Performance-Based Software Sustainment for the F-35 Lightning II

Lloyd Huff and George Novak
Lockheed Martin Aeronautics

The complexity and sophistication of F-35 Air System software and the multiplicity of F-35 missions, versions, and customers, combined with a performance-based contract structure, present unprecedented software sustainment challenges. Understanding how F-35 software will be sustained is the focus of ongoing analysis and planning. This article describes some of the revolutionary conclusions and products of that analysis and provides a look forward to performance-based sustainment of software for the multinational F-35 fleet.

In mid-2007, an article entitled "Lockheed Martin Hopes F-35 Leads to Maintenance Revolution" [1] was released through internet news outlets. The article stated the following:

> Lockheed Martin ... is designing a new kind of maintenance program for the $300 billion F-35 Joint Strike Fighter project, which company officials say could set a new standard for military aircraft operations. The U.S.-led, nine-nation fighter program has "performance-based logistics" built into its purchase plan, giving contractors a big role in maintenance management. ... Lockheed Martin says this maintenance strategy means that logistical support will make up about 50 percent of total program costs, compared to 67 percent of total costs under a less centralized strategy ... Under the maintenance plan, F-35 owners will pay for anticipated operating time. ... Based on how much the aircraft are expected to fly, Lockheed Martin will manage parts inventory, plan overhaul schedules and train the military crews who support aircraft operations. ... The F-35 program marks the first time an entire aircraft has used a performance-based logistics plan. Such pay-by-the-flight-hour maintenance strategies are more common for component systems, or commercial jet operations.

The ramifications of this *maintenance revolution* to software engineering and management are extensive. Initial planning for sustainment of F-35 software began in 2002. In 2005, however, the Office of the Secretary of Defense confirmed that a performance-based sustainment approach would be applied to the F-35 Joint Strike Fighter (JSF) program. This decision focused the planning phase and allowed more detailed analysis to begin. The results of the analysis to date are reviewed in this article. These results are shaping the approach to F-35 software sustainment in order to support air system performance and life-cycle cost savings objectives.

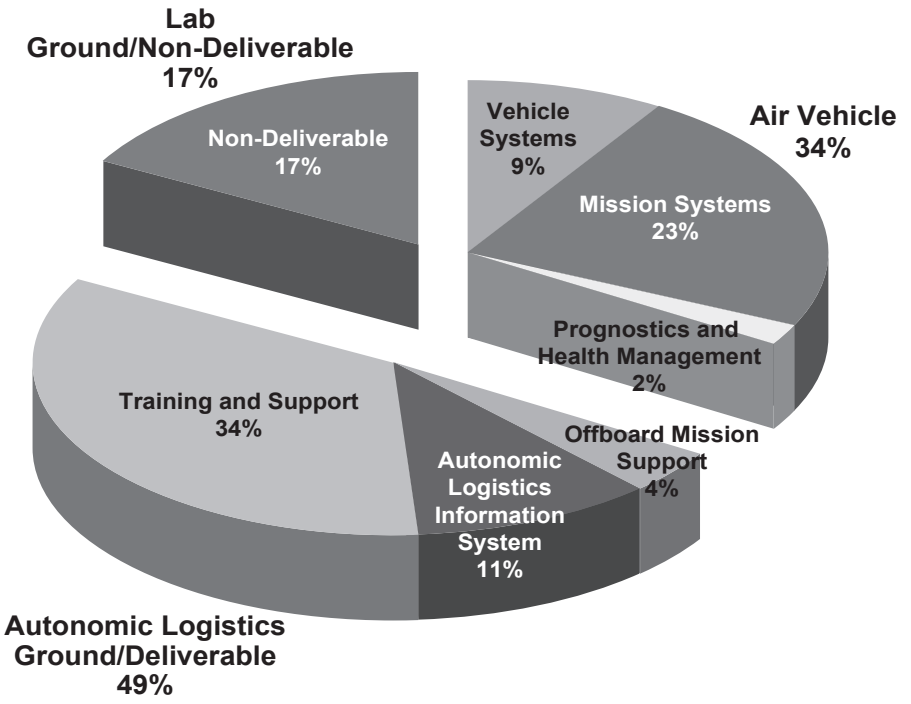
F-35 Program Overview

The F-35 Air System consists of the Air Vehicle (AV), including the propulsion system and the Autonomic Logistics (AL) system. The AV is a three-variant family of fifth-generation[1] strikefighter aircraft consisting of the F-35A Conventional Take-off and Landing (CTOL), F-35B Short Take-off Vertical Landing (STOVL), and F-35C Carrier Variant (CV).

