Theater Battle Management Core System: Lessons for Systems Engineers

Josiah R. Collens, Jr.
jrc@mitre.org

Abstract

The difficulties encountered during development of the Theater Battle Management Core System (TBMCS) provide lessons for systems engineering of large-scale, software-intensive systems. The absence of formal requirements and oversight, coupled with strong pressure for rapid deployment, caused the program to fail its first operational tests and actually delayed its deployment to the field.

The lack of measurable requirements and the need to integrate multiple third-party products and systems made it impossible to establish a system baseline and to test TBMCS in realistic conditions. Thus, significant problems manifested themselves only during official government tests. Moreover, despite nominal authority, the lead contractor had little or no control over the government-furnished elements and commercial off-the-shelf products that TBMCS was to incorporate.

Experience with TBMCS leads to several conclusions. First, the more complex a system, the greater the need for rigor and discipline in engineering processes. Second, well-defined requirements are essential. Third, mandating the incorporation of specific third-party hardware or software may create severe problems for system development. Other lessons highlight the importance of open standards in a heterogeneous information technology environment and of layering with well-defined interfaces to facilitate integration and system evolution.

Introduction

The Theater Battle Management Core System (TBMCS) is an integrated air command and control (C2) system that enables an air component commander to plan, direct, and control all theater air operations and to coordinate with land, maritime, and special operations elements. It encompasses hardware, software, communications links, spares, personnel, training, and other resources. The system is currently deployed worldwide at both the operational and tactical levels and is actively supporting Operation Enduring Freedom and Operation Iraqi Freedom. Figure 1 depicts the operational theater and the interaction among the TBMCS components. A detailed history of TBMCS development can be found in [1].

