Delivering Military Software Affordably

Christian Hagen ■ Jeff Sorenson

Much of the weaponry now used by the U.S. military—advanced warplanes, drones, smart bombs, autonomous vehicles—is driven by software. In the future, the capacity of the Department of Defense (DoD) and the military commands to defend our country will depend more on their ability to develop the best software rather than on the physical design chosen for the weapons. Like it or not, the DoD now is in the software business.

As one Air Force general makes clear, the military forces and DoD have reached their limit on improving the capabilities of our warplanes and other weapons systems by simply changing their physical design. "The B-52," the general explains, "lived and died on the quality of its sheet metal. Today our aircraft will live or die on the quality of our software."

This increased software demand reflects the DoD's need for advanced operational capabilities and requirements. In the future, fighter jets will have to be even faster, more maneuverable, with split-second response. Drones and other remotely piloted aircraft will require greater accuracy and controllability. GPS-guided smart weapons will need more advanced software, continually updated, to remain smart.

As software has become more important, the DoD has begun to see it as a strategic weapon on which its success relies. But to shift emphasis from hardware to software, the DoD must change its perspective, processes, and capabilities if it is to avoid the increasing costs of developing, modernizing, and sustaining

Hagen *is a partner in A.T. Kearney's Strategic Information Technology Practice and is based in Chicago. Hagen advises many of the world's largest organizations across multiple industries—including government and defense contractors.* **Sorenson** *is a partner in A.T. Kearney's Public Sector and Aerospace Defense Practice based in Washington, D.C. Sorenson, the former chief information officer of the U.S. Army, retired as a lieutenant general with more than 20 years of acquisition experience.*

software. So far, this effort, while seeing some success, has been plagued by failures, cost overruns, program delays, and cancellations. Look at the following two programs, for example:

F-35 Joint Strike Fighter. The F-35 already has cost the DoD billions of dollars and has surpassed its delivery date by several years. No definite timeline is yet set for when the Navy, the Air Force, or the Marines will receive a fully operational version of the plane. In the meantime, costs continue to rise. The Pentagon recently confirmed the F-35 program's estimated development and sustainment costs are likely to be $1 trillion over the aircraft's 50-year projected life.

These skyrocketing costs and delays are caused, in part, by the overall size and complexity of the F-35's software. When the JSF first was tested in late 2006, the estimated number of operational source lines of code was about 6.8 million. Recent estimates put the operational plus support lines of code at approximately 24 million (see Figure 1, on p. 32).

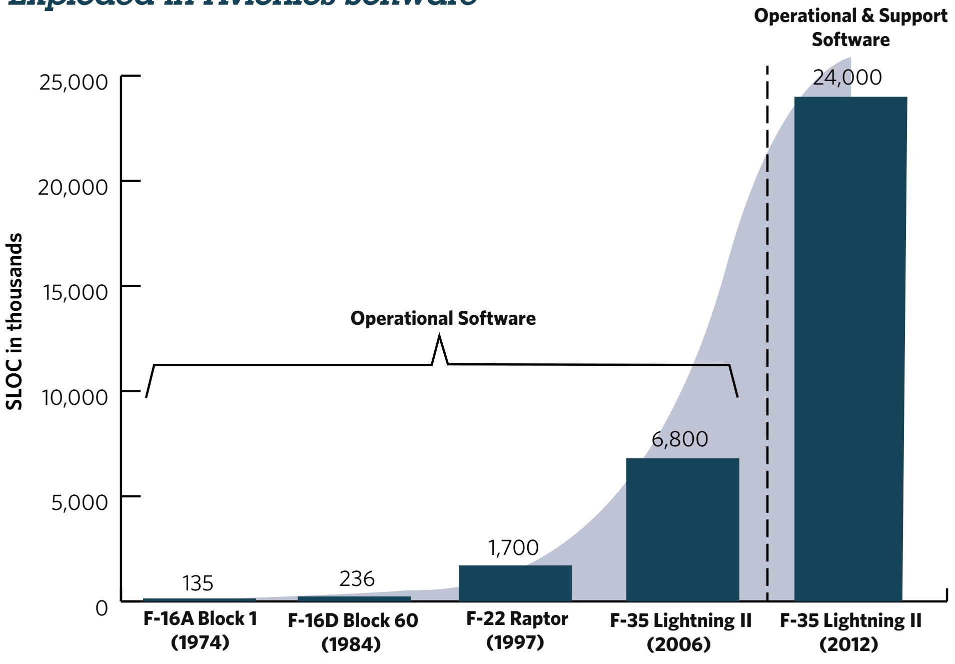
Expeditionary Combat Support System. In December 2012, the Air Force canceled this system after 8 years of development and costs of more than $1 billion. The software program, designed to manage Air Force logistics, required comprehensive change to the "processes, tools, and languages of all 250,000 people in our business at once," according to Air Force director of transportation Grover Dunn. The program was canceled when this change became unmanageable, leaving the Air Force to rely, in part, on its outdated 1970s-era logistics systems.

