WORKING DOCUMENT//DRAFT

Software is Never Done:
Refactoring the Acquisition Code for Competitive Advantage

Defense Innovation Board

SUPPORTING INFORMATION

v1.0, 19 Feb 2019

This document contains the supporting information for the Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) study. This information is in preliminary form and should be read along with the main (draft) report.

Contents:

Appendix B.1: Acquisition Strategy Subgroup Report

8 February 2019

This appendix examines pain points, obstacles, change ideas, and future vision for the Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) Study in the area of Acquisition Strategy and Oversight (i.e., *Acquisition Environment*). In 2017 the Office of the DASD(C3CB) under the ASD(A) commissioned an IT acquisition study with Deloitte. The study recommended the following attributes of an effective and efficient IT acquisition structure:

- *Fast* to incorporate current technology and make efficient use of Agency resources
- *Flexible* and adaptable to support rapid changes in technology and input from stakeholders about capability needs
- *Collaborative* to seek stakeholder involvement and input to be incorporated throughout

In a previous study completed in September 2016, Deloitte also provided key findings on commercial IT practices. Findings were taken into consideration when forming the proposals following in this appendix. The team recognizes that DoD is falling short of the preferred attributes outlined above with the current IT acquisition structure, in addition to multiple statutory, regulatory, and cultural issues that currently hinder an effective and efficient DoD acquisition environment that would benefit from reform.

Pain points

Acquisition Policy Environment. The DoD lacks a cohesive acquisition policy architecture and robust policy for software acquisition. Existing policies, to include tangential or supplemental policies that are integral to the operation of the defense acquisition system, do not fit well together and result in discrepancies, conflicts, and gaps. The defense acquisition system is monolithic, compiled in pieces as needs arose instead of as an integrated and evolving environment. It has proven unable to keep up with or remain ahead of the pace of change and technological advancements that require speed and agility. While it has regularly been revised, the changes tend to be conservative and incremental, requiring the agreement of too many parties protecting narrow interests and who are reluctant to relinquish authority or evolve. The system remains focused on oversight and situational control rather than insight and trust. The policies, practices, and documents become quickly entrenched and manifest themselves in the form of the Department's culture, leading to additional bureaucracy and decreased levels of organizational trust, that are difficult to rapidly reverse. Furthermore, the environment is risk averse, seeking out what is perceived to be the "safest" route to get things done, stifling the innovation and risk- taking that's required to maintain an advantage over adversaries.

As an example, one DoD weapons system program, which is implementing a DevSecOps pipeline to enable agile capability releases, informed us it took 18 months to get approval of a Test and Evaluation Master Plan (TEMP). The process within the TEMP drove them into sequential developmental and operational test - which is antithetical to continuous delivery under the DevSecOps concept.

Governance and Management. The Department lacks a strategic approach that recognizes software's criticality as the backbone and nervous system of the Department's mission and operations, often leading to widespread duplication of capabilities that could be consolidated and scaled at an enterprise level (whether Service-enterprise or OSD-enterprise). This absence of any strategy, compounded by a long-

standing lack of organizational trust in the Department, is exemplified by various situations in the software environment. For example, the lack of reciprocity on matters such as security standards, architecture, and compliance methods – my way is “better” (insert “less expensive,” “more efficient,” “more effective”) than your way, or, “our requirements / processes are unique,” regardless of validity. Further, the DoD issues separate policies on matters such as cloud, architecture, and risk management, with no unified approach at the strategic level. Management and governance of these matters takes the form of prolific numbers of senior working groups (or equivalent) that make few decisions but have frequent meetings. The DoD’s lack of an overarching strategic plan for key technologies, with a robust decision making framework that pushes responsibility and authority down to the lowest executable level, creates inefficiency, duplication, and waste.

Organization and Culture. The DoD lacks an organizational structure with clear responsibility and authority for software acquisition and management; there are confusing roles and responsibilities between DoD CIO, USD(A&S), and the DoD CMO. This state of ambiguity leads to overlap, inefficiency, and unnecessary bureaucracy; and it is replicated at the Service level. The result is a slow, rigid, siloed organization unable to adapt in the present and plan for the future in order to maintain competitive advantage. The DoD is not a change-ready environment and the acquisition system was not designed for rapid change. DoD employees tend to receive change mandates rather than participating in them. A case in point is that when DoD issues a policy, the Services will implement their own supporting version or “supplemental guidance”, which expands the policy and introduces multiple layers of bureaucracy, eliminating any semblance of flexibility that was intended by the original policy issued. For example, the Department issued DoD Instruction 5000.75 in February 2018, a tailored requirements and acquisition approach for business systems. Subsequently, the Army produced accompanying implementation guidance – 91 pages – which introduces additional forms, templates, processes, and time constraints.

Desired (end) state An acquisition system that enables rapid delivery of cost-efficient, relevant software capability through the application of creative compliance and fact-based critical thinking under a logical and minimal policy framework. The Department treats software as a national security capability and continuously retrains the workforce to be able to adapt to an ever-changing technology environment, embraces continuous collaboration between user and developers, embraces changing requirements, accepts and take risks, and deliver adversary- countering capabilities to the warfighter. Executing the approach requires an end state with an efficient contracting environment; a culture that rewards informed risk-taking and fast failures; the use of limits or guardrails instead of prescriptive requirements that limit creativity; outcome-based metrics that focus on value vs. execution against a plan; and a move away from traditional funding models and compliance-driven management.

Obstacles The Department operates with a general lack of urgency regarding its software – it is not recognized or treated as a national security capability. There is an aversion to informed risk-taking regarding new and innovative approaches to doing business and adopting emerging (or even simply relevant) technologies, even though it’s risky, or riskier, to continue using outdated technologies that are not secure or facing obsolescence in the face of evolving threats. Dramatic changes in policy or process are viewed as risky yet our current ways of operation are not despite a known degradation in strategic advantage previously enjoyed over adversaries. The inability to evolve and support rapid changes in technology and input from stakeholders about capability needs is bred through organizational silos and stovepipes that stifle the collaboration necessary to develop and operationalize software. Further, stakeholder involvement is limited by following restrictive controls, timelines, and processes in a sequential manner that impedes progress and results in a lower state of readiness. The duplication of

authorities and responsibilities among organizations both horizontally and vertically, within the defense acquisition system only exacerbates an already complex environment where a protectionist culture is ingrained and the workforce is not incentivized to change. In its endeavors to improve the status quo, "help" from Congress over the past decades translates into entrenched policies, processes, and procedures – "cultural norms" that are difficult to reverse.

Ideas for Change

*Acquisition Policy Environment***.** Define software as a critical national security capability under Section 805 of FY16 NDAA "Use of Alternative Acquisition Paths to Acquire Critical National Security Capabilities". Create an acquisition policy framework that recognizes that software is ubiquitous and will be part of all acquisition policy models. Recommend the creation of a clear, efficient acquisition path for acquiring non-embedded software capability. Reconcile and resolve discrepancies among supplemental policies that lead to conflicts. Consider the following tenets in development of a reformed software acquisition policy:

- Emphasis on quickly delivering working software
- Encourage projects and pilot efforts that serve to reduce risk and complexity - fail fast
- Reimagine program structures and program offices – i.e., accommodate move to "as-a-service" capabilities, agile, micro-services, and micro-applications
- Iterative, incremental development practices based on agile methods
- Rapid adoption of emerging technologies through piloting or prototyping
- Elimination of traditional A, B, C milestones; replaced by more sprint-centric decision points
- Elimination of arbitrary phases or merge phases to reflect rapid, agile development methods
- Tailor in requirements (statutory, regulatory – i.e., documentation) rather than tailor out; start with a minimum set
- No big-bang testing with sequential DT/OT; move to fully integrated test approaches driven by automated testing as well as regular, automated cybersecurity scanning
- Use a "guardrail-based" (upper / lower limit) approach for software requirements rather than defining every requirement up front
- Track value-driven outcome metrics which can be easily and continuously generated rather than measuring execution against a plan

*Governance and Management - Software as an Asset***.** Develop an enterprise-level Strategic Technology Plan that reinforces the concept of software as a national security capability. Include an approach for enterprise-level DevSecOps and other centralized infrastructure development and management, an approach for shared services, and applications management. The plan should recognize how disruptive technologies will be introduced into the environment on an ongoing basis. Ensure appropriate integration of a data strategy and the Department's Cloud Strategy. Examine a Steering Committee approach for management.

*Organization and Culture Reform***.** Examine roles and responsibilities with the intent to streamline reconcile, and resolve discrepancies for software acquisition and management among the DoD CIO, the USD(A&S) and the CMO. Re-focus the software acquisition workforce on teaming and collaboration, agility, improved role definition, career path advancement methods, continuing education and training opportunities, incentivization, and empowerment. Involve them in the change process.

Proposed Legislative/Regulatory Language

Any topic with an "*" was an idea derived either wholly or in part from engagements with the FY18 NDAA Section 873 and 874 agile pilot programs.

STATUTORY

TOPIC	OVERVIEW / ISSUE	STATUTE	PROPOSAL
Acquisition Strategy	Acquisition Strategies mandated by Section 821 of the FY16 NDAA for MDAPs, MAIS, and Major Systems, mandates content for acquisition strategies and authorities, the content in terms of how the provision is mandated does not allow for much flexibility and agility in content.	Section 821, FY16 NDAA	1. Eliminate for all except MDAPs 2.Keep overall definition of content with the A&S (listed as AT&L) in (b)(1) for consistency across the Services 3. Section (b)(2) authority should reside with the Service Chiefs
MDAPs	Specific to the establishment of cost, fielding, and performance goals for MDAPs under section 2448a of title 10 introduced by Section 807 of the FY17 NDAA. Does not distinguish software intensive programs from any other type of program. Also this provision was a reaction to programs not following guidance for affordability already established in the DODI 5000.02.	Title 10 § 2448a through Section 807 of the FY17 NDAA	Eliminate this provision from statute. There is policy which already exists that covers this in the DoDI 5000.02 (note: DSD just signed out a memo on this)
Nunn McCurdy	Nunn McCurdy is not an effective tool for restructuring MDAPs. There is little evidence to show that programs emerging from Nunn McCurdy breaches are drastically restructured for improvement, and rarely are programs cancelled. Some go through more than one Nunn McCurdy (though, a rare occurrence). Perception is a reporting / paperwork bureaucratic exercise that does not positively impact behavior.	10 U.S.C. § 2433	1.Consider elimination of Nunn McCurdy 2. Consider replacement of Nunn McCurdy w/ different focus and outcomes mandated (note: requires additional discussion)

Statutory Definition – Major System	The purpose and intent of this term is confusing. The term is separate and distinct from MDAP and MAIS. Typically ACAT II programs are affiliated with the "major system" designator, but ACAT II is a policy designation not a statutory designation. These systems do not do mandated statutory reporting like MDAPs. Per 10 U.S.C § 2302: "The term "major system" means a combination of elements that will function together to produce the capabilities required to fulfill a mission need. The elements may include hardware, equipment, software or any combination thereof, but excludes construction or other improvements to real property. A system shall be considered a major system if (A) the conditions of section 2302d of this title are satisfied, or (B) the system is designated a "major system" by the head of the agency responsible for the system." Dollar thresholds are defined in 10 U.S.C. 2302d; 41 U.S.C § 109 (the title 10 and title 41 thresholds are different)	10 U.S.C § 2302 10 U.S.C § 2302d 41 U.S.C § 109	Eliminate definition from title 10. Other agencies may use the definition in Title 41; recommend keeping Title 41.

Live fire / survivability / lethality testing*	There is no exemption for software-intensive programs to conduct survivability / lethality / live fire testing to move beyond LRIP OR to modify these requirements to reflect their nature as software intensive programs. Any covered system may require LFT&E. Includes major systems in the definition which may or may not be software programs (per the § 2302 definition). Otherwise, a waiver must be sent to the congressional committees before MS B. *Note: awaiting feedback / add'l info from AIAMD PMO*	Title 10 U.S.C. § 2366; and DoDI 5000.02	First, elimination of the Major Systems from Title 10 U.S.C. § 2302 helps to solve the identified challenges. Further, consider language for Title 10 2366a which allows exemption for software intensive programs, where DOT&E must justify adding the program for oversight with the MDA and must streamline the process. *Note: awaiting feedback / add'l info from AIAMD PMO*

Statutory DOT&E authority*	DOT&E has been able to essentially stop programs as they move through the development (acquisition) process. DOT&E testers are also not often SMEs in the systems they are conducting testing oversight on which negatively impacts testing. 1. Statutory authority assumes use of waterfall methodology; relies on infrequent, major test events instead of the continuous testing that agile uses 2. Also assumes a separate test team (and even organization) as opposed to testers being embedded in an agile team.	Title 10 U.S.C. § 2399	1. DOT&E oversight is only when requested by the SAE or USD(A&S), or Congressionally directed, unless MDAP. 2. DOT&E will utilize, to the greatest extent possible, test data collected through existing test methodologies present in the program and will not recommend or prescribe additional independent one-time test events. 3. One time IOT&Es or cybersecurity test events will not be recommended for software intensive systems unless in specific circumstances if warranted 4. Lead tester from either DOT&E or JITC (preferably both, if JITC is being used as test org) must be a subject matter expert in the subject being tested, similar to how qualified test pilots run test flights (health records, financial systems, etc.)
Clinger Cohen Act (CCA)*	1. CCA compliance process is outdated 2. Has become a time-consuming burden for programs that is layered on top of DoD's robust resources, requirements, and acquisition system. This renders many CCA requirements redundant with other laws, regulations, and policies. 3. Checklist-driven; provides limited strategic value; recognized as more of a hurdle than an enabler to capability delivery	40 U.S.C. § 1401(3)	Exempt the DoD from the Clinger Cohen Act, 40 U.S.C. 1401(3)

Business Systems Acquisition Reform *	DoD has three different governance entities: a business organization (CMO), an IT organization (CIO), and an acquisition organization (A&S) involved in providing oversight of business systems. Further, the annual certification requirement for DBS investments leads to unnecessary delays and is duplicative of the POM in the PPBE process	Title 10 U.S.C. § 2222	1. For the 4th estate combine all three authorities for DBS under the DoD CMO. After one year conduct assessment and make a determination if this should be applied to the Services as well. 2. Eliminate the separate funding certification process from 10 U.S.C. § 2222; or 3. If not eliminated, require the funding certification to be merged into the PPBE process
Configuration Steering Boards (CSB)	Must occur on at least an annual basis per the current statute (MDAPs). The Services tend to implement them for programs other than MDAPs based on 5000.02, and long-standing cultural factors.	FY 2009 NDAA, section 814; DoDI 5000.02	Other boards (or equivalent entities) established by the CAE or as delegated, the PEO or PM may fulfill the requirement of the CSB as long as the board (or equivalent entity) meets at least once a year and addresses the requirements in (c)(1).

Appropriations Accounts supporting IT Acquisition*	Agile acquisition is hindered by the appropriations environment. We must allow for more flexibility in appropriations account definitions for IT programs.	10 U.S.C. § 2214	Proposed language: "Funding for software solution acquisition does not adhere to the same standard development categories as other major programs. Funding approved by Congress for acquisition of a specific software solution may be used for research and development, production, or sustainment of that software solution. Provided that the software solution being acquired is the same software solution for which funding was appropriated, that funding may be accessed without respect to the appropriations account and without engaging in transfer of funds under the standard reprogramming process. If funding for one software solution is used for a different software solution, it must undergo a transfer of funds under the standard reprogramming process."
Expand FAR 39 to cover all IT purchasing regulations*	FAR 39 is too general. Further, for more streamlined acquisition of IT all rules governing it would be contained in one place. Purchasing speed is also too slow. Would allow for government-wide IT best practices and increase commodity / government-wide purchasing.	Title 49, Chapter 1, part 39	Expand the FAR 39 (Acquisition of IT) to allow for one area to drive technology purchases. Unless otherwise stated, no other FAR rules would apply

REGULATORY / POLICY

TOPIC	ISSUE	REG / POLICY	PROPOSAL
Earned Value Management (EVM) *	1. Earned Value Management (EVM) techniques are difficult (resource intensive) to implement; are neither designed nor well suited to effectively on measure an agile project; EVM cannot easily accommodate fluid requirements and shifting baselines. 2. EVM is lagging, not leading. 3. EVM does not measure product quality or user acceptance, which are hallmarks of the agile software development approach.	DFARS Subpart 234.201 DoDI 5000.02 Table 8 OMB Circular A-11 (not high priority)	Revise DFARS Subpart 234.201, DoDI 5000.02 Table 8, and OMB Circular A-11 to remove EVM requirement

FMR rules supporting IT acquisition*	IT/AIS that are not embedded in weapons systems and/or major end item procurements are budgeted according to the investment and expense criteria, these criteria do not enable agile acquisition or recognize the lifecycle nature of IT	FMR Volume 2A, Chapter 1, Section 010212(B)	Rewrite FMR Volume 2A, Chapter 1, Section 010212(B): 1. Acknowledge that, for the purpose of modifying or enhancing software, there is no technically meaningful distinction between RDT&E, Procurement, and O&M. 2. Eliminate the $250,000 barrier between expenses and investments (i.e., stop explicitly tying to a dollar threshold the determination of whether software is an expense or an investment. If the recommendations listed under "Appropriations Accounts Supporting IT Acquisition" are adopted, there should no longer be a need to make this determination for intra-program transfers.)

DoD Interoperability Policy*	Directs various things that should be reconsidered for IT/Software: 1. NR KPP required 2. DoD specific architecture products in the DoDAF format which are labor intensive and of questionable value 3. Interoperability Support Plans (ISPs) required, where DoD CIO can declare any ISP of "special interest" 4. Requires DT authority to provide assessments at MS C 5. Mandates JITC to do interoperability assessments for IT with "joint, multinational, and interagency interoperability requirements"	DoDI 8330.01	Direct revision of DoDI 8330.01 or potentially elimination of it
PfM Policy	Outdated (Sept 2008). Does not consider role of data and metrics, additional portfolios (like NC3) since 2008	DoDD 7045.20	Determine authority for policy; direct revision of DoDD 7045.20

Appendix B.2:

Appropriations Subgroup Report – released previously January 11, 2019

Appendix B.3: Contracting Subgroup Report

v0.2, 6 Feb 2019

The contacting challenges faced by the DoD today are almost entirely cultural. This premise is asserted by instances of excellence throughout the Department where effective contracting methods have been executed (DDS, DIU, Kessel Run).

That said, rather than attempting to battle each cultural challenge as they arise, it is easier to create a new modern acquisition platform from which to execute contracts that starts from a point of “how should it be done” as a product of “what should we be buying”.

The historical acquisition system was created to prevent fraud. The new priority is to establish technical superiority over our adversaries. While the prevention of fraud continues to be, and always will be, important, as a singular priority it serves to undermine the current identified need of speed and efficiency, which results in technical excellence for the Department.