A high degree of designed-in commonality will exist among the three variants (e.g., engines, avionics, crew station, subsystems, suspension, and release equipment and structure). CTOL operations will be a common capability among the variants with unique capabilities for the CV (e.g., catapult and arresting gear compatibility) and STOVL (e.g., vertical launch and recovery, and ski jump compatibility) variants. Key features include a blend of supportable low observable technologies, highly integrated mission systems, interchangeable propulsion systems, interoperability and internal and external carriage of stores. Other examples of commonality include Prognostics and Health Management systems, Institute of Electrical and Electronics Engineers (IEEE) 1394 aircraft bus design [2], and a cockpit incorporating advanced on-board and off-board sensor fusion. Each variant will provide an adverse weather, day/night capability to effectively execute operational missions.

The F-35 AV operates in concert with AL, including AL Information System (ALIS), which uses prognostics and health information from the AV to enable proactive maintenance. AL also features a training system which is concurrent with aircraft versions, missions, and maintenance tasks. And, F-35 AV and ground systems are designed to interoperate with the net-centric combat and logistics environments required for modern combat operations.

Figure 1: *Estimated F-35 Software Sustainment Baseline*



The air system software configuration present at the end of the system development and demonstration phase of the program forms the software sustainment baseline. This baseline, consisting of AV, AL systems, and lab software is currently estimated to be approximately 20 million source lines of code. A break-out by category is shown in Figure 1 (see previous page). Maintaining maximum commonality of this software across all variants and versions is key to achieving program affordability goals.

Figure 2 overlays the relationship of planned F-35 program phases (top of figure) to a standard product life cycle (center), and to the steps involved in development and delivery of product support (bottom of figure), through the end of program. The JSF Development and Low Rate Initial Production (LRIP) phases of the program ensure that the Air System and its support systems are mature as Full Rate Production (FRP) and Initial Operation Capability thresholds are reached. Software sustainment plans and estimates, for the *In-Service* portion of Figure 2, extend 50 years.

