Major Automated Information System (MAIS) Best Practices

Introduction

DOT&E oversees operational testing of 30 DOD Major Automated Information Systems (MAIS) programs.[1] Many MAIS program managers find it challenging to meet cost, schedule, and performance goals. The U.S. Government Accountability Office (GAO) reported in 2014 that, "most selected [MAIS] programs changed their planned cost and schedule estimates, and over half did not fully meet system performance targets."[2] The same report stated that of the 15 MAIS programs the GAO studied, "three of the selected programs reported meeting system performance targets, while eight reported not fully meeting targets, and four did not have system performance data available." All of the 15 programs that GAO reviewed are on the DOT&E oversight list, and DOT&E has gained unique insights into MAIS programs through operational testing.

The purpose of this section is to identify best practices in MAIS acquisition and provide examples of how those were implemented by the systems under DOT&E oversight. The DOD acquisition workforce has sporadically implemented many of the best practices for MAIS programs. A wider, more consistent application of the best practices described in this section, including implementation of an agile acquisition framework, should help DOD more frequently deliver successful MAIS programs that perform well during operational testing and in the field.

Challenges

The challenging nature of MAIS acquisition can be attributed to many factors, but software acquisition reference materials often cite complexity and unstable requirements as the most significant.

- Program complexity. DOD MAIS programs tend to be very complex. Typical MAIS programs have to be integrated into multiple existing enterprises that contain large numbers of interfaces with government and commercial entities, each with its own configuration, database structure, and security requirements. In addition, the program itself most often is an integration of large numbers of commercial off-the-shelf (COTS) and government off-the-shelf (GOTS) components with existing military and commercial networks. This complexity is often paired with an acquisition strategy that requires delivery of a full, mature product in a single development cycle, which often results in delays and performance shortfalls.
- Unstable requirements. DOD systems often have to deal with changing requirements. In many cases, the changes are driven by advancement in technology (e.g., vendors updating hardware, operating system, or database versions) and the program office must either pay sharply increased costs to continue the support or move to the new version with associated changes. At other times, world events and doctrine changes drive the requirements to change (e.g., a system that was intended for use in conventional warfare may need new functions to be used in counterinsurgency warfare). In either case, changes in requirements necessitate changes in software, causing disruptions in the development cycle.

Best Practices

These challenges may be mitigated through MAIS program best practices. In the process of overseeing the operational testing of systems under DOT&E oversight, DOT&E noted the following 10 practices that produced observable benefits to the programs.

Robust Senior-Level Participation

Robust and continued senior-level attention and participation contributed significantly to the success of agile acquisition MAIS programs like the Army's Logistics Modernization Program (LMP), Global Combat Support System – Army (GCSS-A), and GCSS – Joint (GCSS J). Senior leader support was key for securing necessary resources, enforcing updated business processes, and shortening decision cycles.

- Resource help. Agile programs tend to have relatively short delivery cycles. This often means short development-test-deployment cycles. Executing such agile cycles is resource-intensive for the entire acquisition team. A typical agile program deploys an approved release, develops the current release, and plans for the next release, all at the same time. To support such concurrent acquisition cycles, testers must simultaneously prepare evaluation reports from the last release, execute and witness test events for the current release, and conduct risk assessment and plan test events for the next release. One test team usually cannot adequately plan test, and report simultaneously. To reduce the burden, the GCSS-J Program Office provided sufficient resources to form two

[1] Section 2445a of title 10, U.S. Code, defines a MAIS program as a DOD information technology (IT) investment with: 1) program costs in any single year exceeding $32 Million; 2) total program acquisition costs exceeding $126 Million; or 3) total life-cycle costs exceeding $378 Million (all in FY00 constant dollars). DOD Instruction (DODI) 5000.02, "Operation of the Defense Acquisition System," dated January 7, 2015, updates the dollar figures to FY14 constant dollars: 1) $40 Million in any single year, 2) $165 Million total program cost, or 3) $520 Million total life-cycle cost. The Secretary of Defense and the Milestone Decision Authority can also use discretion to designate a program as a MAIS.

[2] GAO report GAO-14-309, "Major Automated Information Systems: Selected Defense Programs Need to Implement Key Acquisition Practices," March 27, 2014, page 16

test teams so that each team could alternate and focus on one release at a time.