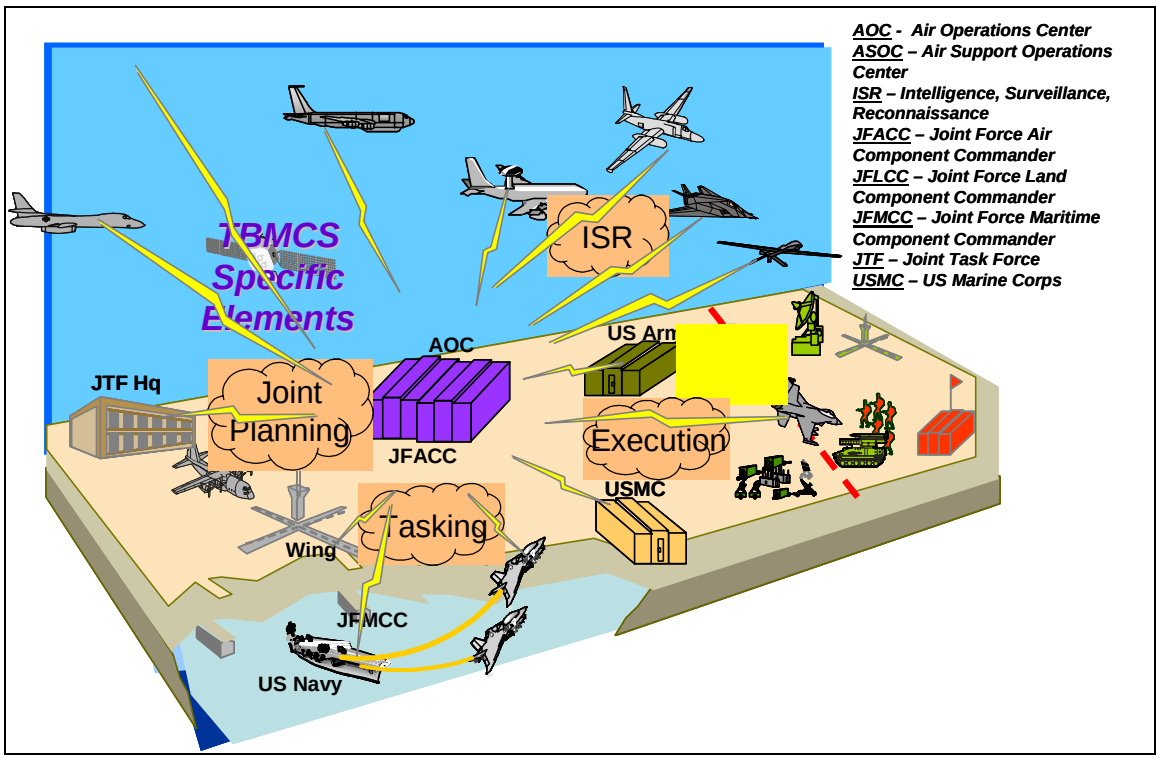

Figure 1: TBMCS Operational Concept

The TBMCS concept responded to user demand for a more streamlined process to generate the Air Tasking Order than that used in Operation Desert Storm. Although the target system would necessarily be highly complex, the Air Force sought to minimize fielding time by eliminating requirements to build to military standards, reducing government oversight, and mandating adoption of best commercial practices. The lead contractor, Lockheed Martin (LM), was designated as the system integrator and was given Total System Performance Responsibility (TSPR). The Air Force System Program Office (SPO) was instructed to provide insight rather than oversight and in essence free LM to find its own path toward producing the system. The ability to reuse software applications across a common infrastructure also became a key program/design driver.

While highly laudable in theory, these approaches ignored the realities of building complex systems for deployment in a combat environment, and had serious consequences for TBMCS. The repercussions affected all aspects of system development, from architectural design to testing. Two aspects had an especially strong impact: the lack of a formal requirements baseline and the mandate to integrate components from multiple sources into an operational system.

Requirements

Although the Air Force led the program, TBMCS involves significant joint service participation; thus, the requirements came from many sources. Figure 2 shows the organizations involved in the TBMCS requirements process in the year before TBMCS underwent its first operational test (OT). Over time, the operational user community fed

requirements into TBMCS from the top down, while various functional components simultaneously drove requirements from their existing implementations back into the system.

