From Projects to People: Shifting the Software Acquisition Paradigm

Dr. Douglas J. Buettner
The Aerospace Corporation

Lt. Col. Chad Millette
United States Air Force

The amount of embedded flight software is growing at a tremendous rate in the National Security Space (NSS) systems under development by the Space and Missile Systems Center (SMC). Problems with Total System Performance Responsibility (TSPR)-era programs like the Space-Based Infrared System (SBIRS) have been aligned with opinions that the DoD has lost the "recipe" for acquiring complex space systems. The software-intensive nature of next-generation space systems necessitates consideration of a new software-intensive system acquisition paradigm to not only take full advantage of the best people that defense contractors have to offer, but to ensure the ability to engineer and build these systems far into the future.

In October 2006, Lt. Gen. Michael Hamel [1], the SMC's Program Executive Officer, briefed the SMC system software growth trend to the National Defense Industry Association Defense Software Strategy Summit (see Figure 1).

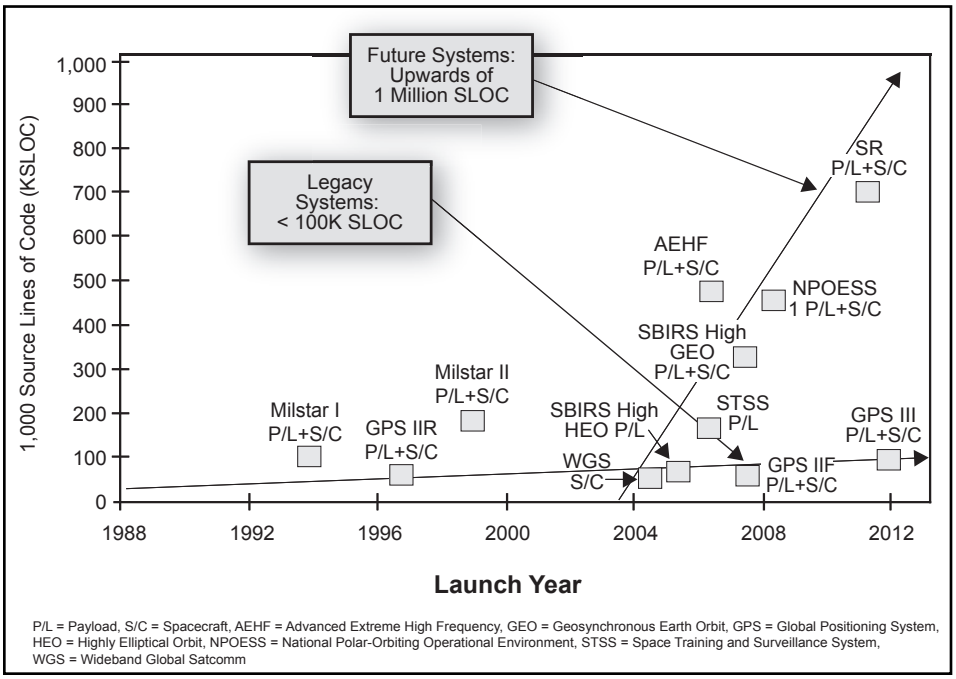
In [2], Buettner and Arnheim described the SMC-wide review of test issues attributed to the TSPR-era acquisition reform policy changes and its impacts on embedded flight software; also provided were space computer technology improvement reasons for the observed growth trend. While the legacy class of vehicles (shown in Figure 1) appear to have a manageable growth trend, the software growth trend for the future space systems (with envisioned systems greater than one million SLOC) is beyond anything our space defense establishment has had to grapple with in the past.

Hamel's presentation supported a broad industry software strategy summit report [3] containing the following recommendations (among others):

- Review and analyze the software engineering, acquisition, and life cycle management initiatives, policies, processes, and plans. This should occur in service branches (Army, Navy, and Air Force), defense agencies, and in other organizations such as the National Security Agency.
- Solicit service branch, major command, engineering center, and Program Executive Office software life-cycle management recommendations.
- Define and publish the DoD's long-term objectives and course of action with associated priorities and resources in a software life-cycle strategy.

In the face of increased software demand, software project difficulties, limited experienced personnel availability, varied standards and processes, and declining budgets, the report recommends that DoD management staff continue aggressively focusing on "software engineering, acquisition, management, and human resource life-cycle challenges through the application of resources and focused action" [3].