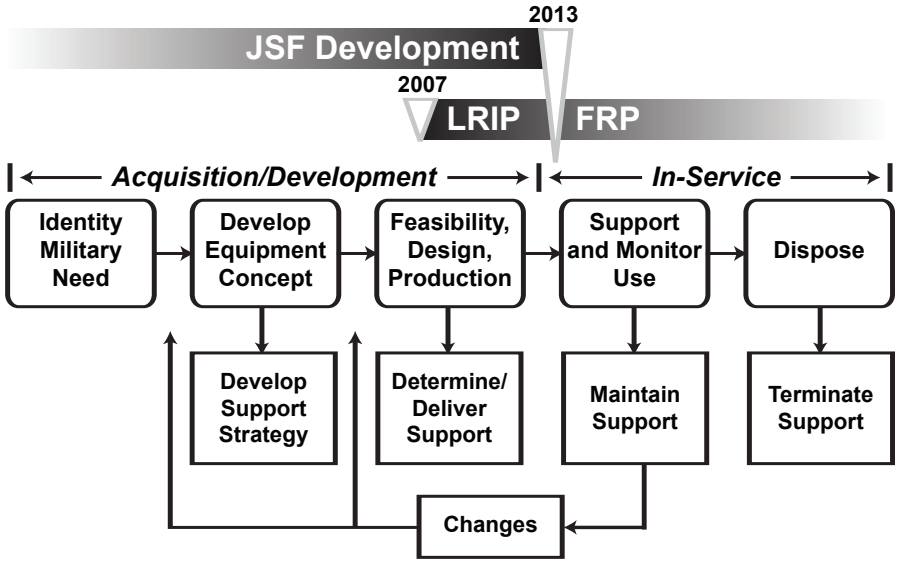
Performance-Based Contracting

The F-35 program includes partner participation by the U.S. Air Force, Navy, and Marine Corps; the United Kingdom; Italy; the Netherlands; Turkey; Canada; Australia; Denmark; and Norway. Additional foreign military sales are under consideration. Warfighters from these militaries will channel their needs through the JSF Program Office (JPO). A joint agreement on F-35 production, sustainment and follow-on development will guide the evolution of the Air System, and sets ground rules for partner participation.

The JPO is the single point of contractual direction from warfighters to JSF principal partners, and to propulsion system contractors. JSF principal partners include Lockheed Martin Aeronautics, (the Product Support Integrator [PSI]), Northrop Grumman, and BAE SYSTEMS. The PSI is, in turn, responsible for managing the F-35 global industrial base of United States and international suppliers and depots.

The performance-based contracting model can be visualized as a *loop*, which begins and ends with the warfighters and flows through the JPO, the PSI, and the global industrial base. Warfighters express their needed capabilities or changes to the JPO, along with their required performance levels expressed in terms of mission effectiveness, aircraft availability, sorties per month, etc. Performance-based contracts from JPO then transfer the risk and responsibility to provide specified performance levels to the PSI and to the industrial base. Metrics quantify air system performance and incentives or penalties. These performance metrics are the basis for a variable pricing component, referred to as *power by the hour*. Payment, under performance based contracting, is thus based on usage instead of breakage. Additionally, price improvement targets/ curves are established to drive reduction in cost over the term of a contract. In the event that design changes are implemented to improve performance, resulting cost reductions are shared between customers (in reduced price), and industry, (in increased profit). The point of performance-based contracts is encapsulated in the term Performance Based Outcomes (PBO). *Warfighters are now contracting for an outcome, or a result, as opposed to contracting for repairs, replacements, supplies, inventory, shipment, or services.*

Figure 2: *Mapping of Standard Life-Cycle Phases to the F-35 Program*



Managing Software in a Performance-Based Environment

Just as performance of the F-35 Air System is predicated on software, so is the success of performance-based contracting. Software is viewed as a crucial commodity among many that must be managed for predictability. This article will proceed to examine the keys to successful sustainment of software in a PBO environment. First, however, the PBO software sustainment domain will be scoped by reviewing boundaries, definitions, and success criteria.

As an entry point to analysis of software sustainment, a boundary graphic (Table 1), was produced to delineate software services funded under PBO, as opposed to that funded otherwise. To summarize the graphic, any software sustainment action taken to maintain the delivered software baseline falls *within* PBO; any software sustainment action which adds or changes functionality to the software baseline falls *outside* of PBO.

These boundaries, however, must be accompanied by precise definitions of software changes, releases, and support services. Software changes are defined in terms of *priority* (routine, urgent, and emergency) and *purpose* (corrective, adaptive, perfective, new capability, performance initiative, technology sustainment, or technology insertion). Software release types are defined in terms of size and *tempo* (such as major or minor block releases, AV maintenance updates, or asynchronous releases of AL software). Typical timespans for development are associated with software change and release categories. These values form the basis for sustainment cost models and a single, integrated master sustainment plan.

Measurement and analysis of F-35 software performance will be driven by performance-based metrics derived from Performance-Based Agreements between the JPO and the warfighters. The top-level metrics are decomposed into a metrics taxonomy. This metrics taxonomy encompasses all influences upon the top-level PBO metrics. Accordingly, software performance is measured and analyzed to quantify its operational performance. The lower-tier software metrics reveal the influence of software on F-35 air system PBO metrics and serve to initiate improvements in performance and predictability. Table 2 provides a look at how software might influence air system performance and how such influences will be tracked.