In June 2012, the Air Force Scientific Advisory Board released a report that highlighted the growing role of software in sustaining aircraft for the long term and noted that software's utility and complexity have grown faster than the Air Force's ability to address it across a system's life cycle. So while hardware sustainment costs will decrease as the USAF reduces the number of aircraft, software sustainment costs will not decrease because a weapon system generally needs the same software sustainment, whether it is a fleet of 10 aircraft or 1,000 aircraft.

As the DoD, the military, and their contractors attempt to build needed software capabilities into our weaponry, they must contend with increasingly complex system requirements and a shrinking budget. This challenge makes the transition to software extremely difficult for modernizing systems and managing sustainment.

The challenge appears even more daunting when one considers the technological, programmatic, and enterprise barriers that, over the next few years, will impact the already dynamic defense and software landscape. A few of these many barriers are the following:

Figure 1. The Number of Source Lines of Code (SLOC) Has Exploded in Avionics Software



Note: SLOC for F-16 and F-22 are at first operational flight. F-35 SLOC figures are from first test flight and current estimates/sources.
Source: P. Judas and L.E. Prokop, "A historical compilation of software metrics with applicability to NASA's Orion spacecraft flight software sizing." Innovations in Systems and Software Engineering, vol. 7 issue 3, September 2011, p. 161-170
Andrea Shalal-Esa, "Pentagon focused on resolving F-35 software issues," Reuters, www.reuters.com/article/2012/03/30/lockheed-fighter-idUSL2E8EU8C420120330, March 2012
Robert N. Charette, "F-35 Program Continues to Struggle with Software," IEEE Spectrum, http://spectrum.ieee.org/riskfactor/aerospace/military/f35-program-continues-to-struggle-with-software/, September 2012

- Application of open architecture paradigms
- Management of self-healing software
- Development of a cyber warfare strategy
- Exploration of mission adaptable programs
- Introduction of fully autonomous vehicles

Regardless of the magnitude of this challenge and its barriers, the overriding considerations must be around software affordability. Few would doubt that together the DoD, the military, and their contractors have the capabilities to develop the software-driven weaponry needed for future combat. But can they develop and maintain this software in an affordable manner?

As the role of software continues to grow, the DoD must drive savings and efficiencies in the way its software is designed and maintained, ensuring that needed savings are realized and performance is preserved. Bringing about these objectives requires the mastery of four critical areas: architecture, commercial software, software should-cost analysis, and organic software sustainment.

Architecture

To achieve the best advantage from its software architecture, the DoD must address two key issues at the start of the design and planning process and then reassess them throughout the maintenance phase. The first issue is how to design the architecture to deliver the performance and capability needed. The second is how to develop that architecture to be sustainable and drive effective maintenance programs and costs going forward.

Designing software architecture against specific capabilities has tight restrictions since many DoD applications cannot tolerate compromise on the ability of a weapons system to perform its objectives. However, several architecture choices are available for reducing the costs of achieving that capability, while also providing cost-effective maintenance and enhancements. For example, the ease or difficulty of enhancing or modifying a weapons system's software is strongly linked to how integrated and tightly coupled the architectures are designed. By starting with a modular and loosely independent design, the DoD can perform maintenance and enhancement efforts more cost efficiently.

Rather than select a tightly coupled and integrated architecture, which requires more effort to develop and maintain across the software development life cycle, the DoD could select a federated, open, or open-federated architecture. A federated architecture's overall effort and associated costs of software design are half or less than those of an integrated architecture. With open architecture's ease in adding, upgrading, and swapping components that conform to agreed-upon standards, developers across multiple contractors can work at the platform level to achieve reduced efforts and costs. Combining the first two approaches into an open-federated architecture leads to an even greater level of efficiency and cost savings, while encouraging development teams to modularize their code and compartmentalize their efforts.

Commercial Software

The custom software traditionally used by the DoD is both expensive and time consuming to develop, especially when compared to software developed by the private sector for commercial uses. For this reason and others, the DoD is turning to commercial software for noncritical, and some critical, applications. As a result, it is instilling both battlefield and support systems with the latest capabilities at a cost significantly below that delivered by custom software.

