 system for theater-level air operations at both the operational and tactical levels of war for joint and coalition contingencies. The program was to evolve by integrating three legacy systems into a single C2 system used by the joint AOC and the theater components. TBMCS was to provide a common and shared air operations and intelligence database, as well as a common suite of tools to plan, manage, and execute the ABP, and was to include a common operational picture for shared situational awareness.

Because there was no CONOPS, and the ORDs of the legacy systems did not have any technical performance metrics, there were no real requirements or performance baseline from which the Air Force could build a system specification. The requirements were expressed in terms of capabilities based on legacy functionality; the tacit guideline was that functionality and performance should not be degraded from those of the legacy systems.

The TBMCS System Program Director (SPD) recognized that it would be very difficult to produce a formal system specification, given the time constraints for releasing the request for proposals (RFP), and instead asked The MITRE Corporation[1] to produce a technical requirements document (TRD) that formed the basis for the contractual requirements baseline.

---

[1] MITRE operates a Federally Funded Research and Development Center (FFRDC) that serves as the Electronic Systems Center's lead systems engineering support organization.

At the same time, leading up to the development of the TRD, the Program Executive Office (PEO) tasked MITRE to perform a systems engineering study of how the Air Force would integrate disparate legacy systems between force- and unit-level operations, and to determine if TBMCS could provide a common infrastructure that would allow the applications to "plug and play" and be shared among the services. In essence, MITRE was asked to define the technical framework that would allow this integration.

The MITRE study concluded that systems supporting air operations for the theater were not integrated, but exhibited considerable commonality and duplication. Therefore, moving to a common set of services was technically feasible. The study recommended the services to be incorporated, including messaging, databases, common operational picture, communications, and security [9]. MITRE's systems engineers prescribed an object-oriented approach to encapsulate the legacy or third-party applications and provide a common message service via an object request broker. This analysis formed the technical approach for TBMCS and was reflected in the TRD.

The requirement to use government-furnished equipment (GFE) for many TBMCS functions had a major influence on the requirements analysis. The government prescribed the use of specific hardware, which varied depending on the service branch that would use TBMCS: the Air Force used Sun Microsystems, the Navy used Hewlett-Packard, and the Marines used a combination of both. The size of the configuration would also change according to the type of deployment. The three types of deployment packages were: Quick Response Package (QRP) – for human relief types of missions, Limited Response Package – for small-scale military operations, and Theater Response Package (TRP) – for large-scale, Desert Storm-like conflicts.

The government also prescribed the specific software applications (COTS and government off-the- shelf [GOTS]) and infrastructure for TBMCS, including the mandate to use the DII COE as the core of the infrastructure. To make matters worse, many of these systems were undergoing parallel development at the time TBMCS was being created. This presented additional challenges, because it required all stakeholders to achieve a reasonable current baseline of the products that would be stable long enough to allow the contractor to integrate them into the larger TBMCS.

Because of these multiple and often conflicting demands, TBMCS was perceived as being slow to adapt. Often, this was due to the disproportionate impact of what might have seemed like a minor issue to an outside observer. For example, at one point TBMCS was based on Solaris 2.5.1, while Sun had released Solaris 2.8 (a.k.a. Solaris 8). The move to these new releases of the Sun operating system was delayed by dependencies on COE products and by the sheer cost of a massive upgrade of COTS products to match this new baseline.

Another major factor affecting the requirements was database consolidation and compliance with the ATO message standard, the United States Message Text Format (USMTF). The standard covered not only the format but also the ATO functionality. TBMCS had to consolidate 13 separate databases into two: one for air operations and one for intelligence.

The last major influence was security and coalition operations. TBMCS had to be able to operate at multiple security levels and support coalition operations, meaning that the system had to be releasable to our coalition partners.

The contractor, together with the SPO engineering staff, formed a systems engineering working group to develop the system specification and prepare for the System Requirements Review (SRR). Given the vast amount of information available during the first months of the TBMCS contract, the team broke down the requirements provided and performed an initial analysis of the candidate solutions based on specific factors, such as the COTS/GOTS products mandated from above. The SRR took place in March 1996, five months after contract award, and the requirements baseline was then placed under configuration control and managed by a tool called a Requirements Traceability Matrix. The initial System Design Review (SDR) was held about six months later.

The timeline for this analysis and design phase did not permit much in the way of prototyping and full trade studies. Such a fast-paced engineering process can work well for prototypes where an opportunity exists to revisit decisions and rework the product, but TBMCS was attempting to define a relatively large system of systems. Thus, the program was forced to review and rework some areas based on the results of formal tests instead of feedback from internal activities.

Once a baseline was established, the TBMCS program was able to move forward by modifying more traditional engineering processes. Figure 3-3 shows how the team adapted an existing engineering process of design/development with periodic reviews to work in an environment where many of the component products in the system are created by organizations outside the prime contractor's direct control. During the periodic reviews the TBMCS SPO and its MITRE engineering team were invited to participate and to initiate additional discussions at an engineering level early in the Quality Points (QP) process [4].

The system requirements baseline was placed under LM configuration control, but the real baseline was the allocated baseline: the requirements were being defined as the system was being built. LM held Initial Design Evaluations to ensure understanding and gain user buy-in, but that was not the equivalent of formal documentation.

The government also did not concern itself with the suitability of the system and its components for operational test. The System Segment Specification (SSS) was updated periodically, but never really reflected the baseline; instead, it was an afterthought and lagged behind the current program requirements. In addition, the government continually changed the allocated baseline with mandated third-party products, which did or did not reflect the agreed-to requirements.

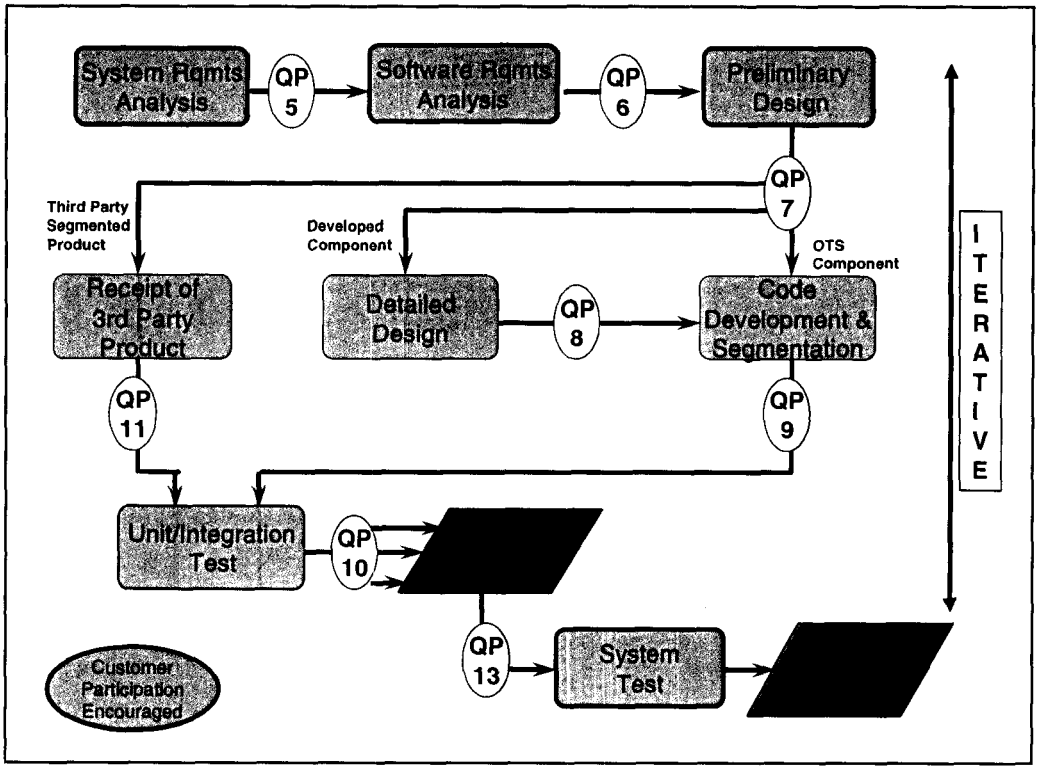


Figure 3-3. QP Process Flow

Thus, the requirements process was very loose and managing expectations was extremely difficult. The implications affected performance at the system-of-systems level because changes in the lower-level requirements did not flow back up to the system level baseline and allow LM to determine the overall impact. In one case the impact only became evident in operational test, which revealed a major problem in the intelligence database that resulted in an eight-month schedule slip.

3.2 TBMCS Learning Principle 2 – Systems Architecture

> The system architecture was defined at too high a level, which had a tremendous impact on system design and development. The government's mandate for a software reuse and use of commercial software products were contradictory and problematic for the system development. The layered system architecture did support system evolution and migration to modern technologies.

Responsibility for designing the system architecture for TBMCS was shared between the government and LM. The key tenets for the system architecture were:

- Permit collaborative air battle planning and execution with automated distribution,
- Provide common situational awareness,
- Support coalition operations,
- Offer shared air operations and intelligence databases,
- Ensure seamless integration between force- and unit-level operations,

- Operate in fixed and deployed locations,
- Support system evolution and facilitate software application reuse – especially with existing legacy systems,
- Leverage commercial hardware and software,
- Provide a standard user interface, and
- Operate on government-furnished hardware and communications infrastructure.

The system employs a layered architecture and has migrated from a UNIX-based client/server to a PC-based, N-tiered, Web-based service-oriented architecture (SOA), as is shown in Figure 3-4 [4].

LM did not develop the initial system architecture in accordance with the C4ISR framework, or use any automated tools to create the architecture. The contractor did perform a functional decomposition, allocate functions to subcomponents, and define the interaction and interfaces among the components. However, the government provided neither an operational architecture nor use cases to describe operations. This definitely caused problems with developing a CONOPS and made it far more difficult for LM to gain a deep understanding of how the system would be employed. That, in turn, resulted in a major performance flaw in the intelligence database that was only discovered at operational test.

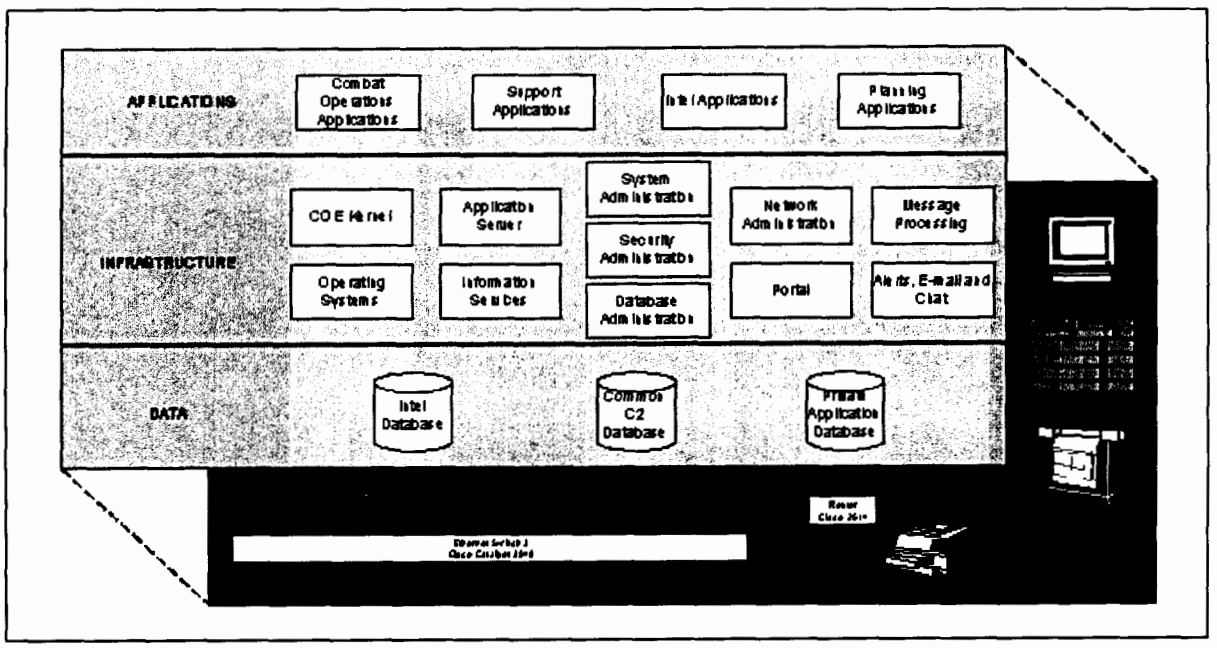


Figure 3-4. N-Tiered Architecture

The original architecture was at a very high level and thus left considerable room for interpretation, which had a lasting effect on system integration. The software architecture was

very immature and was based on the DII COE segmentation process.[2] In the case of TBMCS, segments and the database layer were to interact through a common set of Application Program Interfaces (APIs) or a common information service layer. Figure 3-5 depicts the DII COE structure for TBMCS. Unfortunately, as the architecture was being built, the DII COE was still evolving and the inter-process communication API was not well defined. Similarly, the COTS products under consideration were still evolving and often did not meet performance requirements, especially the requirement for an automated database replication scheme. The third-party applications were designed to a different set of requirements and often did not scale or were incompatible with current versions of the COTS software product baseline, e.g., Netscape browser. Typically, TBMCS was two versions behind the commercial market standard.

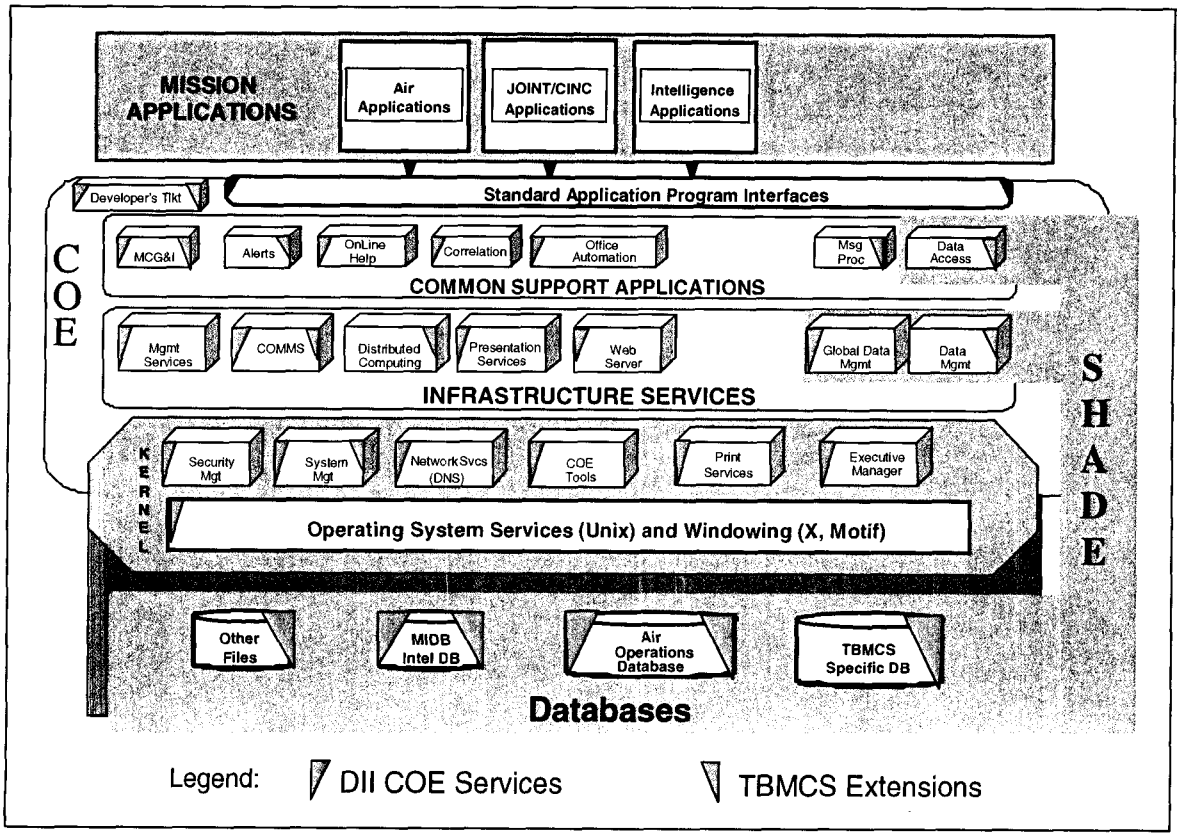


Figure 3-5. DII COE Architecture

The operational data architecture was well defined and understood (see Figure 3-6). The Air Force hardware architecture was also well defined and understood, but was based upon outdated equipment specifications (see Figure 3-7). However, the communications architecture was not well defined or understood until after the failure of the first operational test,

[2] A segment is a solely contained software capability that can operate independently or interact with other segments. Segmentation imposes a set of software development rules on both legacy and newly developed software, with the goal of allowing applications to be easily installed and integrated with the DII COE. It also allows installed applications and COE components to share data [10].

because the contractor was not able to test or integrate on the actual infrastructure until the operational test took place. This led to a major lesson learned.