Figure 1: *Software Growth Trend* [1]



P/L = Payload, S/C = Spacecraft, AEHF = Advanced Extreme High Frequency, GEO = Geosynchronous Earth Orbit, GPS = Global Positioning System, HEO = Highly Elliptical Orbit, NPOESS = National Polar-Orbiting Operational Environment, STSS = Space Training and Surveillance System, WGS = Wideband Global Satcomm

Fundamentally, many of the problems are a side effect of the DoD's current competitive bid acquisition strategy. It is our belief that a number of the problems could be minimized using a paradigm shift away from competing for the software engineering and development aspect of these software-intensive contracts. Hence, we provide supporting arguments and information showing that a number of the issues that we have faced on the SBIRS—and those facing other software-intensive system acquisitions—are side-effects attributable to constraints imposed by the competitive-bid acquisition process. These constraints stress cost and schedule from the onset, resulting in additional rework cycles from the late discovery of quality issues. Furthermore, we will explain how a paradigm shift could minimize these issues for the class of space system acquisitions that are on the future systems software growth trend. The current acquisition paradigm involves a competitive bid (with software as a factor) between teams of contractors in response to a request for proposal (RFP).

The Problem With "Best Value" Bidding

In a Defense Acquisition University (DAU) course class exercise (attended by Millette), students assumed the roles of contractors preparing a bid response to an RFP for a software-intensive system. Students are given three options for software costs: a low-, medium-, and high-cost figure. The evaluation criteria indicated that cost was not specifically a criterion, but it is certainly always considered.

Having the development life-cycle

issues faced by the SBIRS in their background, the group selected the high software cost as a way to mitigate the overall software development risk. The students believed that the high-cost figure would, combined with staying under the life-cycle and unit procurement cost thresholds: reduce overall cost and schedule risk; help inevitable requirements creep, rework, and other typical software cost and schedule drivers; and present a better risk-mitigation position. However, when the other groups briefed their analysis of the proposal—and specifically, why they did not recommend selection—each cited the high software cost as a negative aspect of the proposal.

In the SMC, we learn this lesson often: It is not in the contractor's best interest to bid the actual, expected, or risk-sensitive cost, as the evaluators may not recognize this as a positive and will only focus on the bottom line. The contractors we work with are not devious or intentionally trying to underbid these efforts maliciously; they are simply doing what they believe they have to do in order to secure the work. If one bidder of the group goes with the realistic or conservative cost estimate, they run the risk of being identified as not providing the best value bid for the government.

From this experience—and some of the observed strategies employed by the bidders for the SBIRS program and others—we conclude that contractors will (and do) try to utilize cost-minimization strategies to win contracts. If they are bidding on multi-billion dollar systems, cutting costs to save the government billions of dollars has repeatedly been shown to be a winning strategy. While shaving costs in an attempt to provide the government and the taxpayer with a system for a good value is appreciated, it ultimately raises the question of how such strategies could impact the quality of the NSS mission's critical software component early in a program's life cycle.

In [4], seven different flight software projects contained in an Aerospace Corporation database are reviewed: Two remarkably different software projects are compared in detail using a system dynamics model. Chapters focused on qualitative research and game theory provide a number of insightful government schedule and cost pressure strategy impacts on contractor quality. Also included is a model showing the deleterious schedule impacts from the early life-cycle schedule-driven behavior: minimizing effort in quality-enhancing peer reviews and developer unit testing (that is often perceived by these individuals to be a waste of time). Contrary to the near-term schedule-saving efforts by engineers, the opposite schedule effect occurs in the long-run due to the increased time spent fixing errors that are found later (e.g., during software integration testing).

The application of game theory concepts (see sidebar) suggests how contracts can get into the situation that TSPR policies seemed to accentuate. TSPR policies both reduced government oversight leading to contractor decisions contrary to government's quality expectations, and removed software development standards from the contracts. Software standards are essential on contracts: They result in development and test practice compliance that all contractors use to achieve a rigorously engineered software product with a demonstrated level of quality required by space systems. Thus, when it comes to software quality, strategies for bidding low will inevitably lead to cost and schedule overruns.