Keys to Successful PBO Sustainment of F-35 Software

As stated earlier, the ramifications of per-

formance-based contracting to F-35 software are far-reaching. Looking forward through 2063, factoring in the rate of technological change and considering security and safety ramifications, sustainment of F-35 software quickly moves from *far-reaching* to *prodigious*. As such, the following eight key steps are being taken to manage this commodity.

1. Strive for Commonality

The JSF program, from its inception, has been built upon the following four pillars: affordability, lethality, survivability, and supportability. The extent to which a common software baseline is retained across F-35 variants and F-35 international partners will directly affect overall affordability and supportability. While air system software is tailorable and compatible with each owning service's support environments, continued emphasis on commonality will maximize affordability and supportability for all system users. A common solution, employing minimal infrastructure, provides best value sustainment capability at minimum cost to all parties.

The ultimate goal of all participants, therefore, is to reach consensus on a common sustainment solution and, thereby, minimize the incidence of multiple system/software configurations. However, some unique capabilities will be necessary to satisfy specific operational needs, address sovereign requirements, and alleviate political and industrial concerns. Unique software capabilities will typically occur in two areas: in the AV Mission Systems software configuration (specifically, in weapons controls, pilot-vehicle interface, and communications/interoperability), or as additions to the F-35 software integration and test supporting infrastructure. In either event, F-35 partner countries will have the opportunity to have their changes considered for inclusion in the common baseline before steps are taken to assess the cost and impacts of a unique software change. Any unique functionality will be encapsulated to minimize re-verification expense and the cost will be borne by the partner/partners involved on a *pay-to-be-different* basis.

Covered by PBO Contract	Covered by Follow-On Development Contract
• Maintenance – Corrective Maintenance – Urgent and Emergency Corrections • Performance Initiatives – Operational Administration – Day-to-Day Administrative Support • Sustaining Engineering – Planning – Studies – Standing Boards, Configuration Management – Needs Analysis – Lab and Development – Environment/Infrastructure – Programming Infrastructure – Field Deficiency Assessment • Technology Sustainment	• New Capability • Technology Insertion • Adaptive Change • Perfective Change • Urgent Operational Need • Emergency Operational Need • Production Retrofit

Table 1: *Performance-Based Contracting Boundary*

2. Apply Industrial Engineering Practices to Software

Many parameters must be considered to plan and manage the performance-based F-35 software sustainment domain with a globally dispersed, multinational future fleet. These parameters include the frequency and quantity of software changes, the number of versions, and the time required for development, validation, and distribution. AV load times are important, even more so when viewed over a 50-year period. In this context, F-35 software sustainment emerges as an industrial engineering field, where efficiency, consistency, the elimina-

Table 2: *Software Influence on Performance-Based Metrics*

Performance Criteria	Top-Level PBO Metrics	Software Metrics
Readiness/ Availability	• Aircraft Availability • Mission Capable Aircraft Availability (AA) Rate	• Aircraft Downtime (Software) • Software Mission Capability
Mission Effectiveness	• Mission Effectiveness – Primary Task Not Accomplished; System Condition – Secondary Task Not Accomplished; System Condition	• Primary Task Not Accomplished (Software) • Secondary Task Not Accomplished (Software)
Required Sorties and Flying Hours Accomplished	• Percent Sorties Flown • Percent Flying Hours Flown (* Flown *Ground Abort *Cancelled)	• Percent Sorties Not Flown Due to Software • Percent Flying Hours Not Flown Due to Software
Logistics Footprint	• Logistics Footprint Data Total Change = Support Equipment Change + Personnel Change	• Support Equipment Change Due to Software – Support Equipment Size Change Due to Software – Support Equipment Quantity Change Due to Software • Personnel Change Due to Software – Direct Manpower Change Due to Software – Other Manpower Change Due to Software
Military Level of Effort	• Cannibalizations per 1,000 Flight Hours (FH) • Maintenance Man-Hours per FH • Maintenance Man-Hours per FH (A/C Subsystem)	• No Software Metric Applicable • Software Maintenance Man-Hours per Flight Hour • Software Maintenance Man-Hours per Flight Hour (A/C Subsystem)

tion of waste, and a solid understanding of capacity must be achieved. To help bring order and quantification to the F-35 software industry, modeling and measurement are being employed.