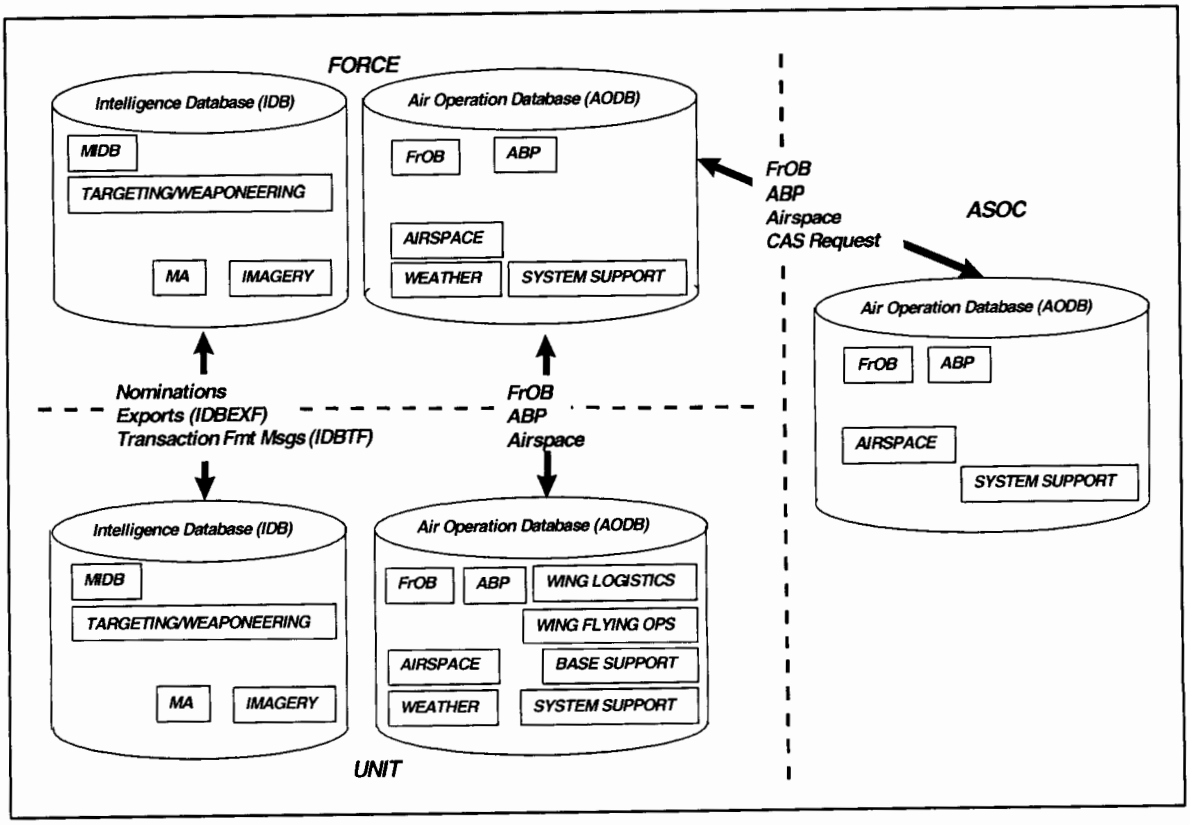


Figure 3-6. Data Architecture

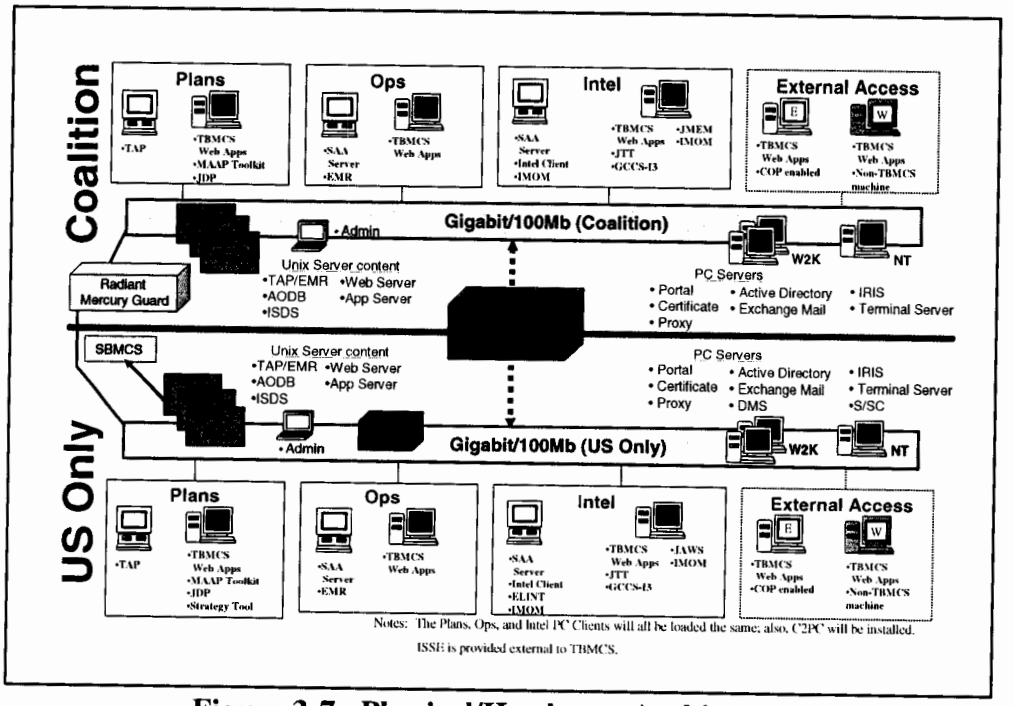


Figure 3-7. Physical/Hardware Architecture

The infrastructure varied from base to base, as well as among the different services. The communication firewall requirements varied from base to base as well. LM eventually put TBMCS on a virtual private network (VPN) that rides on the Defense Information Systems Agency's (DISA's) network and uses guard technology to communicate between U.S. and coalition systems. Figure 3-8 illustrates the current communications architecture.

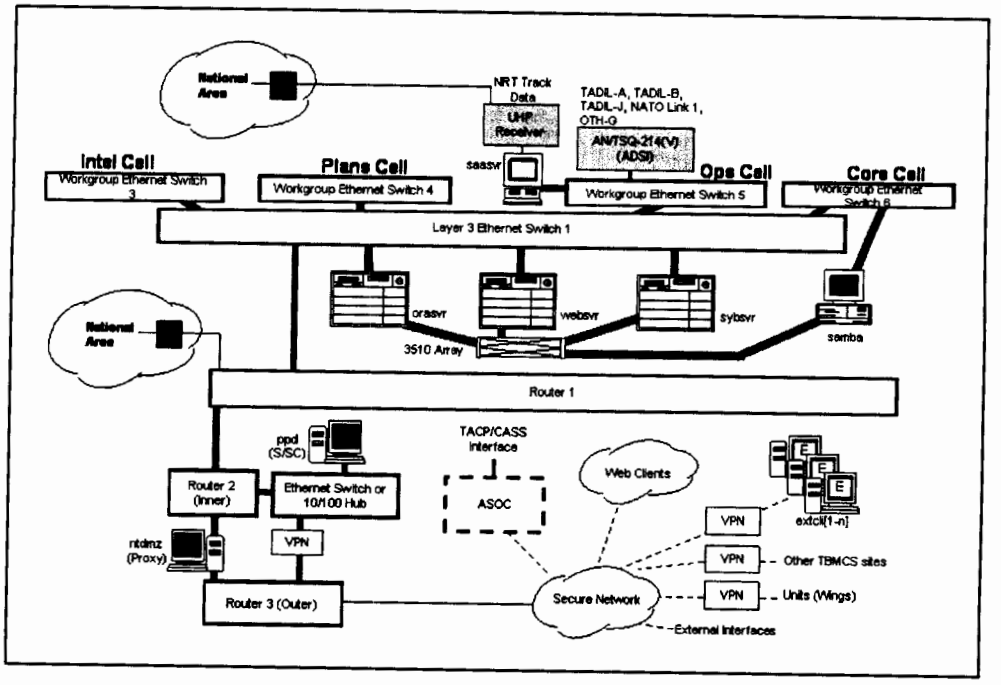


Figure 3-8. Communications Architecture

The initial system architecture produced by LM was conceptually sound, but lacked sufficient detail to enable LM to understand the issues and risks, which therefore were not discovered until well into the design and implementation phases of the program. However, the architectural concept of layers stayed the course over time and greatly facilitated the migration from a client/server to an N-tiered Web-based SOA. Keeping the data separate from the application and having a common infrastructure did establish a plug-and-play environment for application reuse. The concept also promoted legacy isolation, which allowed TBMCS to migrate to an SOA in a relatively short time.

As TBMCS went through its initial development and test phase, the program was under tremendous scrutiny for not having a "to-be" (objective) architecture or a vision for future capabilities. The fundamental reason was that the SPO was so focused on getting TBMCS through operational test that there was no money to develop a "to-be" architecture; the feeling was that there would be no future unless TBMCS passed its operational test. While TBMCS was going through operational test, the Air Force Scientific Advisory Board performed a study on the future of C2 and how to improve integration and interoperability. Dr. Alex Levis, the lead on architecture, recommended that TBMCS have a "to-be" architecture with a roadmap for evolving capabilities. The recommendation was well received and funding was provided to define both an architecture and a roadmap.

The resulting architecture followed the C4ISR framework and produced operational, system, and technical views. A majority of the work, especially the operational views, has since been absorbed by the AOC weapon system. Together with the government, LM developed system and technical views that describe how the current system would evolve into a Web-based system. The new TBMCS architecture still retains the existing databases, but is in the process of moving the applications off UNIX workstations to Microsoft Windows and browser-based clients. The architecture also uses open commercial standards for infrastructure and is migrating away from the DII COE infrastructure. The current architecture has been widely accepted within the Department of Defense (DoD) and user communities.

3.3 Learning Principle 3 – System/Subsystem Design

> The system and subsystem design was severely hampered by the complexity of legacy applications, misunderstanding of the maturity and complexity of commercial and third party software products, and the lack of understanding of how the system would be used and employed by the user.

The government and contractor share responsibility for the TBMCS design, which is based on government-directed software, hardware, and technology focus. The design stems from the family-of-systems approach used on the Global Command and Control System (GCCS) – the system the Joint Force Commander uses to plan and execute a Joint Task Force (JTF) contingency. The plan was to tailor the system depending on the task the user was performing, meaning that a menu of applications would run on a common software infrastructure that could be adapted to each workstation on the basis of user role or profile. Each military service would provide applications from its respective domains that could be reused as part of the GCCS baseline. For example, the Air Force would provide the air operations applications, while the Navy would provide the intelligence applications. The applications would run on both the

Navy's GCCS-M and the Air Force's TBMCS (the Air Force did not adopt the GCCS name for its force- and unit-level air operations system).

A majority of the system design for TBMCS V1.0.1 was downward directed and based on existing capabilities. V1.0.1 was a UNIX-based client/server system. As previously noted, the hardware platform for the Air Force and Marines was Sun Microsystems; for the Navy it was Hewlett-Packard. The communications design was a closed VPN running on the Defense Information Systems Network (DISN). The security design was system collateral Secret, with enclaves for other security levels and coalition operations. Information System Security Engineer guards were used to pass data from one enclave to another. The databases were relational and used COTS products: Oracle for the air operations database and Sybase for the intelligence database. Most of the software was written in C and C++ and was segmented under the DII COE concept, which allows applications to be easily installed and integrated with the DII COE and enables installed applications and COE components to share data [4].

Data sharing and application interfaces were accomplished through a common service layer called Data Access Agents (DAA). The DAA was implemented using a Common Object Request Broker (CORBA): a COTS product called ORBIX that followed the CORBA 2.0 standard. CORBA allows the DAA and other services to access data by using Interface Definition Language (IDL) scripts tailored to specific tasks. There was also a service layer for application-to-application interfaces and for application-to-infrastructure interfaces. Figure 3-9 depicts the service layers for the legacy applications. Unfortunately, the technology was still emerging and the object-oriented concept never really took hold at LM. The IDL was difficult to understand and implement and constantly caused problems during system integration and test with third-party products.

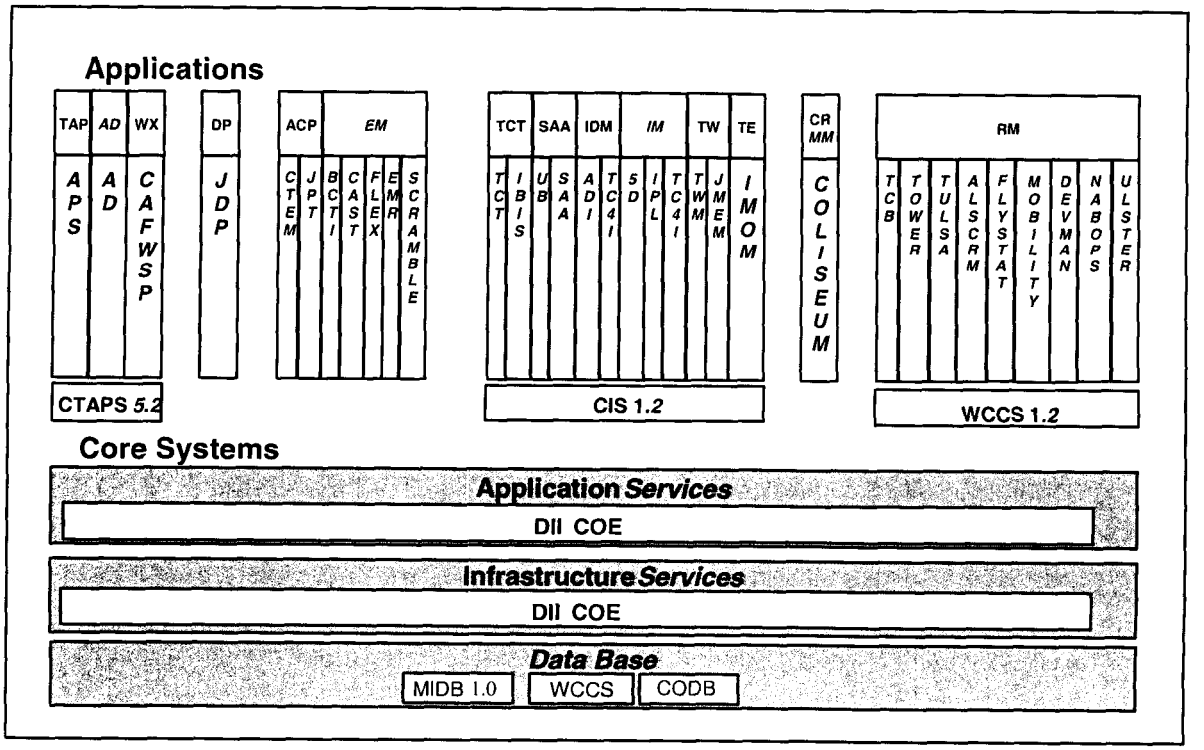


Figure 3-9. Legacy Application Service Layers

After the completion of operational test in August 2000, the Air Force exerted strong pressure for TBMCS to migrate to PC/browser clients and become more Web focused. The Air Force did not fully endorse the UNIX environment and wanted to use PCs as their workstations. Moreover, the DII COE and DAA were brittle and expensive to maintain: the infrastructure could not keep pace with the current versions of commercial information technology products. The TBMCS design therefore evolved from a client/server to an N-tiered architecture operating in a Java environment, as depicted in Figure 3-10.

The design called for migration over time, as shown in Figure 3-11. The first major change focused on moving the remote client from a UNIX client to a PC/browser user interface, which made it possible to access the applications over the network using Java's applet technology. This implementation proved very successful for the CONUS AOC as it responded to September 11, 2001: the flying units could use the browser at their home stations to view their assigned air defense missions.