Reference [4] provides 26 specific recommendations for the government and contractors, including controversial subjects like mandating certifications for software professionals. However, as long as we continue to competitively bid software (as an integral part of NSS systems), the cost and schedule driven aspects faced by the SBIRS program will persist—if not get even worse—in the future. It is our belief that these issues are founded in the government's competitive bid approach, therefore making the current acquisition model unsustainable—even using newer model-based software development methods utilized by the automotive industry pushing cars into the 10 million and 100 million LOC regime. The adoption of these development methodologies into embedded space systems will undoubtedly help; however, due to the nature of the bidding process for these unique and extremely costly systems, we contend that they do not address the fundamental issues[5].

Game Theory and the Bidder's Billion-Dollar Dilemma

Game theory can be used to analyze optimal strategies for action in competitive situations with two or more players of the game[1]. Game theorists use a strategy matrix to analyze each player's strategies when they attempt to take into account the action of their opponent in their decision-making process in order to maximize their payoff[2]. An example of a normal form game matrix with two distinct strategies (*A* and *B*) has Player 1 payoffs of α_1 for the *A* strategy and β_1 for the *B* strategy, while Player 2 payoffs are α_2 for the *A* strategy and β_2 for the *B* strategy:

		Player 2	
		A	*B*
Player 1	*A*	(α_1, α_2)	(α_1, β_2)
	B	(β_1, α_2)	(β_1, β_2)

The game is called a zero-sum game when one player's payoff (win) is the other player's loss. However, the game is called a non-zero sum game if the payoff to the winning player is not from the losing player[3].

We now consider a case where an acquisition game for a very expensive billion-dollar satellite system results in only two potential bidders, with the government acquirer responsible for setting up the rules of the game. Both bidders have two pure strategies: the *A* strategy results in providing a bid for non-recurring research and engineering to build the system that incorporates more costly risk mitigation techniques leading to a politically unacceptable cost plus a substantial fee if they win; and a *B* strategy that results in an acceptable cost plus a substantial fee if they win. If they lose, they simply get reimbursed for their effort to create a bid. Mathematically we write:

$$(\alpha_1 = C + f_1) >> (c + f_1 = \beta_1) \text{ likewise}$$
$$(\alpha_2 = C + f_2) >> (c + f_2 = \beta_2).$$

In this situation, the *A* strategy with its higher cost risk mitigation activities is considered a losing strategy. The highly desired *B* strategy solution (in this case) is a Nash equilibrium[4]. Unless both bidders were required to include the cost for those risk mitigation activities in their bids, the likely outcome is bids that removed these engineering tasks.

In addition to the TSPR policy changes that directly impacted space system software development and testing standards previously mentioned, the cost as an independent variable (CAIV) strategy was envisioned as a method for government to control cost by making it a constraint [14]. Consider the impact of these constraints at the software engineer level: Tough cost, schedule, and quality tradeoff decisions need to be made when trying to hire the people required to complete the contractually obligated design documents, write the software code, and test the software. In addition, staffing is required to adequately review the design, code, and test products. Experience has often shown that sound engineering judgment becomes dominated by what is believed to be *good enough*.

Hence, we propose an alternative software acquisition paradigm that we believe can work to effectively minimize a number of these issues: Remove the competition for low-cost from consideration for the software component of the system acquisition.

Causes of Project Success—and Failure

Ivar Jacobsen, Grady Booch, and James Rumbaugh identify software success factors as dependent on people, process, product, project, and tools [15]; it can also be argued that process, product, project, and tools are also fundamentally dependent on people, and thus people are central to the entire software problem. While establishing an early version of COCOMO, Barry Boehm found that success—in regards to lower costs and on-schedule delivery—was highly dependent on the software team [16]. Considering that contractors are forced to manage and change out the personnel they hire and retain to build our systems (based on the contracts they are able to win), we are faced with a situation where we are dynamically dependent on the number and quality of personnel available at any given time. Even true A-teams under adverse cost and schedule constraints have a high probability of significantly overrunning cost and schedule in order to maintain quality. However, projects driven by cost are not likely to get the best people. The result is a mixed bag of really good people trying to pull along a cadre of less-capable performers that help bring staffing numbers up to the appropriate number the cost/schedule models suggest. These models allow for dialing in the capability of teams; our experience from a number of programs, however, shows that contractors always claim that they have the A-team, that they are CMMI® Level (fill in your favorite contractually obligated number here), and that their team can write the software faster than the speed of light with virtually no defects. When the project is finally over (after either numerous cost overruns or finally being cancelled), these people are recycled onto the next project. Hence, people—more specifically, the assignment, organization, and overall management of people—are the Achilles' heel of software.