An end-to-end software sustainment process model supports the business case analysis of F-35 software sustainment. The model is expressed as an event-driven process model using the Architecture of Integrated Information Systems modeling tool (ARIS). ARIS was selected based on its features, its standardization within Lockheed Martin, and its use in development of interfacing models (F-35 AL operation guides, software integration and test lab modeling, system build, software loading processes, and software distribution). The software sustainment model starts with the field report of a software defect and ends with measurement of performance of the delivered, operational software solution. The model provides a means to understand tasks and capacity constraints, and supports estimation of sustainment costs.

An example analysis area within the model is the process of system build. System build is the packaging of hundreds of lower-tier software products to create a release product set. A complete air system build package is a single, deliverable software product (end item) to the fleet, for installation and operation. It is an organized assembly of vehicle system, mission system, and AL software, along with release documentation, flight clearance, technical data, and other supporting version information to facilitate identification and distribution. System build is a process that will be performed many times during each maintenance release cycle and is critical to both the delivery timeline and overall load integrity.

Tight controls are applied to this crucial *handover* point. The system build/final software integration process is dependent upon safeguards and controls imposed upon lower-tier software builds. All required certifications, qualifications, formats, and approvals must be applied throughout the software build hierarchy. Accordingly, three checkpoints/readiness *gates* were established to ensure that files and artifacts obtained for system build are completed, correctly formatted, fully described, and duly authorized. The gates affirm that all required software components, along with related files or data, are available and have been properly identified, are functionally acceptable, have achieved all required certifications and qualifications, and that interfaces comply with applicable requirements and descriptions. Transfer of files and artifacts through these gates are controlled with checklists, which specify criteria and are administered through a review and approval process. To the extent possible, additional safeguards have been incorporated in tools and workflows.

Apart from measuring the *integrity* of the system build process, a set of metrics is maintained to enable *capacity* planning for system build. As releases are produced, span times, touch times, execution times, delay times, and total effort metrics are tracked for routine and urgent builds. The results are synthesized into cost estimates and process improvement initiatives.

Table 3: *Software Maintenance Benchmarking Participants*

Benchmarking Projects	
British AV-8B	U.S. AV-8B
A-10	B-1B
B-2	C-17
C-130	JE-3
F-15	F-16
F-18	F-22
F-117	P-3C

3. Engage Customers

The F-35 software life cycle was planned in progressive stages. Each stage engaged users and partner country subject matter experts. First, U.S., U.K., and international standards for software maintenance and support, including IEEE, Society of Automotive Engineers, and military standards were canvassed to form a foundation for sustainment planning.

Next, fact-finding was accomplished through a benchmarking study of software maintenance operations across 14 military aircraft programs. A list of software maintenance operations willing to share their expertise is shown in Table 3.

A questionnaire was developed by F-35 Integrated Product Teams (IPTs). The benchmarking study manager used the questionnaire to conduct interviews with 39 representatives from the 14 software maintenance programs. The information obtained from these interviews was compiled and summarized in a report. The study served to identify practices which required consideration for adoption, or avoidance, by the F-35 program. It affirmed the importance of thorough planning, establishment of communication channels and information flows, and compliance with clearly documented processes. The study also revealed that successful support of multinational customers requires focused attention on several elements (e.g., export controls, software storage and segregation, and joint acceptance criteria for software changes).

Following the benchmarking study, results were incorporated into the software life-cycle plan for the F-35. The plan was subject to several rounds of review by graybeard panels. Each round of review was conducted during a 30-day timeframe, beginning with a kickoff, followed by individual preparation. Panelists invested an average of 8.5 hours to study the plan, rate the contents, and prepare their preliminary comments. Deep-dive assignments were allocated to focus specific panelists on selected topics. Panels were then convened for a face-to-face walk-through of topics over the course of several days, and results were summarized in outbriefs. More than 1,000 comments and recommendations were raised and addressed as the result of the graybeard panels. (Responses to several of these recommendations are noted throughout this article.)