Pain Points

Individual contracts are subject to review processes designed for large programs (of which they are likely enabling). This limits the agility of individual contract actions, even when modular contracting approaches are applied. In addition, the acquisition process is rigid and revolves around templates, boards, and checklists thus limiting the ability for innovation and streamlining execution.

Contracts focus on technical requirements instead of contractual process requirements. The contract should address overall scope (required capability), Period of Performance and price. The technical execution requirements should be separate and managed by the product owner or other technical lead.

Intellectual Property (IP) rights are often genetically incorporated without considering the layers of technology often applied to a solution. A single solution might include open source, proprietary software, and government custom code. The IP clauses should reflect all of the technology used.

Desired state

The desired state is an acquisition model that is liberated from the decades of policy and regulations that singularly focus on fraud prevention and provides for efficiency allowing the DoD to keep pace with the private sector and adversaries. This can be accomplished through a new authority Congress establishes a separate *new* authority for contracting for software development and IT modernization.

Obstacles

- Requires act of Congress ⇒ work with Armed Service Committees Staffers
- There is no infrastructure to support this ⇒ establish policy for guidance
- There are no Contracting Officers with specific certifications ⇒ Leverage current certifications

- Could cause confusion on implementation (what applies, what doesn't) ⇒ A&S issues guidance

Ideas for change

Congress establishes a separate *new* authority for contracting for software development and IT modernization

To address "Individual contracts being subject to review processes designed for large programs":

- Treat procurements as investments "what would you pay for a possible initial capability" (cultural).
- Manage programs at budget levels, allow programs to allocate funds at a project investment level (policy).
- Work with appropriators to establish working capital funds so that there is not pressure to spend funds quicker then you're ready (iterative contracts may produce more value with less money) (statute).
- Leverage incentives to make smaller purchases to take advantage of simplified acquisition procedures (cultural).
- Revise estimation models - source lines of code are irrelevant to future development efforts, estimations should be based on the team size, capability delivered, and investment focused (cultural).
- Allow for documentation and reporting substitutions to improve agility (agile reporting vs EVM) (cultural and EVM policy).
- Provide training to contracting officers, program managers, and leadership to understand the value and methods associated with agile and modular implementation (cultural).

To address "*Contracts focus on technical requirements instead of contractual process requirements*":

- Separate contract requirements (scope, PoP, and price) from technical requirements (backlog, roadmap, and stories) (cultural).
- Use statement of objectives (SOO) vs statement of work (SOW) to allow the vendor to solve the objectives how they are best suited (cultural).
- Use collaborative tools and libraries so that all content is available to all parties at all times (cultural).
- Use an agile process to manage structure and technical requirements (cultural).
- Establish a clear definition of done for the end of a sprint (code coverage, defect rate, user acceptance) (cultural).
- Use modular contracting to allow for regular investment decisions based on realized value (cultural).
- Streamline acquisition processes to allow for replacing poor performing contractors (cultural).
- Provide training to contracting officers, program managers, and leadership to understand the value and methods associated with agile and modular implementation (cultural).

To address "*Intellectual Property (IP) rights which are often genetically incorporated without considering the layers of technology often applied to a solution*":

- Establish clear and intuitive guidelines on how and when to apply existing clauses (cultural).
- Educate program managers and contracting officers on open source, proprietary, and government funded code (cultural).
- Have standard clause applications for each of the above that must be excepted vs accepted (cultural).

Proposed Legislative/Regulatory Language

(1) Authority

(a)Additional Forms of Transactions Authorized.—

The Secretary of Defense and the Secretary of each military department may enter into transactions (other than contracts, cooperative agreements, and grants) under the authority of this subsection for the purposes of acquiring Software Development and IT Modernization projects.

(1) The authority of this section—

(A)may be exercised for a transaction for a prototype project, and any follow-on production contract or transaction that is awarded pursuant to subsection (f), that is expected to cost the Department of Defense in excess of $100,000,000 but not in excess of $500,000,000 (including all options) only upon a written determination by the senior procurement executive for the agency as designated for the purpose of section 1702(c) of title 41, or, for the Defense Advanced Research Projects Agency or the Missile Defense Agency, the director of the agency that—

(i)

the requirements of subsection (d) will be met; and

(ii)

the use of the authority of this section is essential to promoting the success of the prototype project; and

(B)may be exercised for a transaction for a Software Development and IT Modernization project, and any follow-on production contract or transaction that is awarded pursuant to subsection (f), that is expected to cost the Department of Defense in excess of $500,000,000 (including all options) only if—

(i)the Under Secretary of Defense for Research and Engineering or the Under Secretary of Defense for Acquisition and Sustainment determines in writing that—

(I)

the requirements of subsection (d) will be met; and

(II)

the use of the authority of this section is essential to meet critical national security objectives; and

(ii)

the congressional defense committees are notified in writing at least 30 days before such authority is exercised.

(C) The authority of a senior procurement executive or director of the Defense Advanced Research Projects Agency or Missile Defense Agency under paragraph (2)(A), and the authority of the Under Secretaries of Defense under paragraph (2)(B), may not be delegated.

(D)Applicability of Procurement Ethics Requirements.—

An agreement entered into under the authority of this section shall be treated as a Federal agency procurement for the purposes of chapter 21 of title 41.

(2) Exercise of Authority by Secretary of Defense.—

In any exercise of the authority in subsection (a), the Secretary of Defense shall act through any element of the Department of Defense that the Secretary may designate.

(A)

Subsections (e)(1)(B) and (e)(2) of such section xxxx shall not apply to projects carried out under subsection (a).

(B)

To the maximum extent practicable, competitive procedures shall be used when entering into agreements to carry out the prototype projects under subsection (a).

(3) Appropriate Use of Authority.—

(1) The Secretary of Defense shall ensure that no official of an agency enters into a transaction (other than a contract, grant, or cooperative agreement) for a SW Development or IT Modernization project under the authority of this section unless one of the following conditions is met:

(A)

There is at least one nontraditional defense contractor or nonprofit research institution participating to a significant extent in the prototype project.

(B)

All significant participants in the transaction other than the Federal Government are small businesses (including small businesses participating in a program described under section 9 of the Small Business Act (15 U.S.C. 638)) or nontraditional defense contractors.

(C)

At least one third of the total cost of the prototype project is to be paid out of funds provided by sources other than other than [1] the Federal Government.

(D)

The senior procurement executive for the agency determines in writing that exceptional circumstances justify the use of a transaction that provides for innovative business arrangements or structures that would not be feasible or appropriate under a contract, or would provide an opportunity to expand the defense supply base in a manner that would not be practical or feasible under a contract.

(2)

(A)

Except as provided in subparagraph (B), the amounts counted for the purposes of this subsection as being provided, or to be provided, by a party to a transaction with respect to a SW Development or IT modernization project that is entered into under this section other than the Federal Government do not include costs that were incurred before the date on which the transaction becomes effective.

(B) Costs that were incurred for a SW Development or IT modernization project by a party after the beginning of negotiations resulting in a transaction (other than a contract, grant, or cooperative agreement) with respect to the project before the date on which the transaction becomes effective may be counted for purposes of this subsection as being provided, or to be provided, by the party to the transaction if and to the extent that the official responsible for entering into the transaction determines in writing that—

(i)

the party incurred the costs in anticipation of entering into the transaction; and

(ii)

it was appropriate for the party to incur the costs before the transaction became effective in order to ensure the successful implementation of the transaction.

(2) Payments

(a)Advance Payments.—

The authority provided under subsection (a) may be exercised without regard to section 3324 of title 31.

(b)Recovery of Funds.—

(1)

A cooperative agreement for performance of basic, applied, or advanced research authorized by a transaction authorized by subsection (a) may include a clause that requires a person or other entity to make payments to the Department of Defense or any other department or agency of the Federal Government as a condition for receiving support under the agreement or other transaction.

(2)

The amount of any payment received by the Federal Government pursuant to a requirement imposed under paragraph (1) may be credited, to the extent authorized by the Secretary of Defense, to the appropriate account established under subsection (f). Amounts so credited shall be merged with other funds in the account and shall be available for the same purposes and the same period for which other funds in such account are available.

(c)Support Accounts.—

There is hereby established on the books of the Treasury separate accounts for each of the military departments for support of Software Development and IT Modernization projects provided for in cooperative agreements containing a clause under subsection (d) and Software Development and IT Modernization projects provided for in transactions entered into under subsection (a). Funds in those accounts shall be available for the payment of such support.

(3)Education and Training.

The Secretary of Defense shall—

(1)

ensure that management, technical, and contracting personnel of the Department of Defense involved in the award or administration of transactions under this section or other innovative forms of contracting are afforded opportunities for adequate education and training; and

(2)

establish minimum levels and requirements for continuous and experiential learning for such personnel, including levels and requirements for acquisition certification programs.

(4)Regulations.—

The Secretary of Defense shall prescribe regulations to carry out this section.

(i)Protection of Certain Information From Disclosure.—

(1)

Disclosure of information described in paragraph (2) is not required, and may not be compelled, under section 552 of title 5 for five years after the date on which the information is received by the Department of Defense.

(2)

(A)

Paragraph (1) applies to information described in subparagraph (B) that is in the records of the Department of Defense if the information was submitted to the Department in a competitive or noncompetitive process having the potential for resulting in an award, to the party submitting the information, of a cooperative agreement for Software Development and IT Modernization projects authorized by transaction authorized by subsection (a).

(B) The information referred to in subparagraph (A) is the following:

(i)

A proposal, proposal abstract, and supporting documents.

(ii)

A business plan submitted on a confidential basis.

(iii)

Technical information submitted on a confidential basis.

(5)Records

Comptroller General Access to Information.—

(1)

Each agreement entered into by an official referred to in subsection (a) to carry out a project under that subsection that provides for payments in a total amount in excess of $5,000,000 shall include a clause that provides for the Comptroller General, in the discretion of the Comptroller General, to examine the records of any party to the agreement or any entity that participates in the performance of the agreement.

(2)

The requirement in paragraph (1) shall not apply with respect to a party or entity, or a subordinate element of a party or entity that has not entered into any other agreement that provides for audit access by a Government entity in the year prior to the date of the agreement.

(3)

(A)

The right provided to the Comptroller General in a clause of an agreement under paragraph (1) is limited as provided in subparagraph (B) in the case of a party to the agreement, an entity that participates in the performance of the agreement, or a subordinate element of that party or entity if the only agreements or other transactions that the party, entity, or subordinate element entered into with Government entities in the year prior to the date of that agreement are cooperative agreements or transactions that were entered into under this section or section xxxx of this title.

(B)

The only records of a party, other entity, or subordinate element referred to in subparagraph (A) that the Comptroller General may examine in the exercise of the right referred to in that subparagraph are records of the same type as the records that the Government has had the right to examine under the audit access clauses of the previous agreements or transactions referred to in such subparagraph that were entered into by that particular party, entity, or subordinate element.

(4)

The head of the contracting activity that is carrying out the agreement may waive the applicability of the requirement in paragraph (1) to the agreement if the head of the contracting activity determines that it would not be in the public interest to apply the requirement to the agreement. The waiver shall be effective with respect to the agreement only if the head of the contracting activity transmits a notification of the waiver to Congress and the Comptroller General before entering into the agreement. The notification shall include the rationale for the determination.

(5)

The Comptroller General may not examine records pursuant to a clause included in an agreement under paragraph (1) more than three years after the final payment is made by the United States under the agreement.

(6)Definitions.

In this section:

(1)

The term “nontraditional defense contractor” has the meaning given the term under section 2302(9) of this title.

(2)

The term “small business” means a small business concern as defined under section 3 of the Small Business Act (15 U.S.C. 632).

(a) Follow-on Contracts or Transactions.—

(1)

A transaction entered into under this section for a SW Development or IT modernization project may provide for the award of a follow-on contract or transaction to the participants in the transaction. A transaction includes all individual SW Development or IT modernization project subprojects awarded under the transaction to a consortium of United States industry and academic institutions.

(2)A follow-on production contract or transaction provided for in a transaction under paragraph (1) may be awarded to the participants in the transaction without the use of competitive procedures, notwithstanding the requirements of section 2304 of this title, if—

(A)

competitive procedures were used for the selection of parties for participation in the transaction; and

(B)

the participants in the transaction successfully completed the prototype project provided for in the transaction.

(3)

A follow-on production contract or transaction may be awarded, pursuant to this subsection, when the Department determines that an individual prototype or prototype subproject as part of a consortium is successfully completed by the participants.

(4)

Award of a follow-on production contract or transaction pursuant to the terms under this subsection is not contingent upon the successful completion of all activities within a consortium as a condition for an award for follow-on production of a successfully completed prototype or prototype subproject within that consortium.

(5)

Contracts and transactions entered into pursuant to this subsection may be awarded using the authority in subsection (a), under the authority of chapter 137 of this title, or under such procedures, terms, and conditions as the Secretary of Defense may establish by regulation.

(b)Authority To Provide Prototypes and Follow-on Production Items as Government-furnished Equipment.—

An agreement entered into pursuant to the authority of subsection (a) or a follow-on contract or transaction entered into pursuant to the authority of subsection (f) may provide for follow-on items to be provided to another contractor as Government-furnished equipment.

Proposed Policy for Implementation:

SOFTWARE DEVELOPMENT USING AGILE BEST PRACTICES.

(a) In General.—This policy governs software development activities within the Department of Defense or military departments to be developed using agile acquisition methods as provided for under NDAA 2020 Section XXX.

(b) Streamlined Processes.—Software development activities identified under subsection (a) shall be developed without incorporation of the following contract or transaction requirements:

(1) Earned value management (EVM) or EVM-like reporting.

(2) Development of integrated master schedule.

(3) Development of integrated master plan.

(4) Development of technical requirement document.

(5) Development of systems requirement documents.

(6) Use of information technology infrastructure library agreements.

(7) Use of software development life cycle (methodology).

(c) Roles And Responsibilities.—

(1) IN GENERAL.—Selected activities shall include the following roles and responsibilities:

(A) A program manager that is authorized to make all programmatic decisions within the overarching activity objectives, including resources, funding, personnel, and contract or transaction termination recommendations.

(B) A product owner that reports directly to the program manager and is responsible for the overall design of the product, prioritization of roadmap elements and interpretation of their acceptance criteria, and prioritization of the list of all features desired in the product.

(C) An engineering lead that reports directly to the program manager and is responsible for the implementation and operation of the software.

(D) A design lead that reports directly to the program manager and is responsible for identifying, communicating, and visualizing user needs through a human-centered design process.

(2) QUALIFICATIONS.—Shall establish qualifications for personnel filling the positions described in paragraph (1) prior to their selection. The qualifications may not include a positive education requirement and must be based on technical expertise or experience in delivery of software products, including agile concepts.

(3) COORDINATION PLAN FOR TESTING AND CERTIFICATION ORGANIZATIONS.—The program manager shall ensure the availability of resources for test and certification organizations support of iterative development processes.

(d) Plan.— DPAP shall develop a plan which shall include the following elements:

(1) Definition of a product vision, identifying a succinct, clearly defined need the software will address.

(2) Definition of a product road map, outlining a noncontractual plan that identifies short-term and long-term product goals and specific technology solutions to help meet those goals and adjusts to mission and user needs at the product owner's discretion.

(3) The use of a broad agency announcement, other transaction authority, or other rapid merit-based solicitation procedure.

(4) Identification of, and continuous engagement with, end users.

(5) Frequent and iterative end user validation of features and usability consistent with the principles outlined in the Digital Services Playbook of the U.S. Digital Service.

(6) Use of commercial best practices for advanced computing systems, including, where applicable—

(A) Automated testing, integration, and deployment;

(B) compliance with applicable commercial accessibility standards;

(C) capability to support modern versions of multiple, common web browsers;

(D) capability to be viewable across commonly used end user devices, including mobile devices; and

(E) built-in application monitoring.

(e) Program Schedule.—Shall ensure that each activity includes—

(1) award processes that take no longer than three months after a requirement is identified;

(2) planned frequent and iterative end user validation of implemented features and their usability;

(3) delivery of a functional prototype or minimally viable product in three months or less from award; and

(4) follow-on delivery of iterative development cycles no longer than four weeks apart, including security testing and configuration management as applicable.

(f) Oversight Metrics.—Shall ensure that the selected activities—

(1) use a modern tracking tool to execute requirements backlog tracking; and

(2) use agile development metrics that, at a minimum, track—

(A) pace of work accomplishment;

(B) completeness of scope of testing activities (such as code coverage, fault tolerance, and boundary testing);

(C) product quality attributes (such as major and minor defects and measures of key performance attributes and quality attributes);

(D) delivery progress relative to the current product roadmap; and

(E) goals for each iteration.

(g) Restrictions.—

(1) USE OF FUNDS.—No funds made available for the selected activities may be expended on estimation or evaluation using source lines of code methodologies.

(2) CONTRACT TYPES.—The Secretary of Defense may not use lowest price technically acceptable contracting methods or cost plus contracts to carry out selected activities under this section, and shall encourage the use of existing streamlined and flexible contracting arrangements.

(h) Definitions.—In this section:

(1) AGILE ACQUISITION.—The term “agile acquisition” means acquisition using agile or iterative development.

(2) AGILE OR ITERATIVE DEVELOPMENT.—The term “agile or iterative development”, with respect to software—

(A) means acquisition pursuant to a method for delivering multiple, rapid, incremental capabilities to the user for operational use, evaluation, and feedback not exclusively linked to any single, proprietary method or process; and

(B) involves—

(i) the incremental development and fielding of capabilities, commonly called “spirals”, “spins”, or “sprints”, which can be measured in a few weeks or months; and

(ii) continuous participation and collaboration by users, testers, and requirements authorities.

Appendix B.4:

Data and Metrics Subgroup Report – released previously January 11, 2019

Appendix B.5: Infrastructure Working Group Report

v0.1, 11 February 2019

Despite several years of effort to "move DoD to the cloud," significant friction still exists for the DoD to easily leverage the required compute, storage, and bandwidth infrastructure that the commercial world so readily enjoys. The major obstacle is not at all technical, but is broadly one of accessibility: the ability to specify, contract for, pay for, connect to, secure, and continuously monitor sufficient modern computing infrastructure. Modern computing infrastructure refers primarily to cloud-based computing technologies and stacks. "Cloud-based" does not necessarily presuppose commercial cloud, but could also be on premises or hybrid cloud solutions. Similarly, "computing technologies and stacks" can run the full spectrum from infrastructure, to platform, to function, to software as a Service (IaaS, PaaS, FaaS, SaaS).

Pain Points and Obstacles

How much cloud do I need? Countless developers and IT professionals have wrestled with this question, and often the answer is to "dive in," move some apps, see what is needed, and then scale and tweak from there. The Department's culture hampers our ability to even take a "leap of faith" like this. We must be able to precisely size and cost our cloud requirements before ever starting to experiment or prototype. It should become more clear why this analysis paralysis exists as the below pain points are outlined and considered.