The next major change brought more Web capability into the AOC by standing up a portal and creating an initial Java 2 Platform Enterprise Edition (J2EE) environment for Web application development, using a Web application server by BEA Systems called WebLogic. The last major change was to adopt the J2EE environment fully and upgrade the software infrastructure (i.e., the DII COE) using current commercial technologies. Adopting open standards and not dictating a specific design implementation has removed many constraints and considerable complexity, and has allowed the contractor to field capabilities at a much faster rate with less program risk.

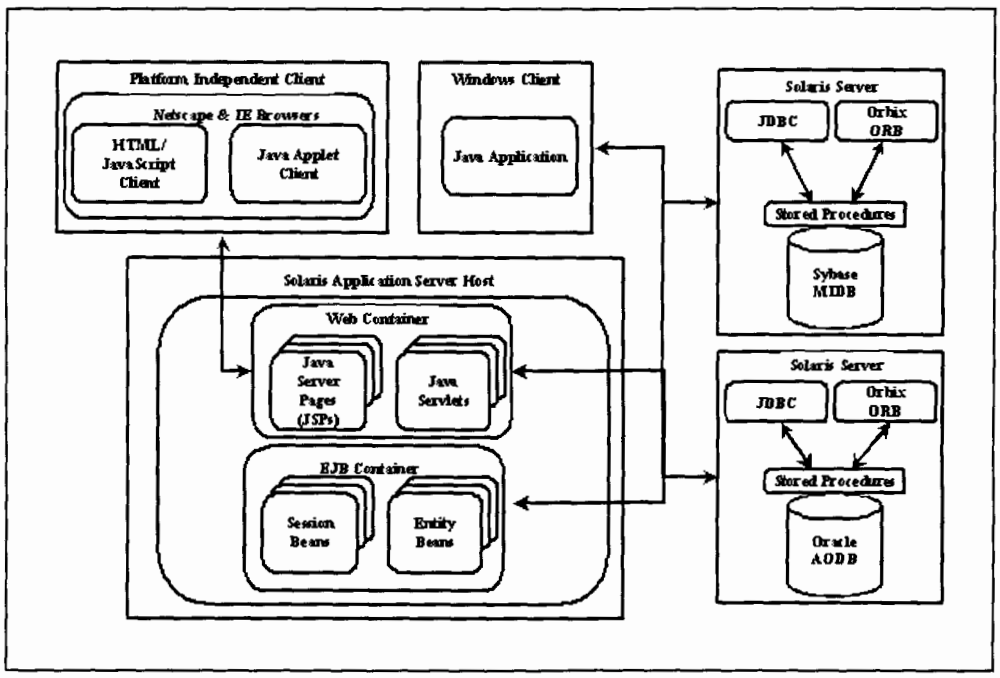


Figure 3-10. Java Environment

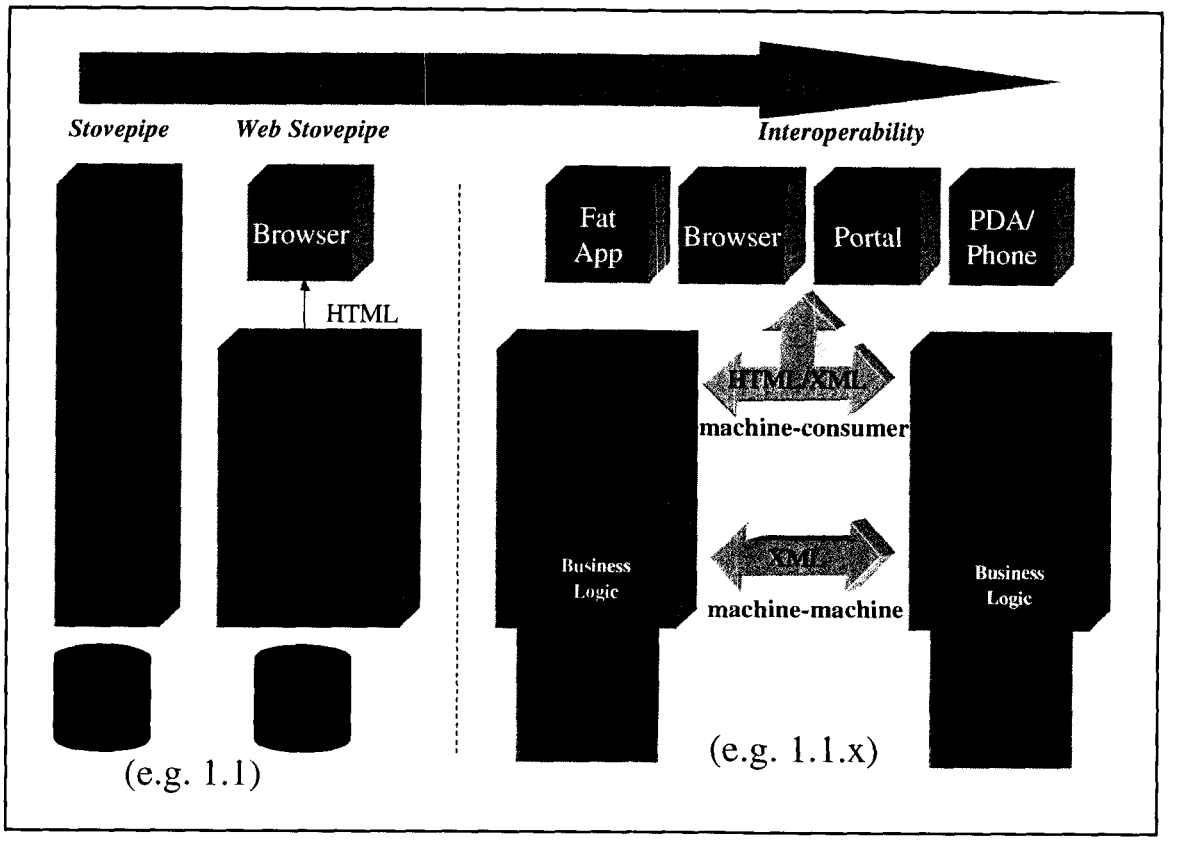


Figure 3-11. Web Migration

3.4 Learning Principle 4 – System Integration and Test

> Systems and interface integration was highly complex. System integration was very difficult because of the lack of detail in the system architecture and the mandate to use government-furnished equipment that was not necessarily compatible with commercial off-the-shelf products. Integrating third party software products were an arduous process and required extensive oversight. The external system interfaces were not managed and were often impossible to test at the contractor's facility.

Integration is one of the key systems engineering processes for TBMCS. The system involves four types of integration: (1) internal interfaces and subcomponents, (2) third-party applications, (3) external interfaces, and (4) databases.

LM is not only the prime contractor and developer but also, and most important, the system integrator. Ninety percent of TBMCS consists of third-party products or GFE, and a majority of the software is third-party or COTS. There are 76 applications and 413 segments involving over 5 million lines of software and two commercial relational databases – one for air operations and the other for intelligence. The system has two hardware baselines, and DISA runs the communications infrastructure. Sixty-four point-to-point external interfaces have connectivity with TBMCS.

As with requirement definition, the contractor and the government share responsibility for system integration. Clearly the contractor has overall responsibility, but the government bears

responsibility as well, especially when it directs that certain products or interfaces be incorporated into the system baseline. A major lesson learned from TBMCS is that if a third-party product does not integrate well – meaning it takes more time and money than the budget allows – the contractor should have the option to develop its own solution. The government, in turn, must provide stable interfaces and an environment that allows the contractor to test them. The government must also manage tight configuration of those interfaces to minimize interoperability problems with the system baseline.

LM uses a highly serialized process for development and integration, as referenced in Figure 3-12. LM defines the integration process in its Software Development Kit (SDK) [4].

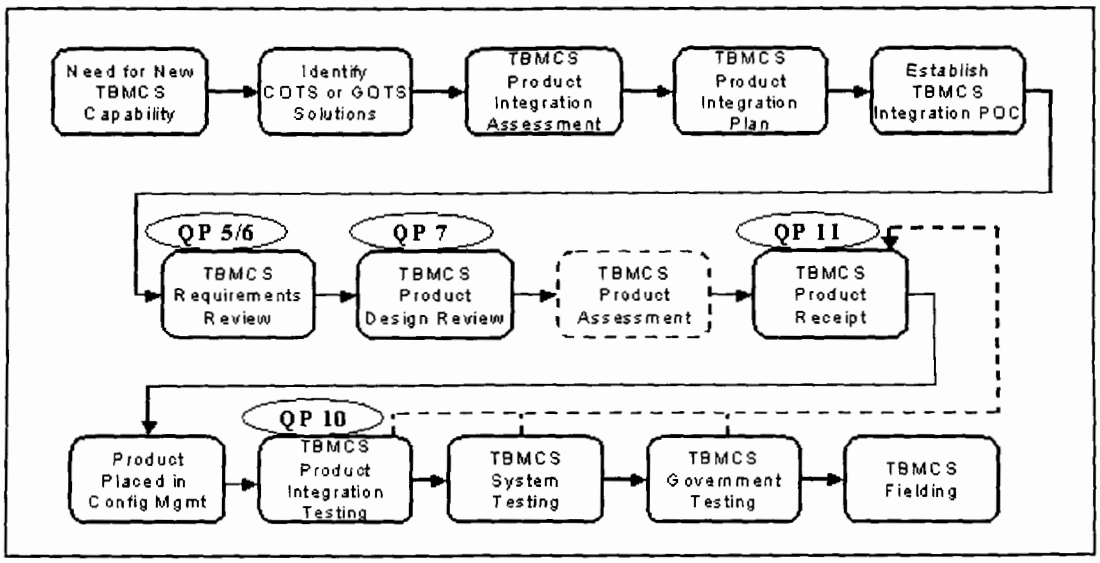


Figure 3-12. Third Party Integration Process Flow

The SDK for third-party integration states:

> Product Integration Process Flow [Figure 3-13] illustrates the flow of integration activities. The optional activities are shown with dashed lines. The activities are associated with a series of Quality Point (QP) product reviews to ensure a complete integration engineering process. QP checklists define the criteria for each phase of integration. At the time a product is identified for integration into TBMCS, the TBMCS Architecture Team will coordinate a product integration assessment. The assessment consists of completing a concise questionnaire that quickly characterizes the product in terms of the technical qualities of the product. Using the integration assessment, a product integration plan is developed, which includes a description of the objective, scope of the integration effort, and a schedule of integration milestone tasks. The completed questionnaire and plan are placed in the Software Development Folder (SDF) for the 3rd party product. A TBMCS product integration engineer will be designated as the Point of Contact (POC) responsible for managing the integration activities.

> The 3rd party product provider is expected to support the QP 5/6/7/11 events, product integration, and testing activities. The TBMCS product integration engineer will coordinate with the 3rd party product supplier to develop a set of specifications that will be used to verify the successful integration of the product (QP5 and QP6). In order to capture the technical context of the product within the TBMCS system and provide a technical description of the product after the specifications have been established, a Software Design Report (SWDR) is generated, which is verified at the QP7. TBMCS will also register the planned segments with the DII COE, if not already done. Then, the required sections of the SWDR and Product Integration Questionnaire are completed to document the technical and implementation details in addition to identifying all of the products to be integrated including the COTS/GOTS/Freeware products. A QP11 is conducted upon the Receipt of a 3rd party product, which consists of verifying the correct version of the product and associated documentation. The Unit Test Cases are reviewed prior to Integration Testing. The installation and configuration steps associated with the product are also reviewed. All software components requiring licenses are verified. The product is placed under SCM control.

This process has clearly improved over time and is now easy to repeat; however, determining the quality of the third-party product and coping with hidden designs during execution remain problems today. The Joint Targeting Toolbox (JTT) provides an excellent example. The services use this application to generate the Joint Priority Integrated Target List (JPITL). JTT, which is designed to support classified operations at the JTF level, works extremely well in standalone mode, but when forced to operate in a collaborative and distributive mode among the different combat components it is slow, erratic, and not user friendly. To make it function properly within TBMCS, LM had to use brute force to make the software interface with the other system components.

The real problem with third-party integration is that the prime contractor does not have control over the configuration of the product. This forces the government to broker changes to the product when issues arise, which often results in delays and increased cost. In the abstract, the requirement to use the DII COE as a common software infrastructure represented a worthy goal; unfortunately, the infrastructure was slow to mature and could not keep pace with commercial information technology, making integration very difficult and expensive.

LM's standard systems engineering process focuses strongly on product teams (see Figure 3-13) [7]. Gregg Hinchman, former chief architect, stated, " ...this is the first program where we (LM) made an effort to not distinguish between system engineering and software engineering ... We instituted IPTs [integrated product teams] ... We became product focused ... We lost the system engineering activities vs. the software engineering activities and [they] became product engineering activities..."[3] In essence, the system engineers were embedded in the development IPTs at the subcomponent level. LM did ask the program office to fund 12 systems engineers to assess performance, but the request was denied due to funding constraints.

---

[3] Gregg Hinchman, personal interview with the author, __ November 2003.

LM performed system testing and string testing, but the tests did not exercise concurrent processes at the system-of-systems level to assess overall system performance. This failure can be directly attributed to the lack of a system CONEMP.

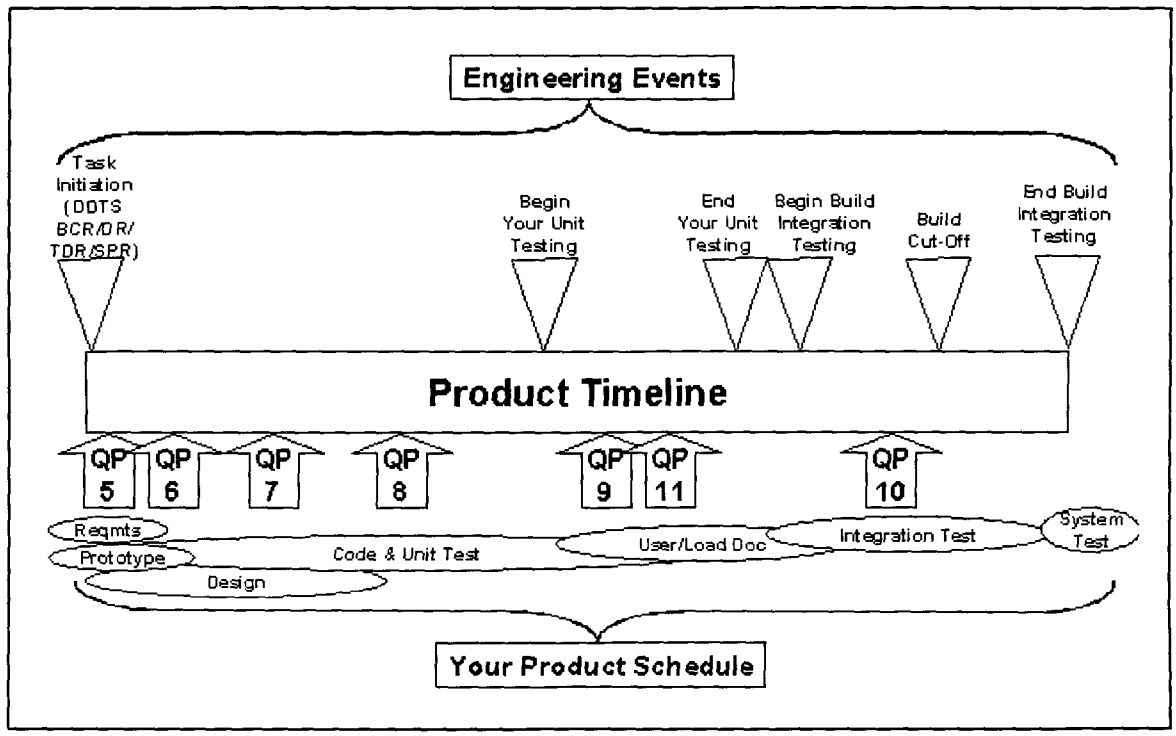


Figure 3-13. Product Timeline

Managing the external system interfaces on TBMCS is very difficult. The government and LM systems engineering team has an Interface Control Working Group [11], jointly chaired by the government and contractor, to deal with this issue. Figure 3-14 illustrates the process. The intent is for TBMCS to manage the interface with its counterpart systems by controlling two documents. The first document is the Interface Control Drawing (ICD), which defines the functional and technical requirements of the interface. The second document is a Memorandum of Agreement that describes the program management and configuration control of the interface.