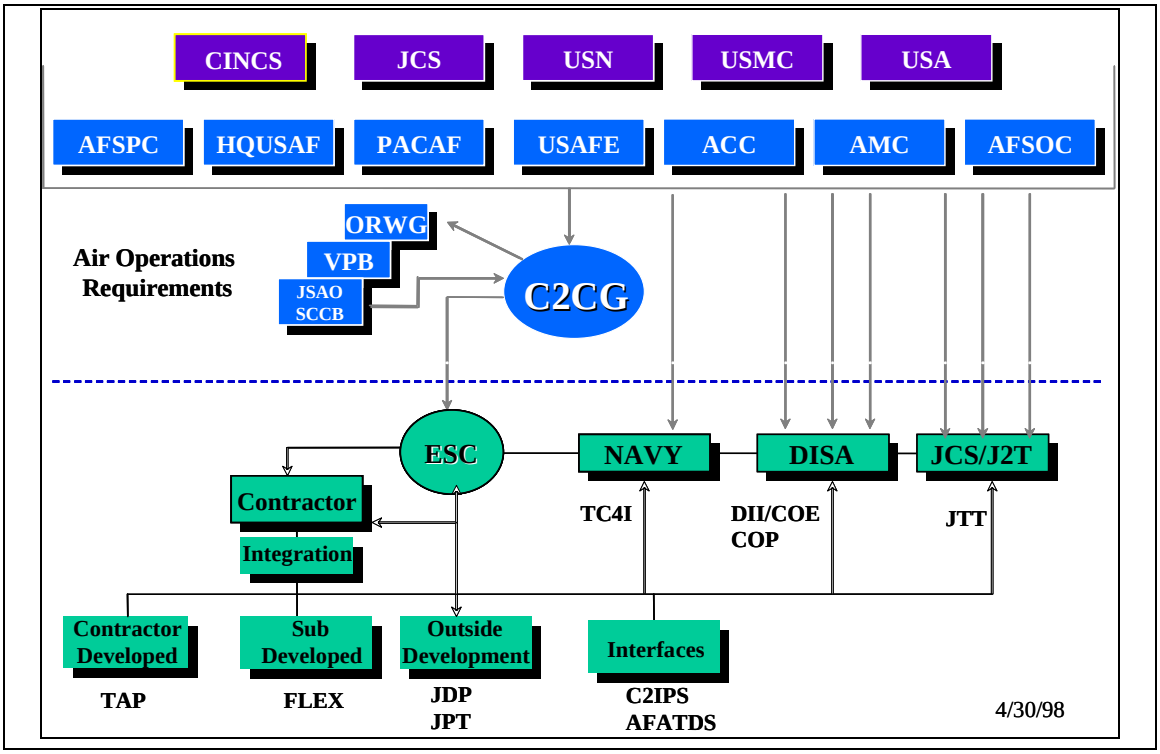

Figure 2: TBMCS Participating Organizations (circa 1998)

The Air Force intended TBMCS to integrate the functions of three systems: the Contingency Theater Automated Planning System (CTAPS), which was under development, the Wing Command and Control System, and the Combat Intelligence System [2]. Because it was considered simply a modernization of existing legacy systems, TBMCS did not undergo the usual formal joint requirement approval process [3]. The user, the Air Force Command, Control, Intelligence, Surveillance, and Reconnaissance Center (AFC2ISRC), believed that TBMCS would not need its own Operational Requirements Document (ORD), because the ORDs for the legacy systems would remain valid and TBMCS requirements could evolve from them. TBMCS would therefore avoid the bureaucratic delays involved in the normal Department of Defense requirements generation and review process and in accreditation as a joint program. However, the ORD specifies the performance and related operational parameters for a proposed concept or system, and the lack of such a baseline proved highly detrimental to TBMCS.

TBMCS also lacked an overarching concept of operations (CONOPS) to define how the system would actually be used in the field. Again, AFC2ISRC believed that the CONOPS for CTAPS would suffice for TBMCS, and therefore tasked The MITRE Corporation to generate a Technical Requirements Document that provided a top-level description of how the system might be employed [4]. The tacit guideline was that TBMCS

functionality and performance should be at least equal to those of the legacy systems. As a result, the system architecture was defined at too high a level, which had a tremendous impact on system design and development.

The system development team attempted to cope with this lack of guidance by breaking down the requirements provided and performing an initial analysis of the candidate solutions based on specific factors, such as the commercial off-the-shelf/government off-the-shelf (COTS/GOTS) products mandated from above. The technical leadership of the program then attempted to create a system definition to meet these requirements. The fully harmonized approach of the TBMCS integration and development team proved essential to gathering the information needed on all of the products involved in TBMCS, but did not overcome the inherent problems of contradictory user demands and a fluid baseline.

The lack of an ORD and a CONOPS for TBMCS also had serious implications for the testing community. The loose requirements process made managing expectations extremely difficult. The criteria for assessing system performance became somewhat subjective and left room for interpretation. The formal, documented performance was not agreed to until the OT plan was approved. Moreover, the requirements continually changed depending on which product the government wanted LM to incorporate into the baseline – a critical problem in itself. 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 their overall impact. In one case the impact only became evident in OT, which revealed a major problem in the intelligence database that resulted in an eight-month schedule slip.

Pressure from the operational community prompted the government to force early operational testing, even though both LM and the Air Force knew the system was not ready. The tests that LM had performed did not exercise concurrent processes at the system-of-systems level to assess overall performance, and did not involve nearly as many simultaneous users. The Air Force leadership therefore wanted the first official tests merely to assess system maturity, but the joint test community insisted that this test be a pass/fail Operational Test and Evaluation (OT&E).