Interactions with graybeard panelists opened the door for visits to seven U.S and international software maintenance operations by F-35 representatives. These visits resulted in useful dialogue with customers and software sustainment personnel from 11 aircraft programs. Again, a standard list of topics and questions were used to ensure consistency.

Finally, with a reasonably mature life-cycle plan in place, subject matter experts from partner countries were directly engaged with JSF contractors in a Software Maintenance and Sustainment Working Group (SMS WG), a team of 40 participants comprised of equal part contractors and customers. The SMS WG is chartered to ensure customer expectations relative to JSF software sustainment are considered in communicating, planning, documenting, contracting, and scheduling of affordable software sustainment solutions.

4. Adopt a Holistic Approach to Sustainment

Earlier in this article, the contractual boundaries between PBO maintenance changes and follow-on development changes were emphasized. While this is keenly important from a funding standpoint, PBO-driven software changes cannot be viewed independently. PBO-driven software changes must be weighed in the context of performance of the global F-35 fleet and balanced for their impact on overall system change capacity of the F-35 enterprise. Ultimately, the effect of any software change will be evaluated in terms

of net worth provided to the warfighter.

F-35 block updates will *bundle* the delivery of new functionality and PBO maintenance changes. Block updates will include a mix of all types of software changes, and may encompass hardware/subsystem changes. As IPTs develop and produce the changes to support F-35 block plans, capacity planning must allow for *both* PBO changes and for new functionality. Accordingly, the processes, definitions, requirements and practices for software *maintenance* planning were merged with the F-35 template for software development plans.

On the plus side, technical solutions are not constrained. System changes, hardware changes and software changes, corrections, and new functionality are all assessed with respect to the end effect on air system performance and affordability.

On the downside, cost accounting within IPTs responsible for producing corrective changes and new functionality requires extreme fidelity to ensure PBO effort is distinguished from follow-on development.

5. Develop Highly Maintainable Systems and Software

Maintainability of F-35 software is based upon an AV with an open and scalable architecture. The *open architecture* allows for expansion with minimal impact to unchanged elements, through use of well-defined, non-proprietary interfaces and protocols. Hardware and software elements are partitioned using loosely coupled, non-time critical interfaces. A *data collection domain*, added to the AV architecture, supports general instrumentation and fault isolation requirements. An *isolation layer* protects the software investment from hardware obsolescence and facilitates multi-use across air system domains.

DOORS (Dynamic Object Oriented Requirements System) databases are populated with requirements and the rationale for their selection, including linkages to the architectural models. Modeling and simulation tools based on architectural constructs are employed to develop and validate the requirements and verify the air system.

Object-oriented design results in smaller configuration items with clearly defined functionalities. *Hardware independence* supports problem accountability and localizes change. This approach results in minimal changes to the overall configuration when adding or deleting functional capability. It also supports change development at the lowest level, minimizes the impact of changes, and enables *focused testing*. *Focused, model-based component testing* provides opportunities for efficiency in an area that typically entails high cost.

Architectural and design integrity is maintained through use of structured, *common systems engineering processes and tools*, which have reduced initial development costs and will support the efficient long-term maintenance of designs. The JSF Systems/Software Engineering Environment (S/SEE) resides on networks of computing equipment which connect F-35 customers, contractors and subcontractors. This shared environment is used across the entire team to foster a unified understanding of the open architecture and its maintenance. The S/SEE includes commercial off-the-shelf (COTS) Unified Modeling Language tools, such as Rhapsody, for enforcing the object-oriented software design and the Signal Interface Management System[2] (SIMS) tool, which forces full interface definition.

Autocoding with JSF S/SEE tools allows software code to be developed to Open System Architecture (OSA) standards. In specific domains, design tools are used to capture requirements, model a functional algorithm and provide source code as an output to implement the modeled design.