In practice, program synchronization is exceedingly hard to achieve. Often a program's schedule slips; therefore, any system that releases an update to the interface must be backward compatible. In some cases, when the systems affected by a change must all update at the same time, a vertical release is required. Synchronizing a vertical turnover can be very complicated, especially for complex interfaces such as Link 16.

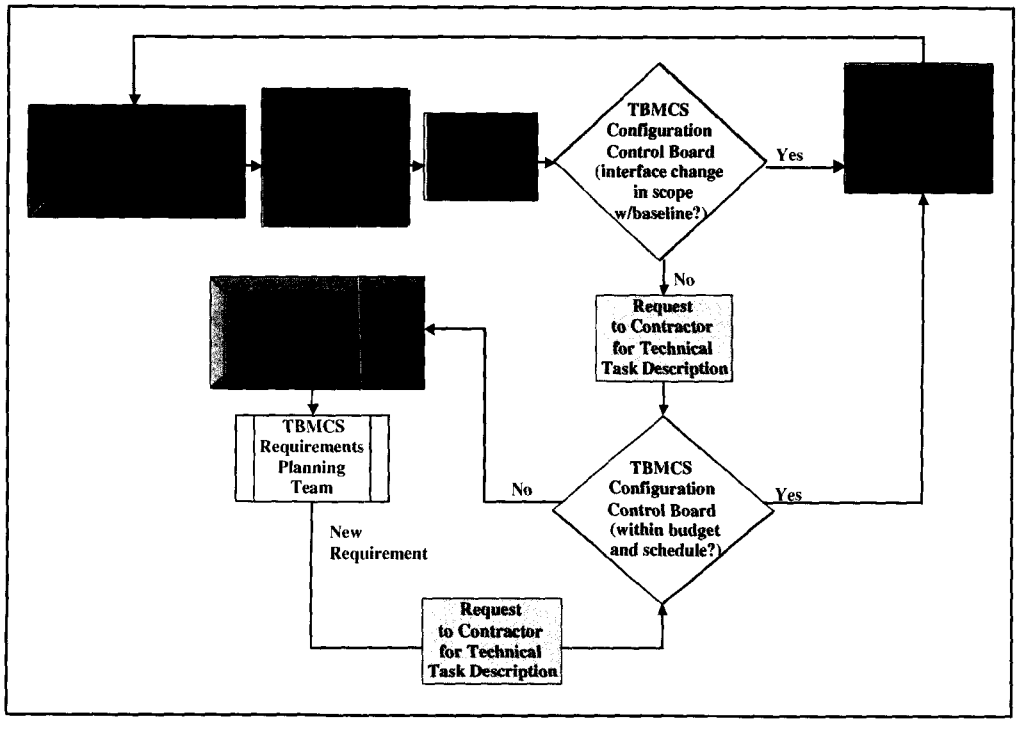


Figure 3-14. Change Process for Current and Future TBMCS External Interfaces

The most extensive integration in TBMCS involves data interoperability. TBMCS has two main databases: the Air Operations Data Base (AODB) and the Intelligence Server Data System (ISDS). The AODB is built according to the USMTF standard, which not only governs text format but also defines the rules for the ATO and ACO. All the military services adhere to this standard, which is typically updated on an even-year basis for planning purposes. For example, TBMCS V1.0.1 was released with USMTF'00. The subsequent release provided USMTF'02, which had a richer definition for the ACO and provided more geometric shapes for restricted airspace zones (also known as Airspace Space Control Measures). The JCS C4 Directorate (JCS/J6) manages the standard and holds interoperability working groups to control and update it. For the most part, the USMTF standards are backward compatible and interoperability in the operational theater is very good.

The ISDS contains the Military Intelligence Database (MIDB) maintained by the Defense Intelligence Agency (DIA). The MIDB is the national database for worldwide targets of opportunity. DIA updates it every six months. This creates difficulties for TBMCS. Typically a TBMCS release takes place every 18 months; hence, its ISDS server could be three releases behind. This can result in interoperability problems, some of which require emergency patches to fix incompatibilities.

As noted, TBMCS is migrating from a client/server object request broker architecture to a Web-based, N-tiered information services architecture. The integration strategies for these two types of systems are very different. In the client/server architecture, the software applications are tightly coupled to a common software infrastructure and brute force is often the only method for integration. By contrast, the Web approach is loosely coupled and supports open standards that facilitate options for third-party integration. As the publish/subscribe method gains popularity,

the number of point-to-point interfaces will decrease; thus, the burden of interface management should lessen as well.

3.5 Learning Principle 5 – Validation and Verification

> The lack of a firm requirements baseline made validation and verification very difficult. The program was schedule driven and often ran parallel test processes with out clear measures of success. Not being able to replicate the operational environment prior to acceptance test created severe problems.

The contractor and the government share responsibility for TBMCS verification and validation. Figure 3-15 depicts the process and relationships [4].

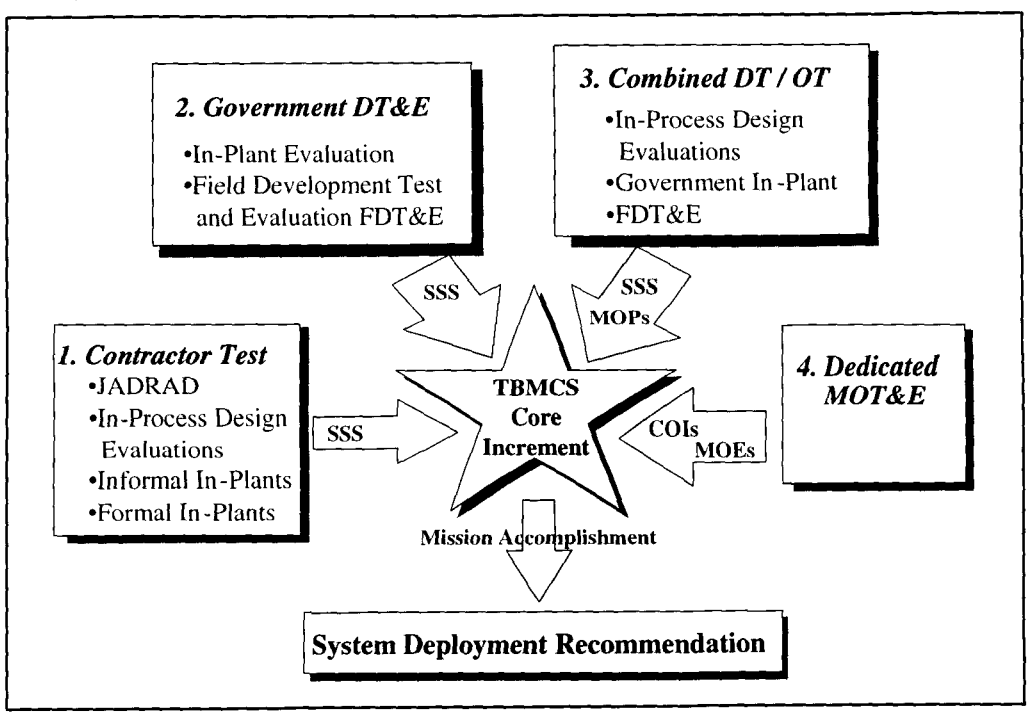


Figure 3-15. TBMCS Test Relationships

The test methodology supports a building block approach for the integration of a family of systems, culminating with an operational test assessing critical operational issues (COI). The test process follows four major steps or events. For the most part the processes are serial, except when external influences compress the schedule and force parallelism in the process, which typically results in unacceptable performance. The first major test process starts with the contractor verifying and validating the system requirements defined in the SSS. In the case of TBMCS, LM carries out a complete functional decomposition and generates a test case for every system requirement. This quintessential building block approach progresses from unit-level testing through subsystem testing, which leads to system integration testing that includes third-party products, and culminates in system-level testing. LM tests performance and external interfaces along the way as appropriate, but these are usually evaluated as part of the system-level test. Development IPTs carry out the relevant unit-level, subsystem, and integration tests; an independent test organization within LM performs the system-level testing. Initially, the test

organization reports to the version development manager, but after two operational test failures, the test organization reports directly to LM's program manager. After the contractor has completed the system and interface testing (to the extent possible), the government and contractor hold a Government In-Plant test to evaluate performance and assess system maturity prior to the formal Development Test and Evaluation (DT&E) run by the government.

In the sprit of acquisition reform, the Air Force exercised minimal oversight of the contractor's test procedures: representatives of the SPO were invited to watch the tests, but did not comment on or approve the test cases and procedures. However, the contractor's and the government's test cases did not always align, and sometimes the government test community and the contractor exercised the system differently. As it happened, testing in the contractor facility had serious shortcomings, of which the most important was that the tests did not accurately represent the operational environment. Hardware limitations and restricted access to external wide area communications and live interfaces meant that the contractor could never replicate how the system would operate in the field. These limitations had a direct impact on performance testing and the ability to assess the system's overall operational effectiveness and suitability.

Early in the program, the live interfaces and performance were not tested prior to formal government test events. This has since changed, and the contractor and the government now perform a field test prior to the formal government test.

The second step in the test process was the government-run DT&E. A Combined Test Force (CTF), with representatives from each of the services and the contractor, performed shared testing and provided independent reporting for the different services. For developmental testing, the Air Force test agency was the 46th Test Squadron; for the Navy it was Space and Naval Warfare Systems Command; for the Army it was the Army Test and Evaluation Command (ATEC); and for the Marines it was the Marine Corps Systems Command.

TBMCS development testing starts in plant and migrates to field test over the span of six months. The government runs the test, drawing on contractor support for discrepancy analysis and trouble shooting. The test is based on the contractor's system test cases, but has a stronger operational flavor and a more realistic operational environment.

Initially, the contractor's system test and government's development test for TBMCS (without a field test) were carried out in parallel and led to an operational test, which the system failed. The process has since changed. The tests are now run serially and build upon each other, culminating in a developmental field test. The third and fourth steps constitute the operational test.

TBMCS used the same CTF concept as in developmental testing, but this time the service operational test agencies led the test event. The Air Force's test agency was the Air Force Operational Test and Evaluation Center (AFOTEC), the Navy's was the Operational Test and Evaluation Force, the Army's was ATEC, and the Marines' was the Marine Corps Operational Test and Evaluation Activity. Depending on the level of change and associated risk, the test event was either a combined development test/operational test (DT/OT) or an MOT&E. The original plan was to test the core over three evolutionary releases, each release increasing in capability, with the third release undergoing the Title 10 USC operational test. When the system was five years late and had failed the first operational test, the approach changed to test V1.0.1 as the core system baseline.

There was tremendous political pressure to make TBMCS the system of record (SOR) for the year 2000. At the time of the OT&E, TBMCS was three years late in delivery and the Air Force wanted desperately to deploy the system in the field and retire CTAPS. Therefore, the fundamentals for planning systems engineering tests were completely compromised for the first operational test, called a Joint Functional Acceptance Test. To say the least, the events leading up to operational test were not based on sound systems engineering principles.

- The tests did not adhere to the entrance and exit criteria.
- The test events did not build on each other.
- The baseline continually changed with software modifications.
- The live interfaces were not tested.
- The contractor had never tested the system configuration, especially the communication infrastructure and connectivity, in an operational context.
- The contractor had tested the system at the QRP level, but the operational test was at the TRP level; as a result, no one could guarantee the system performance.
- The testing events ran in parallel, meaning that LM was running its 600-plus system test cases while the government was carrying out its DT&E.
- User expectations varied from service to service.

Needless to say, the system never had a chance to pass.

After TBMCS failed the operational test, the SPD used a risk management tool called a red team to investigate the issues, derive lessons learned, and recommend a way ahead for the next operational test. The red team was composed of senior military officers, DoD contractors, and representatives of FFRDCs. The team made six specific recommendations:

1. Lock down the baseline and gain consensus on the capabilities.
2. Provide more rigor and discipline in the systems engineering process.
3. Re-baseline the schedule and make the test events serial, with entrance and exit criteria.
4. Allow the contractor to test the system in an operational field setting.
5. Formalize the operational test process.
6. Manage the risk at the officer (O-6 and above) level.

All six recommendations were accepted.

To manage the risk leading up to the next test event, the SPO adopted AF-MAN 63-119, *Preparation for Operational Test* [12], as a template for preparing the system for operational test. Under this new structure, AFOTEC could not start the system test until the PEO certified the system was ready. In turn, the PEO would not certify the system for operational test until the SPO lead engineer certified the system was technically ready for OT&E.

The basic tenet was that TBMCS had to be able to perform the minimal legacy Air Operations Functions carried out in AOCs. No new functionality would be tested. Figure 3-16

breaks the COIs down into MOEs and MOPs. These clear definitions created a path that the contractor and test community could follow.

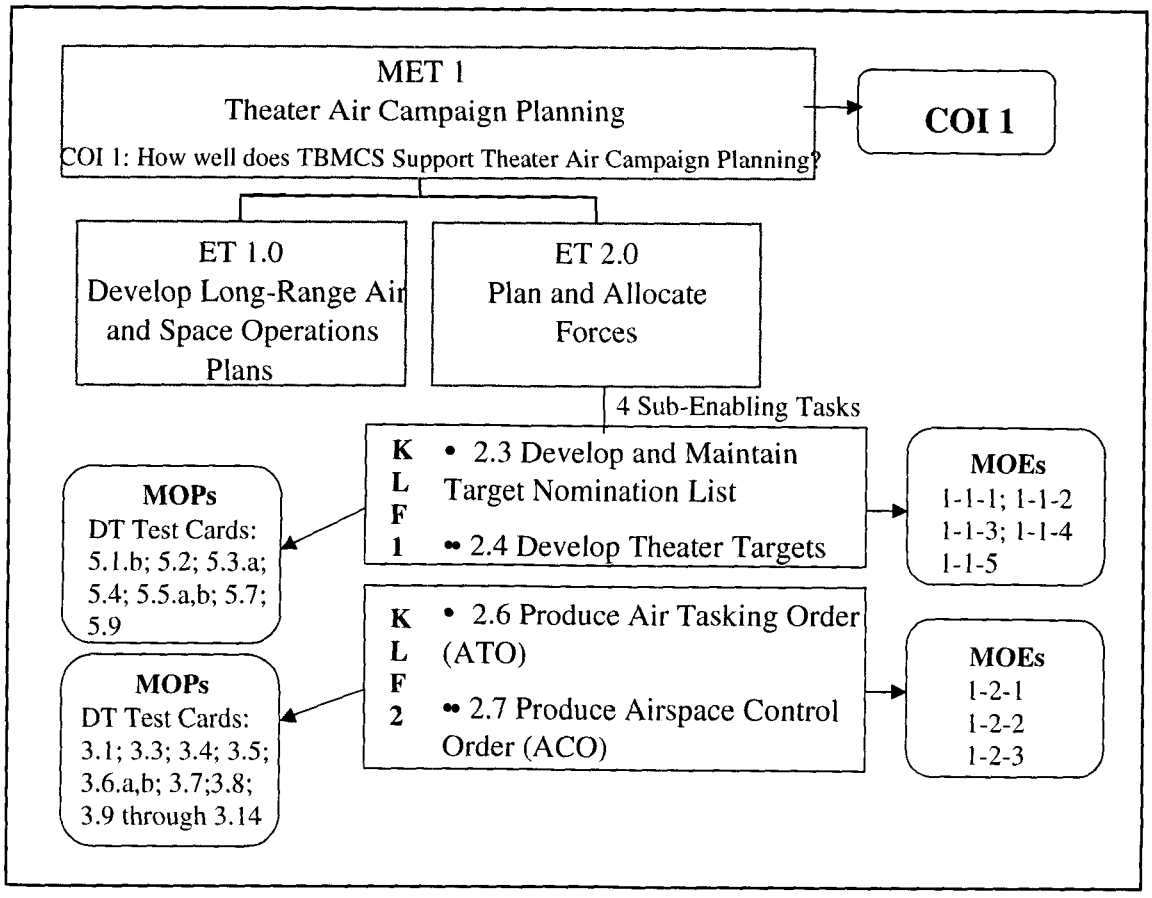


Figure 3-16. Mission-Essential Task (MET) Decomposition

Once the functionality was agreed upon, which was no easy task, the system baseline was locked down prior to the start of in-plant development test. Any changes had to be validated by a formal configuration control board chaired by the SPD and lead engineer. Most changes at this time remedied software deficiencies. The schedule was re-baselined and the test events were serialized to build on each other, with specific entrance and exit criteria. Additional test events were incorporated in the schedule to allow the contractor to test the system in an operational field setting.