How do I buy cloud? Oh, just head on over to FedRAMP, pick an approved provider, sign up and you're on your way… FedRAMP? Is that a cloud? What about GovCloud, cloud.gov (not the same thing by the way), and MilCloud (is that version 1.0 or 2.0?)? What's the difference between AWS GovCloud and Azure Government? Can I just sign up with a credit card like a normal private citizen and start hosting my compute and data in the cloud? Sadly, the answer is a definitive and resounding NO! Even if you know which "government-approved" cloud you're moving to, it's just not easy to contract for it or buy it.

There is not space here to answer all these rhetorical questions. For a good description of the difficulty of buying cloud, please refer to the DoD Cloud Acquisition Guidebook at https://www.dau.mil/tools/t/DoD-Cloud-Acquisition-Guidebook. Here the Defense Acquisition University (DAU) outlines the multiple activities that need to be accomplished to contract for cloud services. Starting with the dreaded IT Business Case Analysis (BCA), moving on to applying the DoD Cloud Security Requirements Guide (SRG - more on this soon), to getting an Authority to Operate (ATO), ensuring DISA approves of your Boundary Cloud Access Point (BCAP) and your Cyber Security Service Provider (CCSSP), and lastly to applying the DFARS supplementary rule to your cloud contact. No friction here right?

How do I know my cloud is secure? Easy. FedRAMP pre-evaluates and approves Cloud Service Providers (CSSPs) for Information Impact Levels (IILs) 2, 4, 5, and 6 (don't ask about levels 1 and 3; apparently we over specified and they aren't necessary any longer). Whew, now things are making sense… Not so fast, the FedRAMP IILs are for US Government cloud use, but not DoD![1] We need FedRAMP+ for DoD use, and DISA doesn't evaluate Cloud Service Providers (CSPs), only Cloud Service Offerings (CSOs). Huh? Be sure to go through the DoD Cloud Computing SRG, ensure those extra security controls are in place for FedRAMP+, and you're on your way. Again,

[1] Don't ask… we know DoD is part of the US Government.

not so fast Program Manager (or small business owner)! How are you and your customers going to access the fancy new cloud you just finally got on contract?

How do I access my cloud? The cloud, sort of by definition, implies ease of access, right? The National Institute of Standards and Technology (NIST) definition in SP 800-145 defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." Well, if you're a DoD user, you need to ensure you've got a BCAP in place between your application/service and your users. It's OK and accurate to immediately envision bottleneck and single point of failure here.[2] Mis-configuring and under-provisioning BCAPs is the norm rather than the exception, so even with all that compute and storage in the cloud that you somehow ran the contracting gauntlet to get, you're going to severely lack adequate bandwidth and likely suffer from significant latency. Friction++.

How do I pay for cloud? The best part of cloud computing is that I can only pay for what I use. A true consumption-based cost model. Just like a utility. Not so for Government and DoD though. The Anti-Deficiency Act doesn't allow us to pay for cloud computing like a utility. A common way around this is to pay a third party contractor to buy the cloud service for us. This results in a situation where we estimate the highest charges we could ever incur in a year, add a bit of padding to that (say 20-30%), pay the third party, and we've paid for our cloud. What happens if we don't use it all up by the end of the year? Nothing (i.e. no refunds). Money spent. The third party contractor makes (quite?) a bit of extra profit for "taking the risk off the government." So much for consumption-based payments.

Desired State

The ability to provision, pay for, consume, access, and monitor cloud computing (compute, storage, and bandwidth) the same way any commercial organization does. It is understood that there are unique DoD security requirements, but that should only affect cloud pricing (say 1.5 to 2 times commercial, worst case), and not any of the other procedures to easily access cloud computing technologies and resources.

Obstacles

Significant obstacles remain to easily leverage commercially equivalent compute, storage, and bandwidth infrastructure. Contracting, security procedures (not necessarily requirements), network access (i.e. a modern technological approach to BCAP), and billing all loom large. The most important of these is the DoD's inability to contract and pay for cloud computing on a consumption basis.

Ideas for change

Establish a DoD enterprise ability to procure, provision, pay for, and use cloud that is no different from the commercial entry points for cloud computing. The Joint Enterprise Defense Infrastructure (JEDI) Cloud initiative is a bold attempt at this solution and should be awarded. Cloud.gov (which is ironically hosted in GovCloud) is another promising program that is already very straightforward to provision and buy, but is limited to IIL 2 data and applications. The objective cloud procurement and billing contract must include the ability to truly pay for consumption of cloud services and not be

[2] There are better ways to do this, like zero trust networks. The commercial world has some really good examples and architectures that don't require this man-in-the-middle attack called a BCAP which actually breaks end-to-end encryption by design…

artificially limited by the Anti-Deficiency Act. Modern software demands the ability to consume and pay for cloud services just as we do any other utility.

In addition to this, the DoD should establish a common, enterprise ability to develop software solutions in the “easy-to-acquire-and-provision” cloud that is fully accredited by design of the process, tools, and pipeline. Said another way, the DoD should stop the security accreditation of individual applications, but *should instead invest in accrediting the ability to produce software*. The pipeline, automated tooling, procedures, and operational monitoring and auditing of software should be the focus and target of security accreditation, not each individual application and version of an operating system or application.

Another essential and necessary, though not sufficient, change that must occur is to adopt modern commercial approaches to software and system security in the cloud that does NOT involve BCAPs, Internet Access (choke) Points (IAPs), or CSSPs that cannot be performed entirely by trusted commercial entities. The DoD must adopt modern cloud security approaches such as zero trust networks[3], micro-segmentation, and eliminate the perimeter approach to network security and trust that is based on assigned IP address or network connection point. Perimeter-based security cannot scale to accommodate the bandwidth, traffic, and latency demands of modern cloud access, applications, and services. Furthermore, it is a failed architectural practice that has proven to be readily exploitable by adversaries and is especially vulnerable to insider threats.

[3] https://www.oreilly.com/library/view/zero-trust-networks/9781491962183/ch01.html

Proposed Legislative/Regulatory Language

Provide explicit policy and guidance that allows cloud computing resources to be acquired and paid for by consumption and on demand. This will require an amendment to or reconsideration of the Anti-Deficiency Act to consider compute, storage, and bandwidth as a utility.

The following are excerpts from the 809 Panel report that can help address the recommendations made here.

> Panel 809 - Volume 3 of 3 January 2019 Implementation Legislative Branch Revise appropriation law and budgeting rules to address the unique aspects of buying consumption-based solutions. Recommendation 49 provides the flexibility necessary for these changes. Executive Branch Create a new subcategory of services called consumption-based solutions in FAR Part 37, Service Contracting, and add a reference (pointer) in FAR Part 39, Acquisition of Information Technology.43 Agency-specific regulations, policies, and guidance regarding service contracting are not applicable to contracts for consumption-based solutions or hybrid contracts when the primary purpose is to procure consumption-based solutions. The following is the definition of consumption-based solutions: Any combination of hardware/equipment, software, and labor/services that together provide a seamless capability that is metered and billed based on actual usage and predetermined pricing per resource unit, and includes the ability to rapidly scale capacity up or down. Consumption-based solutions must be measurable/meterable on a frequent interval customary for the type of solution (e.g., hourly, daily, weekly). The contractor is required to notify the government when consumption reaches 75 percent and 90 percent of the contract funded amount. New services or features can be added to contracts for consumption-based solutions at the discretion of the contracting officer without conducting a new competition, provided the amount of these new services or features does not exceed 25 percent of the total contract value. Update the Product Service Code (PSC) data architecture to accommodate consumption-based solutions as a new data type. **Add a new contract type called fixed-price resource units to FAR Subpart 16.2. The fixed-price resource units contract type: Establishes a fixed price per unit of measure (e.g., one hour of computing resource as shown in Table 3-1 below). Sets a ceiling for the overall contract value against which consumption of individual resource line items will be charged. Is the preferred contract type for consumption-based solutions, and when used for those procurements should not require special approvals. Can be incrementally funded.**
>
> 43 The term consumption-based solutions was chosen in favor of consumption-based services because lessons learned from utility services contracting indicated that including the word “services” would cause confusion and result in attempts to improperly apply all Service Contracting (i.e., FAR Part 37) rules to the new purchasing category.
>
> Sets a maximum unit price for each resource unit and captures price reductions when commercial catalog prices are reduced. Is permitted for use under commercial item/service acquisition in FAR Part 12: Acquisition of Commercial Items.
>
> Develop IT solutions training and a corresponding certification/designation for DoD acquisition professionals based on the existing DITAP, which is part of the FAC-C Core-Plus specialization in digital services. Refresh training content and individual certifications at least annually. Include instruction on how to conduct cost/price analysis for consumption-based solutions. This training curriculum is for commercial IT solutions and does not apply to weapon systems acquisition.

Note: Draft regulatory text can be found in the Implementation Details subsection at the end of Section 3. SECTION 3: IT PROCUREMENT Due to the limited interaction between commercial and DoD information technology (IT) markets, the two now operate at substantially different paces of technological advancement. Because the commercial IT market has outpaced the DoD market for decades, DoD regularly acquires outdated and inferior technology, often at higher prices and slower rates. DoD's slower acquisition pace has a direct effect on warfighting capability in a defense era defined by technological edge. Warfighters, and their support commands, are often operating with less functionality and at higher operating costs. This market 1 GAO, Weapon System Acquisitions: Opportunities Exist to Improve the DOD's Portfolio Management, GAO-15-466, August 2015, Highlights, accessed November 26, 2018, https://www.gao.gov/assets/680/672205.pdf. Report of the Advisory Panel on Streamlining and Codifying Acquisition Regulations Volume 3 of 3 | January 2019 Page EX-4 | Volume 3 Executive Summary segregation is caused by the vastly different way in which DoD and the wider federal government acquire IT. Rather than operating in the private-sector market of readily available options, DoD often creates detailed, intricate and unique requirements for its IT systems and services. DoD must acknowledge its acquisition system suffers from processes and procedures that are obsolete, redundant, or unnecessary and work to move quickly enough to keep pace with private-sector innovation. The recommendations in Section 3 offer strategies for transforming DoD's IT acquisition from both the top down and bottom up. Strategic revisions to how DoD understands and acquires IT are integrated with smaller-scale changes that restore efficiency to routine processes that have become bogged down by layers of bureaucracy. None of the actions recommended in Section 3 alone will solve the challenges associated with IT market segregation; however, together they offer a series of changes that can better align DoD acquisition with private-sector practices. Allowing DoD to buy in a manner similar to private-sector companies will reduce barriers to sellers in the marketplace. Rec. 43: Revise acquisition regulations to enable more flexible and effective procurement of consumption-based solutions. Rec. 44: Exempt DoD from Clinger–Cohen Act Provisions in Title 40. Rec. 45: Create a pilot program for contracting directly with information technology consultants through an online talent marketplace.

Appendix B.6: Sustainment / Modernization Subgroup Report

v0.2, 11 Feb 2019

Improving the materiel readiness of our fielded weapon systems and equipment is an imperative across the Department in accordance with the new National Defense Strategy.[4] The time is now to shift from our traditional, hardware-centric focus and identify what core[5] means for software intensive weapon systems and associated software engineering capabilities. Software is a foundational building material for the engineering of systems, enabling almost 100 percent of the integrated functionality of cyber-physical systems, especially mission- and safety-critical software-reliant systems. More simply, these systems cannot function without software.

For fielded weapon systems and military equipment, software life-cycle activities follow somewhat predictable cycles of corrective, perfective, adaptive, and preventative modifications while major modifications drive new periods of development. Software development activities, even those following agile methods, encounter a phase where the program transitions from adding new features to supporting and sustaining day-to-day use and operations. At that point, development changes and signals a move to "sustainers" within the organic industrial base. Therefore, sustainment may be defined as the sum of all actions and activities necessary to support a weapon system or military equipment after it has been fielded.

Prioritizing the transition to software sustainment during requirements and engineering development is critical to timely, effective, and affordable sustainment, regardless of how software engineering organizations are structured and resourced. Software sustainment organizations must be engaged and embedded at the earliest design stages to ensure we can keep pace with new capabilities as systems become operational. Lastly, access to software source code, emphasizing an early focus on designing for sustainment, and investment into establishing and modernizing system integration laboratories, are just a few of the challenges faced by the DoD software enterprise.

Pain points

Applying a hardware maintenance mindset to software hinders the DoD's ability to better leverage the organic software engineering infrastructure. DoD maintenance policies and maintenance-related Congressional statutes have traditionally been optimized for hardware and are difficult to change due to long standing policies, practices, inertia, and incentives. The goal of hardware maintenance is to repair and restore form, fit, and function. This mindset does not align well with the ever evolving nature of software. The scope of software engineering for sustainment mitigates defects and vulnerabilities, fact-of-life interface changes, and add new enhancements. Software is never done and any time it is "touched," it triggers the software engineering development life cycle which produces a new configuration. Therefore, any system that is dependent on software to remain operational, is always in a state of continuous engineering during sustainment (or O&S phase of the life cycle).

DoD's acquisition process is not emphasizing an upfront focus on design for software sustainment and a seamless transition to organic sustainment. It is critical that software be designed to be more affordably sustained with high assurance and the ability to integrate changes and enhancements

[4] "Summary of the 2018 National Defense Strategy" (Washington, DC: Department of Defense, 2018), https://dod.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf.

[5] As defined in 10 USC 2464, *Core logistics capabilities.*

more rapidly to provide a continual operational capability to the warfighter. Moreover, software must be decoupled from hardware to the greatest extent possible in order to enable leveraging rapid and continuous hardware improvements. We need to place increased emphasis in acquisition on designing in software sustainability with a consistent emphasis on how DoD contracts for software as well as the span of requirements, architecture, design, development, and test. Additionally, this includes making provisions for timely access to the necessary range of software technical data to enable timely and effective organic software engineering and rapid re-hosting. It is essential that the DoD and industry work collaboratively to meet the increasing software sustainment demand.

Public Private Partnerships (PPPs) provide one means to leverage DoD and industry capabilities as a team to deliver warfighter capability. However, PPPs and other options are not being considered up front and leveraged across DoD as an inherent element of the acquisition and engineering strategy of programs. This team strategy may facilitate mutual access to the technical data inherent in executing the software development life cycle.

Limited visibility of the DoD organic software engineering infrastructure, capabilities, workload, and resources. Title 10 USC 2464 establishes a key imperative for DoD to establish core Government Owned Government Operated (GOGO) capabilities as a ready and controlled source of technical competence and resources for national security. DoD's focus has traditionally been on hardware and therefore there has seen significant Service and DoD enterprise focus on hardware GOGO capabilities and infrastructure for core. However, there has been significantly less upfront acquisition focus and visibility on what core means for software intensive systems and the associated GOGO software engineering capability. For the traditional DoD hardware-centric model, core capability is based on individual weapon systems or platforms at the depot level. All systems operate interdependently in a net-centric environment, where force structure and execution of mission capabilities are products of a system-of-systems capability. In a software intensive environment "Go to War" analysis of what core means as it relates to software requires more strategic thinking about core than just focusing on individual weapon systems or platforms (aircraft, ship, tank, etc.) as hardware. The hardware-centric focus on weapon systems likely underestimates the scope and magnitude of what should be considered a core requirement in a software intensive systems operational environment.

Desired State. Require government integrated software sustainment participation from the very beginning of development activities.

Ideas for Change

- Title 10 USC 2460 should be revised to replace the term software maintenance with the term software sustainment and a definition that is consistent with a continuous engineering approach across the lifecycle.
- DoD should establish a capability for visibility into the size and composition of DoD's software sustainment portfolio, demographics, and infrastructure to better inform enterprise investment and program decisions.
- A DoD working group should be established to leverage on-going individual Service efforts and create a DoD contracting and acquisition guide for software and software sustainment patterned after the approach that led to creation of the DoD Open Systems Architecture Contracting Guide.

- Acquisition Strategy, RFP/Evaluation Criteria, and Systems Engineering Plan should address software sustainability, re-hosting, and transition to sustainment as an acquisition priority. The engineering strategy and plan should engage software sustainment engineers upfront and co-locates government software sustainment engineers on the contractor software development teams to enable effectively and timely transition to an organic sustainment capability.
- The definition of "core capabilities" in 10 USC 2464 should be revisited in light of warfighter dependence on software intensive systems to determine the scope of DoD's core organic software engineering capability, and we should engage with Congress on the proposed revision to clarify the intent and extent of key terminology used in the current statute.
- The DoD should revise industrial base policy to include software and DoD's organic software engineering capabilities and infrastructure. Start enterprise planning and investment to establish and modernize organic System Integration Labs (SILs), software engineering environments, and technical infrastructure; invest in R&D to advance organic software engineering infrastructure capabilities.
- **Revisions to the Definition of Depot-Level Maintenance and Repair**

Section 2460 of title 10, United States Code, is amended—

(1) in subsection (a), by striking "maintenance classified by the Department of Defense as of July 1, 1995" and inserting "sustainment and software engineering (including requirements definition, architecture, design, development and coding, integration and test, and all other related software engineering-related activities) for fielded software to correct faults and vulnerabilities, make continuous capability upgrades, improve performance or other attributes, or adapt the product to a modified environment without regard to the type of system, funding source, means (organic software engineering, contractor, Public Private Partnership, etc.), and organizational location and alignment"

§2460. Definition of depot-level maintenance and repair

(a) IN GENERAL.—In this chapter, the term "depot-level maintenance and repair" means (except as provided in subsection (b)) material maintenance or repair requiring the overhaul, upgrading, or rebuilding of parts, assemblies, or subassemblies, and the testing and reclamation of equipment as necessary, regardless of the source of funds for the maintenance or repair or the location at which the maintenance or repair is performed. The term includes (1) all aspects of software ~~maintenance classified by the Department of Defense as of July 1, 1995,~~ sustainment and software engineering (including requirements definition, architecture, design, development and coding, integration and test, and all other related software engineering-related activities) for fielded software to correct faults and vulnerabilities, make continuous capability upgrades, improve performance or other attributes, or adapt the product to a modified environment without regard to the type of system, funding source, means (organic software engineering, contractor, Public Private Partnership, etc.), and organizational location and alignment, as depot-level maintenance and repair, and (2) interim contractor support or contractor logistics support (or any similar contractor support), to the extent that such support is for the performance of services described in the preceding sentence.

(b) EXCEPTIONS.—(1) The term does not include the procurement of major modifications or upgrades of weapon systems that are designed to improve program performance or the nuclear refueling or defueling of an aircraft carrier and any concurrent

complex overhaul. A major upgrade program covered by this exception could continue to be performed by private or public sector activities.

(2) The term also does not include the procurement of parts for safety modifications. However, the term does include the installation of parts for that purpose.