Emphasis is placed on leveraging COTS, and *supportability* (aggressive preparation for tool obsolescence). Configurations of S/SEE tools are managed in consonance with air system software releases to ensure build repeatability and maintainability.

Maintainability of F-35 software is also supplemented by software *reuse and multi-use*. (Reuse is the use of pre-existing code; multi-use is the reuse of code in multiple areas.) Software with proven maturity and reliability, from various sources including legacy aircraft, government-furnished equipment, software and databases, different F-35 variants, off-the-shelf software, subsystem software, and third party suppliers, is reused to the extent possible. Reuse objects are also used as starting points for development of other, closely related objects that may be required for different F-35 domains. Availability and quality of documentation and source files are considered as part of reuse analysis and determination. *Preconditioning* of software targeted for reuse may be performed as necessary in order to ensure maintainability. If candidate code does not meet OSA standards or object orientation, but provides needed functionality with an effective design, the design and algorithms (only) are reused and recoding is performed using architectural patterns to develop software which provides better long-term maintainability and lower total ownership cost than pre-existing software that is *wrapped* to be OSA compliant.

6. Manage Off-the-Shelf Software

The up-front savings realized through use of off-the-shelf software is frequently offset by risk and expense incurred later in the product life cycle. (Off-the-shelf software includes COTS, modified off-the-shelf [MOTS], government off-the-shelf [GOTS], freeware, public software, and related categories of non-developed software.) Accordingly, special emphasis was placed on managing off-the-shelf software. Through a collaborative effort with the JPO, a process document entitled, "Off-The-Shelf Software in the JSF Software Lifecycle" [3] was produced and deployed. The process document describes actions, over and above standard software process requirements applicable to software developed specifically for the JSF Program, which must be taken to ensure off-the-shelf software components are configuration managed throughout their life cycle. It also identifies the functions/roles responsible for those actions. Instructions in the process are partitioned according to their applicability to a 4-phase life cycle, along with generally applicable rules, guidelines, and warnings.

Based on results of a 2005 collaborative evaluation of process implementation, "Off-The-Shelf Software in the JSF Software Lifecycle" was revised and updated to include a system for classification of off-the-shelf software projects (small, medium, or large projects) based on specified criteria. Suggested tailoring of process requirements based on project category is included along with examples. The revised process also incorporates a requirement for a generation of a compliance matrix by off-the-shelf software projects, to ensure that all applicable requirements, including license and distribution controls, are adequately addressed.

7. Plan for the Unexpected

Warfighters are keenly interested in how the F-35 AL global sustainment solution will respond to urgent operational requests, or to emergencies. To answer these concerns, software sustainment scenarios have been developed by the F-35 SMS WG. The scenarios contain sufficient detail to describe the activities required for non-routine situations. The scenarios are used to exercise ARIS process models and make an up-front determination of the cost and time required to perform all needed activities.

8. Analyze and Refine the Software Sustainment Business Case

The eight steps featured in this article have covered a lot of territory. But the steps

begin and end with a focus on money. Step 1 focused on achieving affordability through commonality. This final, 8th step addresses the business case analysis of F-35 sustainment, annual global sustainment total ownership cost estimates, and the software cost estimation practices that support these analyses. Each of these interconnected activities uses a spiral development approach with each spiral providing increased fidelity of data, inclusion of decisions from across the program, updates on configurations and reliability projections, and comprehensive detailing of the business offering.

Business Case Analyses (BCAs) define F-35 global sustainment policies and processes. These analyses answer three basic questions: 1) What individual tasks must be performed during sustainment?, (2) who (government or contractor) should perform those tasks, based on *best value?*, and (3) does the task allocation support established performance standards and provide sufficient savings? BCAs address, for example, a pricing architecture for PBO sustainment, international taxes and tariffs, industrial base capacity and responsiveness, and, of course, software sustainment.