The contractor tests and DT&E, including a field test, were all successful. All stakeholders were elated that the system was truly ready for operational test – at least that was what they thought. AFOTEC suspended the operational test because of a contention problem with the intelligence database. The intelligence cell could not produce the target list in the prescribed time because the database locked up when the data was accessed from different applications. AFOTEC delayed the test to let the contractor isolate and correct the problem.

The problem resulted from a design flaw that was not discovered until OT&E because that test represented the first time the system was exercised under real-world conditions, meaning that parallel activities needed access to the database. Moreover, the intelligence product suite was GFE from the GCCS, used primarily by the Navy, and had not been tested at the scale the

Air Force was using: the Air Force had 30 target analysts, while the Navy only had 8 or 9. The DT&E field test also had not stressed the system in a way that mirrored real-world conditions, because the processes were exercised serially, not in parallel. Therefore, a major lesson learned in TBMCS was the need to exercise the system in a true battle rhythm, rather than using test cards and canned scenarios as was done in the contractor's facility.

After the first OT&E, the SPO provided tighter oversight and established a strict risk management policy. In addition, the SPO developed a CONEMP and helped the contractor redesign its test cases to ensure the performance testing reflected the level at which the system would be used for both the AODB and the ISDS. The SPO also created another test event called a combined DT/OT. The DT portion was similar to previous DT events, but this time an operational test was added that would be conducted at the same scale as the Title 10 test. In addition, the contractor was allowed to participate in the test. The timeline and test events leading up the operational test are depicted in Table 3-1 [8].

Because the users in most cases did not know the basics of C2 for air operations, the training basically involved "buttonology": press *this* button to get *this* result. Typically it took two to three days for the cell chiefs to get their cells operating at battle rhythm precision. Also, the TTPs and CONOPS varied depending which numbered Air Force was at the test, so obtaining repeatable results was difficult at best.

Finally, TBMCS passed its operational test and was declared the system of record for the JFACC to plan, execute, and manage air operations in a theater of war. Thus, another valuable lesson learned was the importance of having the same operators at both tests and giving them system spin-up time.

Table 3-1. MOT&E Timeline

EVENTS	DATES
Ver 1, In-plant Design Evaluation (IDE) Build 3	Dec 97
Ver 1, IDE Build 5	May-Jun 98
Ver 1, Build 6.3 In-Plant Testing	Nov 98
Ver 1, FDT&E Phase 1 (Build 7.0)	Dec 98
Ver 1, FDT&E Phase 2 (Build 7.1)	Jan-Feb 99
Ver 1, OT/JFAT(Build 7.2)	Feb-Mar 99
Ver 1.0.1 Build 2.1.1 In-Plant Testing	Jul 99
Ver 1.0.1 Build 2.1.1 Infrastructure Test	Jul-Aug 99
Ver 1.0.1 Build 2.1.2 Combined DT/OT	Sep-Oct 99
Ver 1.0.1 Build 2.1.2 DT/OT	13 Sep – 17 Oct 99
Certification of System Readiness for MOT&E	21 Dec 99
Ver 1.0.1 MOT&E System Admin Training	8 – 17 Nov 99
Ver 1.0.1 MOT&E Infrastructure Setup	29 Nov – 12 Dec 99
Ver 1.0.1 MOT&E Operator Training	15 - 20 Jan 00
Final Test Readiness Review for MOT&E	21 Dec 99
Test Team Training	21 Jan 00
Ver 1.0.1 MOT&E Ver 1.0.1 Build 2.1.3 Execution	22 -31 Jan 00
Deficiency Review Board	15 Feb 00
Ver 1.0.1 MOT&E ISR	16 Feb 00
Government In-Plant Test	16 – 21 Apr 00
Government In-Plant Regression Test	16 – 20 May 00
Ver 1.0.1 Build 2.1.3.5 System Build and Configure	10 – 26 May 00
User Validation	31 May – 2 Jun 00
Future Events:	
Field Development Test	3 – 10 Jun 00
System Reconfigure (USN and USMC)	28 Jun – 12 Jul 00
Final System Checkout	13 – 16 Jul 00
MOT&E Dry Run	16 – 22 Jul 00
Go/No-Go Decision	24 Jul 00
MOT&E Resumption	25 – 31 Jul 00
Deficiency Review Board	15 – 18 Aug 00
Ver 1.0.1 MOT&E ISR	22 Aug 00
Ver 1.0.1 MOT&E Final Report	16 Nov 00

4.0 SUMMARY

The lessons learned from TBMCS can be directly applied to other software-intensive programs that require the integration of vast numbers of third-party products with GFE, such as hardware and communications. A key lesson is that there is no substitute for a well-defined systems engineering process. In the case of TBMCS, external influences drove a relaxation of discipline and rigor on the systems engineering process. In fact, the need for rigor and discipline in the process is even greater when the program lacks sufficient detail in the requirements, architecture, and system design, or when the contractor and government underestimate the complexity of software reuse and third-party integration, as demonstrated in V1.0.1. Giving the contractor TSPR when over 90% of the program content is GFE is a flawed strategy. The contractor cannot be held accountable for performance if the contractor does not control all of the system components that affect performance. Perhaps having performance defined as goals instead of requirements was the only possible approach in this particular case, but it should certainly not be adopted by other programs as a standard.

The original approach of evolving the system's baseline over three software releases never was implemented. A key lesson was that an evolutionary approach will only work once a baseline is established. The pressure from the user community to first fix the legacy system (CTAPS) greatly impaired the program, resulting in a three-year schedule slip and budget overruns totaling tens of millions of dollars. From the start, the program was always trying to catch up to the original plan. Then, continued pressure from the user community to field V1.0.1 basically forced relaxation of test entrance and exit criteria, resulting in a failed operational test. Testing a complicated system takes time and the process needs to be serial, with well-defined entrance and exit criteria.

The lessons learned from the difficulty in fielding V1.0.1 had a very positive impact on the program's current systems engineering environment. TBMCS systems engineering processes have evolved to become mature and repeatable. As the TBMCS program developed, roles and responsibilities shifted between LM and the government. They became predominantly shared functions after the core baseline, V1.0.1, passed operational test and was approved for system fielding in October 2000. If portrayed in the Friedman-Sage context, all nine F-S processes would be seen as shared between the government and contractor. The degree of responsibility varies for each process, but the overall process is orchestrated as a team approach. The operational capability of TBMCS in Operations Enduring Freedom and Iraqi Freedom demonstrates the success of the current approach, as does the contractor's ability to field four subsequent releases in the short span of three years since the release of V1.0.1. The key lessons learned for the systems engineering processes requirements, architecture/design, integration, and verification/validation are described below.

Lessons Learned: Requirements

Some of the lessons learned from the development of V1.0.1 are:

- The government cannot expect the contractor to control the system and functional requirements baseline, especially when the government will perform DT and OT testing.
- The user must specify the operational requirements and concept of operations.
- The government must control the system and functional baseline.

- The contractor must show complete traceability of all system elements to the allocated baseline.
- Spiral development does not obviate the need for a rigorous and disciplined requirements process.

Since August 2000, when V1.0.1 passed its operational test, the requirements process and structure have changed in several ways:

- LM has a separate architecture/SEIPT that maintains the system-of-systems view and reports to the program manager.
- There is a new requirement IPT representing the joint users, SPO, and contractor engineers.
- TBMCS now has an ORD and a CONOPS.
- The government owns and controls the requirements.
- Test is factored in as part of the planning process.
- The SPO and the contractor manage each upgrade jointly.

Lessons Learned: System Architecture and Design

The V1.0.1 system architecture and design were really dictated by the operational users. For example, the direction to use the DII COE and the GCCS requirement came from the Navy. Thus, on the one hand, the government gave the contractor free rein; on the other, it dictated to the contractor what to do and to use. The decision to leverage legacy applications with modern information technologies created a dichotomy: some of the mandated products did not directly scale for Air Force operations, others proved incapable of operating over the austere communication channels used by the Marines and the Navy.

Software reuse was not as straightforward as originally thought. The Air Force requirements varied from those of the other services and had direct impact on the overall design, especially as it related to the DII COE. For example, analysis showed that the current GCCS message processor was not robust enough to handle the message load for an AOC. After considerable and time-consuming attempts to improve the product, LM eventually replaced it with a commercial product called IRIS.

Another major difference was the profile manager. The Air Force assigned workstations by type (e.g., planner 01) and not by username. This created a tremendous ripple because permissions, applications, and alerts were all driven by user profile. Thus, LM had to write a profile manager to meet the Air Force requirements.

Enough changes were made to the TBMCS software infrastructure to warrant a separate baseline – a variant of the DII COE baseline. The plan to use common products as the system infrastructure was flawed and very restrictive, because the COTS upgrade cycle was always at least two versions ahead of the TBMCS baseline. The application baseline was also affected. A particular application requested by the user might be very difficult to integrate into the system because it was either not segmented (as required by the DII COE) or its COTS infrastructure was more current than that of TBMCS. This led to extensive overruns in integration cost and schedule. ***A major lesson learned, therefore, is to use open standards and not to specify***

particular commercial products as the software infrastructure. One size does not fit all systems!

It is also essential to understand the maturity of the third-party products specified in the system design. LM's processes were not quite mature enough to flag immature third-party products. Unfortunately, proof-of-concept demonstrations and user-developed applications did not always transition into production-quality products. LM did use an SDK, but for V1.0.1 it was maturing at the same time as the design was evolving. In some cases, the government would direct LM to form an associate contractor agreement, and in especially high-risk cases LM would make the developer a subcontractor. ***The lesson learned is to build in an assessment process that allows the integrator either to build the software application or replace a required product with another.*** Often the process and schedule did not permit such an assessment.

The design had the excellent feature of using layers and thus isolating the applications from the data. This facilitated integration of applications, but for V1.0.1 the interface layer was immature and difficult to follow. Therefore, LM had to devote considerable time to debugging the infrastructure and the application services layer. As the system matured and interfaces settled down, the design proved very valuable for migration to the N-tiered Web-based architecture. Applications could be replaced with minimal impact on the other applications and databases. ***The lesson learned is to define the public interfaces up front and make them available to the third-party developers.***

Initially, TBMCS did not have a vision that the program could follow. LM did include a top-level vision in its proposal and assumed that the work on TBMCS would proceed according to that vision after contract award. However, the government was more focused on the tactical level than the strategic level. Months after contract award, the government instead directed LM to fix and field the legacy system CTAPS. Three years and 2000 software bug fixes later, CTAPS was fielded. TBMCS never recovered, 70% of the resources had been consumed, and, hence, the architecture effort was not funded. The remaining resources were spent on getting TBMCS V1.0.1 tested and fielded as soon as possible.

In 1999, three years after contract award, the government finally agreed that TBMCS needed a vision and a roadmap to achieve the vision. Jointly, the government and LM built a "to be" architecture and defined a roadmap to support Joint Vision 2010. The architecture provided the framework to guide the evolution of TBMCS from the V1.0.1 baseline to its current state. LM's chief architect now ensures that the proposed design is consistent with the defined architecture, which serves as a communications tool and is integral to the planning process for subsequent releases. ***A lesson learned from V1.0.1 was to define a comprehensible architecture and to characterize the constraints and configurations of the system.*** Migrating from the client/server architecture to the N-tiered architecture has truly given TBMCS new life.

Changes made after V1.0.1 were:

- Creation of a "to-be" architecture and roadmap for network centric operations,
- Creation of a shared architecture/systems engineering IPT,
- Migration from a restrictive common commercial product infrastructure to an open system standards approach,
- Adoption of the publish/subscribe approach to simplify the complexity of the external interfaces, and

- Publication of a mature SDK for third-party development based on open commercial standards.

Lessons Learned: System Integration and Test

The current integration and test processes for TBMCS are entirely shared and well integrated between the government and contractor. The operational test process for TBMCS is very expensive and involves many moving parts. Scheduling a test requires an enormous amount of preparation and planning by the government and the contractor. The cost to run an operational test was $5 million. Getting to the test and not passing is not a good practice to adhere to. Unfortunately TBMCS failed twice (the second time the test was officially "suspended," but essentially the entire test had to be re-run). There are several lessons learned.

- The test community never really bought into acquisition reform.
- ***The SPO must take ownership in managing the risk for DT and OT.*** It does not have to run the tests, but clearly has to orchestrate them. Asking the contractor to perform that role was a mistake.
- ***System engineering must play a major role in planning the tests and managing technical risks.***
- ***There is no substitute for a well-defined requirements baseline.*** Managing user expectations on TBMCS capabilities was a nightmare. Getting everyone to agree on the pass/fail performance criteria was a Herculean effort.

Another major lesson learned is that ***testing is a building block process***. The processes must be run in a serial mode with well-understood entrance and exit criteria. The test planning process for V1.0 was completely overruled because of schedule considerations. The contractor was performing integration tests while the government was running development tests. Also, the contractor could not guarantee any kind of success because the system was being tested in a completely unfamiliar environment. Thus, having the contractor test the system in an operational setting is essential. In the first test for V1.0, the contractor was not able to test the integration of the GFE communications infrastructure with the system, which meant that the testers could not isolate issues to the system or the infrastructure. Subsequent tests required that the contractor be allowed to integrate the system with the communications infrastructure prior to the start of the test.

Another major lesson learned is to ***ensure that external interfaces have been fully tested in a real-world environment at both the functional and technical levels***. The contractor did not have the capability to conduct a live test of the interfaces in-plant. As a prerequisite prior to the start of any operational test, each interface was tested with known inputs and outputs to ensure the interface was working properly, but simulation was not always a good indicator of performance.

The final lesson learned is that ***system developers must understand how the system will be employed***. A detailed CONOPS and a corresponding concept of system employment are essential. As described in previous sections, the main processes in building and managing air operations overlap; therefore, not testing those processes prior to an operational test was a major mistake. This is where an operational architecture with use cases would have been especially beneficial. The contractor had a good understanding of the processes internal to an operational cell (e.g., planning), but did not understand the dependencies and interactions among the cells

and the implications for the system. Unfortunately, this was not discovered until the second operational test. The primary contributors to this problem were the absence of a CONOPS, the lack of previous testing in a battle rhythm, and the unprecedented number of users, which far exceeded any encountered in DT or system test.

TBMCS has made several improvements to the test planning since V1.0.1, and the program office continues to take a proactive role in managing risk. The processes are now serial. Entrance and exit criteria for each are well understood. Stress testing is done in DT and is representative of the real operational load, to include interaction among cells. Interfaces are tested live as a prerequisite. Finally, field test is part of the contractor and DT testing prior to operational test.

5.0 REFERENCES

1. Air Force Institute of Technology, "System Engineering Concepts: Illustration Through Case Studies," Wright-Patterson AFB, OH: Air Force Institute of Technology, 19 January 2003, [On-line]. URL: http://cse.afit.edu/Friedman-Sage%20Framework.doc
2. Defense Acquisition University, *Systems Engineering Fundamentals*, Fort Belvoir, VA: Defense Acquisition University Press, January 2001.
3. Contingency Theater Air Control System (TACS) Automated Planning System (CTAPS) ORD, TAF 305-88 (8 Feb 95); Wing Command and Control System (WCCS) ORD, CAF-AFSOC-TAF-340-88 (22 Jun 95); Combat Intelligence System (CIS) ORD, CAF 306-93-I-A (23 Jan 95);
4. Lockheed Martin Integrated Systems and Solutions, *Software Developer's Kit for Theater Battle Management Core Systems (TBMCS) Spiral 1.1.3*, Revision 6, Contract No. F19628-95-C-0143, Colorado Springs, CO, 15 September 2003.
5. Air Force Electronic Systems Center, *TBMCS Technical Reference Document*, Contract # F19628-R0027-94, Hanscom AFB, MA, 3 November 1994.
6. Lockheed Martin Integrated Systems and Solutions, TBMCS Overview Briefing, 19 March 2004.
7. Joint Publication 3-30, *Command and Control for Joint Air Operations*, Office of the Joint Chiefs of Staff, Washington, DC, 5 June 2003.
8. Air Force Operational Test and Evaluation Center, *Theater Battle Management Core System (TBMCS) Version 1.0.1 Multiservice Operational Test and Evaluation (MOT&E) Test Plan*, Kirtland AFB, NM, 17 July 2000.
9. The MITRE Corporation, *Functional Description (FD) for Theater Battle Management Core Systems (TBMCS), Including CTAPS and WCCS*, Bedford, MA, 16 May 1994.
10. Defense Information Systems Agency, *Defense Information Infrastructure (DII) Common Operating Environment (COE) Segment Developer's Guide (SDG)*, Version 2, 30 August 2001.
11. Air Force Electronic Systems Center, ESC/ACF Program Office, *ICWG Charter*, November 2003.
12. U.S. Air Force, Air Force Manual 63-119, *Certification of System Readiness for Dedicated Operational Test and Evaluation*, 22 February 1995, [On-line]. URL: http://www.e-publishing.af.mil/pubfiles/af/63/afman63-119/afman63-119.pdf
13. U.S. Department of Defense, *Military Standard 2167A, Defense System Software Development*, Washington, DC, 29 February 1988.
14. Secretary of the Air Force, Air Force Instruction 63-123, *Evolutionary Acquisition for C2 Systems*, Washington, DC, 1 April 2000.
15. Headquarters Air Combat Command, Air Combat Command Instruction (ACCI) 13-150, *Air Operations Center*, Langley AFB, VA, 7 March 1995.