Appendix B.7: Requirements Subgroup Report

v0.7, 7 Feb 2019

The Department of Defense (DoD) in 2003 institutionalized the identification and validation of requirements via the Joint Capability Integration and Development System (JCIDS). Created to support the statutory responsibility of the Joint Requirements Oversight Council (JROC), it is one of three processes (Acquisition, Requirements, and Funding) that support the Defense Acquisition System (DAS). Considered revolutionary in its design, moving DoD from a threat-based to a capability-based model, it has begun to show its age in today's era of software-intensive systems intending to leverage agile software practices. These evolving agile practices upend traditional industrial-age process attempts to credibly and accurately predict a future 15-20 years away, necessitating unimaginable precision and foresight upfront in support to capability development. The requirement process, writ large, must adapt to support delivering capabilities at the speed of relevance; processes, cultures, and expectations of the Service and Joint Force requirement communities.

Pain points

A byproduct of top-level requirement flow down is rigidity and over specificity at the derived requirements level, that greatly hinders agile software design. Capability validated by the JROC does not proscribe requirement allocation to either hardware or software solutions. However, the resulting flowdown of derived requirements incorporated into the source selection/contract award and the subsequent allocation of these between hardware and software by the prime can ultimately discourage software design flexibility. The decisions, often made years before software coding even begins, locks the prime and the government into a proscribed path that often does not produce the desired warfighter capability within the needed time frame. Preserving software design flexibility must be a key component throughout the requirements validation process. "Requirers" will need to learn to settle for "less" not "more" at capability need inception.

Too often exquisite requirements, intended to be 100 percent correct, are levied on a system that in turn drives extensive complex software requirements and design, affecting development, integration, and system test. Today's requirements process more closely mimics the "big-bang" theory often vilified by industry, government, and Congress. As the warfighting community loses faith in the acquisition community's ability to meet their commitments through timely incremental improvements, the temptation to "gold-plate" a requirement becomes more prevalent. Likewise, as the acquisition community is forced to defend shifting warfighter priorities in budget deliberations and Congressional engagements, the temptation to "lock requirements down early" permeates acquisition strategies. With both of these choices in play, exquisite requirements must be described perfectly at capability inception in order to maintain a low-risk acquisition program - obviously an impossible outcome.

Data sets are siloed within programs - a common Law of Requirements is that programs of record (PoR) try to avoid dependencies with other PoRs. By tying SW to a PoR, it becomes nearly impossible to transfer that code across systems and data environments. Data "lakes," "pools," and "ponds" will be the foundation for future weapon system data repositories, and the requirements process must be flexible enough to accommodate this new archetype. Breaking

from the past mold of tying software code to a program of record and a specific data environment frees the program manager from the arduous task of integrating seams across multiple PORs.

Example. The Navy operates forward at sea and on-shore at maritime operations centers (MOCs). Command and control between sea and shore is a key aspect of how they fight – they need shared battlespace awareness at aligned actions across distributed units at best. However, the systems afloat and ashore are not always the same because ships need systems that are hardened for combat at sea. If a new algorithm can help manage supply and logistics on the cloud ashore, it may not run the same at sea because different system exists afloat. Extrapolating across Services, the USAF writes an algorithm to optimize F-16 maintenance, however it is highly unlikely that the Navy can pick it up and apply it to F-18s. This depends on the vertical integration of the algorithm, data, and system (PoR).

Desired State. Go from Sailor (Airman, Rifleman, etc.)-stated need to software delivery in their hands within days to support future conflicts. This necessitates a process for concept/requirements determination/setting that takes advantage of the agility in software development and software products to increase the agility and modifiability in our systems. Requirements flow down must also maintain a broad-based approach into the lowest levels of design. We also note that one of the overarching agile principles is that “increments are small.” Fast requirements, fast deployments and fast test cycles for usefulness are tough to accomplish with huge, monolithic software projects. Start small, stay small! Finally, recognizing that documenting and contracting for a moving target is not easy but must be done.

Obstacles. Breaking the tyranny of siloed PoRs will require a concerted effort across the Department, Combat Support Agencies, and will require Congressional engagement and support. Considerable cultural barriers must also be overcome as the algorithms themselves become capability, and the methods used to document, validate, and maintain currency enter the mainstream. Complexity and dependencies among multiple elements prevent widespread usage of Family-of-Systems (FoS) and System-of-Systems (SoS) requirement documents. Government requirements and acquisition communities take on extra oversight burden when they take a FoS or SoS approach because they have to manage all the pieces coming together effectively. Lastly, current statutory guidance does not promote, encourage, or reward the use of agile software development practices or environments.

Ideas for Change

- The Joint Staff should consider revising JCIDS guidance to separate functionality that needs high variability from the functionality that deemed “more stable” (e.g., types of signals to analyze vs. allowable space for the antenna). Then implement a “software box” approach for each, one in which the contours of the box are shaped by the functionality variability

- OSD should consider identifying automated software generation areas that can apply to specific domains

- The Joint Staff should consider revising JCIDS guidance to document stable concepts, not speculative ideas.

  - Specifying needed capabilities is important up front, however it must be acknowledged that initial software requirements need to be "just barely good enough" for the situation at hand or, in other words, "document late"

  - Acknowledge that software requirement documents will iterate, iterate, iterate. JCIDS must change from a "one-pass" mentality to a "first of many" model that is inherently agile delegating approval to the lowest possible level

- The DoD should consider instituting a distributed model-based approach to requirements development extended across the enterprise

  - The model should be used to develop result-based metrics for requirement evaluation

- The Joint Staff should consider revising JCIDS guidance to focus on user needs, bypassing the JCIDS process as needed to facilitate rapid software development. Guidance should specifically account for user communities (e.g. Tactical Action Officer (TAO), Maritime Operations Center (MOC) director) that do not have one specific PoR assigned to them, but use multiple systems and data from those systems to be effective

- OSD and the Joint Staff should consider creating "umbrella" software programs around "roles" (e.g. USAF Kessel Run)

Potential DRAFT Legislative/Regulatory Language

No recommendations at this time.

Appendix B.8: Security Accreditation/Certification Subgroup Report

v0.1, 28 Jan 2019

The Department's current Security Certification and Accreditation (C&A) process is a complicated and time-consuming process that is measured in months and years. The process is typically seen as a serial process that occurs after development with a checklist mentality. While this fits with a waterfall approach to development, the Department is changing to an agile, DevSecOps approach. The overall security paradigm must change from one where updates to software happen optimistically on a yearly basis to one where software is updated weekly or daily in response to emerging threats and this is recognized as more secure than the slow, static process. Additionally, we must strive to accredit the process, tools, and platforms to allow and enable continuous authority to operate (ATO) when software changes meet the required thresholds.

Pain points

Complex, time-consuming, and misapplied process. Although developing and operating software securely is a primary concern, the means to achieve and demonstrate security is overly complex and hampered by inconsistent and outdated/misapplied policy and implementation practices (e.g. overlaying historical DoD Information Assurance Certification and Accreditation Process (DIACAP) process over Risk Management Framework (RMF) controls for individual pieces of software versus system accreditation). The sense is that the Certification and Accreditation (C&A) process is primarily a "check-the-box" documentary process, adds little value to the overall security of the system, and is likely to overlook flaws in the design, implementation, and the environment in which the software operates.

No way to calculate total costs of C&A process. The Department needs to be able to calculate the true and component costs for implementing the RMF and C&A in order to identify inefficiencies, duplicative capabilities, and redundant or overlapping security products and services that are being acquired or developed. Absent a set of metrics it is difficult to prioritize risk areas, investments, and evaluating risk reduction and return on investment.

Lack of top-down security requirements. The Department has not decomposed security requirements from an enterprise level to a mission level to a functional implementation level. Programs waste resources implementing security controls that should be inherited.

Lack of automation. The C&A process is predominantly a manual process which makes it a very low process. Programs must plan in terms of months and years to get a product through the security accreditation process. This slow process does not provide the warfighter the timely, modern solutions that are needed.

Desired state

Accredit the process, not the product. Done correctly, security is applied from the beginning of software development using automated tools. Before transitioning into operations, an Authorizing Official (AO) reviews the process under which the software was developed and accepts the risk as determined from various scans and tests. The AO signs a Continuous Authority to Operate (ATO) so that as long as the process remains intact and is continuously operationally monitored, the subsequent software releases are accredited.

Obstacles

Two primary obstacles are culture change and workforce skills. The current security culture is that security is a checkbox activity at the end of the development process. As RMF is implemented, this is beginning to change the culture of security from compliance to continuous risk assessment. However, the process is still very manual. The culture change needs to include using automation to speed up risk assessment and continuous risk monitoring of operational software.

The other obstacle is the security and accreditation workforce skill set. While tools can provide reports and speed up security activities like scans and code analysis, it takes a particular skill set to understand those inputs and recommend or make a risk decisions. The current security workforce must be trained in these new skills.

Ideas for change

Embrace DevSecOps. The Department should embrace DevSecOps (not just DevOps) and provide the necessary resources to develop the common software components and automation to assemble, test, accredit, and operate software systems. DevSecOps also includes policy-supported processes, certified libraries, tools, and an operational platform (with appropriately instrumented run-time software), and a toolchain reference to implementation to produce “born secure” software.

Automate, Automate, Automate! The Department needs to provide automated tools and services needed to integrate continuous monitoring with the development lifecycle, enable continuous assessment and accreditation, and delegate decision making at the lowest level possible. Examples of automation are using static code analysis during the “build” stage, running automated unit tests, functional test, regression tests, integration tests, and resiliency/performance tests during the “test” stage, using dynamic code analysis, fuzzing scans, running container security scans, STIG compliance scans, and 508 compliance scans during the “secure” stage, and running continuous monitoring tools and ensuring logs are being pushed to the appropriate entity during the “monitoring” and “operational” stages.

Define top-down implementation requirements. The Department needs to ensure that each Joint Capability Area (JCA) flows-down its strategy, best practices, and implementation requirements/guidance for security and accreditation to allow the Component responsible for implementing the software to appropriately tailor RMF and plan the development, accreditation, and operation of the software. Furthermore, each JCA should endeavor to clearly state its risk profile and tolerance so that the RMF can be applied effectively and appropriately mitigate identified risks.

Education is necessary at all levels. As security is “baked in” to software during the development process, people must be educated about what that means as different tools look at different security aspects. They must also be educated in what it means to bring different security reports together and make a risk decision, both during development, and continuously during operations.

Culturally, people must learn to appreciate that speed helps increase security. Security is improved when changes and updates can be made quickly to an application. Using automation, software can be reviewed and updated quickly. The AO must also be able to review documentation and make a risk decision quickly and make that decision on the process and not the product and document it in a “Continuous Authority to Operate.”

Appendix B.9: Test and Evaluation Subgroup Report

v0.3, 11 Feb 2019

The fundamental purpose of DoD test and evaluation (T&E) is to provide knowledge that helps decision makers manage the risk involved in developing, producing, operating, and sustaining systems and capabilities. While colloquially referred to as a single construct, T&E is composed of two distinct functions: obtaining the data and assessing the data. This distinction is important because the T&E community will report "pain points" in both functions. There are also two major types of test: Developmental Test (DT) and Operational Test (OT). DT, by nature, is "experimental," performed on behalf of the Program Management Office (PMO), supporting a formative evaluation and identifying design elements that will drive mission critical capability to inform the evolution of component and system design. OT is "evaluative," performed by and on behalf of the warfighter, supporting a summative evaluation of system capabilities to support warfighting missions across the operational envelope.

Because T&E has historically occurred toward the end of, often, a long and costly acquisition process (e.g., requirements, design, development, etc.), it can be perceived as simply adding time and cost to an already late and over-budget effort; PMOs therefore can view this "last step" T&E as simply making the situation worse. And if T&E finds a system substantially defective, necessitating expensive re-engineering of the design late in developing, it adds to the perception that T&E simply adds cost and time to project execution. A continuous iterative T&E model is clearly called for, occurring alongside design and development, where T&E can both; catch defects early so they can be solved quickly and cheaply and inform/shape system requirements based on early feedback from the warfighter. Experience shows that active, early involvement by independent testers – combined with a PMO who responds to the independent testers' advice – makes a positive difference to program outcomes. We have seen this in modern iterative approaches, such as agile development, applied effectively in the DoD, especially in Major Automated Information Systems (MAIS).[6] Taken together, these observations point to the need to move away from what can be a linear waterfall process segregated by siloes, to a more iterative and collaborative model that fuses all development, test, processes, tools, and information to enable the continuous delivery of tested capability. T&E can then be viewed as saving time/cost in development, instead of adding time/cost.

Pain Points and Obstacles

The DoD lacks the enterprise digital infrastructure needed to test the broad spectrum of software types and across the span of T&E to support developmental efficiency (in DT) and operational effectiveness (in OT). Digital models of test articles (e.g., "Digital Twins") are not always available and not built to common standards. T&E environments, including threat surrogates or models, are often program-focused and funded, with short-term development goals and narrowly-scoped capabilities defined by the program. Building (and re-building) representative T&E environments is time and cost prohibitive for individual programs and results in duplicative infrastructure investments across DoD. Moreover, current T&E practices in the Services, including those focused on software-intensive systems, do not adequately test systems in Joint and Coalition environments, nor do they consistently use appropriate risk-based, mission-focused testing.

[6] FY16 DOT&E Annual Report.

The DoD lacks the enterprise data management and analytics capability needed to support the evaluation of test data in accordance with the pace of modern iterative software methods. As data required to make informed acquisition decisions continues to grow due to higher resolution measurements, higher acquisition rates, and other additional requirements for software intensive systems (e.g., interdependency, need to operate in system-of-systems, family-of-systems, Joint, and Coalition environments, etc.), the need for a T&E infrastructure to collect, aggregate, and analyze this data must likewise evolve to keep pace. More timely data fusion will require improvements in data management techniques, access speeds, data access policies, data verification techniques, and the availability of more intelligent and agile tools. Without this infrastructure, and within the current paradigm, we are failing to adequately gather and analyze these highly diverse and complex datasets, which leads to invalid assessments of acquisition program progress and system performance, undercuts mission readiness, and places warfighters at risk. This gap becomes an even more prominent choke point in an iterative cycle. Thus, even if we mitigate the first pain point with modernized realistic test environments, and had the capability to collect the appropriate mix/quantity of data in testing, we would still not have the analytics horsepower to turn around an assessment to support the pace of an Agile/DevSecOps iterative cycle.

The DoD lacks the resources needed to adequately emulate advanced cyber adversaries, to support fielding of trusted, survivable, and resilient software-intensive defense systems. Various oversight entities (e.g., NDAAs, GAO Reports, etc.) have acknowledged this gap, and past DOT&E Annual Reports have documented a significant number of adverse cyber findings in OT that should not require an operational environment to discover. While the gap exists now (in the absence of modern software methods), it will become an even more prominent choke point in a rapid development and operational fielding paradigm. We do not have the advanced cyber test resources (manpower, methods, and environment) to support a true Agile/DevSecOps approach to developing, testing, and fielding the broad range of software-intensive systems needed by DoD now and in the future, in an environment increasingly populated by advanced cyber adversaries.

The DoD lacks a modern software intellectual property (IP) strategy to support T&E in a rapid software development and fielding environment. Overcoming this pain point is critical to overcoming all of the three previously described pain points. Specifically, none of the previously described pain points is fully achievable without sufficient access to necessary technical data associated with the software deliverables. Software acquisition processes are and will continue to be suboptimal (with respect to time and risk) without access to relevant technical data and this gap will become an even more prominent choke point in an Agile/DevSecOps-based paradigm without that access. A modern software IP strategy must include access to software environments (e.g., source code, build tools, test scripts, cybersecurity artifacts/risk assessments, etc.) so tests are repeatable, extendable, and reusable. This strategy will also have to strike a balance with the IP rights of the innovator (usually industry) to ensure continued engagement of DoD with leading-edge technology organizations.

A modern software IP strategy would support the three previously described pain points via:

- Enhance our ability to operationalize the concept of “digital twins,” with sufficient access to the source code of a given system (balancing DoD and innovator IP rights), so as to be able adequately represent that system.
- Support the instrumentation of software-intensive systems as needed during testing.
- Support cyber vulnerability assessments and the assignment of risks to residual vulnerabilities, via access to system data (e.g., code, technical data, etc.).

Desired state

While the DOD does a fair amount of “integrated testing” now (across DT and OT), that is not the same as “integrating T&E with the Voice of the End User continuously and alongside software development.” T&E must strive for continuous software testing, automated and integrated into the development cycle to the fullest extent possible, across the entirety of the DoD’s software portfolio. The qualifier, “fullest extent possible” is important, as many experts have acknowledged that no single “one size fits all” approach will work best across the entire DoD software portfolio all of the time.[7,8] In this envisioned state, independent testers would work alongside developers and operators to help software development programs succeed and deliver capability at the speed of need. T&E would no longer be perceived as “slowing things down” or “costing money post-development” because it occurs toward the end of a highly linear and inefficient process, but would instead be associated with saving time and money during development. This vision, applied across the entire DoD software portfolio (i.e., beyond just IT or MAIS) requires the right kinds of tools, architectures and standards (see first three pain points), access to the right kind of data (see second and fourth pain points), and an ability to partner with and work alongside the developer, while yet maintaining independence and objectivity in our assessments.

Ideas for change

Build the enterprise-level digital infrastructure needed to streamline software development and testing across the full DoD software portfolio. Beyond the DevSecOps platform (or Digital Technology concept), the DoD requires a digital *engineering* infrastructure to streamline integration and testing. This suggests that the DevSecOps platform must be made available to all DoD software developers and:

- Integrated with (systems-level) model-based/digital engineering infrastructure, including digital twin(s),
- Integrated with existing T&E infrastructure (e.g., open-air ranges, labs, and other test facilities),
- Integrated with comprehensive tactical/mission-level infrastructure, and
- Available to others who could benefit (e.g., analysis, training, planning, etc.).

Even with this kind of complete testing infrastructure providing the capability to collect the appropriate mix/quantity of data in testing, we would still not have the analytics horsepower to turn around an assessment sufficiently rapidly to support the pace of an Agile/DevSecOps iterative cycle. We must develop the enterprise knowledge management and data analytics capability for rapid analysis/presentation of technical data to support deployment decisions at each iterative cycle.

Finally, to advance our cyber test resources such that we can achieve overmatch to our most capable adversaries while yet supporting the pace of the modern software development, the DoD should expand DOT&E’s current capability to obtain state-of-the-art cyber capabilities on a fee- for-service basis. This provides a straightforward way to acquire skilled cyber personnel from leading institutions (e.g., academia, university affiliated or federally funded research and development centers, etc.), to help the DoD to keep pace with advanced cyber adversaries.

[7] 2018 Defense Science Board Task Force on Design and Acquisition of Software for Defense Systems.

[8] Boehm and Turner, 2009. Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley. Boston, MA.