"... a properly managed labor pool could provide cost advantages as well as a method for identifying and retaining the best engineering talent."

Oftentimes, projects are saddled with the following problems, all leading to late life-cycle schedule and cost overruns:

- Early life-cycle personnel lack the fundamental knowledge required to successfully write requirements or to design, build, and mathematically test complex real-time satellite control software.
- Management does not appreciate the need for following documented engineering processes.
- Cost- and schedule-driven decisions, mandated by government action, remove numerous prudent risk-mitigation steps.

Once upon a time, we could hide our software foibles behind extremely visible hardware issues, but not any longer.

Establishing a National Systems Engineering Laboratory

Quality and schedule could be met (at least within a consistent cost and schedule margin) if we fundamentally shift our acquisition paradigm: from program-by-program cost and schedule management to a focus on the quality of people used to feed our engineering teams. This would be accomplished by establishing what we call a National Systems Engineering Laboratory (NSEL). In it, the private sector (the big system integration and Systems Engineering and Technical Assistance contractors) are initially reimbursed to provide these facilities with their very best software personnel (management and engineers). The NSEL would also be cooperatively staffed with selected personnel from our universities, federally funded research and development centers (FFRDCs), and government services.

As new systems are being competitively designed by select senior private sector staff (competing the hardware and model-based software designs based on system requirements), they are supplemented by NSEL personnel and high-performing university students working behind strict firewalls. NSEL and student personnel would have the experience and training to provide an initial set of documentation and prototypes to acquisition staff. The winning contractors, supplemented with these NSEL (and now more mature) student personnel, would build, from the preliminary design and prototype, the final software.

Standard systems and software development process tools—such as the Agile, Spiral, or incremental models—are used insomuch as they are tailored by the engineers themselves to follow the best practices brought forth from industry and academia, based on each system's size and type of effort. Personnel are trained and incentivized to both follow these processes and also suggest process improvements as lessons are learned and technology advances. Overtime policy can be set to maintain schedule, but never to the detriment of quality. Incentives are provided to ensure creation of only the end-products necessary for designing the system and the documents that must be handed off to the next development phase or as required to maintain the system.

Prestige combined with attractive pay and high quality of life, NSEL site locations can be used to attract the best of the best. Contractor payouts are used to entice industry to bring to the table for consideration their best processes, software designs, and existing code used in other systems. Once the final system design has been selected, the pool of available top-notch engineers can draw from a wealth of software designs and prototype code to build the final flight code. Existing systems in use can draw from the same pool of engineers to maintain these systems, as needed.

Another consideration is utilizing university students and fresh graduates as a significant labor source. Software-inten-

® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

sive systems are usually dependent on the development and maintenance of significant specialized software utilities and tools to support the development effort. Select students using an open-source software paradigm could interact with the top-notch NSEL engineers (as their customers) to develop the required tool suites. While open-source code in our defense systems is usually fraught with security concerns, a properly managed labor pool could provide cost advantages as well as a method for identifying and retaining the best engineering talent. Expanding this open-source tool suite support effort to include actual system code could be investigated once the paradigm takes hold.

This alternative paradigm would alleviate the dilemmas facing prospective bidders on software-intensive acquisition efforts (exemplified in the DAU example). Under the NSEL paradigm, underbidding the software development cost would be unnecessary because it would not be a direct labor charge to the contractor. The late life-cycle software development personnel would be supplied by the NSEL, and could tap into prototypes from our universities, the contractor community, and the engineered design for the target system. This would ultimately lead to more predictability in the cost and schedule of the software development efforts, as the NSEL would employ high-quality people using disciplined processes tailored for the specific acquisition underway.

The paradigm—that funds an NSEL as a national asset, and removes the software cost consideration from the bidding process—includes the following:

- There will be competition between engineering teams.
- Early software activities will provide risk mitigation for the construction phase of the contract.
- Independent government and support staff will evaluate the engineering designs and estimated construction costs for the different systems.
- NSEL staff will be included on each team, with the expectation of working in seclusion from NSEL members on the other team.
- Teams will individually utilize the pool of available sub-contractors—with products and services determined by the systems and software engineering each team requires to build the system.
- The engineering and software prototyping staff is selected based on merit, capability, and need.
- Software build and test staff is selected in part from the NSEL staff on the losing team and from engineering/software prototyping staff. They will test the constructed software system or augment the software development and test phase, based on staffing requirements.