6.0 LIST OF APPENDICES

Appendix 1 - Completed Friedman Sage Matrix for TBMCS

Appendix 2 - Author Biography

Appendix 3 - Acronyms

Appendix 4 - Background and History of TBMCS

Appendix 5 - Risk Assessment and Management

Appendix 6 - System and Program Management

Appendix 1

Completed Friedman Sage Matrix for TBMCS

Table A1-1. The Friedman-Sage Matrix for TBMCS

Concept Domain	Responsibility Domain		
	1. SE Contractor Responsibility	2. Shared Responsibility	3. Government Responsibility
A. Requirements Definition and Management	Initially, LM responsible for generating the System Segment Specification based on legacy ORDs and TRD. The technical performance measurements were goals and not requirements. ,	Currently, LM and government have an SE IPT to define and manage requirements for current and future releases. User has an on-line database of operational requirements for all stakeholders to access.	Did not develop a system specification. Defined a TRD that specifically directed a concept and technology focus. No firm requirements baseline, lack of detailed CONOPS and CONEMP.
B. Systems Architecting and Conceptual Design	System architecture defined at too high a level, impacting development and integration. Layering approach very good. No "to be" architecture and roadmap.	Jointly developed "to be" architecture to support Network Centric Operations. Evolved from a C/S and mandated products to an N-tiered open system standards architecture.	Mandated use of certain 3rd party commercial hardware and communication infrastructure products. Mandated products immature. Reuse concept good in principle, but difficult to execute.
C. System and Subsystem Detailed Design and Implementation	Heavy influence of legacy systems, Underestimated complexity and maturity of 3rd party and COTS products, Negative impact of no CONEMP on system design. Able to evolve baseline to Web construct.	Jointly developed Web-based system design: more open and flexible, facilitates Cots upgrades and 3rd party integration. Good user acceptance and better understanding of risks and product maturity.	Drove initial design and immature 3rd party products, redirected design to accommodate different GFE software and hardware products, e.g., GCCS-13 and Navy hardware.
D. Systems and Interface Integration	System architecture and design had negative impact on integration. Internal interfaces not well documented, some managed by sub-contractors, could not replicate an operations environment in-plant.	LM and government hold joint ICWGs; LM able to integrate at government facilities prior to test; LM can simulate or test most interfaces in-plant; LM and government have much tighter control on 3rd party and risk mitigation plans.	GFE not well controlled, big impact on integration, most interfaces have ICDs and MOAs, communications infrastructure not consistent by service and by base.
E. Validation and Verification	LM responsible for system-level test, test cases, and procedure flow from SSS; system test at functional test executed in serial, not parallel as in operational test; test environment was problematic.	Combined test force with representation from LM and government; integrated test plan, each test building off the previous; able to determine level of testing based on risk assessment; LM system test at government facilities.	Responsible for DT and OT; combined testing but independent reporting for each service. Initially no ORD or detailed CONOPS; impact on test expectations.
F. Deployment and Post Deployment (post launch)	Was not able to reproduce an operational environment for testing and debugging purposes.	Government provides operational facilities to support contractor integration and test.	Operational test environment did not correspond exactly to a real-word environment. Level of operational realism determined by level of change for new system baseline under test.

Table A1-1. The Friedman-Sage Matrix for TBMCS

Concept Domain	Responsibility Domain		
	1. SE Contractor Responsibility	**2. Shared Responsibility**	**3. Government Responsibility**
G. Life Cycle Support	Contractor is responsible for software licensing, manages a two-tiered help desk, and provides mobile training teams.	Mobile training teams, system administration.	Schoolhouse for initial training. Government provides all hardware and communication infrastructure.
H. Risk Assessment and Management	Technical risk managed at the lower levels via tech reviews, no formal process at the PM level; GFE and 3rd party high-risk items difficult to manage.	Have a formal joint management process between LM and the government, manage risks at the program level and disposition on a monthly basis.	Acquisition reform, initially minimal oversight, give LM TSPR, increased oversight and managed risk at the program level for operational test.
I. System and Program Management	Corporate requires a system engineering management plan for organization process reviews and products.	Run an IPT; contractor evaluated on system engineering as part of award fee; government and contractor have joint management plan.	Supports the initial requirements process, architecture, design, development, test, and risk management.

Appendix 2

Author Biography

JOSIAH R. COLLENS, JR.

Josiah R. (Josh) Collens is currently the Director of Engineering for C4ISR Enterprise Integration at The MITRE Corporation. He has worked at MITRE for over 15 years, serving as a systems engineer on assignments that included the Modular Control Equipment (MCE) system for tactical command and control, Primary Simulation Trainer (PST) for Weapons Controllers, Peace Shield (an air defense system for the Royal Saudi Air Force), and the Integrated Maintenance System (IMDS) for unit-level maintenance. He was also the Air Force Weather re-engineering architect and the lead engineer for the Theater Battle Management Core System (TBMCS), which provides theater operational air and space planning and management.

Josh received a Bachelor of Arts degree in mathematics from The Citadel, The Military College of South Carolina, in 1982. He received an academic scholarship to the Air Force and was commissioned as an Air Force officer in 1982. As an active-duty officer he was a computer programmer/system analyst for the Joint Surveillance System, which provided air defense for the North American continent. He separated from the Air Force in 1986 after fulfilling his academic scholarship obligation. He then earned his Master of Science in Computer Information Systems from Boston University, Metropolitan College, in 1992.

Appendix 3

Acronyms

ABP	Air Battle Plan
ACC	Air Combat Command
ACO	Airspace Control Order
AFC2ISRC	Air Force Command, Control, Intelligence, Surveillance, and Reconnaissance Center
AFI	Air Force Instruction
AFMC	Air Force Materiel Command
AFOTEC	Air Force Operational Test and Evaluation Center
AOC	Air Operations Center
AODB	Air Operations Data Base
API	Application Program Interface
ASOC	Air Support Operations Center
AT&L	Acquisition, Technology, and Logistics
ATEC	Army Test and Evaluation Command
ATO	air tasking order
ATO	Air Tasking Order
C2	command and control
C4ISR	command, control, communications, computers, intelligence, surveillance, and reconnaissance
CAOC	Combined Air Operations Center
CIS	Combat Intelligence System
CM	configuration management
CMM	Capabilities Maturity Model
CONEMP	concept of employment
CONOPS	concept of operations
CORBA	Common Object Request Broker
COTS	commercial off-the-shelf
CTAPS	Contingency Theater Automated Planning System
CTF	Combined Test Force
DAA	Data Access Agents
DIA	Defense Intelligence Agency
DII COE	Defense Information Infrastructure Common Operating Environment
DISA	Defense Information Systems Agency
DoD	Department of Defense
DT	Development Test
DT&E	Development Test and Evaluation
ESC	Electronic Systems Center
FLEX	Force Level Execution
GCCS	Global Command and Control System
GFE	Government-Furnished Equipment
GOTS	government off-the-shelf
ICD	Interface Control Drawing
IDL	Interface Definition Language
IPT	integrated product team

ISDS	Intelligence Server Data System
JCS/J6	Joint Chiefs of Staff, Command, Control, and Communications Directorate
JFACC	Joint Force Air Component Commander
JTF	Joint Task Force
JTT	Joint Targeting Toolbox
KLF	Key Legacy Function
LM	Lockheed Martin
LM-IS&S	Lockheed Martin Integrated Systems and Solutions
MAJCOM	Major Command (Air Force)
MCF	Mission Critical Function
MET	Mission Essential Task
MIDB	Military Intelligence Database
MOE	measure of effectiveness
MOP	measure of performance
MOT&E	Multi-Service Operational Test and Evaluation
MTT	Mobile Training Team
ORD	Operational Requirements Document
OSD	Office of the Secretary of Defense
OT	Operational Test
OT&E	operational test and evaluation
PC	personal computer
PEO	Program Executive Office
PMD	Program Management Directive
POC	point of contact
QP	Quality Point
QRP	Quick Response Package
RFP	request for proposals
RPT	Requirements Planning Team
SAF/AQ	Air Force Chief Acquisition Executive
SAIC	Science Applications International Corporation
SDIPT	Spiral Development Integrated Product Team
SDR	System Design Review
SEIPT	System Engineering Integrated Product Team
SEMP	Systems Engineering Management Plan
SOA	service-oriented architecture
SOR	system of record
SPD	System Program Director
SPO	System Program Office
SPR	Software Problem Report
SRR	System Requirements Review
SSS	System Segment Specification
SVRD	System Version Requirements Document
TAC	Tactical Air Command (now ACC)
TBMCS	Theater Battle Management Core System
TRD	Technical Requirements Document

TRP	Theater Response Package
TSPR	Total System Performance Responsibility
USMTF	United States Message Text Format
VPN	virtual private network
WCCS	Wing Command and Control System
Y2K	Year 2000

Appendix 4

Background and History of TBMCS

Project Genesis/Origin

The genesis of TBMCS dates back to lessons learned from Desert Storm in 1991. Generating and disseminating the daily 3000-sortie air tasking order (ATO) for that conflict was laborious and very time consuming. Moreover, the Air Force could only deliver the ATO to other components, specifically to maritime components on aircraft carriers, by flying hard copy to them via helicopters. The applications used at the time were not battle tested and could not scale to the level of war experienced in Desert Storm. Most of them had been developed via limited research and development efforts initiated by the Major Commands (MAJCOMs) or research laboratories.

After the end of the conflict, DoD formed a Tactical Battle Management General Officer Steering Group, composed of representatives from the operational commands, to improve tactical C2 over the practices used during Desert Storm. The group identified shortcomings and provided a roadmap for theater C2 [15]. On the basis of these findings, Tactical Air Command (TAC) started an in-house development program (using Operations and Maintenance funds) to build a system called Contingency Theater Automated Planning System (CTAPS). The primary function of CTAPS was to construct large ATOs and disseminate them quickly to the other service components and aircraft wings. Specifically, the system was to automate and integrate airspace deconfliction, air battle planning, and ATO generation, and automatically disseminate the ATO and Airspace Control Order (ACO).

Originally, CTAPS was to be the umbrella program, with its key components derived from previous software systems: the Computer Assisted Force Management System and Airspace Deconfliction System. Follow-on components included two applications developed by the Air Force Research Laboratory at Rome, New York: the Advanced Planning System and Force Level Execution (FLEX).

In 1992, when TAC became ACC, the director for requirements recognized that CTAPS was too big and complicated for a using command to manage as an in-house project and recommended that it be transitioned to AFMC. ESC assumed ownership and established a formal program office in 1993 under the direction of the C2 PEO. Because ACC wanted the capability developed and fielded as soon as possible ESC retained Science Applications International Corporation (SAIC), the original developer under TAC, after the transition.

The C2 PEO, John Gilligan, had a much broader vision for CTAPS. He wanted it to be the all-encompassing theater C2 system, and believed that industry should have the opportunity to compete for this system. In mid-1993, he issued a Program Management Directive (PMD) for a system called Theater Battle Management Core Command and Control System. The PMD called for program consolidation and the resources necessary to pay for a standardized, secure, automated C2 decision support system that would be deployed worldwide. The systems to be consolidated were the Wing Command and Control System (WCCS), CTAPS Command and Control Information Processing System – a system built for Air Mobility Command, and the Command Tactical Information System – a system developed by the 11th Air Force.

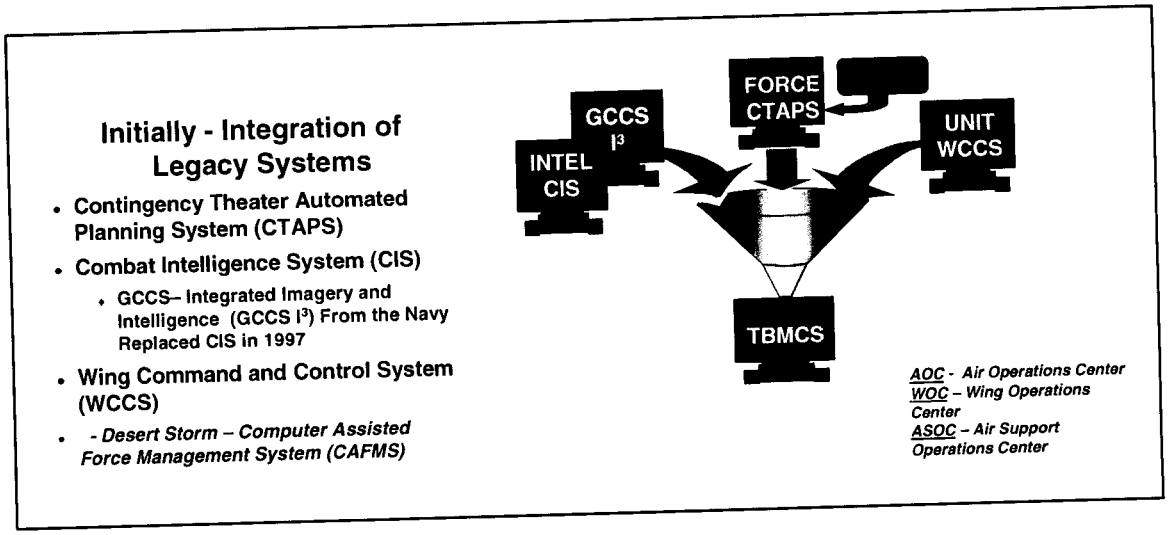


Figure A4-1. Initial Intent of Program

Unfortunately, the consolidation never took place and legacy systems continued their separate evolution, which had a direct impact on the development of TBMCS. In the course of developing the Request for Proposals (RFP) for the system originally envisioned, ESC changed the consolidation and integration strategy. The final acquisition strategy encompassed three systems: CTAPS, WCCS, and an intelligence system called Combat Intelligence System (CIS) (see Figure A4-1 above). In addition, the ability to reuse software applications across a common infrastructure became a key program/design driver. As the program started to crystallize and the acquisition Program Element was established, the name changed again to Theater Battle Management Core System.

Timeline of the TBMCS Program

TBMCS development proved long and arduous. The original acquisition strategy called on the contractor to deliver three increments with increasing capability in the years from 1995 to 2001, culminating in operational test after the third release.

External influences drive the decision process in most large-scale and highly complex acquisition programs, and affect the program schedule for both technical and programmatic reasons. TBMCS was no different: it did not deliver its first version until August 2000. It should be noted, however, that as of 2004 four spirals have been produced.

Pre-proposal Competitive Phase

In the summer of 1993, ACC conducted a user evaluation of CTAPS Version 5.0. The test results received mixed reviews. The ACC users considered the test a qualified success and recommended that the SPO fix the reported system anomalies and field the system as soon as possible. The SPO disagreed, believing the contractor was in trouble, and considered the test a failure. The SPO therefore developed a plan to produce a Version 5.1 within one year that would be more robust and would solve the problems detected in the summer test.

Once ACC formally transitioned CTAPS to ESC in October 1993, the emphasis shifted to quickly developing and deploying the CTAPS capabilities to the field and then folding those capabilities into the TBMCS baseline after TBMCS contract award. In parallel, the SPO developed an acquisition strategy that called for a single contractor to act as the system integrator

and enable TBMCS to subsume CTAPS. That strategy had three key components which had long-lasting effects on the SE processes and development of the program.