Appendix B.10: Workforce Subgroup Report

V0.3, 7 Feb 2019

DoD's workforce (civilian, military, and supporting contractor personnel) is our most valuable resource. The workforce's capacity to apply modern technology and software practices to meet the mission is the only way we can remain relevant in increasingly technical fighting domains, especially against our sophisticated peers, Russia and China.

Improved management of the Department's software acquisition talent will also drive success across the other subgroups and sections of this report. Policies, processes, and bureaucratic practices are never a sufficient substitute for competence.

The Department's challenges are well documented and well known by the software acquisition and engineering professionals who suffer most from the accrued technology, cultural, and leadership debt. The Workforce Subgroup identified prevalent pain points, but focused on providing concrete and actionable solutions for improving the recruitment, retention, development, and engagement of the workforce.

Pain Points

The Department's reputation as an employer is a weakness rather than a strength. Candidates base their employment decision on a variety of factors, but the organization's reputation and day-to-day work are chief among their considerations. The demand, and competition with the private sector, for an experienced and qualified workforce, is increasing as threats to our data security become more sophisticated. DoD has a reputation as an antiquated employer that rewards time in grade rather than competence and most often outsources its technical execution. Technical employees often serve as oversight or move away from "hands-on-keyboard" as they advance in their careers; no longer contributing to creative or innovative execution.

The Department does not adequately understand which competencies and skill sets are possessed and needed within its software acquisition and engineering workforce. Without the ability to distinguish the workforce, the DoD cannot effectively drive human capital initiatives. Furthermore, there is no enterprise-wide talent management system to manage the workforce (e.g., geographically, skills, etc.), which leads to bureaucratic silos and the inability to leverage the Total Force.

The Department has not prioritized a comprehensive recruiting strategy or campaign targeting civilians (90 percent of the acquisition workforce) for technical positions. When candidates do apply, they face an "overly complex and lengthy hiring process (that) frequently results in the Government losing potential employees to private sector organizations with more streamlined hiring processes," according to the President's Management Agenda.[9]

There is no comprehensive training or development program that prepares the software acquisition and technical workforce to adequately deploy modern development tools and methodologies within our dynamic environments. Hiring top technical talent into the Department will never be a silver

[9] "President's Management Agenda: Modernizing Government for the 21st Century," (Washington, DC: Office of Management and Budget, April 2018), 20, https://www.whitehouse.gov/omb/management/pma/.

bullet. The Department also needs to consider how to equip, reward, promote, and empower its existing workforce.

The Department is unable to leverage modern tools that are common in the private sector and our personal lives (e.g., cloud storage, collaborative software, etc.) due to bureaucratic barriers. Top talent expects access to these tools to meet mission demands, and their absence may discourage qualified candidates from applying or staying. Although the Department has pockets of innovation and entrepreneurship within rapid fielding offices across the services, this culture has not scaled to the larger acquisition programs and offices. Long-cycle times, bureaucratic silos, and information-hoarding prevail.

Desired State

The Department requires a workforce capable of acquiring, building, and delivering software and technology in real time, as threats and demands emerge. This workforce should resemble successful technology companies that must move quickly to meet market challenges. They do so by promoting an agile culture, celebrating innovation, learning from calculated failures, and valuing people over process.

The Department's workforce embraced commercial best practices for the rapid recruitment of talented professionals. Once on boarded quickly, they will use modern tools and continuously learn in state-of-the-art training environments, bringing in the best from industry and academia, while pursuing private-public exchange programs to broaden their skill sets.

Obstacles

The bureaucratic culture of the Department creates significant barriers compared to a commercial sector ecosystem that moves at the speed of relevance. These barriers are now ingrained within the institution, perpetuating a risk-averse environment that represents the most significant obstacle to reform. While there are minor legislative solutions to achieving the desired state, we believe that the Department has the necessary authorities and flexibilities, but has shown lack of impetus to move to the modern era of talent management.

While small pockets of expertise and progress exist, the Department as a whole lacks sufficient understanding of current software development practices and talent management models that support them. Studies on the workforce dating back 35 years that show "limited evidence these different efforts had any lasting impact or resulted in meaningful outcomes."[10]

Ideas for Change

Foundational. Taking into account history and the significant challenges with changing the culture in a bureaucracy, the Department should *empower a small cadre of Highly Qualified Experts and innovative Department employees to execute changes* from this report. This cadre is empowered with the authority to create, eliminate, and change policies within the Department for organizations beyond themselves. If needed, create a software acquisition workforce fund similar to the existing

[10] McLendon, Michael H.; Shull, Forrest; Miller, Christopher, "DoD's Software Sustainment Ecosystem: Needed Skill Sets," (Naval Postgraduate School, Monterey, California, April 30, 2018).

Defense Acquisition Workforce Development Fund (DAWDF). As called out by the Defense Science Board, the purpose of this fund will be to hire and train a cadre of modern software acquisition experts. This fund should also be used to provide Agile, Tech, and DevSecOps coaches in Program Offices to support transformations, adoption of modern software practice and sharing lessons across the enterprise.[11]

Workforce Foundations. The Department must develop a core occupational series based on current core competencies and skills for software acquisition and engineering. This occupational series should encompass all workforce roles required for modern software development and acquisition - engineers, designers, product managers, etc. Additionally, the Department should create a unique identifier or endorsement of qualified (experience & training) individuals who are capable of serving on an acquisition for software. This includes the development of a modern talent marketplace (and associated knowledge and skill tags/badges) to track these individuals. The competencies for this series should be flexible enough to evolve alongside technology, something that has constrained the 2110 IT Series.

Contractor Reforms. Defense contractors develop the majority of software in the Department. The Department should incentivize defense contractors that demonstrate modern software methodologies; this may take the form of software factory demonstrations and rapid software delivery challenges when evaluating proposals. Additional consideration should be given to contractors with demonstrated excellence creating commercially successful software.

Recruitment and Hiring. The Department must overhaul its recruiting and hiring process to use simple position titles and descriptions, educate hiring managers to leverage all hiring authorities, engage subject-matter experts as reviewers, and streamline the onboarding process to take weeks instead of months. The Department needs to embrace private-sector hiring methods to attract and onboard top talent from non-traditional backgrounds (e.g., hackers and entrepreneurs). Too often, these types of candidates are passed over or require special authorities to join the Department, due to lack of education or regular pay stubs. Furthermore, the Department must develop a strategic recruitment program that targets civilians, similar to its recruitment strategy for military members. This includes prioritizing experience and skills over cookie-cutter commercial certifications or educational credentials.

Development, Advancement, Engagement, and Retention. The Department must pilot development programs that provide comprehensive training for all software acquisition professionals, developers, and associated functions. Programs should be built in partnership with academia and industry, leveraging commercial training solutions rather than custom and expensive Federal solutions. This will include continuing education courses to help the workforce stay current and ensure technical literacy across the acquisition workforce. The Department must emphasize promoting and rewarding those that have proven both commitment and technical competence. Continually looking outside the Department is demoralizing and insulting to existing professionals that demonstrate innovation, excellence, and the ability to deliver already. The Department should incentivize and provide software practitioners access to modern engagement and collaboration platforms to connect, share their skills and knowledge, and develop solutions leveraging the full enterprise.

[11] Design and Acquisition of Software for Defense Systems," Defense Science Board, Feb. 2018, https://www.acq.osd.mil/dsb/reports.htm

Finally, the Department should encourage greater private-public sector fluidity within its workforce. Federal employees who come from the private sector bring with them best practices, modern methodologies, and exposure to new technologies. Federal employees who leave bring their understanding of our unique mission and constraints, helping the private sector develop offerings and services that meet our needs.

Proposed Legislative/Regulatory Language

1. **Establishment of a Core "Digital Delivery" Occupational Series.** *Modifying Existing Language* - Title 10, §1721. Need to add this Core Occupational Series to the list of "Designation of Acquisition Positions" or *Consider Using Existing Language:* Title 10, §1607 to add this occupational series fit within this established Defense Intelligence Senior Level model.

2. **Empower Implementation Cadre.** *New Legislation* - This will be critical to avoid a repeat of the past 35+ years of continuous admiration of the problem.

3. **Contractor Reform.** Adjust future NDAA's to add incentives for defense contractors to use modern development practices. (See FY18NDAA / §§873 & 874)

4. **Modernize Position Description and Hiring Practices.** *Modifying Existing Language* - Title 5, Part III, Subpart D, Chapter 53, the addition of this pilot program needs to be added.

5. **Develop a Modern Academy.** *Modification Language* - Title 10 §1746: This section should be added under the Defense Acquisition University, however, the HQE Cadre from Proposal #1 will lead the development of this pilot training program. Note: Tied with FY18 NDAA §891

6. **Private-Public Sector Fluidity.** *Modification Language* - Title 5, §§3371-3375: Expand the Inter-Government Personnel Act and allow more civil service employees to work with non-Federal Agencies and Educational Institutions. *Modification Language* - Title 10, §1599g: Expand the Public-Private Talent Exchange Program and modify the language to reduce the "repayment" period from 1:2 to 1:1 ratio.

7. **Computer Language Proficiency Pay.** *New Language* - Title 10, §1596a - Use this language to create a new Computer-language proficiency pay statute.

8. **Develop a Strategic Recruitment Strategy for Civilians.** New Legislation

9. **Pilot a Cyber Hiring Team.** *New Legislation* - Team will have all the necessary authorities to execute recommendations called out in this report. The team will serve as a Department-wide alternative to organization's traditional HR offices and will provide expedited hiring and a better candidate experience for top tier cyber positions.

10. **Establish Workforce Fund.** *New Legislation* - Similar to DAWDF, but the primary use will be for hiring and training a cadre of modern software acquisition experts.

Appendix C: Analysis the Old Fashioned Way: A Look at Past DoD Software Projects

v1.0, 6 Jan 2019

The Department has been building and buying software for decades. The study's initial idea was to take a cutting edge machine learning tool, hook it up to the Department's databases, and do an analysis across all of the plentiful software data collected over the years.

Unfortunately, initial attempts at analysis quickly led to the realization that the Department had never strategically collected data on its software. The data that have been collected cover only a subset of the systems the Department acquires and are typically collected by hand, with all the potential for erroneous or missing values that that implies. The granularity at which data are collected also does not typically support insight into specific questions of acquisition performance. Without massive data calls, enormous amounts of PDF scanning, and an impossible number of non-disclosure agreements, a comprehensive analysis would not be possible.

Instead, the SWAP members broke the analysis into two main efforts:

1. Analysis of the available data in order to test the board's hypotheses as they evolve. Subject Matter Experts who are familiar with the existing data and its constraints explored the available data in search of insights that would confirm or refute the board's hypotheses about DoD software acquisition performance. These results are described in this appendix.

2. Application of cutting edge machine learning and other modern analytical techniques to datasets from outside of the DoD, to support reasoning about the type of insights that could be gained and reported, if the Department had access to more comprehensive data about its software. These results are described in Appendix D.

C.1 Data Used in This Analysis

The focus of this study is on software-intensive programs – and the specific software scope within these programs – presenting top-level insights into software acquisition performance. We focused our analysis on a few major data sources collected by the Department, which can provide insight on these issues.

The data in our first source are known as Software Resources Data Reports (SRDRs). The SRDR data were selected for use because they are specifically focused on the software activities of DoD acquisition programs. The SRDR is a contract data deliverable that formalizes the reporting of software metrics data and is the primary source of data on software projects and their performance. The SRDR reports are provided at the project level or subsystem level, not at the DoD Acquisition Program level. The data points included in the analyses reported here are representative of software builds, increments, or releases. In many cases, there are multiple data points in the set that represent different subsystems or projects from the same program.

The SRDR applies to all major contracts and subcontracts, regardless of contract type, for contractors developing or producing software elements that meet specific criteria[12] and with a projected software effort greater than $20M.

SRDR reports are designed to record both the estimates and actual results of new software development efforts or upgrades, with the goal of supporting cost estimation. The reports collect many characteristics about software activities in both structured and unstructured formats. The primary data analyzed in our work were size, effort, and schedule. Notably absent from the SRDRs are any data about quality. Defect data have been optional until recently and hence were not reported.

Other data sources used to explore some of the assumptions and recommendations of the DIB are the IPMR (Integrated Program Management Report) and SAR (Selected Acquisition Report) datasets. Programs in these datasets fall into the category of Major Defense Acquisition Programs (MDAPs). These datasets include:

1. Software development effort measured in labor hours, software size, and development activity duration metrics delivered as mandated respective to contractual agreements.

2. Software development performance as identified within each contract report. However, each contract contained common elements supporting both software and non-software activity on contracts. These were treated in proportion to the weight of software activity cost on contract. These reports contain data for measuring contractor's cost compared to budget baselines on Department acquisition contracts as well as projections of cost at completion.

3. Planned and executed schedule milestone dates reported to the Department at the aggregate program level as required by acquisition policy. This information is included as a part of a comprehensive summary of total program cost, schedule, and unit cost breach information.

These software development effort metrics, contract performance, and program level schedule data represent the best source of product development, contract cost, and schedule performance information available on various projects throughout DoD. In addition, these datasets are also independently validated by agencies within the Department and subject to audits that require maximum fidelity to accounting standards.

It is worth noting that these datasets provide the best available information on DoD software acquisition, but are mainly limited to contract cost and budget performance (versus technical functionality performance) and were collected by hand. This scenario seems to address larger structural and cultural problems:

- The Department has no real acquisition data system that holds anything more than top-level data on our largest programs.

[12] Specifically, "within acquisition category (ACAT) I and IA programs and pre-MDAP and pre-MAIS programs, subsequent to milestone A approval."

- There is no automated collection of acquisition data, despite the fact that software tools and infrastructures, from which data can be automatically extracted, are integral parts of the state of the practice in the software industry.

- For much of the limited software-specific data that we do have (for example, source lines of code, or SLOC), this study has argued that they do not provide meaningful technical insight. Metrics like SLOC are not what the private sector would use to assess and manage programs.

- Leadership often relies on experience and trusted advisors because timely, authoritative data are not available for real analysis.

C.2 Software Development Project Analysis

One area of analysis focused on the SRDR data to describe, at an enterprise- or portfolio-level, what the Department is able to say about its software based on the software-specific data. As described above, SRDR data are more project- or subcomponent-focused versus program- or contract-focused; indeed, it is not easy and perhaps not possible to create a program-level understanding of software activities from the SRDR data.

The results reported here address 3 three questions:

1. How well do software projects perform in terms of effort and schedule?

2. Is there a difference in project performance related to the size of the project and the use of agile development?

3. How long do software projects take to reach completion?

The source of the data was the May 2018 compilation file published by members of the Software Resources Data Report Working Group. This file contains 3993 submissions that yielded 475 initial reports of planning estimates, 598 reports of final actual values, and 295 pairs of initial and final reports. Upon further investigation, 131 pairs contained full lifecycle information and therefore serve as a better dataset for studying effort and schedule growth. Thus, while we base our conclusions in this section on the best available data for software, it is important to keep in mind the data represent only a small subset of the Department's software.

The results presented below were primarily based on common statistical methods. Although a variety of additional explorations were conducted, the results were not found to be stable or to have achieved high confidence. These included dynamic simulation modeling, causal learning, and analysis with repetitive partitioning and regression trees.

Software Project Effort and Schedule Performance

In the current DoD acquisition lifecycle, substantial effort goes into defining requirements upfront in extensive detail, and projecting the cost and schedule for achieving the capabilities so described. Despite that, it is often said that the Department has problems acquiring the software capabilities it needs within budget and schedule. This analysis explored whether there was support for this conventional wisdom.

DoD projects in the dataset generally do indeed experience substantial effort growth. As seen in the following figure, the median number of estimated hours is 22,250 while the median number of actual hours is 30,120. (Note that the vast majority of points lie above the green line, indicating that actual values were greater than estimated.) The median rate of growth is 25%. However, there are some projects that expend less than their estimated effort, sometimes by a substantial amount as reflected by the points within the red circle. Unfortunately, based on the data reported we cannot discern whether they delivered the full committed functionality or not.

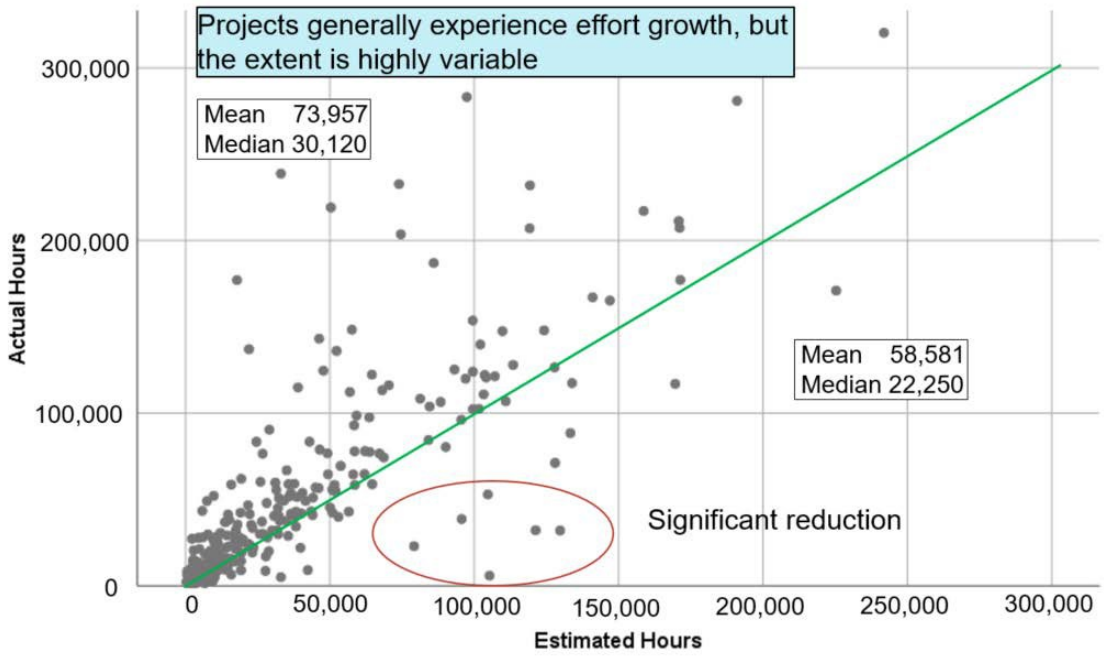


Figure 1. Estimated and actual project hours for project with less than 300,000 estimated hours.

The growth in project duration is generally not as large as the growth in effort. The median planned duration is 28 months and the actual duration is 34.9 months. The median growth in duration is 12%.