- Enforcement of updated business processes. Users tend to be comfortable with the business processes or tactics, techniques, and procedures (TTPs) they have been using. Unfortunately, new TTPs and business processes are inevitable with significantly new capabilities for a couple of reasons. First, new software often will not support established business processes and TTPs without customization, and the risk in a MAIS program tends to correspond to the amount of customization. Customization can cause deviation from the initial design of the COTS and GOTS software. Such a change necessitates not only new code writing, but also may change the way the software interfaces with other systems or modules. Second, the use of outdated business processes and TTPs increases the risk of not using the new software to its maximum value. The advantages of automation are eliminating manual steps and reducing human decision points. Some users might resist such automation, but avoiding automation can negate the benefit of the new technology. Thus, once decision-makers agree there is a need to change TTPs and business practices, they must help implement them by enforcing their use and providing the necessary resources for training. The Army's LMP performed well during its recent operational test in part because of the rigorous user training the program manager provided well prior to the test.
- Shortened decision cycles. The acquisition process for MAIS programs require OSD-level decisions, which can often mean lengthy staffing processes. This is very difficult for programs that deploy more than one release per year. Many programs successfully developed a model where they adequately informed decision-makers without lengthy staffing processes. One such method is simultaneous staffing of acquisition decisions vice a step-by-step iteration of signature process. This method is not always practical, but can work well if senior-level leaders participate in the acquisition. For instance, LMP Increment 2 grouped seven releases into three waves. Each wave grouped one to three releases based on a risk assessment. The acquisition decision makers made production and fielding decisions for waves rather than individual releases. This way, decision makers still managed risks without excessive, time-consuming staffing processes.

Flexible and Disciplined Requirements Management

Program sponsors for the majority of MAIS programs document their requirements with the Joint Capabilities Integration Development System "IT Box" model. With the IT Box, requirements are specified in an Information System Initial Capability Document (IS ICD) and Information System Capability Development Document (IS CDD).[3] The program sponsors describe more details of the IS ICD and IS CDD requirements in Requirements Definition Packages and further define the capability for each release in Capability Drops.[4]

One advantage of agile acquisition and the IT Box is the flexibility to adjust the priority and urgency of requirements. Program sponsors document requirements at the beginning of the acquisition program when the software developers and users know only a rough outline of the program. As the system matures, users and developers might realize some of the requirements are not consistent with the best use of the system's capabilities. The threats or the doctrine may change, and in response, the program may need to develop a capability earlier than originally planned. A software module might encounter significant challenges that could ultimately influence the acquisition timeline. In such cases, the IT Box provides the requirement governance body with the authority to decide whether to leave that capability for a future release, or to add resources to complete that capability.

Many MAIS programs implement commercially available agile framework products. Most agile frameworks state requirements in terms of user stories, which are a small segment of functionality that a user wants. The capability to execute a user story is delivered in a sprint, or a small segment of software. The user stories are combined into an epic, which is a larger description of how the user intends to use the system. The capability to execute the epic is delivered in a release composed of multiple sprints.

Compared with typical requirements in a system specification such as "system ABC must be able to perform XXX task within YY seconds," epics and user stories provide a more operational context such as "the user must be able to receive X input and produce Y product in time to support Z task."[5] The user story not only provides performance goals for each task, but also provides operational context of how those tasks work together to produce a desired outcome.

A user story allows the program sponsor to frame a feature in terms of its benefits for a particular user. A well-written user story helps developers design software that delivers specific benefits. A pitfall a program can easily fall into is breaking epics into tasks rather than user stories. In those cases, development and testing processes becomes task-focused (doing things) instead of delivery-focused (creating value). For a coherent and consistent understanding of requirements in operationally relevant terms, it is important to describe requirements in terms of value to the user rather than tasks; e.g., a user story should be, "user must update unit location before the next planning update cycle," rather than, "user must be able to update the unit location in less than 4 seconds." This way, developers and testers can both understand the importance and operational consequence of each step.

---

[3] Manual for the Operation of the Joint Capabilities Integration and Development System (JCIDS), February 12, 2015, page D-29

[4] Ibid., page D-34 and figure D-4

[5] Defense Acquisition University (https://dap.dau.mil/glossary/Pages/2752.aspx) defines system specification as "a description of the system-level requirements, constraints, and interfaces (functional, performance, and design) and the qualification conditions and procedures for their testing and acceptance. The System Specification, initially reviewed at the System Requirements Review (SRR), ultimately becomes part of the functional baseline that is confirmed at the completion of the System Functional Review (SFR)."

