Convergence of “Hard Spots” in the Army Tactical Command and Control System (ATCCS)

Raymond A. Paul, Technical Director for Software
U.S. Army Test and Evaluation Command

Abstract

This paper presents a case study in large-scale system integration, the Army Tactical Command and Control System (ATCCS). ATCCS is being acquired by the Army under the Evolutionary Acquisition concept. The classical system engineering strategy for system integration is to ensure interoperability of system components. However, this paper presents the concept of "convergence" as a better method to ensure effective integration of software components in the Evolutionary Acquisition process. In order to achieve convergence of software products, the Army Program Manager must implement three management initiatives:

1. accurate definition of system requirements;
2. Continuous Evaluation (CE) of system development progress;
3. continuous identification, assessment and management of software risk.

Army Tactical Command and Control System (ATCCS)

The Army Tactical Command and Control System (ATCCS) is the US Army Command and Control (C2) system which will manage all automated tactical functions on the battlefield. ATCCS is the most complex C2 system in the US Army, incorporating all the resources, procedures, and personnel essential for US Army commanders to manage their operations. As shown in Figure 1, ATCCS is actually a network of computer resources which will link a family of major automated Battlefield Functional Areas (BFA). The five BFAs are:

1. Maneuver;
2. Fire Support;
3. Intelligence and Electronic Warfare;
4. Air Defense;
5. Combat Service Support.

Figure 1: Army Tactical Command and Control System

U.S. Government Work Not Protected by U.S. Copyright
Three types of communications systems support and interconnect the five BFAs:
(1) Area Common User Systems;
(2) Data Distribution Systems;
The Program Executive Officer for Command and Control Systems (PE0 CCS) has the responsibility to field ATCCS as an effective C2 system. To achieve an acquisition strategy and coherent system, PE0 CCS must provide centralized management of the five Program Managers who are responsible for production of the subordinate BFA systems of Figure 1. The role of ATCCS and the significance of software integration in the overall system is best described by the Deputy Program Executive Officer, Robert Giordano [1]:

"Because we provide the command and control, we track all the weapons systems. So we are, in essence, the Army's battlefield architect and systems engineer. We play a role well beyond the development of the ATCCS components."

ATCCS once was viewed as a mystery of five dissimilar systems. The fact it's now being viewed as an integrated, cohesive entity that provides force integration and puts capability on the ground in a phased approach is attributable to the significance of the program executive officer and CHS [Common Hardware/Software] programs. What we're putting inside those computers is a single architecture that will permit considerable reuse of software."

**Evolutionary Acquisition**

The PEO CCS provides engineering and management support to transition from the five independently developed and emerging BFA Control Systems to an integrated (and evolving) tactical C2 system. The BFA Control Systems are in various states of completion, from the definition phase to the deployment phase, and a number of changes to these systems are also under consideration. In order to achieve the large scale integration which is required to build ATCCS, the Army has implemented a process of "evolutionary acquisition" [2]. An evolutionary acquisition strategy is recommended for systems which exhibit any of the engineering characteristics shown in Figure 2. This strategy will define an objective ATCCS, plan an evolutionary "block-by-block" approach leading to the target system, ensure user and developer testing supports the evolutionary approach, and provide discipline to the total acquisition process.

A major challenge of the ATCCS program is to manage system integration in a systematic manner. Implicit in this challenge is the recognition that the Army cannot field a system as large as the ATCCS all at once. Instead, the Army has identified the block approach that allows ATCCS system fielding in increments. The Program Manager will quickly field a core capability, collect user feedback during an incremental block upgrade program, and build towards the objective ATCCS. The ATCCS System Description defines "Block A" as the capabilities which were fielded through 1991. Block A is characterized by the initial automated integration of C2 among the five BFAs and the fielding of new wide-area communication systems. "Block B" is the phase in which the full capability required in the ATCCS System Description will be achieved. Block B will provide the overall automation capabilities desired for the ATCCS in the 1992 through 1996 timeframe. At the onset of Block B, the Army will require that new automated C2 component systems conform to a common set of standards, and much of the C2 hardware and software will be standardized across the battlefield.