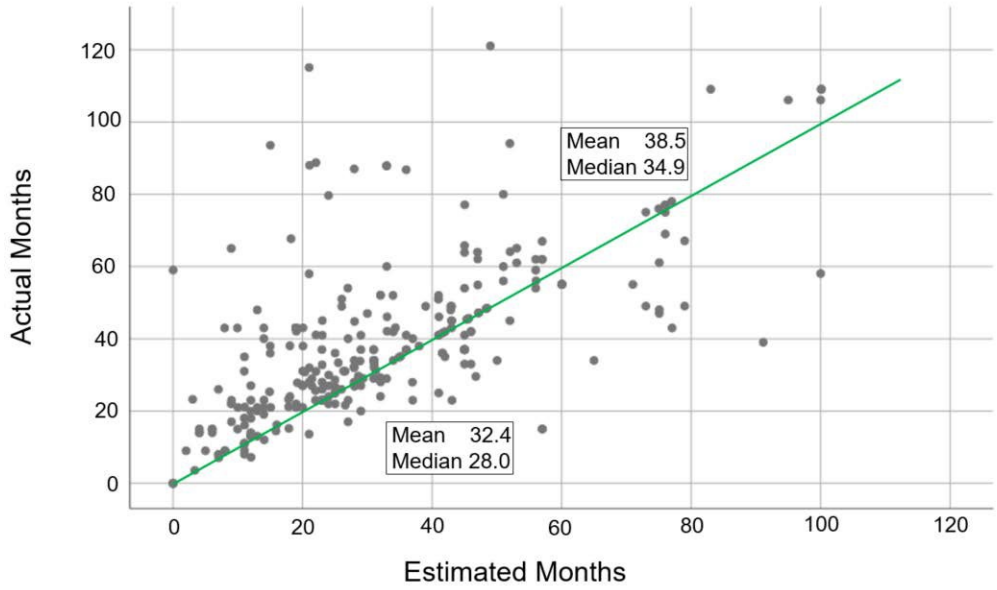


Figure 2. Estimated and actual project duration.

Interestingly, effort and duration growth are only weakly correlated and the highly skewed nature of their distributions means that averages create a more negative impression of performance than may be warranted. That is, the average exaggerates the degree of growth across the portfolio of projects. Nonetheless, in the data we have available, overruns of effort and duration are the norm.

Does Project Size Affect Performance?

The DIB has recommended that software programs should start small. The next analysis examined the historical data available to test whether small programs performed better than large ones, at least in terms of delivering capabilities on time and within budget.

To perform this analysis, projects were categorized in terms of their estimated equivalent source lines of code (ESLOC)[13] and effort. ESLOC is not collected but computed from the detailed SLOC measures that are collected: ESLOC combines the different sources of lines of code, new, modified, reused, and autogenerated, into a single count. Projects that were in the lower and upper quartiles on both effort and ESLOC measures were labelled as small and large projects respectively. This yielded 53 small and 55 large projects. An analysis of variance was conducted for growth in effort and duration.

The results found that small projects do not outperform large projects. Large projects do have less effort growth on a percentage basis but more growth in terms of raw hours. Surprisingly, schedule growth is very similar. Variation in performance overwhelms any apparent difference and the results do not achieve statistical significance.

[13] Elsewhere in this report, we reflect on the problems inherent with using SLOC as a measure. However, this is a key measure that has been collected historically by the department and so represents the best available data for this analysis.

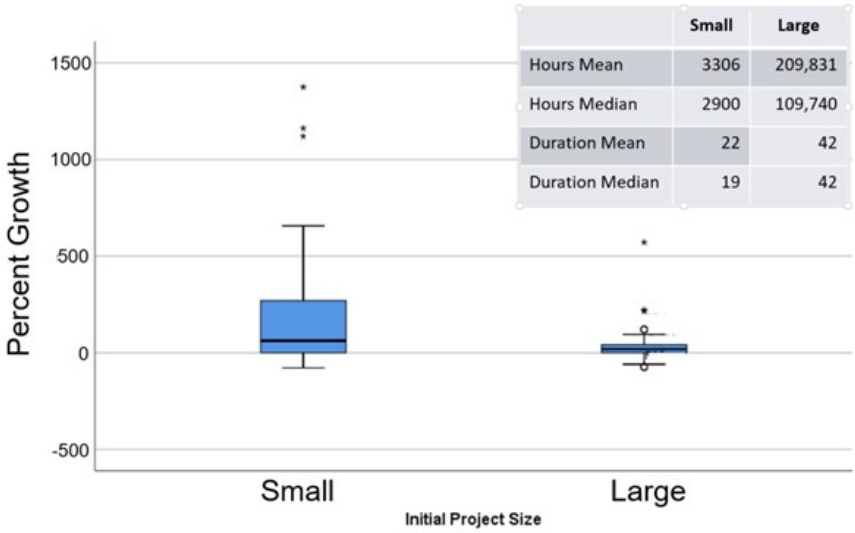


Figure 3. Effort growth by project size.

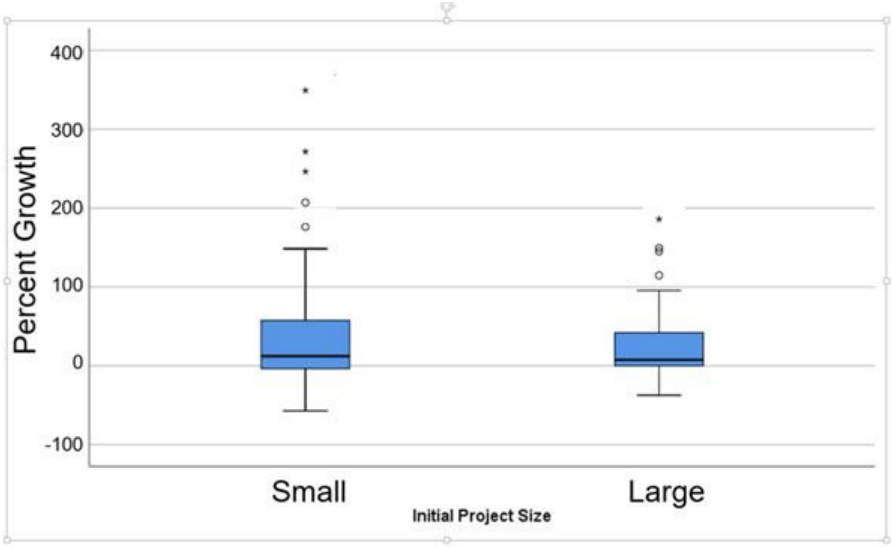


Figure 4. Duration growth by project size.

The fact that small projects still experience the same growth as large projects does not negate the advice that projects should start small, iterate often, and be terminated early if unsuccessful, since this can still result in significant savings in costs for projects that are not performing well.

Do Development Approaches Affect Performance?

There is much interest in the software development community and the DoD in the use of Agile methods. While the most recently updated SRDR form explicitly calls out measures for Agile projects, this has not been the case for the historical SRDR data upon which these analyses rely. Furthermore, the identification of the development approach is captured in an open text field. This necessitated interpretation and grouping of the entries in order to perform this analysis. A significant number of projects reported using “Waterfall,” “Incremental,” “Spiral,” or “Iterative” approaches. The remainder suggest use of a customized or hybrid approach. For the analysis here, “Waterfall” is compared to “Incremental,” “Spiral,” and “Iterative” projects.

Again, using ANOVA, the results indicate that effort growth does not significantly vary by development approach. However, duration growth is significantly less for projects using incremental development approaches as compared to waterfall (28% v 70% on average).

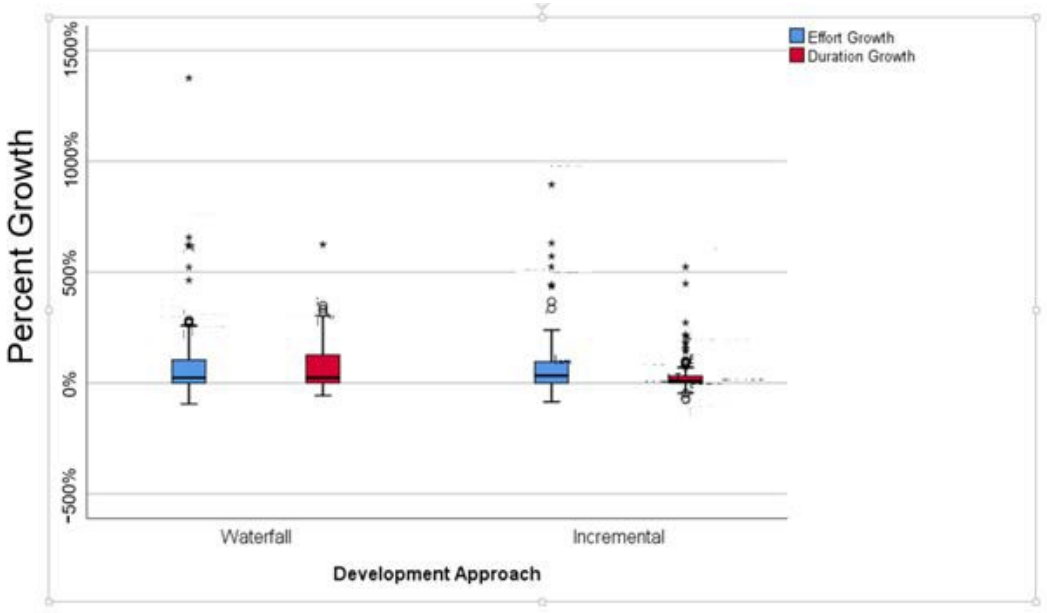


Figure 5. Effort and duration growth by development approach.

How Long Does It Currently Take to Complete a Project/Deliver Software?

As can be seen in the following figure, it is very rare for a project to complete in 12 months or less. Out of 371 projects used for this analysis, only 21 (6%) completed in this timeframe.

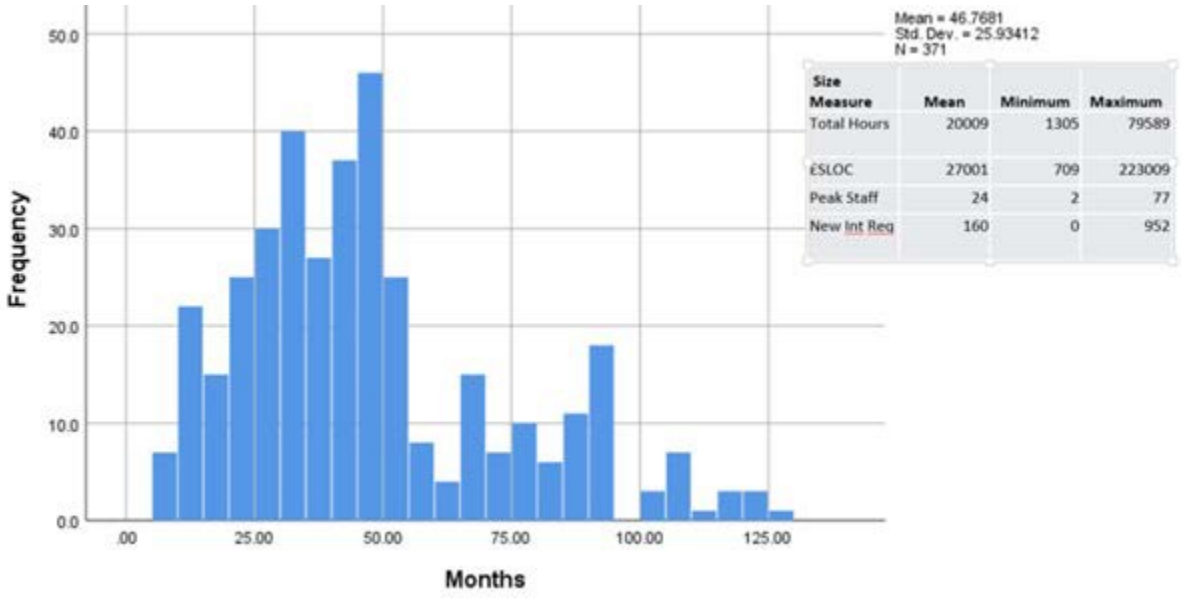


Figure 6. Actual duration for 371 AIS, Engineering, and Real-time projects.

Additional Insights from the SRDR Data

The preceding analyses were guided by the recommendations and proposed measures in DIB authored documents. In the course of performing those analyses, other questions and issues were posed and investigated. Briefly, these findings are:

1. Extreme variability in project performance confounds the identification of statistically significant results. This was noted above and is most likely actually due to performance and reporting inconsistencies.

2. Planned values can be useful for establishing expectations regarding reported actual effort and duration. That is, planned and actual values tend to be highly correlated with each other.

3. Planning for reuse is associated with significantly more schedule growth as compared to projects that do not plan for reuse.

The last one deserves more explanation as it is a somewhat counterintuitive result. Based on 275 projects that reported either no plan for code reuse or did plan for code reuse, the growth analysis showed no statistically significant differences in effort growth, but a significant difference in the amount of duration growth. Projects planning for code reuse had 52% duration growth as compared to only 20% for those that did not plan for code reuse. This phenomenon has been noted before and attributed to over-optimism about the amount and ease of code reuse. As the ability to reuse code falls short, unplanned effort and time go into producing new or modified code to compensate for the unrealized code reuse. Why effort growth is not significantly different is but likely at least partially related to the extreme variability in the performance measures.

Recommendations for Improving SRDR Data for Use

Issues regarding the data quality of SRDR data used here hampered the analyses. As is noted earlier, there is a substantial reduction from the number of submissions in the system to the number of usable records. At its most extreme there are 131 high quality pairs (262 records) out of the 3993 submissions included in the compilation dataset. That is, roughly 93% of the data is discarded.

The following recommendations are offered for improving SRDR data for use in addition to supporting the needs of the DOD cost community. Briefly, they are:

1. Leverage data collection and reporting from automation within the software environments (software factory). Minimize the need for manual entry and transformation.

2. Capture information about the quality of the delivered system.

3. Make the data more broadly available and encourage analyses into DoD software challenges (DIB Recommendation A6).

4. Identify the information needs of the stakeholders and intended users of the data beyond the cost community.

C.3 Software Development Data Analyses

A second investigation focused on cost and schedule performance data reported on recently completed and ongoing software development efforts within DoD. As these data provided insights *within* programs (and allowed understanding how values changed over time), we expected that this analysis would allow for deeper dives that could better explain how software acquisition occurs in programs.

This information was extracted from IPMRs, which are deliverables required by most contracts. The team also reviewed SARs for the large ACAT I programs to gain perspective on programs as they evolve over time.

Poor Data Quality and Inconsistent Data Reporting

There are approximately 130 ACAT I programs reporting research and development (R&D) contract performance over the past 10 years. We discarded from our analysis:

- Contracts for which the first IPMR report showed 65% (or about two-thirds) completed in work scope, reasoning that too much of the work had occurred before data collection began;
- Contracts for which the latest IPMR reported work that was less than 70% complete, reasoning that we would not have the ability to evaluate a significant portion of work completed.

146 contracts (35%) did not meet these data quality criteria out of the total of the 413 ACAT I program development contracts for which we have data (Figure 7). The fact that more than one-third of contracts do not meet this criterion implies that DoD would benefit from improving the quality and consistency of software development performance reporting. DoD cannot comprehensively assess the performance and value of the billions of dollars in investment without insight into a third of the complete portfolio.

Additionally, there are many data that are of limited utility due inconsistencies related to reporting. These have to do with problems with filing the mandated regular reports, and a lack of contextual data (i.e., metadata) being collected in a readily analyzable form. The DIB Software Metrics Recommendations contain recommended best practices on data collection and metrics definitions to not only capture data, but to establish standards meant to enhance software development performance.

Cost and Schedule Data

The resulting list of contracts was prioritized based on the budget assigned to the software-specific development efforts, and the top 46 contracts with the largest budgets were included in this study. These 46 contracts covered roughly half of the total dollar scope for all development programs in our dataset, and thus provided a reasonable sample size for our analysis. In addition, 35 contracts for smaller ACAT II and ACAT III software intensive Command and Control (C2) and Automated Information System (AIS) programs were included in this analysis. This resulted in the study capturing 81 total contracts valued at $17.9B in software development cost over the past

10 years (2008-2018). This study did not attempt to qualify or quantify the reasons for cost and schedule growth, recognizing that growth is not always indicative of poor performance by the program and/or contractor.

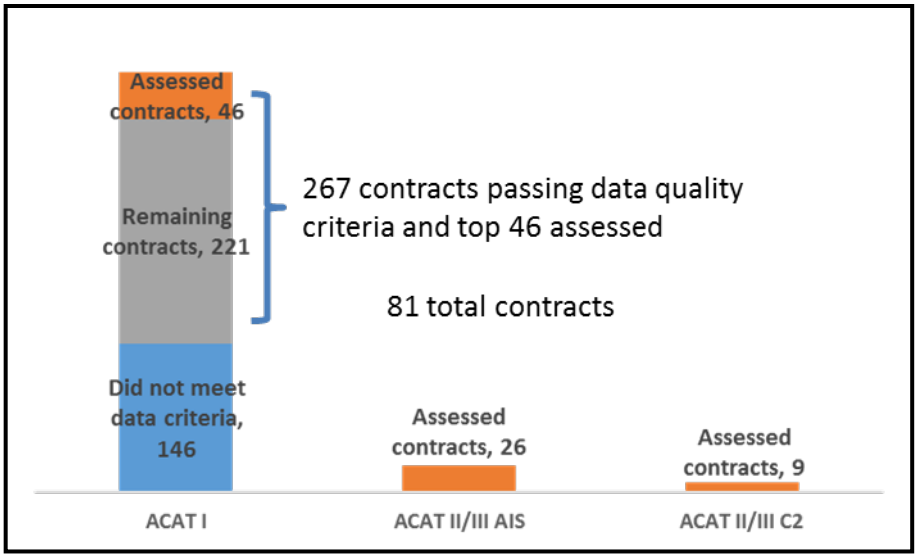


Figure 7. Results of Contract Selection Process

The 81 total contracts included in this analysis covered the portfolio of DoD programs, including software intensive C2 and AIS programs as well as aircraft, radars, land vehicles, and missile weapon systems, as shown in Figure 8.

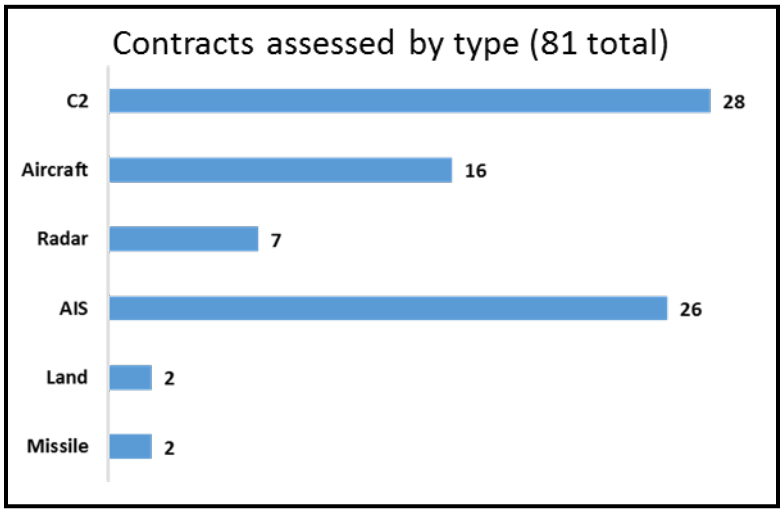


Figure 8. Contracts Analyzed by Weapon System Type

Large Software Cost Growth

The analysis of IPMR data found that on average, the contracts experienced 138% cost growth. The total combined value of the software development budgets within these contracts was $7.6B at the time of initial reporting. By the time these contracts reported the latest (or in some cases,

final) performance baseline, the software development budget total grew by $10.4B. Based on the analysis completed, significant software development cost growth was experienced across all platform and program types, resulting in a second observation: In general, the DoD struggles to minimize software development cost growth across the complete portfolio of projects. Figure 9 provides a summary of the 81 contracts evaluated, organized by project and by platform type. Note that the cost growth of “C2 Program A05” was truncated in the figure as it was an outlier in the analysis.