System Integration and Interfaces

TBMCS involved four types of integration: internal interfaces and subcomponents, third-party applications, external interfaces, and databases. Fully 90 percent of TBMCS consisted of third-party products or government-furnished equipment (GFE), and a majority of the software was third-party: GOTS or COTS. TBMCS incorporated 76 applications, 64 point-to-point external system interfaces, and 413 segments involving over 5 million lines of software, as well as two commercial relational databases. The system had two hardware baselines, and the communications infrastructure was run by the Defense Information Systems Agency. In the abstract, the requirement to use the DII COE as the common software infrastructure represented a worthy goal; unfortunately, the infrastructure could not keep pace with commercial information technology, making integration difficult and expensive. The most extensive integration in TBMCS involves data interoperability, and the two primary TBMCS databases – the Air Operations Data Base and the Intelligence Server Data System – follow different standards and are

updated at different intervals. The government also mandated the use of specific hardware, which varied depending on the service branch that would use TBMCS.

This situation led to severe difficulties. The SPO tasked the contractor to integrate disparate legacy capabilities by using open standards with a common user interface. The architecture should allow flexibility for new capabilities to evolve. In theory, LM was the system integrator and had TSPR; in practice, third-party integration meant that LM had little control over the configuration of TBMCS. Determining the quality of a third-party product and coping with hidden design flaws during execution proved highly problematic (and remain problems today). This forced the government to broker changes to the product when problems arose and often resulted in delays and increased cost.

Integrating immature third-party applications also demanded extremely high levels of resources. A particular application requested by the user might be very difficult to integrate into the system either because it did not fit into the DII COE or because its COTS infrastructure was more current than that of TBMCS. This led to extensive overruns in integration cost and schedule. Occasionally, LM had to reduce applications in functionality or replace them with other products to achieve integration and operational capabilities.

Thus, program synchronization was exceedingly hard to achieve. Many of the mandated systems were undergoing parallel development while TBMCS was being created, which meant that all stakeholders had to achieve a reasonable current baseline of the products that would be stable long enough to allow LM to integrate them into the larger TBMCS. For example, at one point TBMCS was based on Solaris 2.5.1, while Sun had already 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.

LM did not have the capability to conduct a live test of the external system interfaces in-plant. Each interface was tested with known inputs and outputs to ensure it was working properly, but simulation did not always reflect performance under realistic conditions, and this led to failures during the actual test.

Results

Not surprisingly, TBMCS failed its first OT in March 1999. As a result, the Air Force established a new baseline for TBMCS and more government oversight was brought to bear, including mandatory oversight by the Office of the Secretary of Defense (OSD). The SPO and LM 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 established a serial test process with entrance and exit criteria for each test event. In September 1999 the user community reduced the Key Legacy Functions (KLFs) that TBMCS had to perform to five essential capabilities.

The program was then able to move forward by modifying more traditional engineering processes. The SPO–LM team adapted an existing engineering process of design/development with periodic reviews to succeed in an environment where LM lacked direct control over the component products. A System Design Review that bridges

the operational and engineering activities has been held for each release since the release of the core baseline (V1.0.1).

Despite these changes, the second OT, which began in January 2000, was suspended because of a problem that prior tests had failed to reveal because the system had never been exercised in a true battle rhythm. At this point, the SPO chief system engineer assumed responsibility for the technical integrity of the system. With LM help, the SPO developed performance tests that reflected a realistic operational battle rhythm. These became part of the formal developmental test (DT) process that TBMCS would have to pass before proceeding to Multi-Service Operational Test and Evaluation (MOT&E).

These new processes, coupled with the reduced number of requirements, enabled TBMCS to pass its MOT&E in July 2000. TBMCS Version 1.0.1 received a favorable fielding decision in October 2000. The following year, the user community approved a TBMCS CONOPS. 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, and produced a TBMCS ORD, which the Joint Requirements Oversight Council approved in February 2002. The ORD defined the objective TBMCS with the understanding that TBMCS would field spirals with increasing capability, which would eventually produce the objective system.