For the Distributed Common Ground System – Army (DCGS-A) FOT&E, DOT&E evaluated the system primarily based on the user's ability to execute "vignettes" – a series of user actions that accomplishes the mission. For instance, one of the vignettes required the brigade equipped with DCGS-A to identify a facility that manufactured IEDs, and locate and designate the facility to be targeted. The Army program sponsors developed 10 such vignettes for FOT&E. The program sponsor, in concert with combat developers and the brigade, further divided the vignettes into steps for specific DCGS-A users.

Change Management that Starts Early and Continues Throughout the Process

Military users cannot always adapt to commercial practices. In such cases, the program office should work closely with the users to refine business processes. For example, the GCSS – Marine Corps (GCSS-MC) Program Office spent many months with system designers and tactical users, exchanging ideas and designing new business processes that retained the power of new software while accommodating specific military requirements such as limited bandwidth on the move, limited ability to carry heavy hardware, and unit personnel changing over with military rotations. The process was iterative; approved procedures did not always work out the way users and engineers expected. In such cases, users and engineers needed to retune business processes and software to accommodate the military missions.

After deploying the new software, the GCSS-MC Program Office fielding team worked extensively with users during the fielding process so that individual adjustments could be made for specific users. Similarly, another program, GCSS-J, coordinated early with the users to describe their workflow in terms of user stories, and continued dialog with the users after fielding to make requested changes. Such adjustments can be as simple as redesigning the look of the display and writing patches to adapt the software. In some cases, extensive adjustments ended up as a new function to be delivered in the next available software drop, pending approval by decision-makers.

Architecture Description in Accordance with the DOD Architectural Framework

A well-designed and sufficiently detailed architecture is a prerequisite for effective development and employment of enterprise software. This is no different than needing a detailed blueprint for a building before construction and for maintenance. The more complex a program is, the more the developer and maintainers need the architecture description. The DOD architectural framework provides an outline for documenting the architecture.

Sufficiently detailed workflow information (as provided in the system view and operational view architectural products) should be coordinated with users to develop user procedures and training. Such coordination allows discussion regarding how the system can be integrated into user's doctrine and procedures, or to modify the doctrine, procedures, and user training to take advantage of the technology.

During the development and sustainment phases, the program office should update architectural products to ensure consistency with user procedures and updated interfacing systems. The updated architecture should also remain consistent with user stories that describe the updated procedures and interfaces.

Mature Doctrine and Training Development

It is easy to fall into the trap of mistaking the purchase of tools with providing solution to a problem. In reality, tools do not help the user unless users know how to use the tools to accomplish the mission. For DOD systems, successful programs tend to have doctrine that describes how the system fits into the overall military operations. The doctrine in turn becomes the basis of developing TTPs that describes in more detail how the users should employ the functions the system provides. The doctrine and TTPs then should be integrated into a training program so that users have necessary knowledge to operate and maintain the system.

- TTPs. While the program manager should make the transition to a new MAIS program as seamless as possible, the reality of automation and optimization can demand change in the way the military does things. For instance, whereas the old process may have been to place an order for a part first and have the financial office check that order against available funds second, the new software may pre-check the funds balance as a part of processing the order. To take advantage of new capabilities, system sponsors and users must develop and train doctrine and TTPs. GCSS-A incrementally fielded capability with sufficient time to develop the TTPs so that the users received systems with clear instructions on how to use the system to accomplish the mission.
- Training. User training for new system capabilities should include not only how to do an individual task, but also how to work with the new capabilities as a team. The training must include sufficient practice sessions to get used to new TTPs and for each unit to develop its own operating procedures. The DCGS-A Program Manager dedicated almost a year to gradually increasing the scope of training, starting with individual training and culminating in a brigade free-play training exercise.

Iterative Developmental Tests that Start Early

MAIS programs typically have one prime vendor that integrates hardware and software components from multiple vendors. The program office should have a coherent strategy to find and fix problems as each software component is developed and delivered, because software engineers can find and fix problems more quickly before a software module is integrated into a larger and more complex program. Isolating the root causes of a problem can be very difficult after the software has been nested with other vendors' products. In addition, the prime vendor may have to redo the integration work after receiving an updated software module.