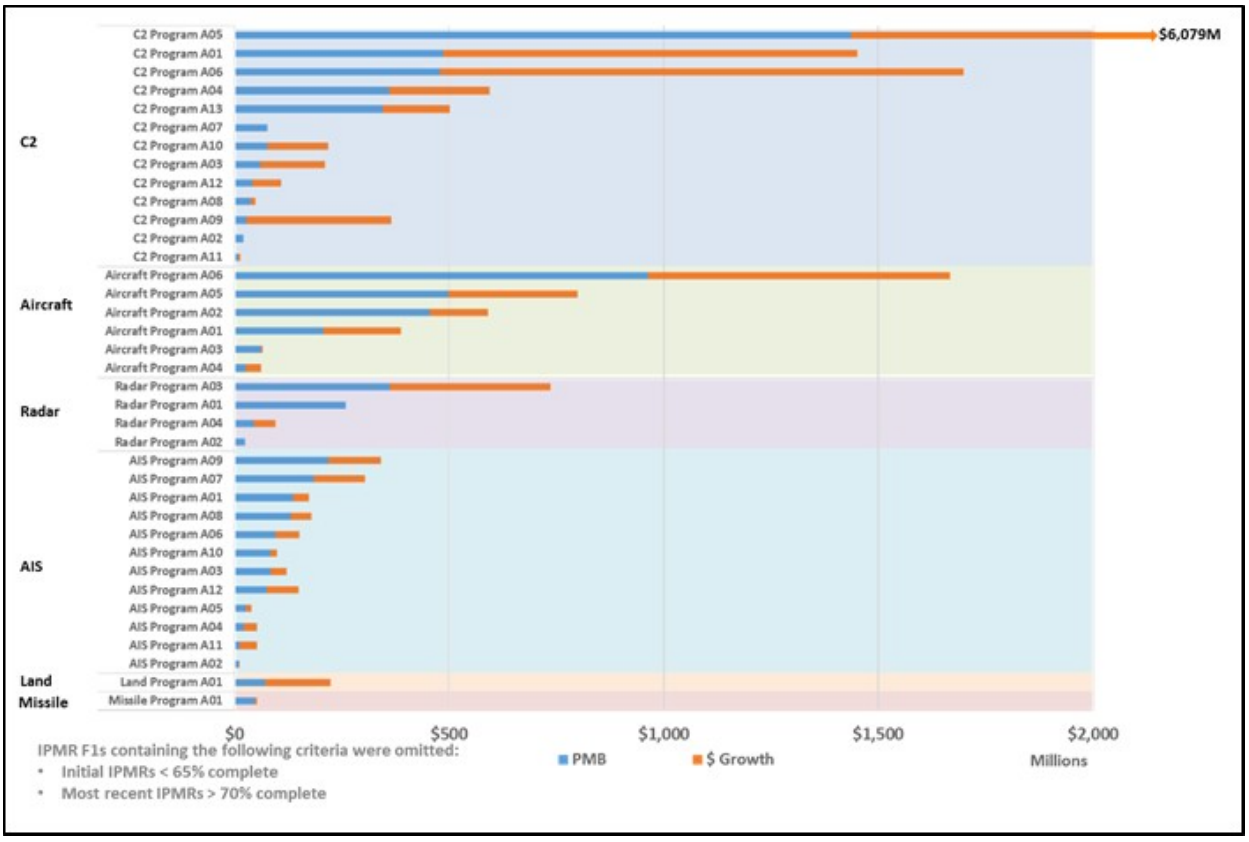


Figure 9. Contract Software Development Cost Growth by Program and by Platform

The study team used information provided by SARs and other relevant acquisition documentation to calculate project schedule growth. Figure 10 illustrates both dimensions of cost and schedule performance and identifies programs for which actual performance exceeds more than twice the baseline cost and schedule. Two programs, “AIS Program A01” and “C2 Program A02,” experienced cost or schedule growth so extreme that the bounds of the diagram axis plots were exceeded. This figure also supports the second observation that recent software development programs experience significant cost growth. The DIB SW Commandment 3 addresses cost growth by advocating that software budgets be planned upfront to support the full lifecycle versus the current funding lifecycle, defined around Planning, Programming, Budgeting, and Execution (PPB&E).

Figure 10. Software Development Cost Growth vs. Program Schedule Growth

Long Planned Durations and Frequent Re-baselining

The third study observation results from a deeper look into programs with high cost growth. This research found that in numerous instances, program baselines shifted (re-baselined) during the contract period of performance. The contracts with what appear to be significant “re-rebaselining” (i.e., multiple recurring increases to the expected cost) were analyzed in further detail.

SAR program milestones and available open source data were evaluated to provide a scale of time and functionality. It is observed that the software development effort crosses the same percent complete, as defined by the Earned Value Management (EVM) metric as the ratio of Budgeted Cost of Work Performed (BCWP) to Budget at Completion (BAC), multiple times. This represents an incremental method of adding cost, which is presumably associated with the addition of technical scope and requirements, which can result in a doubling or tripling of the total original budgeted value of the software development effort.

Figure 11 provides an example of this behavior, showing the “C2 Program A01” program effort that appears to re-baseline several times. The software development effort crosses the same percent complete point multiple times.

DIB Software Commandment 2 provides the recommendation that software development should begin small, be iterative and build on success; otherwise, be terminated quickly. DoD programs that take this approach are likely to see an improvement in performance once scope and requirements can be delimited through successful iteration. The behavior demonstrated in Figure 11 seems to indicate that to some extent, at least some programs are already behaving in an iterative way that better suits the technical work of software evolution. Unfortunately, our reporting mechanisms are not suited to reflect this reality, and in fact cannot differentiate a reasonable approach to incremental development from problematic cost or schedule growth. Looking just at the top-line numbers, these instances could be interpreted as excessive cost growth on the program, representing a problem from the Department’s point of view since the predictability of performance against cost and schedule baselines are normally taken as indicators of success.

What this scenario seems to point to is a need to improve our metrics collection to better reflect the underlying technical reality of software, where good performance often leads to a demand for new capabilities and new scope, as well as better educating our decision makers about how to interpret the results.

Thus this example provides more information about associated reporting issues tied to observation 5, that budgets should be contracted to support the full, iterative lifecycle of the software being procured with amounts definitized proportionally to the criticality and utility of the software.

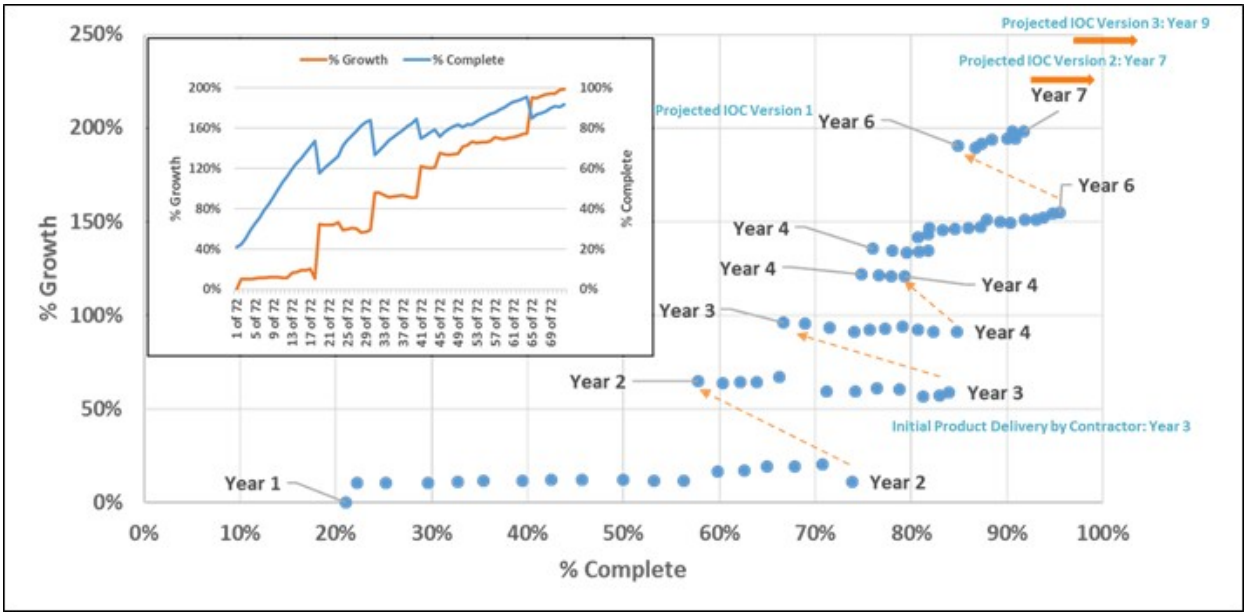


Figure 11. C2 Program A01 Performance Measurement Re-baselining

Agile Software Development Can Improve Program Performance

This study researched the performance of agile development methods that are implemented in existing programs. IPMRs do not explicitly state the type of development effort being used (incremental, agile, etc.). However, an article published in the journal Defense Acquisition provided an instance where agile development was applied and considered a success story. Although this article did not name the program, we were able to identify the most likely candidate, “Aircraft Program A05,” by matching the timeline presented in the article against the timeline of contracts that we could see in the program data.

The IPMR data for this program are shown in Figure 12. The contract work completed using an agile approach are shown in blue and represent a 21% cost reduction when compared to the initial budgeted value. This is in contrast to the contracts that seem to adopt a waterfall development methodology, i.e., contracts with planned long durations, which are shown in shades of orange and represent a 129% cost growth compared to the initial budgeted cost.

This analysis supports the fourth study observation that agile development may reduce cost growth compared to more traditional waterfall approaches. The DIB SW Commandment 2 also

advocates that agile approaches seen in commercial development result in faster deployment of functionality and cost savings which we observe in this instance.

Though a comparison of cost is one facet of performance, more research is required to increase the certainty that better overall performance and results were achieved with agile methods.

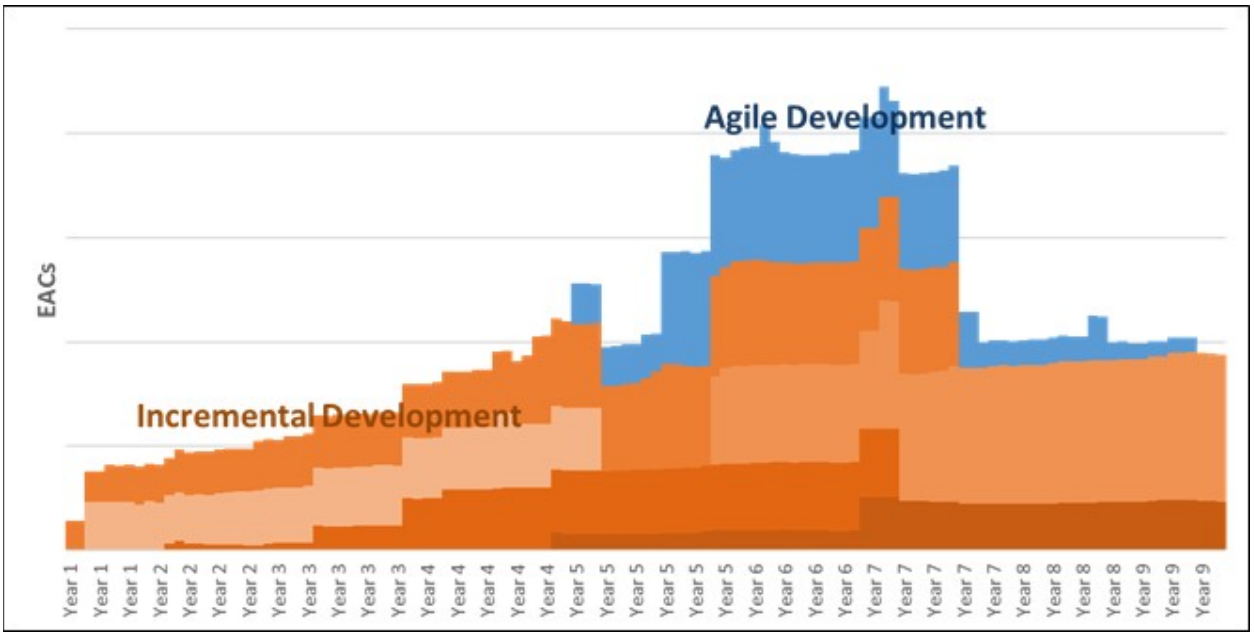


Figure 12. Aircraft Program A05: Incremental vs. Agile Development Efforts

Cost and Schedule Analysis Summary

In important ways, this analysis was typical of other efforts that aim to use Department data to examine the performance of acquisition. Due to the limited nature of the data available, our best analyses typically take months to create, with substantial time needed to find the data, to collect them, and to compile them into a structured format from multiple siloed and restricted systems.

The observations taken from data analysis of DoD program cost and schedule performance support the supposition that the current state of software acquisition is highly problematic and unsustainable relative to affordability and functionality. The DIB SW Commandments 2, 3, and 4 provide recommended measures to contain growth and increase the opportunity for cost savings by detaching software development from a hardware manufacturing industrial model and integrating software development and operations to quickly provide functionality to users and meet changing needs dictated by a dynamic global environment.

The preceding sections have described specific conclusions from the analyses our team conducted. Equally important, however, are the types of analyses we were *unable* to conduct given the data that were available.

A notable omission is that the Department is unable to address questions of *how much* software it has. Not in terms of software size but in terms of an index of how many important software systems have been acquired or are being sustained by the Department: There is no DoD or Service framework for describing the types of software intensive systems, or any inventory /

catalogue of the software in use. As a result, it is challenging to comprehend the scope and magnitude of the DoD software enterprise, and to design appropriate solutions for issues such as infrastructure or workforce that can meet the magnitude of the problem. Although done at a smaller scale, NASA's software inventory is an example of such an inventory model that is used to make strategic decisions for a federal agency.[14]

There is a large and growing body of work on software analytics, the automated or tool-assisted analysis of data about software systems (usually collected automatically) in order to make decisions. Conferences such as Mining Software Repositories[15] and Automated Software Engineering[16] annually showcase the best of the new research in these areas, and these methods are having a practical impact in commercial and government environments as well. A summary of software analytic applications lists several important questions that can be explored in this way: to name just a few, "using process data to predict overall project effort, using software process models to learn effective project changes, … using execution traces to learn normal interface usage patterns, … using bug databases to learn defect predictors that guide inspections teams to where code is most likely to fail."[17] Without access to its own software data, the DoD is missing the opportunity to exploit another area of research that could provide practical benefit for improving acquisition.

In a later section of this report (Appendix D), we provide the results of a small study that was undertaken to demonstrate potential practical impacts that could be achieved if software data access could be possible in the future.

[14] NASA Engineering Handbook (https://swehb.nasa.gov/display/7150/SWE-006+-+Agency+Software+Inventory#_tabs-6).

[15] https://2018.msrconf.org/

[16] http://ase-conferences.org/

[17] T. Menzies and T. Zimmermann, "Software Analytics: So What?," in *IEEE Software*, vol. 30, no. 4, pp. 31-37, July-Aug. 2013. DOI: 10.1109/MS.2013.86

Appendix D: Machine Learning Exploration

Linda Harrell, John Piorkoski, Phil Koshute, Erhan Guven, Marc Johnson (JHU/APL)
Vladimir Filkov, Farhana Sarkar, Guowei Yang, Anze Wang (UC Davis)
Steven Lee (Rotunda Solutions)
v0.2, 18 Feb 2019

D.1 Introduction

The Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) study chartered an exploratory study to explore the use of modern tools in data analytics and Machine Learning (ML) to provide insights into cost, time, and quality of Department of Defense (DoD) software projects. The data analytics and ML effort were performed by a team from academia (University of California Davis (UC-Davis)), a university affiliated research center (The Johns Hopkins University Applied Physics Laboratory (JHU/APL)) and industry (Rotunda Solutions). Since a suitable DoD data set was not available, the three teams leveraged existing data sets that were readily available to perform ML experiments and quickly get results.

ML models were created to predict the cost, time, and other aspects of software projects and gain a deeper understanding of the potential impact of project characteristics on overall project budget and effort. The models were trained with different data sets and were constructed to predict different performance metrics throughout the software development lifecycle.

The JHU/APL team developed ML models to predict software project duration and effort using the commercially available International Software Benchmarking Standards Group (ISBSG) Development and Enhancement (D&E) Repository of completed software projects. The UC-Davis team developed ML models to forecast software project duration, effort, and popularity using the publicly available GitHub repository of open-source projects. Finally, Rotunda Solutions created a defect density ML model to capture the code complexity and predict potential risk of code modules using a publicly available NASA dataset.

Additionally, the Rotunda Solutions team identified a number of opportunities for harnessing ML and Artificial Intelligence (AI) to improve the software acquisition process during different phases of the procurement cycle. This research effort is referred to as the Opportunities for Analytic Intervention. Rotunda Solutions also started development of a conceptual mock-up to explore some of these opportunities.

Overall, the three ML model development approaches demonstrated promising results aimed at improving predictions of software cost, time, and quality during different life-cycle phases.

- The JHU/APL team identified features (software metrics) that can support predictions of duration and effort at the project onset and shows that ML models have very good accuracy even with as few as 5 to 15 important features, most of which can be easily collected. It also shows how the prediction accuracy increases slightly by also including the effort expended in different life-cycle phases (e.g., planning, specification, design, build, test, and implementation). Since this analysis addresses the whole software lifecycle, the APL effort is referred to as the Software Life-Cycle Prediction Model.
- The UC-Davis team shows how monitoring of software development activities over time via automated tools that capture metrics (such as the number of lines of code, the number

of commits, and team size) can support accurate forecasts of duration, software effort (SWE), and software popularity. Additionally, the UC-Davis analysis showed that the ML models could obtain very good forecasting accuracy only 6 months after code development has started. Hence the UC-Davis ML model can serve as an early warning indicator. Since this analysis leveraged data obtained during software development activities to forecast future outcomes, it is referred to as the Software Development Forecasting Model.