Global sustainment cost estimates are formulated annually. These estimates integrate estimates from all IPTs, functions, team member companies, and subcontractors involved in the F-35 global sustainment solution and the pilot program for performance-based sustainment. Annual cost estimates are consistently produced, fact-based, and supportable. They are reconciled with the JPO affordability cost analysts and are finalized and formally approved at JSF cost summit events. The integrity of these annual global sustainment cost estimates is critical to the success of affordable, PBO sustainment of the F-35 fleet.

Parametric software sustainment cost estimates are developed for inclusion in the annual global sustainment estimates using output from the System Evaluation and Estimation of Resources – Software Estimating Model (SEER-SEM) tool. Software sustainment cost estimates align with Cost Analysis Improvement Group Element 6.5, "Software Maintenance Support." They are not, however, fully representative of all costs associated with performance-based software sustainment and are subject to ongoing refinement and updates. Updates provide greater detail and direct estimates for software integration and test activities, software lab *keep warm* costs, and greater fidelity with respect to license costs for off-the-shelf software included in deliverable F-35 software products. In the absence of *actual* data, ground rules and assumptions are documented and version-controlled to describe cost areas which are included in, or excluded from, the F-35 software sustainment business case.

Finally, long-term software sustainment cost estimates entail software maintenance *and* software growth estimates. Once a software release (which, as we have seen, will contain fixes and new functionality) is distributed to the field, it becomes the new *maintenance baseline* and PBO contracting for when the new release takes effect.

Conclusion

The decision to apply a performance-based sustainment approach to the F-35 has caused fundamental changes in the approach to Air System sustainment. Traditional roles and responsibilities are shifting. An increased risk is transferred to contractors who are now responsible for system availability and mission success. This has precipitated a new approach to software sustainment. While results are years away, the F-35 software community has put a foundation in place for PBO software sustainment. Construction on that foundation continues, day by day.◆

References

1. Christie, Rebecca. "Lockheed Martin Hopes F-35 Leads To Maintenance Revolution." DOW Jones Newswires 12 June 2007.
2. IEEE. "IEEE Standard for a High Performance Serial Bus." IEEE 1394, 1995. "Amendment One." IEEE 1394a, 2000. "High-Speed Supplement."IEEE 1394b, 2002.

Notes

1. Fifth Generation Fighter features these attributes: advanced stealth, information fusion, high agility, enhanced situational awareness, new levels of reliability and maintainability, and network-enabled operations.
2. SIMS is a Lockheed Martin Aero internally developed interface management tool, based on a commercial relational database and used on multiple aircraft platforms.

About the Authors

Lloyd Huff is a Lockheed Martin senior fellow for software and avionics where he is currently engaged in the development of the F-35 software sustainment solution. Recently, Huff served as director of JSF Software Management, JSF Software Proposal Lead, and X-35 STOVL First Flight Deputy Lead. He has 28 years of aerospace experience including F-16 multiplex buss design, LM Site Lead at Hill AFB, 50% Software Cost Reduction Lead, CMM Level 4 Tool Development Lead, and Principle Investigator for Multiplex and Fiber Optic Research. Huff holds bachelor's and master's degrees in computer science from the University of Kansas.

Lockheed Martin Aeronautics
P.O. Box 748, MZ 1523
Fort Worth, TX 76101
Phone: (817) 821-9196
Fax: (817) 763-1737
E-mail: lloyd.a.huff@lmco.com

George Novak is a Lockheed Martin software senior staff member where he is responsible for planning the long-term global sustainment of software under a performance-based logistics contracting model. Novak is a member of the software management team responsible for air system software builds and software life-cycle planning for the F-35 air system, and is co-chairman of the F-35 software maintenance and sustainment working group. He has held a variety of positions on military aircraft and commercial telecommunications programs and has served as a lecturer and independent consultant in the area of software process improvement. Novak has a master's of business administration in industrial management from the University of Dallas.

Lockheed Martin Aeronautics
P.O. Box 748, MZ 2306
Fort Worth, TX 76101
Phone: (817) 763-3863
Fax: (817) 763-1737
E-mail: george.j.novak@lmco.com