Besides lower cost and shorter time to develop, commercial software offers the DoD several advantages for modifying or retrofitting existing platforms into weapons systems for the future. For example, with it, the DoD can do the following:

- Adopt proven best practices that are already free of potential problems.
- Achieve cost-effective, faster development by relying on a large pool of experienced developers who may be between 50 percent and 100 percent more productive than those proficient with DoD custom development standards.
- Obtain new technology that offers more advanced capability and increased performance—technology made possible by the wider range of tools for coding applications now available for commercial software developers.
- Capitalize on the continual maintenance of commercial software to gain new capabilities with each software release as well as a faster, more coordinated refresh strategy.
- Coordinate upgrades across weapons platforms by using the common architectures, code, and maintenance efforts offered by commercial software.

While the DoD can significantly benefit from using commercial software, its ability to both recognize the shift away from custom hardware and to manage the vendors and solutions making it possible will be critical over the next decade. In addition, it will need to determine the right balance between relying on contractors to drive innovation and using the government's own organic resources to perform standard maintenance.

Software Should-Cost Analysis

Given the complexity of most software development efforts, it's not surprising that even those who are experts at estimating hardware-related efforts often struggle to correctly gauge and manage software development costs. Their inaccurate estimates often prove to be costly in time, money, and lost capabilities.

Estimating total software costs for weapons systems is far more complex than for typical commercial projects, which seldom are completed within budget or provide promised functionality. Military cost and schedule overruns often result from a poor estimate of the software development effort and complexity. Nevertheless, senior DoD and military leaders stress that they can accurately estimate the required work and cost by using standard techniques of software effort and cost modeling and, thus, can avoid the overruns often associated with software development.

Within the DoD, as in private enterprises, a software should-cost review can provide a better estimate than that achieved without such an analysis. A should-cost review—which enables everyone involved to question the traditional ways of doing business and to improve value-chain efficiency— offers important advantages. It can save the DoD millions of dollars on key projects by estimating costs more accurately for the software capabilities being produced, integrated, and tested. Additionally, it can break the cycle of history-based cost estimation and improve transparency and affordability—making clear what the costs of a program should be in an efficient, highly competitive environment.

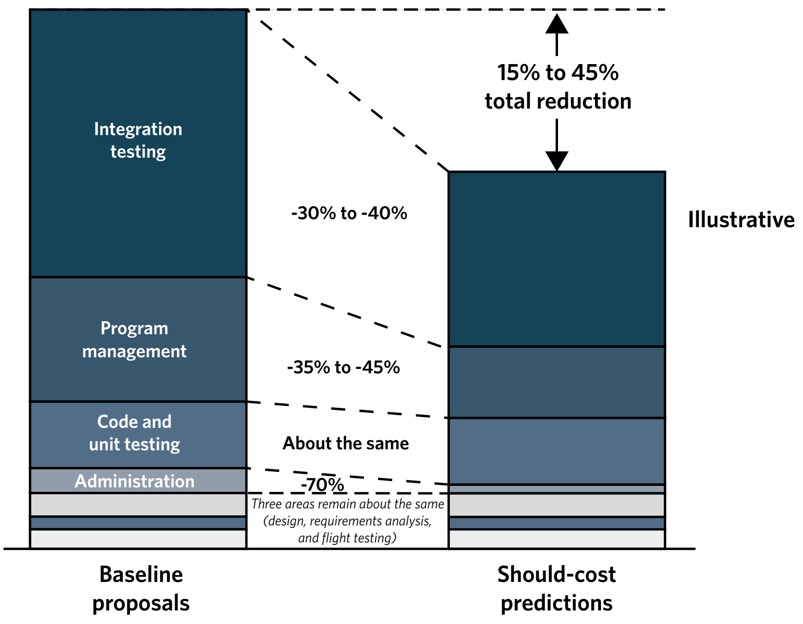
Furthermore, this win-win proposition for both the DoD and its suppliers is a valuable tool for reducing costs without eroding supplier profits or cutting capabilities.

Of course, the DoD cannot save money merely by conducting a should-cost analysis. Instead, it must incorporate the review's results and implement its key initiatives to bring about expected savings. That's why, to make sure the review is not just a theoretical study, a successful should-cost analysis includes a detailed action plan with specific milestones for re-evaluation and an aggressive implementation mindset. It also contains the following five actions for delivering accelerated, maximum benefits.

Bring best practices to bear. The should-cost analysis must be approached with an aggressive attitude for making changes and moving beyond the status quo. Ask "What if?" and "Why not?" Then ask these questions over and over again. Find the best practices that can be adopted for a competitive environment. Focus on comparable and relevant benchmarks as well as determining the most efficient cost-to-deliver program requirements.