Database Interfaces and Commonality

MAIS programs typically ingest data from multiple sources to produce new database products. If data sources provide inaccurate data, the resulting product will be inaccurate. The program may not be able to ingest the data if a data source provides data in a different format. To minimize such risks, the

LMP Program Management Office (PMO) conducted trading partner test (TPT) as well as process and data integrations test (PDIT) events before government developmental test (DT) and operational test (OT) events. The TPT ensured interfaces with trading partner systems worked as intended, and the PDIT ensured that the end-to -end processes worked well. Many programs do adequate interface tests that are similar to a TPT, but they neglect to test an entire process as done in the PDIT. An early test of process and data in a controlled environment makes it much easier to identify and fix root causes of any discrepancies. The TPTs and PDITs provided the LMP PMO early opportunities to discover shortfalls and implement necessary adjustments.

The LMP PMO put management focus on data integration. Conducting PDITs before DT and OT events helped ensure LMP was ready to ingest and use accurate data from the data sources. The PDITs helped LMP avoid one of the most common causes for logistics system failures: nomenclature inconsistencies. For instance, when a user needs to know how many M1A1 tanks are in the unit's inventory, the database should be capable of counting all M1A1s. Unfortunately, one database may call it M1A1; another database may call it Abrams Tank; and another database may call it "tank, main battle, armored." Even worse, some databases may track the data at the component level (such as engine, transmission, or gun mounts) rather than the platform level such as M1A1. Given the variety of source databases, the LMP database manager had to first correlate all of these terms with a common term before the system could return an accurate count for the query. Even when the database manager succeeds in this difficult task, if the database manager is not careful, a query for "Abrams tank" may count all of the M1A2s as well as M1A1s. If the intent was to count M1A1s, the count would be wrong. The database manager must find a way to work with all of the existing databases and either build interfaces or modify databases. LMP managed this challenge by conducting well-designed, two way data integration tests to identify and fix the interface issues.

DCGS-A is an intelligence system that exploits intelligence, surveillance, and reconnaissance data to produce actionable intelligence. The system accomplishes this through an intelligence fusion process that combines information from a large number of sources. The fused intelligence can only be as good as the accuracy of the data it uses. The Army quickly found that synchronizing databases is a daunting challenge and created the Tactical Entity Database (TED) that combines and organizes data from hundreds of sources into specific entities. An entity may be a person, building, organization, or equipment. By organizing large and disparate information into a coherent database, information can be correlated and associated so that an analyst can get a clear picture of what is in the unit's area of responsibility.

Even after the creation of TED, DCGS-A had more database challenges to overcome. In unconventional warfare, the database has to record many items that do not have standard nomenclatures, or item names. An example is a brand new type of IED. For some purposes, such as route planning, the unit would find it more useful to group all such devices as IEDs. For other intelligence purposes, the unit may need to identify specific types of IED, and must create a new item description to document that type of IED. The new nomenclature needs to be designed so that DCGS-A can still recognize it as an IED when a user queries for total number of IEDs. In addition, the creator of the new nomenclature must ensure all other DCGS-A users are aware of such item description. The Army conducted extensive unit-level training to define and teach when to create new nomenclature, how to create the nomenclature, and how to share the new nomenclature with other users.

DCGS-A followed the intelligence fusion process that begins with the fusion level 0, or "Normalization," step. Normalization is the process where DCGS-A users enter data from multiple sources into TED. If a soldier reported seeing a truck with a machine gun mounted in the back, the data entry person would first look to see if such an item is on the pull-down menu. If not, the data entry person must decide whether to create a new item or call it the most similar item such as armored personnel carrier with machine gun. This step determines the value and accuracy of all processes that follow.

DOT&E evaluated DCGS-A to be not operationally effective after the IOT&E in 2012, but evaluated the system to be operationally effective after the FOT&E in 2015. Many factors contributed to the difference, but one of the most significant improvements was TED. A major contributing factor was that the Army conducted a series of extensive training events, including unit-level training, so that the unit was able to develop and train with detailed procedures and processes.