The predicted end-result is a higher quality software product that is staffed with the best people available. However, it should be mentioned that the number of new software-intensive systems we could build as a nation would be constrained by the number of NSEL staff. Yet, we view this as an acceptable engineering alternative to the CAIV-approved approach under TSPR.

The NSEL is first and foremost tasked with building and maintaining quality systems, with a strong eye towards successfully designing and building cost- and schedule-acceptable solutions. Under this premise, quality staff can, with time, create their own training and competitive hiring policies for their engineering positions. In this manner, the processes developed and promulgated by these staff tend (through generations of engineers) towards a level that can keep up with demand. While problems will undoubtedly arise, this self-contained, continual learning environment will foster and lead towards solutions for these issues.

We also propose that NSEL directors for functional areas in engineering are hand-selected for prestigious appointments from academia, the FFRDCs, and private industry. Their hiring will be based on current requirements for government positions—and will not be hired via political appointment. The directors' primary role will be to resolve technical and management issues—with the national need, which is always at the center of their decision-making process.

Conclusions

An NSEL for defense acquisition strategy is an alternative system acquisition concept that is based on a grass-roots or grounds-up negotiation between the engineering disciplines. It has the potential to take the DoD boldly where no one has gone before—allowing for acquisition, next-generation-embedded, software-intensive systems. This grass-roots process is designed to provide the best quality-minded engineers needed to yield engineered systems with a consistent cost and schedule. We also believe that this concept is required to mitigate the software-intensive, system-driven people factors that have plagued a number of our system acquisitions—leading some to believe that we have lost our space acquisition recipe for success. We acknowledge that the concept must pass through the normal gamut of politically driven negotiations. Hopefully, during this process, the concept for building a national capability consisting of the best of the best—and a method to identify and retain our university engineering talent—is not lost. We've even heard of an even more drastic strategy: using a draft to nationalize our software development and system engineering talent. Short of this controversial and unlikely option, creating a prestigious system engineering research and development laboratory—retaining and nurturing the world's best engineering talent—is a sound method, fundamentally based on the talent-retention successes of our national laboratories. Our national goal should be to attract the best personnel to this field and we suggest that subtleties such as funding details, redundant locations, and other issues can all be politically negotiated and worked as this concept is matured.◆

References

1. Hamel, Michael. *Growth Trend in System Software*. Proc. of the Defense Software Strategy Summit. Arlington, VA. 18-19 Oct. 2006 <www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Content/ContentGroups/Divisions1/Systems_Engineering/Hamel_10_18.pdf>.
2. Buettner, Douglas J., and Bruce L. Arnheim. *The Need for Advanced Space Software Development Technologies*. Proc. of the 23rd Aerospace Testing Seminar. Manhattan Beach, CA: 10-12 Oct. 2006.
3. Office of the Deputy Under Secretary of Defense for Acquisition and Technology Systems and Software Engineering. *Defense Software Summit Report*. Final Draft, Vers. 1.0. Washington D.C.: Pentagon. Oct. 2006 <www.acq.osd.mil/sse/docs/NDIA-Defense-Software-Summit-Report-October-2006.pdf>.
4. Buettner, Douglas J. "Designing an Optimal Software-Intensive System Acquisition: A Game Theoretic Approach." Ph.D. Dissertation. Astronautics and Space Technology Division, Viterbi School of Engineering, University of Southern California, Sept. 2008.
5. Schinasi, Katherine V. "Military Space Operations: Common Problems and Their Effects on Satellite and Related

Software Defense Application

The article proposes a National Systems Engineering Laboratory that can help other DoD entities with issues that the authors have observed in their National Security Space software-intensive system acquisitions. This article first details the problem of cost mitigation bidding strategies used by DoD contractors, and then recommends solutions through an NSEL. Removing this competitive bid process for the software engineering through the creation of a prestigious systems engineering lab—staffed by our nation's best engineers, created solely to provide quality design and engineering services—is one possible solution.

Acquisitions." Letter to The Honorable Jerry Lewis. U.S. General Accounting Office. GAO-03-825R Satellite Acquisition Programs. 2 June 2003 <www.gao.gov/new.items/d03825r.pdf>.