**Integration Through Interoperability**

Traditionally, Army engineers have ensured integration of large-scale systems through "interoperability" of system components. Interoperability is defined in JCS Pub 1 [3] as: "the ability of systems, units, or forces to provide services to and accept services from other systems, units, or forces and use the services so exchanged to enable them to operate effectively together... Interoperability requires compatibility... Interoperability is achieved when the data exchanged between compatible systems is applied to perform functions which result in effectiveness over and above that achievable by either of the systems alone."

As a means toward achieving software standardization, the Army has embarked on a program of mandating the use of ATCCS standards for interfaces, data communications protocols, message text formats, databases, data element dictionaries, application software and software engineering development environments. Standardization to a single programming language is a significant step in software interface compatibility. The Ada programming language will become the single, common computer programming language for all mission-critical military applications.

The Army view of "interoperability" has historically focussed on hardware compatibility and compatibility of
communications protocols and message formats. The Common Hardware/Software (CHS) acquisition program is the underlying approach to ensure interoperability of ATCCS program components. The CHS program objective is to field modern computer systems through a consolidated acquisition of compatible, Nondevelopment Item (NDI) hardware and software products. These products are described in Figure 3. The goal of CHS is to facilitate the interconnection of the control systems to receive near real-time, critical, cross-functional information over Army Common User Systems, combat net radios, and the Army Data Distribution System.

In particular, the common hardware and software acquisition of the ATCCS program will provide the various ATCCS project managers with a standard family of computer, peripherals, operating systems, utilities, and application software. The Project Manager for Common Hardware Systems (PM CHS) will acquire these system components and make them available to system developers as the building blocks for developing all ADP systems under the ATCCS program. Using the CHS components as a basis, the ATCCS project managers will insert their own battlefield functional area-unique software, add any unique hardware, and assemble the components into systems which meet the needs of each battlefield functional area. These project managers will develop their systems under evolutionary acquisition by building their systems, testing them, fielding a capability, gaining user feedback, and incorporating the feedback into the next increment of the system.

While fielding and ensuring interconnection of computers on the battlefield, the Army must cope with a high degree of diversity. For example, the Army will field software that is common across all battlefield functional areas and software that is unique to each individual area. Some users will have specialized computers that are embedded into sensor systems; such as word processing. Some computers will have access to only one type of communication system, while others will have access to several. The security requirements of these systems will vary because the ATCCS must serve users with different security clearances.

**Integration Through Convergence of “Hard Spots”**

However, to provide the level of automated support required by today’s sophisticated Army users, ATCCS must provide more than hardware and software compatibility. For example, users performing the same function at different command echelons or in different environments may have different needs regarding the granularity (resolution), quality, timeliness, or perishability of data. Ultimately, the software must provide the degree of intelligence required to support all users in all environments.

In order for a software system to be “everything to everyone”, early and continuous user involvement is needed for effective system and software requirements definition and soldier/software interface refinement. That is, in order for the software to adequately support system users, users must identify common and critical mission functions or “hard spots”. Once the users have identified hard spots (such as those shown in Figure 4), it becomes possible to strive for “convergence” of those hard spots with respect to the end users’ perspective. Convergence, as opposed to interoperability, is defined as “movement toward union” [4]. ATCCS can then be viewed as an integral system as opposed to a family of interoperable systems.

In order for the integral system vision to become a reality, ATCCS must be an “open” system. Only an open system architecture can effectively support the evolution and convergence of user requirements brought about by evolving threats; evolving force structure, tactics, and doctrine; and rapidly evolving hardware and software technologies. As

---

**Common hardware** will be procured over a five year period and will include:
- a Handheld Terminal Unit (HTU);
- a Portable Computer Unit (PCU);
- a Transportable Computer Unit (TCU);
- compatible NDI peripheral devices.

**Common software** will be acquired from the commercial market and will include:
- operating systems;
- data base management systems;
- Ada language programming tools;
- communications programs;
- training programs;
- maintenance diagnostic programs.