Database accuracy and currency cannot rely on software solutions alone. Proper data integration and interfaces tend to be the most accurate predictors of program success for networked MAIS systems. Accordingly, program managers should first identify and document all database and interface requirements in architectural products, monitor progress via interface and data integration tests, and implement procedures and training programs to ensure users maintain the databases properly.

A Robust Developmental Test with Operationally Representative Interfaces and Networks

Automated developmental testing is critical to gain efficiency and accuracy. Automated acceptance and regression tests provide an efficient and reliable option to verify that a code change works as intended without breaking anything. However, program offices must avoid using automated testing as a replacement for a comprehensive DT. Automated testing is a prerequisite step to make sure coding is done correctly; it is not a validation of the software's ability to support the user's mission.

Many complex MAIS programs perform well in DT and fail to perform in OT. Two contributing factors cover the majority of the difficulties seen during OT:

- Network connectivity and congestion. Most DT labs use a hardwired network with unlimited bandwidth, but during OT the system uses a tactical network with limited bandwidth. The limitations can cause the network to time-out, resulting

in a system failure. DT labs should emulate the expected operational networks as accurately as possible and simulate tactical network bandwidth, connectivity, and congestion.
- Interfacing systems. Each of the interfacing systems may have peculiarities which are not well understood during DT. Operational interfaces may have software patches to compensate for problems experienced during operation and thus work differently from the initial design. These differences might be enough to cause the system under test to fail to support the user's mission. DT labs should have the latest versions of the key interfacing systems and use as much operationally realistic data as possible.

Persistent Maintenance of the Cybersecurity Plan of Actions and Milestones

An enterprise network requires MAIS programs to interface with multiple outside programs, which often include commercial systems. Allowing such connections is inherently risky from a cybersecurity perspective, and often makes it impossible to eliminate all vulnerabilities. Thus, it is important to identify, document, and continue to monitor those risks. A cybersecurity Plan of Actions and Milestones (POA&M) is the best tool to identify and document cybersecurity vulnerabilities and the mitigations for them. The POA&M should clearly identify all of the vulnerabilities by priority and urgency, the proposed corrective actions, responsible organization and person, and the milestone to achieve correction. It should include vulnerabilities associated with interfacing systems, and should not be a document that is approved once and put away; the threats are dynamic, as are the network environments.

Continual awareness of emerging cybersecurity threats, realistic adversarial testing of the system against those threats, and implementing mitigations for vulnerabilities should be an ongoing process supported by decision-makers with the authority to require corrective actions. With appropriate leadership's focus, MAIS programs with extensive cybersecurity vulnerabilities have successfully resolved them. For example, the Navy's Consolidated Afloat Networks and Enterprise Services (CANES) program had hundreds of significant cybersecurity vulnerabilities as it entered into IOT&E, but successfully tracked and fixed a sufficient number of them to be more secure against cyber-attacks. The CANES program will have to continue to maintain its POA&M to discover and fix cybersecurity vulnerabilities as the threats and the network continue to evolve.

Thorough Tracking of Software Problems in a Comprehensive Database and Senior-Level Review of Priorities

Agile development requires decision-makers to quickly modify the priority and urgency of functions from one release to another. For the decision-makers to make an informed decision on a short decision cycle, they need to understand the development status and challenges. Even within the release cycle, decision-makers may have to change the amount of resources devoted to a particular function. Therefore, the decision-makers need to know the number of open software problems by criticality and urgency, as well as the time and resources needed to resolve software deficiencies. If correcting a problem requires a long time and interferes with the fielding schedule, decision-makers should consider mission impact against the time and resources required to fix problems. This will help to decide whether to defer the delivery to the next release or rearrange resources to more quickly solve the problem. Both GCSS-A and LMP have good processes for senior-level Army leaders to review and prioritize fixes to software problems based on user input.

Implementing Best Practices through Agile Acquisition

The best practices identified in this report can help to improve the success of MAIS programs and should be applied broadly. In order to maximize the effectiveness of these practices, DOD should pursue the agile acquisition approach. Incremental software delivery is one aspect of agile acquisition and has already been implemented with some success. However, DOD can do more to accommodate agile software development. Using proven commercial agile frameworks is a good way to systematically integrate the best practices.