Perform a rigorous analysis. Acquire an in-depth understanding of the root-cost drivers and efficiency potential for supply chain, manufacturing, program management, overhead, and other major areas. Detail the savings potential to support the conclusions and drive tangible actions.

Figure 2. Software Should-Cost Modeling Identifies Significant Potential Savings



Source:: A.T. Kearney analysis

Establish the right incentives. The proper incentives—those that benefit both the government and its suppliers—will inspire workers to challenge the status quo and act with appropriate urgency. Therefore, set up incentives to encourage performance that improves suppliers' profits while lowering government costs. Seek a collaborative effort to better achieve the best results.

Translate opportunities into tangible action. Convert cost-management opportunities into realistic action plans with clear timelines and responsibilities. Use multiple cost-cutting approaches, such as negotiation, investment, joint-process improvement, and contract restructuring.

Track performance against the cost-reduction plans. Implement a target assurance program to identify cost-reduction targets and milestones. Review progress regularly to understand performance slips and ensure that mitigation steps are in place. Make sure progress is transparent, credible, and well managed.

Since a should-cost analysis provides insight into cost drivers and forces accountability (especially across contractors and suppliers), it offers a tangible value proposition for avionics software development. With it, the DoD can shed light on the development process, isolate opportunities at the subcomponent level, increase the fact base for negotiation—and reduce the long-term cost of software development by 15 percent to 45 percent (see Figure 2). On the F-22 increment 3.2A program, according to a recent DoD Better Buying Power fact sheet, the Air Force successfully identified and implemented cost-saving initiatives of 15 percent (equaling $32 million) to address areas in the software development process that were above industry benchmarks.

Organic Software Sustainment

With software's growing importance in weapons development and design, software maintenance has taken on a greater role in the post-development work needed to enhance and sustain weapons platforms. This work can be done far less expensively by a stable organic sustainment organization than by primary contractors—in part because wages and overhead are lower, and payback periods are fewer than 5 years. Such cost-cutting opportunities can be found in a number of programs, as the following projected savings illustrate:

- Operational programs, 25 percent
- C4I (command, control, communications, computers, and intelligence), 25 percent
- Ground systems, 20 percent
- Training systems, 20 percent
- Diagnostics and repair, 25 percent

For the DoD to achieve these savings, much of the contractor sustainment efforts must be supplemented with more cost-effective and efficient government-led sustainment. This approach—which follows a recent mandate that much of the sustainment be done in-house to ensure captive capacity during war—will enable the government's organic resources to focus on updating the many legacy platforms now undergoing service-life extension programs. Equally important, it will give contractors the freedom to concentrate on modernizing to keep pace with rapid advances in sensor and weapons technology.

Data rights. If the DoD is to increase its in-house software maintenance, the government must have the appropriate rights to the data and all appropriate personnel must be capable of using them. Achieving these objectives requires that (1) the government earmarks as a critical expenditure the funding needed for data rights acquisition, (2) the DoD increases software data-rights training for all appropriate personnel so they understand the use and relevance of these rights, and (3) the DoD confirms that all acquired data and data formats are usable throughout the system's life.

Skills. Furthermore, the DoD must give its organic resources the capacity and skill to conduct tomorrow's sophisticated sustainment activities. Are these resources ready for the challenge? They probably are more ready than is typically assumed. But to find out, the DoD should evaluate their potential and their organizational readiness to accept future workloads.

Also, DoD should create a capability roadmap that includes plans for building out the specific skills required over the next 5 to 10 years and details the organic actions needed to maintain the weapons systems. Such a roadmap would govern schedules and priorities and ease the transformation required to meet crucial software maintenance goals.

Organic software sustainment organizations are likely to need several key in-house skills over the next 10 or so years if they are to succeed in developing and supporting the critical work of the future. Besides continuing their work (in some instances) in test stand and control, and verification and validation, they will need the ability to handle advanced work in operational flight programs, advanced ground control stations, and command and control systems. The DoD will need software architecture, design, and engineering skills, and will have to use them earlier in the software develop life cycle. And it will need new skills in requirements analysis, functional design, and software architecture.

As software becomes the driving force behind most weapons systems, and the DoD shifts its emphasis away from hardware design, the department is challenged to find the best, yet least expensive, way to develop and sustain its software. By mastering the four key areas of architecture, commercial software, should-cost analysis, and organic sustainment, the DoD can achieve this transformation more efficiently and can deliver modern software-powered weapons systems to our armed forces more affordably.

The authors can be contacted at **christian.hagen@atkearney.com** *and* **jeffrey.sorenson@atkearneypsds.com**.