**Figure 3: Common Hardware Software Elements are the Basis of ATCCS Interoperability**

**Figure 4: Examples of “Hard Spots” which must be Identified to Manage the Convergence Process**
shown in Figures 5 and 6, design for convergence must consider multiple aspects of the system architecture and system operational requirements.

With increased resource constraints, increased pressure is brought upon the military community to use Nondevelopment Item (NDI) products. Thus, ATCCS must have the built-in capability to migrate to new technologies and architectures in a cost-competitive manner. The computer software must be the leading edge of technology insertion in order to meet the challenge is to define the standards for a multi-level, secure open system.

Software Test and Evaluation Panel (STEP)

The concept of requirements and functional convergence, as implemented in an open system environment, is in its infancy. Concurrent with further development and maturation of this concept, the Army is pursuing a strategy of imposing increased discipline to the software T&E and development process. This strategy is being developed by the Army Software Test and Evaluation Panel (STEP), which was initiated in September 1989 by the Commanding General, US Army Operational Test and Evaluation Agency (OTEA). The immediate objective of STEP was to address the problem that most of the delays in Army operational tests had been caused by immature software. The underlying cause of the problem was that the technical and management resources for software testing had significantly lagged behind the complexity of embedded software in Army systems. STEP was eventually chartered as an Army-wide effort to investigate and improve the state-of-the-art of Army software test and evaluation (T&E). The long-range objective of STEP was to discipline the software development process by developing software T&E capabilities to:

1. Demonstrate the progress of software at specified intervals;
2. Prevent immature software from entering user system tests or being deployed.

STEP membership has involved a cross-section of the Army community to determine both short-term and long-term solutions to current software T&E problems. STEP focused attention not only on software T&E, but also considered related issues in software development and deployment. Three of the STEP initiatives have immediate application to the capabilities for requirements convergence:

1. Software Requirements Definition;
2. Software T&E Improvements;

Army End User Software Requirements Definition

In the "downsized" Army of the future, the user will be expected to "do more with less". As resources are increasingly constrained and military systems become more and more software-intensive, software will become the "Force Multiplier". For military software systems, the ultimate measure of success is whether a system meets the expectations of the end user (see Figure 7). The Army user expects software to work as part of the total system, in his environment, when he needs it—when his life depends on it.

In order to effectively serve the user, it becomes necessary to know the user and his environments. The Army user's thoughts and software requirements are defined by his personal experience; by his particular mission; by the specific threat or enemy; and by the environment. The ATCCS will be required to perform as the same Army system for a variety of users in a wide range of operating environments. Because the ATCCS users will have different priorities and needs, it is necessary to identify common functions.
The Software Must:
- Enable the System to Perform the Mission(s);
- Be Robust and Fault Resistant;
- Be "User Friendly" to Facilitate Learning and Continued Operator Proficiency;
- Be Protected from Unauthorized Alteration;
- Be Designed to Operate in the User's Environment and Under Expected Stresses (Multiple Stress Factors);
- and
- Provide an Added Value to the System.

The "requirements process" never ends for large-scale integration systems under the evolutionary acquisition concept. Typically, these systems are characterized by having intensive user interfaces and interdependence with other systems. They are generally large and complex and they operate in a dynamic environment. The delineation of requirements for such systems is often incomplete, inconsistent, and specified at varying levels of detail, all of which significantly contribute to the risk of the development. Some full-scale development contracts for such systems are awarded with incomplete and ambiguous requirements, as the time and effort needed to improve upon requirements definition is frequently underestimated. Requirements errors are frequently not discovered until much later in the development and acquisition process, resulting in cost and schedule growth.

The Army is currently addressing requirements definition problems by staffing a Pamphlet, "Operational Requirements for Automated Capabilities" [5]. This Department of the Army Pamphlet will provide the Army manager with procedures to develop a User Functional Description (UFD) which will reflect the software in terms of the end user's perspective. The DA Pamphlet will also define procedures to facilitate alternative software acquisition strategies, such as rapid prototyping.