6. GAO. *Defense Acquisitions: Improvements Needed in Space Systems Acquisition Management Policy*. GAO-03-1073. 15 Sept. 2003 <www.gao.gov/new.items/d031073.pdf>.
7. GAO. *Defense Acquisitions: Despite Restructuring, SBIRS High Program Remains at Risk of Cost and Schedule Overruns*. GAO-04-48. 31 Oct. 2003 <www.gao.gov/new.items/d0448.pdf>.
8. GAO. *Space Acquisitions: Stronger Development Practices and Investment Planning Needed to Address Continuing Problems*. GAO-05-891T. 12 July 2005 <www.gao.gov/new.items/d05891t.pdf>.
9. GAO. *Space Acquisitions: DoD's Goals for Resolving Space-Based Infrared System Software Problems Are Ambitious*. GAO-08-1073. 30 Sept. 2008 <www.gao.gov/new.items/d081073.pdf>.
10. Singer, Jeremy. "SBIRS Report to Include Update on Health of Defense Support Program." *Space News*. 8 Dec. 2004.
11. Lockheed Martin. *First SBIRS Satellite With New Flight Software Completes Key Test at Lockheed Martin*. 13 Jan. 2009 <www.lockheedmartin.com/news/press_releases/2009/11309ss_sbirs.html>.
12. Defense Industry Daily. "Despite Problems, SBIRS High Moves Ahead." 12 Sept. 2009 <www.defenseindustrydaily.com/Despite-Problems-SBIRS-High-Moves-Ahead-With-3rd-Satellite-Award-05467/#more-5467>.
13. GAO – Applied Research and Methods. *GAO Cost Estimating and Assessment Guide: Best Practices for Developing and Managing Capital Program Costs*. GAO-09-3SP. Mar. 2009 <www.gao.gov/new.items/d093sp.pdf>.
14. Perry, William J. "Acquisition Reform." *Annual Report to the President and the Congress*. Chapter 14. Mar. 1996 <www.dod.mil/execsec/adr96/chapt_14.html>.
15. Jacobson, Ivar, Grady Booch, and James Rumbaugh. *The Unified Software Development Process*. Reading, MA: Addison-Wesley, 1999.
16. Boehm, Barry W. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, 1981.

Notes

1. See James O. Berger's *Statistical Decision Theory and Bayesian Analysis* (1980 edition, page 310).
2. See Harold W. Kuhn's *Lectures on the Theory of Games* (2003 edition, pages 5-6).
3. See Philip D. Straffin's *Game Theory and Strategy* (1993 edition, pages 3-6).
4. See Straffin (pages 65-68) and Roger B. Myerson's *Game Theory: Analysis of Conflict* (1997 edition, pages 97-98).
5. We do not believe that current methods will solve the well-documented software development issues that have plagued government acquisition. These issues are documented in the popular press and in numerous GAO reports; see [5, 6, 7, 8, 9, 10, 11, and 12]. Furthermore, a recent GAO cost estimation and assessment guide documenting the best practices for developing and managing capital program costs singled out the SBIRS as a case study [13].

About the Authors

Douglas J. Buettner, Ph.D., is the systems director of the Flight Software Department in the Aerospace Corporation's Space-Based Surveillance Division. Buettner has more than 20 years of experience including contracts from NASA's comet sample return mission, all phases of defense industry software development, and quality assurance management for venture capital-backed startup companies. He has bachelor's and master's degrees in physics from Oregon State University and, as an aerospace fellow, completed his doctorate in astronautical engineering (with a focus on computer science) from the University of Southern California.

Phone: (310) 336-5658
E-mail: douglas.j.buettner@aero.org

Lt. Col. Chad Millette is the Deputy, Operational Requirements Branch, HQ AFMC/A5C. His previous assignment in the SBIRS Wing included responsibilities overseeing the development of the embedded flight software for the SBIRS program. He has a bachelor's degree in mathematics from the University of Arizona, a master's degree in software engineering administration from Central Michigan University, and a master's degree in systems management from the Air Force Institute of Technology. He is also a certified Project Management Professional and a Software Development Certified Professional.

4375 Chidlaw RD
Wright-Patterson AFB, OH 45433
Phone: (937) 656-3961
E-mail: chad.millette@wpafb.af.mil