First, the SPO decided not to task the contractor with developing a new system, but with integrating disparate legacy capabilities by using open standards with a common user interface. The architecture should allow flexibility for new capabilities to evolve. To support this strategy, The MITRE Corporation[4] developed a technical strategy that used an object-oriented approach to facilitate the "plug and play" integration of legacy software applications and enable them to run on the government-provided DII COE.

Second, the user community did not produce either an overarching CONOPS for how TBMCS would be used in the field or a new Operational Requirements Document (ORD). The plan was to modernize the legacy systems and use the existing CTAPS, WCCS, and CIS ORDs instead. By adopting this strategy, the Air Force afforded TBMCS the opportunity to avoid both the normal DoD requirements generation and review process and the possibility of becoming a joint program, which clearly would have delayed the acquisition.

Given that the SPO did not have an overarching CONOPS and only had a loose collection of legacy ORDs, generating system requirements was difficult at best. The TBMCS SPO director decided not to develop a system specification, but instead generate a Technical Requirements Document (TRD) that provided only a very top-level description of how the system might be employed and formalized the technical strategy for TBMCS. Because of the ambiguous requirements, ESC built flexibility into the contract to allow the contractor and the government to generate requirements by collaborative efforts. This drove the acquisition strategy of enabling the system to evolve by delivering three increments, each with increasing capability. An interesting sidelight, discussed in Section 3, is that the contractor was required to produce a system specification, but the key system performance parameters were not binding; the contract treated them as goals. These decisions had a long-lasting impact on the engineering and testing of the system.

Third, as noted above, Air Force acquisition strategy was undergoing reform. The impetus to reform had come from the operational users, who complained that acquisition programs were taking too long, and that by the time systems were fielded they either did not meet the need, or the threat or the technology had changed or advanced. To meet this challenge, SAF/AQ and the C2 PEO decided to minimize the burden of oversight and allow the contractor greater flexibility in producing the system. Part and parcel of this thinking was to let the contractor work more directly with users and allow the requirements to evolve over time. It should be noted that this approach was not accepted by all of DoD, especially by the operational test and financial management communities. This also had significant oversight implications.

In some respects, TBMCS was clearly ahead of many programs and would not have survived without a strong PEO who was willing and able to confront the DoD scrutiny. TBMCS was the "poster child" for acquisition reform: the SPO would point the contractor in the right direction and get out of the way. The government would provide minimal oversight and use the award fee as the incentive for the contractor to perform in accordance with the contractual technical, cost, and schedule commitments.

---

[4] MITRE operates the DoD C3I Federally Funded Research and Development Center (FFRDC) and serves as ESC's lead systems engineering support organization.

TBMCS Proposal Phase

On 4 November 1994, ESC released the TBMCS RFP for bids. Darleen Druyan, SAF/AQ, was the Source Selection Authority. The primary bidders were Hughes Aircraft, Logicon, Unisys Eagan, Raytheon/SAIC, and Loral Command and Control Systems. The PEO answered all of the bidders' questions and continually asked for their inputs and critiques on the government approach. The source selection was novel because the government not only evaluated the bidders' written proposals, but also conducted numerous in-plant visits to assess their engineering and management capabilities, especially software integration. The bidders were also required to develop and present a two-day "live demonstration" focused on their operational and technical approach to meeting the requirements of the TRD. An additional selection criterion was based on the contractors' past performance and software maturity level, e.g., Software Engineering Institute Capabilities Maturity Model (CMM) rating. A rating of 3 or higher was required because of the legacy reuse strategy, a topic that was explored in Section 3.

The source selection requested three best and final offers before awarding a Cost Plus Award Fee contract in October 1995 to Loral, Colorado Springs, Colorado. The period of performance was six years and the contract had an estimated value of $180 million. The award decision was based on best value to the government: Loral was 25% less costly and ranked second on the technical rating behind Unisys Eagan. The TRD defined the ceiling of the contract as delivering three major versions of the system, but did not preclude the contractor from delivering several incremental or maintenance releases.

Interestingly, during the proposal evaluation period Loral purchased Unisys Eagan, requiring Loral to establish a "firewall" between its two competing business units (Eagan and Colorado Springs). In 1997, Loral was in turn purchased by Lockheed Martin, which subsequently folded its Colorado Springs operation into Lockheed Martin Mission Systems (LMMS), headquartered in Gaithersburg, Maryland. In January 2004, LMMS was dissolved and incorporated in Lockheed Martin Integrated Systems and Solutions (IS&S).

TBMCS Development Phase

Immediately following contract award, the user community sponsored several senior-level meetings to establish TBMCS development priorities. The main priorities were to shorten the ATO development cycle and to integrate operations and intelligence into the ATO process at all levels. TBMCS Version 1.0 was to be fielded 18 months after contract award. Yet, although the acquisition emphasis was supposedly on developing the new integrated TBMCS, the government directed LM to make completing and maintaining the latest legacy CTAPS version its top priority.

Five key influences deferred actual delivery of TBMCS until August 2000. One was the legacy system CTAPS. The operational user was very frustrated with the performance of CTAPS 5.1, wanted improved capability immediately, and did not want to wait for the first version of TBMCS. The SPO, at the direction of the PEO, requested LM to assume responsibility for completing the in-process development of CTAPS Version 5.2 and for fielding and maintaining the system. As a result, the contractor was forced to shift a tremendous portion of the resources planned for TBMCS to completing CTAPS 5.2, which was finally fielded in March 1997. This change in direction cost TBMCS three years in schedule and about 70% of its available resources.

The second major influence was the requirement to integrate TBMCS into the DII COE and align it with Global Command and Control System (GCCS) applications. Both the DII COE and GCCS were very immature, which required LM to hire the Defense Information Systems Agency's (DISA's) contractor as a subcontractor to make necessary fixes to the DII COE.

The third influence was the immaturity of many of the third-party software applications. Integrating these applications demanded extremely high levels of resources. Occasionally, LM had to reduce applications in functionality or replace them with other products to achieve integration and operational capabilities.

The fourth major influence was requirements creep. The SPO never really established a firm baseline until after TBMCS failed its first major operational test. The System Program Director (SPD) described it as TBMCS "trying to solve world hunger."

Last, the program was completely schedule driven. The operational community exerted tremendous pressure for TBMCS to be the Year 2000 (Y2K) SOR rather than the current system, CTAPS. The government therefore forced the contractor to conduct early operational testing, despite knowing the system was not ready. This basically violated the fundamental systems engineering principles of effective test planning, risk assessment, and definition of external system boundaries. The Air Force leadership wanted only an assessment of system maturity, but the joint test community treated this test as a pass/fail Operational Test and Evaluation (OT&E).

Understandably, TBMCS failed its first operational test in March 1999. As a result, the government initiated an effort to reinstitute an upgraded CTAPS as the SOR for Y2K certification. TBMCS was also added to the OSD oversight list. TBMCS was re-baselined, and more government oversight was brought to bear, including mandatory oversight by OSD. The SPO adopted AF MAN 63-119, *Preparation for Operational Test,* processes to assist in certifying that the system was ready for operational test [12]. The SPO and the contractor adopted joint systems engineering processes to help manage the risk, and developed a bottom-up schedule based on the maturity of the system. In addition, the SPO–contractor team removed the parallelism from the schedule and established a serial test process with entrance and exit criteria for each test event.

TBMCS went to its second operational test in January 2000, but the test was suspended due to a contention problem affecting the intelligence database. The problem had not been discovered in prior tests because the system had never been exercised in a true battle rhythm, with the targeting and execution processes operating in parallel. The SPO then applied even more government oversight and established a zero risk tolerance approach. The SPO chief systems engineer assumed responsibility for the technical integrity of the system and for recommending when the PEO should certify the system for operational test. With LM help, the SPO developed performance tests that reflected a realistic operational battle rhythm. These became part of the formal development test process that TBMCS would have to pass before proceeding to Multi-Service Operational Test and Evaluation (MOT&E).

The operational test resumed in July 2000 and TBMCS passed. TBMCS version 1.0.1 received a favorable fielding decision by the JCS/J6 in October 2000 and was designated the SOR.

Shortly thereafter, the Air Force decided that TBMCS should become Web enabled and migrate from a UNIX platform to a personal computer (PC) end-user (client) device. The Air

Force also adopted a new development methodology under which the SPO delivers spirals of capability (which some would say was the original development concept). This process has proven very successful, as TBMCS is now producing its fourth spiral in less than four years. TBMCS also is leading the way in delivering the latest in Web and information services technologies as the system evolves to support network centric warfare.

Operational Use

TBMCS is now deployed worldwide as the J6-mandated joint system that the JFACC uses to plan, manage, and execute the Air Battle Plan (ABP). It has demonstrated rich functionality: it can produce a very complicated integrated ABP for execution by the component commanders. Table A4-1 highlights the success TBMCS achieved during Operation Iraqi Freedom in terms of sorties planned, managed, and flown. The size of the ATOs/ACOs produced in Operation Iraqi Freedom well exceeded the system performance parameters.

Table A4-1. Operation Iraqi Freedom Sortie Count

Total Sorties Flown	41,404
USAF	24,196
USMC	4,948
USN	8,945
USA	269
United Kingdom	2,481
Australia	565

TBMCS initially was not well received because of the UNIX interface (required by the government in the RFP) and the complicated nature of the system. LM has made tremendous progress in simplifying the user interface and reducing system complexity. Figure A4-2 shows where TBMCS is currently deployed to the Combined Air Operations Center (CAOC) at Al-Udeid in Qatar.

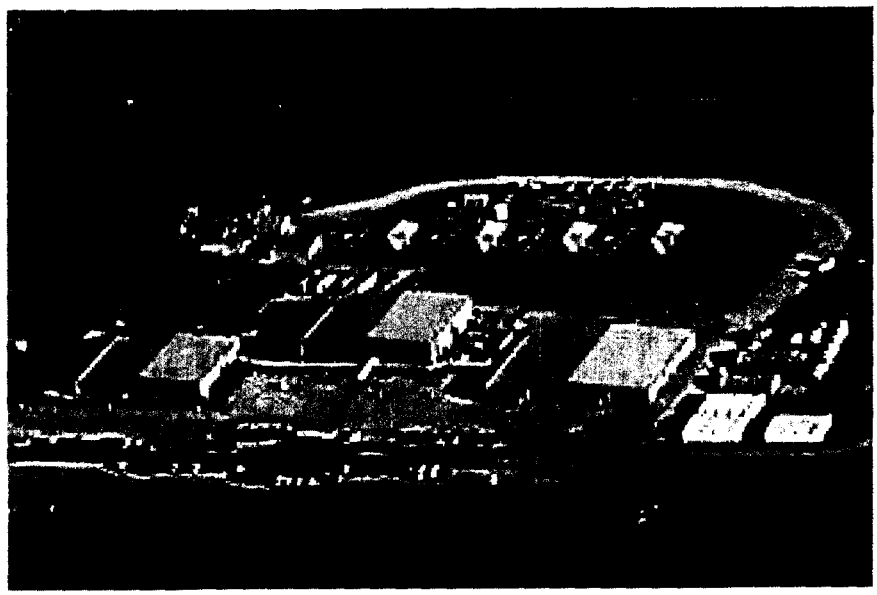


Figure A4-2. Al-Udeid, Qatar – Combined Air Operations Center (CAOC)

Deployment and Post-Deployment

Responsibility for deployment and post-deployment is shared between the government and contractor; this topic was covered in Section 3.3, on testing. Originally, deployment was solely a government activity. After TBMCS failed the first operational test it became apparent that the contractor needed access to government facilities to adequately test the system in an operational environment. The process today is very much a shared one. Depending on the amount of change to the baseline, that process will dictate the operational scenario and test environment.

Life Cycle Support

Although specific steps and tools have been refined since the inception of TBMCS, the overall processes related to life cycle deployment and post-deployment support have remained stable. The initial vision of how to field and maintain the system, train users, and provide help desk/reachback support has proven effective and thus the process has remained relatively unchanged. The following sections explore the current methodology, highlighting good practices as well as areas that could benefit from future focus.

COTS Licensing

The TBMCS contractor procures the COTS licenses required to develop, field, and maintain TBMCS. This approach has given the contractor the ability quickly to assess the needs of the system from the initial development stage through the life cycle of the product, which includes deployment. The contractor can review the current state of deployment sites to make early preparations for technology refresh of the products, including upgrades and new COTS purchases prior to a system deployment. LM takes advantage of large quantity purchase discounts for the COTS products and applies them to the deployment costs as well as to the development, integration, and test environments. However, despite these discounts and the ability to leverage corporate purchase agreements, providing and maintaining licenses at worldwide locations does not come with a small price tag. Labor associated with tracking and maintaining software licenses costs less than labor to develop engineering software, but once the costs of new license procurements are added, COTS maintenance imposes a comparable expense on development.

This approach can also cause problems, as the contractor is often merely a pass-through entity. Legal stipulations on what is and is not required in the software license agreement create a gray area that often requires attention, especially in the case of foreign military sales. Freeware and shareware products also tend to make the licensing exercise more cumbersome, as the distributor has no incentive to make changes to the license agreement.

Kitting and Fielding

Preparations for a TBMCS fielding involve on-going configuration/data management (CM/DM) practices. LM and the customer compose a fielding priority list in advance of a fielding decision to minimize any delay in shipment and to ensure that the rollout plan serves destinations with the greatest operational need first. Lists of points of contact (POCs) at the receiving location are updated continuously so that inaccurate data on recipients does not slow the fielding of releases, service packs, etc. Prior to shipment, reference documents, training materials, and software baselines are appropriately marked, prepared, and shelved for the

fielding decision. In this way, CM is maintained for all spirals, service packs, and COTS/GOTS for tracking purposes as well as to ensure the ability to ship efficiently and in a timely fashion.

Training

The TBMCS program approaches training from the perspective of getting the most impact from the limited funding available. As a result, training does not reach everyone who needs it and more robust training materials could be developed if the funding profile were increased.

Development of training materials lags the software development cycle only briefly. Training developers attend QP events to remain current with the anticipated new and updated applications and to assess impacts to training materials for both Web-based training and Mobile Training Teams (MTTs). They finalize outlines of the material in conjunction with the maturity of the application. However, screen captures are typically not taken until the application is complete, which reduces any rework required if screens are altered. The time between software baseline completion and the fielding decision gives training developers a 30-day window after fielding to complete their training materials, and they often do so earlier.

Web-based training is viewed as giving the greatest "bang for the buck," as materials can be mass produced on CDs for distribution at little cost. Training CDs are produced for each spiral release and distributed with the fielding kits.

MTTs, although costly, provide the most user-specific training possible. The contractor pre-coordinates with the site's training POC to determine what type(s) of students will attend the training session; thus, the contractor can specialize the training modules for the students' needs. In addition, this pre-coordination often lets the instructor know if the site is not familiar with the most recent SOR, so that difference training would need to be expanded to include several version hops. MTTs also provide trainers feedback on the quality/quantity of training via end-of-course surveys. The valuable input received through these surveys is often incorporated into future training modules.

Research is being conducted into providing new methods of training, such as distance learning. Such an approach is anticipated to be more cost effective, with only a slight reduction in quality when compared to MTTs, but at this time data is still incomplete.

Help Desk

TBMCS uses a tiered help desk approach to problem resolution for fielded systems. Figure A4-3 depicts the decision flow from problem identification through closure. The TBMCS contractor operates as the single Tier II Help Desk. Rather than describe the flow, which functions as designed, this section concentrates on the areas needing further refinement.

When the ticket originators initially report their problem, they use standardized guidelines to assign a priority level ranging from Low to Critical (there are five levels in total). Trouble tickets are then placed in the work queue at each help desk level in accordance with their priority. To date, low-priority trouble tickets have not waited inordinately long in the queue and sites have not "over prioritized" to get attention to their problems, but in theory either could occur if the queue becomes too backed up from the user's viewpoint. The advantage of handling tickets in this manner is that emergency problems are not caught in the pipeline and receive the immediate attention they require.