To support this strategy, the Air Force designed a new requirements process, still in use today, that established a Requirements Planning Team (RPT) and created an on-line database to house all TBMCS requirements. The database provides a central focal point that allows all stakeholder representatives to take part in the process. The program also adopted Air Force Instruction (AFI) 63-123 for spiral development and stood up a Spiral Development Integrated Product Team (SDIPT) comprising users, testers, program managers, and system engineers [5]. The SDIPT uses prioritized requirements generated by the RPT to produce a spiral plan that covers capabilities, cost, and schedule for LM. In addition, LM and the SPO established a System Engineering Integrated Product Team that is responsible for the architecture and performs the requirements analysis for the spiral plan.

The lessons learned from the difficulty in fielding V1.0.1 had a positive impact on the program's current systems engineering environment. The requirements process for TBMCS has now matured into a relatively disciplined and repeatable process tied to a specific spiral of capability. While the government controls the requirements, it shares most roles and responsibilities with LM. The SPO and the contractor manage each upgrade jointly, and test is factored into the planning process.

TBMCS performance in Operations Enduring Freedom and Iraqi Freedom demonstrates the success of the current approach. The program is producing its fourth spiral in five years, and leads the way in delivering the latest in Web and information services technologies as TBMCS evolves to support network centric warfare.

Analysis: Lessons Learned

Experience with TBMCS provides a comparison to the modern systems engineering theory and practices taught in leading universities today and leads to several conclusions.

The lessons learned from TBMCS apply directly to other software-intensive programs that require the integration of vast numbers of third-party products with GFE, such as hardware and communications.

Requirements

The requirements process for TBMCS V1.0.1 was profoundly flawed from the start. The acquisition community had a utopian vision of a single modern, integrated, joint C2 system, but had no operator requirements to support it and no CONOPS that described how the system would work as single integrated capability. It took five years to complete the initial TBMCS baseline; in fact, the SPO never established a firm baseline until after TBMCS failed its first major OT. As a result, the test community and the other military services found it difficult to determine what capabilities TBMCS would provide and how the system would be used.

The strategy for developing and fielding TBMCS capabilities was predicated on evolutionary acquisition, but spiral development does not obviate the need for a rigorous and disciplined requirements process. The government wanted to field capabilities to the operator quickly by delivering capability over three increments, culminating in an OT. However, for such an incremental approach to succeed, a program must first establish a baseline from which the system can evolve. TBMCS lacked such a robust baseline, and this had tremendous impact on cost, schedule, and performance.

Initially, TBMCS did not even establish a vision that the program could follow. Three years after contract award, the government finally agreed that TBMCS needed such a vision and a roadmap to achieve it. Jointly, the government and LM designed a target architecture that 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.

Third-Party Integration

It seems intuitively obvious that assigning TSPR to a contractor when over 90 percent of the program content was GFE was a flawed strategy. Contractors cannot be held accountable for performance unless they control all of the system components that affect performance. In theory, the government gave the contractor free rein; in practice, it dictated to the contractor what to do and what equipment to use. The government's mandate for software reuse and use of commercial software products were contradictory and problematic, although the layered system architecture designed by LM did support system evolution and migration to modern technologies. 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. The significant GFE requirements for both physical equipment and other crucial system components meant that LM always needed government support to deliver TBMCS increments.

Software reuse was less straightforward than originally envisioned. 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. The TBMCS software infrastructure changed sufficiently to warrant a separate baseline. Moreover, 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, which led to extensive overruns in integration cost and schedule. This illustrates the importance of using open standards, rather than specifying particular commercial products as the software infrastructure.

It is also essential to understand the maturity of the third-party products specified in a system design. Unfortunately, proof-of-concept demonstrations and user-developed applications did not always transition into production-quality products, and the process and schedule did not permit such an assessment. Development programs must build in an assessment process that allows the integrator either to build the software application or to replace a required product with another if the third-party product does not integrate well – meaning that it takes more time and money than the budget allows.