Incremental Software Delivery and Agile Acquisition

To overcome challenges associated with program complexity and requirements instability, DODI 5000.02 includes an acquisition model suitable for incremental software delivery.[6] Compared to a traditional "waterfall" model, where all of the functions are developed and delivered in one lengthy and monolithic acquisition cycle, incremental delivery allows each increment to focus on a selected set of functions, which reduces complexity. In addition, each increment takes a shorter time, and thus reduces the chance of requirement changes.

In a 2015 report, the GAO claimed:

> About half of the [selected 20 MAIS] programs that met or planned to meet this condition had been positioned to do so because they had been restructured and split into smaller, incremental programs, which is consistent with a Defense Science Board recommendation, Office of Management and Budget (OMB) guidance, and a statutory requirement to use incremental contracting to the maximum extent practicable for major IT acquisitions.[7]

However, working on multiple software releases, which often overlap, brings its own set of challenges – including difficult coordination among the key stakeholders and increases in redundancies and resource requirements. To help overcome these challenges, many MAIS programs adopted agile acquisition.

Agile acquisition (also known as agile software development) is an approach to software development that is built around a set of guiding principles established by the nonprofit Agile Alliance. This approach's practices and methods are in large part intended to improve efficiency, responsiveness to changing needs, and quality. Essential elements of agile acquisition include:
- Delivering working software quickly and improving/adapting it incrementally in frequent releases

---

[6] DODI 5000.02, page 11, paragraph 5c(3)(d)

[7] GAO report GAO-15-282, "Defense Major Automated Information Systems: Cost and Schedule Commitments Need to Be Established Earlier," February 26, 2015, page 15

- Collaborating directly with users
- Minimizing governance processes

Agile acquisition is only appropriate after the basic infrastructure is in place. While agile acquisition gives flexibility for adding or enhancing functions and applications, building a network infrastructure requires a deliberate and logically sequenced plan. For most DOD MAIS programs, network infrastructure is so complex and interrelated that there is not much flexibility, and this lack of flexibility nullifies the benefit of agile acquisition. A large system may have an infrastructure software component that is necessary for verification testing of other system components.[8] A program should have a working infrastructure that satisfies the Information Exchange Requirements and network protocol requirements, and have a sufficiently detailed architectural description to ensure each software module fits into the overall enterprise.

Additionally, a MITRE report advises:[9]

> … it is absolutely critical that the development of the architecture precede sprint development.[10] Alternatively, a program can initially use a traditional approach to build the initial increment that meets the baseline architecture requirements. Once the program has established the baseline and framed the overall conceptual design, program managers can consider shifting to an agile approach for subsequent increments that build additional functionality into the operational baseline.

For instance, DCGS-A and DCGS-Navy first delivered stable infrastructure with Increment 1, and are now moving to agile acquisition for Increment 2. In both cases, the first phases of Increment 2 improve data infrastructure before adding newer applications.

Implementing a Proven Agile Framework Product

Most successful commercial software developers use proven agile software development framework packages. Popular agile development framework products include Scrum, Extreme Programming, and Scaled Agile Framework (SAFe). These products systematically incorporate the best practices discussed in this section, and make it easy for MAIS programs to implement good ideas from both government and commercial developers. Scrum and SAFe are the approaches most often implemented by MAIS program managers.

The agile acquisition frameworks share common attributes: an integrated team approach that integrates users, developers, and testers; flexible management of requirements priority and urgency; small segments developed and tested before combining into larger segments; and many concurrent activities.

While the commercially available agile frameworks help build good acquisition structure, leaning how to use the frameworks is not easy. The program office needs to plan sufficient resources to train acquisition stakeholders. Air Force DCGS is starting to implement SAFe for its Open Architecture development and has heavily invested time and resources to train not only the program office, but everyone in the acquisition community – such as requirement owners, testers, and program sponsors. Such training is essential for the team approach; it is impossible to collaborate until everyone shares a common language and frame of reference.

---

[8] Carnegie Mellon University, Software Engineering Institute report, "Considerations for Using Agile in DoD Acquisition," 2010

[9] The MITRE Corporation technical paper, "Defense Agile Acquisition Guide: Tailoring DoD IT Acquisition Program Structures and Processes to Rapidly Deliver Capabilities," March 2014

[10] A "sprint" is a regular, repeatable work cycle in agile methodology during which work is completed and made ready for review.