The User's Functional Description provides guidance on identifying, analyzing and documenting operational requirements for automated capabilities; it forms the foundation for the traceability of system requirements to the software specifications. As depicted in Figure 8, this allows determination of the: (1) criticality of faults detected; (2) development progress in terms of user capabilities; and (3) adequacy of testing for managing operational risks. The user

Figure 7: The Army Software User Has Many Expectations from the Product

Figure 8: The UFD Allows Tracing of Operational Requirements to Software Specifications
provides enough definition to bound the problem, but not so much as to inhibit the design effort. The immediate benefit is the ability to tell if the software does what the user needs. The long-term benefit is that the same guidance is provided to both the software designer and the tester and evaluator.

**Software T&E Improvement – Continuous Evaluation**

STEP has also proposed a set of procedures to improve the current Army software development and T&E process [6]. STEP proposed that the software T&E process be improved by ensuring software product and process evaluation throughout the software life cycle. The STEP recommendation is to apply the established Army concept of Continuous Evaluation (CE) to the software development process. CE is the process by which all available data sources are used to develop evaluations and to report on the operational effectiveness and suitability of a system, from concept definition through deployment. The process includes evaluation of training and life cycle support requirements, operational concepts, assessment of requirements definition, and analysis of all technical, user and contractual test and evaluation results. The objective is to assess progress of the software toward satisfying operational requirements. Additionally, CE includes verification of fixes through Follow-on Operational Test and Evaluation with intensive follow-up, including post fielding data surveys.

We commonly find that software which has undergone a lengthy development process does not work properly when it is delivered to the user. This is similar to a student’s going through 12 grades of school and surprising everyone with low SAT scores. To avoid such surprises, the school system uses a series of regular report cards, as shown in Figure 9. This process of continuous student evaluation allows early detection and correction of problems.

In the Army acquisition system, an independent evaluator monitors the developing product under the Continuous Evaluation concept. The evaluator provides reports at each of the system reviews, as shown in Figure 10. Continuous evaluation is a process which recognizes the current state of software maturity, and implements early evaluation of all sources of information. This allows the Army Program Manager to identify problems in time to make corrections.

Software evaluation in an evolutionary acquisition places increased emphasis on early design and risk analysis under the CE concept. The lack of a complete software requirements specification requires a test activity to continuously monitor the evolution of software design against currently projected user requirements.

Software CE entails continuously measuring software maturity and its readiness to perform the proper user functions.

![Figure 9: Report Cards Provide Continuous Evaluation](image)
in the operational environment. In an evolutionary acquisition the evaluator cannot check formal test results against a detailed software specification. With an evolving technical baseline, CE must provide certification of the effectiveness and suitability of the software product at any point during development.

**Software T&E Improvement – Continuous Measurement**

The STEP also recommended a mandatory set of software metrics to allow continuous evaluation during the life cycle. The mandatory set of 12 metrics, shown in Figure 11, will be collected in each Army software development program. The Army metrics effort is supported by development of a central database for Army-wide metrics collection which will provide baseline status indicators of software requirements, management, and software quality.

The immediate benefit of metrics application in an evolutionary acquisition is that managers will have a better understanding of progress and convergence to system requirements. A long-term benefit of reporting metrics is that they can improve milestone reviews and enable decisions to be based on facts. Eventually our understanding of how standard metrics correlate with various software developments can become the basis for planning tools.

The overall process of software development would be supported by management through measurement. Figure 12 shows how current project metrics can be compared with both the project plan and the corporate knowledge from the historical database. If deviations are out of norms, the program can be revised to prevent significant operational failures before the system is put in the hands of users for either operational test or fielding.
Management Metrics
- Cost
- Schedule
- Computer Resource Utilization
- Software Engineering Environment

Requirements Metrics
- Requirements Traceability
- Requirements Volatility

Quality Metrics
- Breadth Of Testing
- Depth Of Testing
- Fault Profiles
- Design Stability
- Reliability
- Complexity

REFERENCES
3. US Department of Defense, Joint Chief of Staff Publication #1 (JCS Pub1), "DOD Dictionary of Military and Associated Terms".

Figure 11: STEP Has Developed a Set of Metrics Which Allow Continuous Measurement of Software

MANAGEMENT THROUGH MEASUREMENT

Figure 12: Metrics Allow Continuous Evaluation Based on Facts