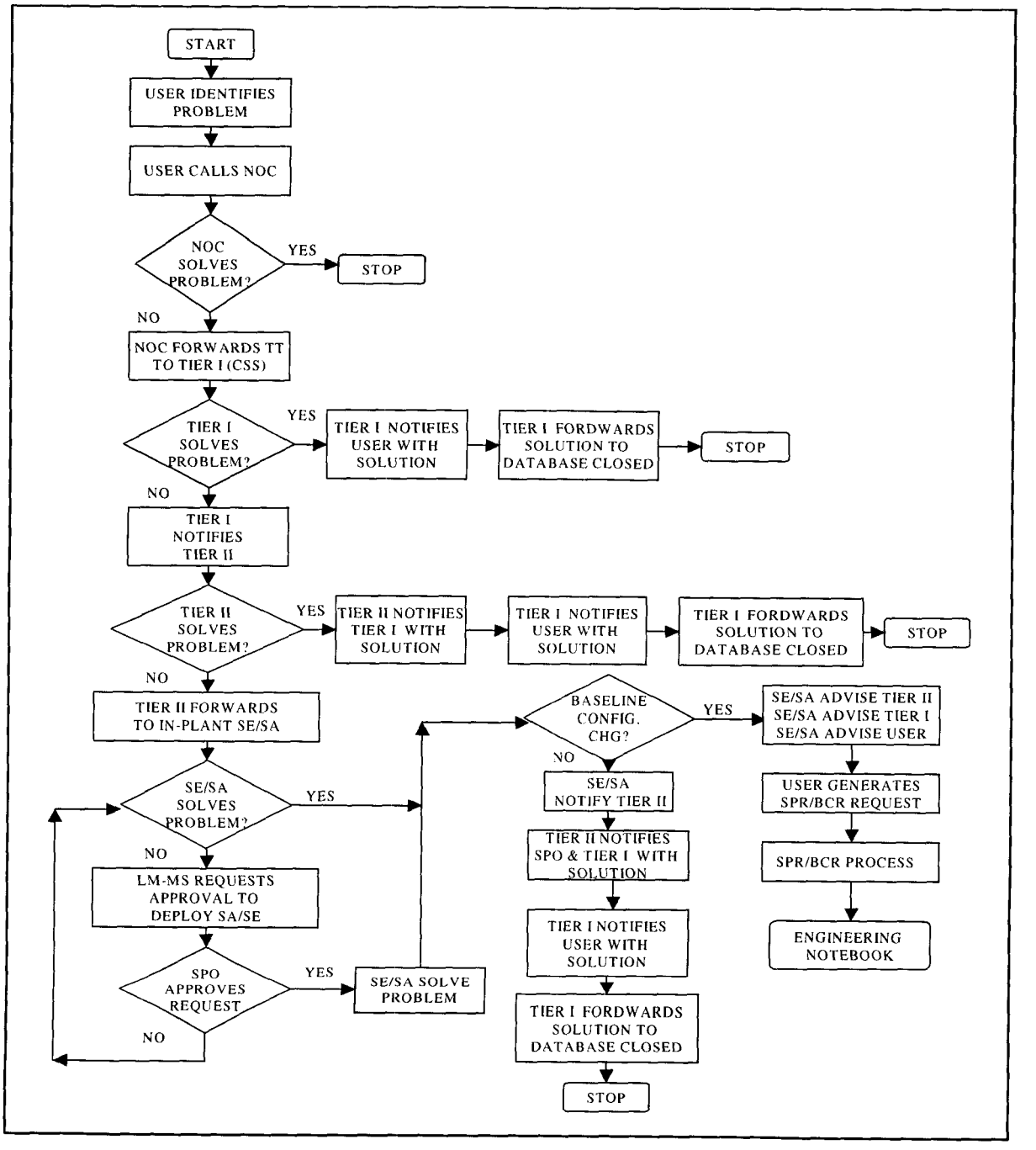


Figure A4-3. Problem Decision Flow

Each help desk tier uses its own unique database and numbering system for trouble tickets. Although the different help desks have standardized on the same database platform, they do not actively share trouble ticket descriptions and resolutions. Thus, information potentially useful to other sites is not readily available. In addition, it is theoretically possible that multiple help desk locations might be solving similar problems simultaneously.

Lastly, the Software Problem Report (SPR)/Baseline Change Request process does not always work at optimum effectiveness, especially if the problem is specific to one site and

requires a significant amount of software coding/design. Currently, SPRs must be generated by the user, sponsored by an advocate, and placed high enough in the priority queue to impel changes in implemented software. Low-priority changes and/or limited funding can prohibit a change from ever reaching the cutoff level for software development.

Appendix 5

Risk Assessment and Management

Risk management for TBMCS evolved from a contractor-only activity to a joint government/contractor process. The original acquisition strategy gave LM total system responsibility; the government would reward performance based on a potentially generous award fee. After the series of test failures and schedule re-baselines on the initial TBMCS version, the government took a much more proactive role in assessing and managing risk for the program. Finally, after the completion of the V1.0.1 operational testing in August 2000, a shared process was developed for TBMCS, which remains in use today. Figure A5-1 shows this shared closed-loop process.

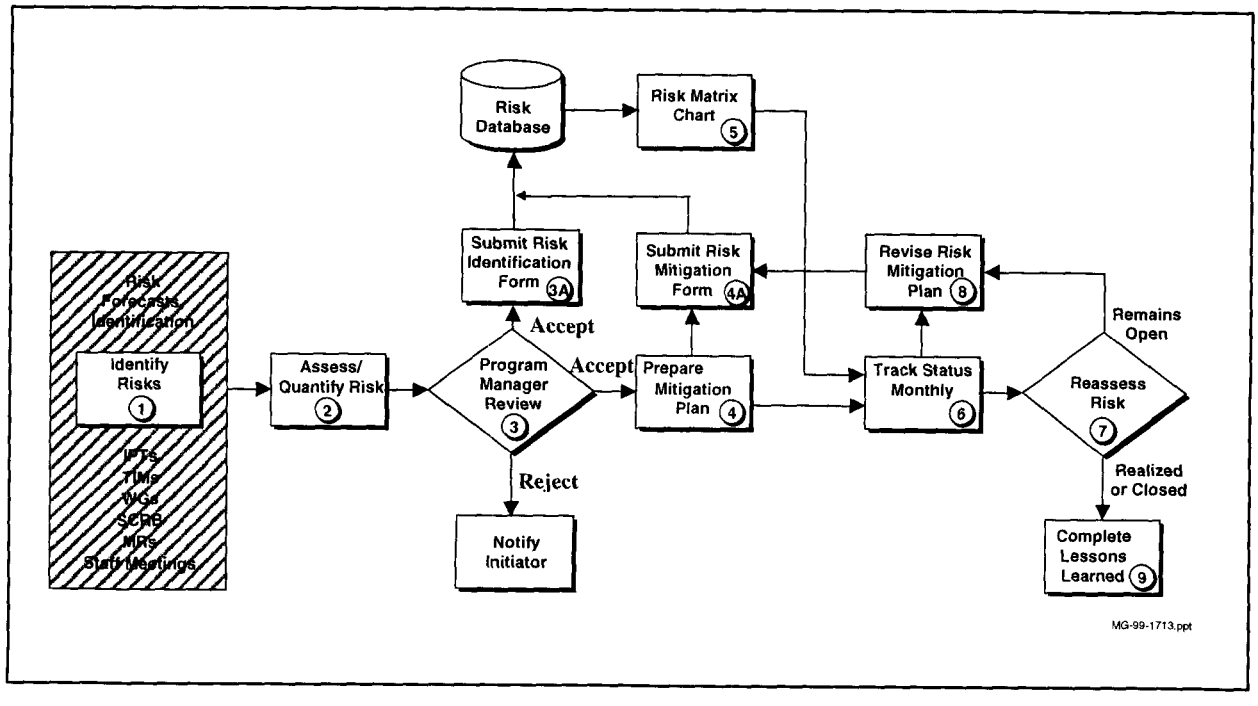


Figure A5-1. TBMCS Program Risk Management Process

LM risk management was based on a two-tiered approach. Many of the technical risks were managed by the implementing engineering organization at the IPT level and addressed as part of the QP Review process. As the system development progressed through the QP process, LM managed risk items implicitly by tracking progress in terms of cost, schedule, and performance. Risks considered to have a high probability of affecting the contractual cost and/or schedule baseline were elevated to the program level for program manager cognizance and direct participation in the mitigation activities. It was incumbent on the IPT leaders to determine which risks to manage at which level. Clearly, the program manager did not have time to address every technical risk personally.

Throughout the TBMCS lifecycle, the three major performance risk areas for TBMCS were:

- Floating requirements and user expectation management,
- Maturity of third-party products, and

- Government-provided software infrastructure, system integration, and test.

At first, because of the early implementation of acquisition reform, contractual system performance requirements were expressed only as goals. LM was committed to satisfying these goals and built performance monitoring into part of its QP process. This meant that LM would continually assess performance congruent with the software build schedule, but at the subcomponent level only. Originally, LM had proposed to the government a 12-person team to model and assess end-to-end performance, but the SPO could not afford the proposal and rejected it. Organizationally, LM had two performance engineers positioned in the development team, but they did not provide an overarching system view of how the system would operate from an end-to-end perspective.

In several instances, the performance risks were masked and not discovered until very late in the version development cycle. Key contributors to this problem included the lack of a formal CONEMP and CONOPS. Since the legacy requirements and CONOPS from the original CTAPS program sufficed for the government, the LM systems engineers were left to extrapolate from them how the system would be employed in the real world. LM system testing was focused on single-thread functional testing and not on end-to-end, multi-threaded testing with concurrent operational processes. Unfortunately, LM did not fully exercise the system in the way the user would actually employ it, resulting in the latent design flaw in accessing the MIDB (as described in Section 3.4). By the time the problem was fixed the schedule had slipped eight months. CONEMPs have since been developed and are exercised as part of the development test process. In a lesson to other developments, LM separated the systems engineering organization from the development organization to ensure end-to-end performance testing.

A second major risk area was government-directed third-party applications, specifically an application called FLEX. FLEX was a Rome Laboratory Advanced Concept Technology Demonstration to show that the AOC could monitor execution of the ATO. The concept was correct, but the lesson learned from the painful transition from a prototype to a production-quality application was a very expensive one. The application provided an automated interface between the AOC and the operational flying units, and gave the JFACC near-real-time updates on the status of ATO execution. FLEX worked superbly as a demonstration with a handful of users, but could never scale to an upper bound of 150 to 200 simultaneous users. After three years of development at a cost of $20 million, the product was dropped and replaced by an application developed in-house by an LM subcontractor. To minimize risk in integrating third-party software products, LM published its SDK for vendors to follow. In some cases, when risk was known to be very high or quality was a concern, LM would hire the vendor as a subcontractor and contractually require a specific capability and performance.

In March 1999, after TBMCS failed the first operational test and PEO leadership changed, the government took on a much stronger oversight role and began managing risk at the program level with all key stakeholders. As discussed in Section 3.4, the SPO adopted AF MAN 63-119 as a guideline. The new PEO wanted tighter control than before and required quantifiable metrics for the award fee. At the time, the program was three-and-a-half years late and $40 million over budget, and required more discipline and rigor in the systems engineering process. Some of the specific changes made required a government-run configuration control board to control the system baseline and LM to report earned value. Program performance progression and risks were managed at the senior level (O-6) on a biweekly basis; the results were used by a second, monthly, management process at the general officer level.

Managing the risks and applying more rigor and discipline proved a positive step toward passing operational test. After TBMCS passed MOT&E, the SPO and LM adopted a formal shared risk management process documented in a risk management plan. The contractor and SPO now meet on a monthly basis to discuss the risks, mitigation plans, and disposition. The risk management plan gives greater insight at the program manager and chief engineer levels and leads to better management of overall system development.

Appendix 6

System and Program Management

Both the government and LM recognize and support the essential role of systems engineering in TBMCS program development and management. Systems engineering performance is an evaluated factor in the TBMCS Award Fee Plan. LM company policy requires a robust systems engineering program, which is critical to achieving and maintaining company ISO certification and high CMMI ratings. Accordingly, the TBMCS Program Management Plan specifies systems engineering as a critical element for successful contract implementation and assigns responsibilities for ensuring active systems engineering implementation.

The TBMCS Systems Engineering Management Plan (SEMP) provides detailed systems engineering guidance specific to the TBMCS program. It prescribes the management and engineering processes for planning, implementing, and controlling the systems engineering activities across the TBMCS program. It is routinely updated to reflect changed and/or new policies and practices. The key elements addressed are:

- Systems engineering organization, roles and responsibilities,
- Formal systems engineering reviews and products,
- Major systems engineering functions and processes,
- Systems engineering controls, baselines, and boards,
- Integration of specialty engineering,
- Metrics, QPs, and status, and
- Risk management.

Organizationally, systems engineers are embedded in practically all TBMCS program entities. The Systems Engineering IPT resides at the top level of the program. The TBMCS chief architect, who reports to the TBMCS program director, leads this team and is responsible for all systems engineering effort on the program. The IPT performs the principal engineering integration function to ensure unity of technical effort and control of the engineering baseline across the entire TBMCS program. Key IPT responsibilities include:

- *Requirements* definition, management, and control;
- *Version planning* to include content, prioritization, budget estimates, and design reviews;
- *System architecture* definition and baseline control;
- *External interface* definition, design, and control via Interface Control Documents (ICDs);
- *Security engineering*;
- *New technology* planning and infusion;
- *Specialty engineering*; and
- *Trade studies.*

It is also noteworthy that, in the spirit of the IPT philosophy, systems engineers are embedded in each of the software development/integration IPTs to ensure the integrity of the system requirements, architecture, and other version baselines.

The TBMCS organizational structure ensures that systems engineering is formally represented in all front-end engineering activities as well as in the software development, integration, and test phases of overall product development. Systems engineering therefore serves as a unifying factor across the system lifecycle to guarantee the integrity of the requirements, architecture and system performance baselines.

Systems engineering also serves as a key communications, planning, and issues resolution activity both internal and external to the TBMCS program. The SEIPT, which includes members representing the TBMCS contractor, government acquisition SPO, and user (AFC2ISRC), was established as an informal body to accomplish technical and programmatic (budget and schedule) planning and resolve the difficult issues in advance of the AFI 63-123 formal Joint Requirements Planning Team and SDIPT activities. The SEIPT, which meets weekly, has proved very effective in resolving the technical and programmatic issues related to version planning and implementation in advance of the formal version approval activities that are required to initiate contract actions with specific requirements, budgets, and schedules.

In addition to the TBMCS version-related requirements and design reviews, the SPO also requires quarterly Program Management Reviews. The purpose of these reviews is twofold: (1) assess contract performance against the contract baseline, and (2) review and evaluate future plans from both the programmatic and technical perspectives. The SEIPT is responsible for outlining future plans, presenting alternatives, identifying programmatic and engineering issues, and providing resource estimates. Action items are routinely generated from these reviews.

Systems engineering implementation on the TBMCS program generated several key lessons learned:

- Both the government and the contractor must recognize the need to fully fund critical systems engineering activities. As a key example, in the initial years of the program, the government did not provide funding for the SEIPT, but instead embedded funding for those critical systems engineering activities in the budget lines for product development. This tended to reduce emphasis on the SEIPT's advanced planning functions and made these activities less than effective. In the later years of the program, the SPO has provided a unique budget line to support the SEIPT, which resulted in an extremely robust implementation of these necessary systems engineering activities.
- Version requirements, schedules, and funding must be linked early in the process to ensure effective use of the contractor work force and meet user expectations for product delivery. This is a joint government–contractor systems engineering activity. Delivering a version product that meets user expectations for functionality, cost, and schedule depends upon several key factors in the engineering development cycle: (1) establishing a firm requirements baseline, (2) prioritizing the requirements so trade-offs can be accomplished, (3) estimating what can be delivered by the contractor within the budget and schedule targets, and (4) completing the RFP, proposal, and contract authority actions. The SPO, user, and contractor must work closely together,

on a prescribed schedule, to complete these actions or the contractor work force will not be used effectively and user expectations will not be met.

Systems engineering is the glue that holds program implementation together, while keeping it on a track consistent with design, functionality, and performance baselines. It plays a key role in both engineering implementation and program management. Systems engineering inputs are critical to the program risk management process. To implement systems engineering activities successfully, systems engineering policies and practices must be documented in a program-specific SEMP and then adhered to religiously. Systems engineers must be embedded throughout the program organization to ensure the establishment and integrity of system baselines. The government and contractor must recognize the need to provide adequate resources for systems engineering activities throughout the program life cycle.