The government must provide stable interfaces and an environment that allows the contractor to test them. External interfaces must be fully tested in a real-world environment at both the functional and technical levels. If a program's schedule slips, any system that releases an update to the interface must be backward compatible.

Scheduling

Schedules, requirements, and budgets must be realistic. 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. Months after contract award, the government diverted LM from developing TBMCS and instead directed the contractor to fix and field the legacy system CTAPS. This led to a three-year schedule slip and consumed 70 percent of the budget allocated for TBMCS. TBMCS never recovered. The remaining resources were devoted to testing and fielding TBMCS V1.0.1 on an accelerated schedule.

System Integration and Test

The government did not concern itself with the suitability of TBMCS and its components for formal testing. The System Segment Specification that governs testing never really reflected the baseline; instead, it lagged behind the current program requirements. In addition, the government continually changed the allocated baseline with mandated third-party products, which did not always reflect the agreed-to requirements. Continued pressure from the user community to field V1.0.1 resulted in a failed OT and later basically forced relaxation of test criteria. The program was forced to review and rework some areas based on the results of formal tests instead of feedback from internal activities.

External interfaces must be 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, and simulation was not always a good indicator of performance. Having the contractor test the system in an operational setting is essential.

TBMCS's two failures in OT and the subsequent remedial actions indicate that the SPO must take ownership in managing the risk for DT and OT. System engineering must play a major role in planning the tests and managing technical risks; again, there is no substitute for a well-defined requirements baseline. Obtaining user agreement on the pass/fail performance criteria was a Herculean effort. Moreover, testing is a building block process that must be run in a serial mode with well-understood entrance and exit criteria. For TBMCS, schedule considerations overruled the test planning process, so that LM was performing integration tests while the government was running development tests.

Finally, system developers must understand how the system will be employed. Again, a detailed CONOPS and a corresponding concept of system employment are essential. For TBMCS, not testing the overlapping processes involved in building and managing air operations prior to an OT was a major mistake, and violated the fundamental systems engineering principles of effective test planning, risk assessment, and definition of external system boundaries.

TBMCS has made several improvements to the test planning since V1.0.1, and the SPO continues to take a proactive role in managing risk. The processes are now serial, with well-understood entrance and exit criteria. Stress testing during DT reflects the real operational load, to include interaction among cells and live testing of interfaces. Finally, field test is part of the contractor and DT testing prior to operational test.

Conclusion

Rather than serve as an exemplar for reduced oversight and relaxed standards, TBMCS teaches the lesson that nebulous requirements demand especially rigorous systems engineering processes. The Air Force's well-intentioned attempt to reduce bureaucratic burdens on system development proved inappropriate to a complex system-of-systems integration program. In the case of TBMCS, external influences drove a relaxation of discipline and rigor in the systems engineering process. In fact, the need for a detailed and accountable process increases 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.

The lessons learned from the difficulty in fielding TBMCS V1.0.1 had a very positive impact on the program's current systems engineering processes, which have evolved to become mature and repeatable. The demonstrated success of TBMCS in Operations Enduring Freedom and Iraqi Freedom testifies to the success of the current approach, as does the contractor's ability to field four subsequent releases since the release of V1.0.1.

References

[1] Collens, J.R., Jr., and B. Krause, *Theater Battle Management Core System – Systems Engineering Case Study* (Wright-Patterson AFB, OH: Air Force Institute of Technology, 2005).

[2] 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).

[3] Department of Defense Instruction 5000.2, *Operation of the Defense Acquisition System*, Washington, DC: 12 May 2003 [On-line]. URL: http://www.dtic.mil/whs/directives/corres/html/50002.htm

[4] Air Force Electronic Systems Center, *TBMCS Technical Reference Document*, Contract # F19628-R0027-94, Hanscom AFB, MA, 3 November 1994.

[5] Secretary of the Air Force, Air Force Instruction 63-123, *Evolutionary Acquisition for C2 Systems*, Washington, DC, 1 April 2000.