- The Rotunda Solutions defect density model automatically processed code files and output code complexity metrics to aid efficient resource allocations and risk mitigation.

Interestingly, despite the differences in the approaches taken by JHU/APL and UC-Davis, the teams shared similar conclusions. For instance, both teams identified the team size and the project timing as being important features for the predictions.

Section D.2 of this document describes the methodology applied to the APL Software Life-Cycle Prediction Model and the UC-Davis Software Development Forecasting Model. Section D.3 summarizes the major findings of all three analyses. Section 4 offers implications of these study results for DoD programs.

D.2 Methodology

The approaches taken for the APL Software Life-Cycle Prediction Model and the UC-Davis Software Development Forecasting Model were complementary. Table 2.1 summarizes key aspects of the two approaches. These aspects include:

ML Techniques. Both studies leveraged readily available commercial or open- source ML techniques. This enabled the teams to meet the task's quick reaction turn-around timeline and also ensures that DoD government personnel and contractors can apply a similar approach when they develop their own prediction models for software projects. Although the teams developed several types of ML models, this report focuses on those with the best results: the APL Random Forest (RF) and the UC-Davis Neural Network (NN) models.

Data Sets. The APL team leveraged the 2018 International Software Benchmarking Standards Group (ISBSG) Development and Enhancement (D&E) Repository of completed software projects. This diverse database contains thousands of software projects that are described by a rich set of features that span the whole software lifecycle, but most of these projects have less than one year in duration or less than two years of effort. The UC-Davis team mined the GitHub collaborative project development and repository site, which contains historical trace data captured from millions of open-source software projects. The resulting database includes hundreds of thousands projects of various sizes. Its feature set is not as rich as in the ISBSG database, but it automatically tracks development metrics including commits, discussions, and other activities.

Target Variables. The APL team focuses on predicting software project duration and effort, two of the three metrics of greatest interest to the DIB. On the other hand, the UC-Davis team aims to predict the project duration (via its proxy months committed), the number of software commits (which is an incomplete proxy for software effort), and the number of stars (which is an indicator of the popularity of a project in GitHub).

Project Tiers and Boundaries. Large differences between proposal estimates and actual outcomes for software development duration and effort cause the biggest challenges for the DoD; small deviations are much more manageable. To reflect this perspective, both studies gathered their target variables into discrete tiers with boundaries shown in Figure 2.1.

Performance Metrics. Both studies assessed the performance of their models with confusion matrices (which shows the distribution of predictions in terms of predicted and actual tiers) and overall accuracy.

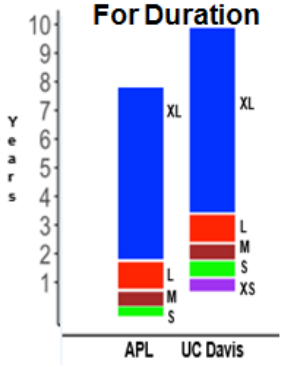
Table 2.1. Key Aspects of APL and UC-Davis Studies

Parameter		**APL Software Life-Cycle Prediction Model**	**UC-Davis Software Development Forecast Model**
Data Set		2018 ISBSG D&E Repository	2018 GitHub Repository
Number of Projects (after preprocessing)		2,818	Approx. 127,000
Number of Features (after reduction)		176	36
Target Variables for …	Duration	Project Duration	Months Committed
	Effort	Effort	Total Number of Commits
	Popularity	N/A	Number of Stars
ML Techniques		Off-the-shelf (NB, SVM, RF)	Off-the-shelf (MR, NB, RF, NN)
Results: Overall Accuracy; Confusion Matrices		Overall accuracy: Yes Confusion Matrix: 4 tier	Overall accuracy: Yes Confusion Matrix: 5 tier
Prediction Snapshots		Early concept development and procurement; Software development in process	After 6 months of software development ; Most recent software development
Feature Reduction		Yes	Yes

Definitions: NB = Naive Bayes, SVM = Support Vector Machines, MR = Multivariate Regression, NN = Neural Networks

APL Classification Tiers

Project Size	Software Effort
Small	2 months to 0.999 year
Medium	1 year to 1.999 years
Large	2 years to 7.999 years
Extra Large	>= 8 years



UC Davis Classification Tiers

Project Size	Number of Commits	Number of Stars
Extra Small	30 - 90	0 - 2
Small	91 - 140	3 - 7
Medium	141 - 217	8 - 22
Large	218 - 384	23 - 90
Extra Large	385 - 3469	91 - 31168

Figure 2.1. Classification Tier Boundaries

Prediction/Forecasting Snapshots. APL made predictions at two project phases (snapshots). The first snapshot is at onset, which includes features that are available or can be estimated during the concept, proposal, and procurement stage. The second is after software development has

been underway; it can include additional features as they become available. UC-Davis made predictions at three snapshots, corresponding to the time elapsed for each project: 6 months from first commit, 12 months from first commit, and most recent snapshot (1/1/2018). The most recent snapshot is taken to be the actual outcome (even if the project is still under development). For simplicity, the results with the 12-month snapshot are not discussed herein.

Feature Importance Ranking and Reduction. The APL RF and UC-Davis NN models both determined feature importance by evaluating the importance of each feature to the overall accuracy prediction and developed corresponding models with only the top ranked features.

Pre-Processing and Feature Selection. The pre-processing actions taken by the APL and UC-Davis are discussed in separate reports.

Project Context (Cluster) Creation. To fine-tune their predictive models, UC-Davis used an Autoencoder NN to group projects into four similarity clusters (i.e., contexts). A separate model NN was trained for each cluster. This technique allows for greater accuracy when project context is known early on, by, for example, tracking project metrics from the start.

D.2 Key Results and Findings

APL Software Life-Cycle Prediction Model

Table 3.1 shows the performance of the APL models that predict software project duration and effort with all features included. Even with minimal data cleaning, model tweaking, or sensitivity studies, and using a very sparse and unevenly distributed data set, the ML models predict a project's size tier with an overall accuracy ranging from 57% to 74%. These are impressive results for a quick-turnaround exploratory analysis.

As expected, the prediction estimates once development is underway are better than the predictions at program onset. This is because additional features, such as the effort expended in various life-cycle phases, help to improve predictions. However, with the features included in this analysis, the improvement was slight.

Even when the ML model does not correctly predict the size of the software project, the prediction is most often in adjacent tiers rather than significantly further away. This is evident in the confusion matrix in Table 3.2 and the additional confusion matrices provided in separate reports. This is important because it indicates that incorrect predictions still tend to be fairly close (e.g., an extra large project predicted as large or vice versa).

Table 3.1. Performance Summary for APL Prediction Models (with all features)

Model	Overall Accuracy
Predicting Duration at Project Onset	57%
Predicting Duration after the Project is Underway	58%
Predicting Effort at Project Onset	68%
Predicting Effort after the Project is Underway	74%

Table 3.2. APL Confusion Matrix for Predicting Effort as Project is Underway (with all features)

Accuracy values are shown as a percent of all projects of a given class		Predicted Class			
		S	**M**	**L**	**XL**
Actual Class	**Small (S)**	80	18	2	0.1
	Medium (M)	23	59	18	0.6
	Large (L)	2	20	73	5
	Extra Large (XL)	0.1	0.8	14	85

Table 3.3 identifies the most important features that influence the predictions. Naturally, the ranking of importance for each feature varies slightly for the predictions of duration and effort and for the two different phases (at project onset versus while the software development is underway), but the discrepancies are generally slight. Encouragingly, the features in this table are generally easy to obtain or estimate: function point standards, team size, software type, project implementation date, scope, programming language. The only feature category that is time consuming to gather is the functional size estimate. Each of the features in these tables is further described in the APL report.

Table 3.3 Most Important Features for ML Accuracy Predictions

Category of Feature	**Most Important Features**	**Project Phase**
Software Size	Functional Size, Relative Size, Adjusted Function Points	Project Onset
Standards for Function Point Estimates	Function Point Standards, Count Approach	Project Onset
Team	Maximum Team Size, Team Size	Project Onset
Type of Software	Industry Sector, Organization Type, Application Type, Business Area	Project Onset
Timing	Year of Project, Implementation Date	Project Onset
Scope	Project Activities, Development Type	Project Onset
Programming Language	Primary Programming Language, Language Type, Development Platform	Project Onset
Incremental Effort	Effort in the Planning Phase, Effort in Specify Phase, Effort in Design Phase, Effort in Build Phase, Effort for Implementation, Effort in Test Phase	When the Project is Underway
Cost	Total Project Cost	When the Project is Underway

Figure 3.1 depicts the accuracy prediction with small subsets of the most important features, and shows how the accuracy increases as additional features are added. This figure shows that although the database includes 176 features, very good predictions can be obtained using only as few as 5 to 15 features. These features are captured in Table 3.3.

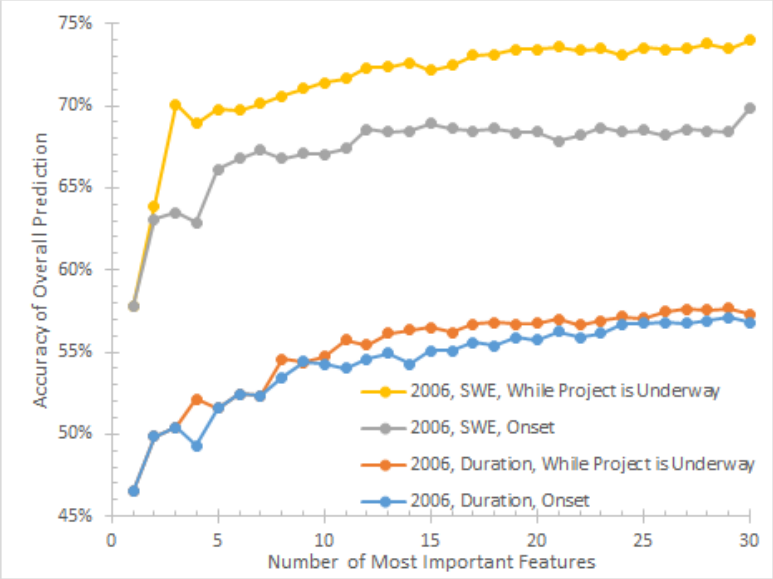


Figure 3.1. Accuracy of APL's Software Project Duration and Software Effort (with reduced, prioritized feature set)

The APL Software Life-Cycle Prediction model results clearly show that ML models can quickly be developed and trained using only a relatively small number of projects, a very small number of features, and a large amount of missing data. Furthermore, the resulting predictions for a software project's duration and total effort can be reasonably accurate at the project onset, and can then improve slightly over time by tracking the effort that is expended over the lifecycle. Only about 5 to 15 features are required to achieve reasonable predictions. The most important features for the predictions were identified; most of them are easy to obtain or estimate.

UC-Davis Software Development Forecasting Model

UC-Davis developed models that predict project duration, number of commits, and popularity using all available historical data of completed projects in the January 2018 snapshot, starting from the first commit of software. Table 3.4 shows the best-case overall prediction accuracies that can be obtained with these models and all of this data. The best-case overall accuracy of the prediction estimate for project duration is 84% and the best-case overall accuracy of the prediction estimate for the number of commits is 72%. Predictions for popularity were less accurate. These results indicate that the features in the GitHub database will be very useful for predicting software project duration and to a lesser extent the predictions for the number of commits. It appears that additional features will be necessary to improve the predictions for software popularity.

Additionally, Table 3.4 also shows that the best-case overall accuracy results for these models vary for different context clusters of similar projects. For instance, the accuracy values for each target variable increase within certain clusters; accuracy is greater in Cluster 1 by 16% for project duration and by 24% for number of commits and in Cluster 4 by 13% for popularity. These increases suggest that clustering projects based on similar context can increase the best-case prediction accuracy and that different models may be necessary to best predict different project contexts. The descriptions of these different clusters are not available at this time, but it would be valuable to investigate this further in order to understand the project characteristics that distinguish the clusters.

Table 3.5 shows the best-case overall accuracy of the UC-Davis models that use only the 9 most important features from the full project lifetime. These results are very close to those of the models that use all available features, indicating that the reduced feature set is sufficient for accurate predictions.

Table 3.4. Full Lifetime (Best-Case) Prediction Accuracy

Target Variable	All Projects	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Number of Projects	126,799	21,462	31,918	55,065	18,354
Project Duration (months committed)	84%	99.5%	83%	80%	78%
Number of Commits	72%	96%	70%	62%	69%
Popularity (number of stars)	49%	46%	48%	42%	62%

Table 3.5. Full Lifetime (Best-Case) Prediction Accuracy with Reduced Feature Set

Target Variable	**All Features** (All Clusters)	**9 Most Important Features** (All Clusters)
Project Duration (months committed)	84%	84%
Number of Commits	72%	74%
Popularity (number of stars)	49%	48%

Table 3.6 shows the accuracy results of the forecasting models, which predict the target variable in the final snapshot using features from a snapshot taken 6 months after project starts. These results are averaged over each of the 4 clusters (i.e., include 126,799 projects). These forecasting results show that data from only the first 6 months into a project can predict future outcomes, reaching accuracies of approximately 50% for both project duration and number of commits.

Table 3.7 identifies the most important features that influenced the UC-Davis predictions and forecasting. This table shows that features related to teams and commit activity are the most important for the UC-Davis models.

Table 3.6. Forecasting Accuracy (Averaged Over All Clusters)

Target Variable	Prediction of target variable at last snapshot given 6 month snapshot	Prediction of target variable at last snapshot given all data
Project Duration (months committed)	53%	84%
Number of Commits	50%	72%
Popularity (number of stars)	41%	49%

Table 3.7. Most Important Features for the UC-Davis Predictions and Forecasting

Feature Category	Most Important Features
Commit Activity Data	First Commit Date, Months Committed

Team Member Data	Team Size, Number of Commenters, Number of Pull Request Mergers, Average Months Active, Standard Deviation (SD) Months Active, Average Commits per Month, SD Commits per Month

In summary, the UC-Davis analysis shows excellent results for being able to forecast project duration and the number of commits only 6 months into a project. Only 9 features are required to achieve these forecasts. The most important features for the predictions were identified; all of them easily obtained with automation tools that track software development activities. Additionally, UC-Davis uncovered clusters of projects that if better understood could lead to improved models and accuracy predictions.

Rotunda Solutions Investigation of Opportunities for Analytic Intervention

The Rotunda Solutions effort focused on identifying strategic opportunities to leverage ML and AI at key points in the overall DoD procurement process. It extended academic research and state-of-the-art quality management principles to identify opportunities to improve the likelihood of successful software development outcomes. It also developed initial conceptual mock-ups to explore potential applications, including a defect prediction platform.

Rotunda Solutions adopted a basic stage-gate model to represent the general structure and stages of a DoD procurement and project development effort. Multiple opportunities are identified in each stage where analytics, ML, and other modern techniques can assist project managers. First, analytics can provide metrics and insights to support the project manager's yes/no/hold decision for whether the project should move to the next development stage. Second, analytics and ML can facilitate the search and interpretation of DoD procurement and development data sets so that decision makers have better access to historical data. Third, analytics can be run on this historical data to provide insights that can inform future projects. The application of modern techniques within a basic stage-gate model for a typical DoD procurement and development project can be envisioned as follows.

Stage 1: Idea Generation/Need Analysis. Analyze the internal unstructured documents from the program office and communications between suppliers and procurement officials. Then apply problem identification analytics to define the problem to be solved, considering the following 5 major groups/factors: need spotting, solution spotting, mental invention, market research, and trend. The literature shows a clear trend in savings of time and resources during the development process by maximizing the effectiveness of the idea generation stage.

Stages 2 and 3: Proposal Development and Response. Analyze internal unstructured documents from the program office and communications as they relate to proposal development and response. Use qualitative techniques such as focus groups, in-depth interviews, and surveys to determine factors associated with development success and failure. Additionally, use natural language processing (NLP) techniques to prepare the documents for further analysis. Both methods can identify key mechanisms and characteristics of software development success.

Stage 4: Contract and Award. Identify keywords through analysis of prior software contracts. Use NLP and topic extraction on legal documents surrounding the final selection of the supplier, contract vehicles, set-asides, and all stipulations to determine content. This can increase the ease of detecting associations between numerous demographic and supplier characteristics and

software development performance. It also provides the ability to build a grading system and general profile of contractors and their performance on projects.

Stage 5: Software Development. Gather representative data regarding project management metrics, code base, and development metrics, and compile a list of metrics that can help identify the likelihood of success of a DoD software development project. This helps the DoD in two ways: first by identifying projects that are likely to succeed or fail in each stage; and second by informing cost and time estimates for future software acquisition projects. Alternatively, analyze code to inform the development of ML tools to assist project managers and developers understand the state of their code. Potential benefits of this analysis include tools that can rapidly identify errors and increase efficiency for automation, audits, process checkpoints, and standardization.

Stage 6: Implementation. Harness available information on users, development, delivery personnel, and performance metrics of the software system. Measure the efficacy of the deployed or implemented software systems through metrics such as dependability, system performance, extensibility, and cross-platform functionality. This provides a post-mortem analysis of the efficiency and effectiveness of the software and the development process, allowing DoD to learn from past experience and increase the likelihood of future development success.

Conceptual Mock-Ups

Rotunda Solutions aims to help the DoD in four ways: (1) understand the potential impact of variables, decisions, and project characteristics on project budget and effort, based on historical data of similar projects; (2) make data-informed project decisions pertaining to the adjustment of project structure, methods, and other details; (3) create and explore what-if scenarios to promote better planning; and (4) encourage transparency and traceability of factors and decision-points affecting project performance. To this end, a number of concepts offer potential for further development and exploration. For instance, the concept of an “intelligent” burn-down chart is especially intriguing. Given sufficient sprint data and historical trend data, effort estimation tools and ML algorithms can be leveraged to make real-time predictions and issue alerts when estimates of team effort needs a closer review. Also, a defect prediction algorithm may be able to support risk mitigation activities and improve resource allocations.

Focus Area: Defect Prediction Platform

Software defect prevention is an essential part of the quality improvement process; timely identification of defects is important for efficient resource allocation, increased productivity, and risk mitigation, yet complete testing of an entire system is generally not feasible due to budget and time constraints. Studies show that the majority of software bugs are often contained within a small number of modules. To more rapidly identify these modules, Rotunda Solutions developed a system to automatically process code files and output code complexity metrics. They built off extensive industry research and tested representative NASA software modules using NN, SVM, Gaussian mixtures, and ensembles of ML techniques. The NN model performed best and was selected for production.

The NN model consists of 8 hidden layers, each layer becoming smaller until converging on a single probability to represent the existence of defects in the file. This model learns to assign importance weights to each of the 17 features and to combine these features in non-linear ways

to identify any potential defects. The NN can then be used to give a probability of defects for future files. This could help the management team in three ways: (1) to recognize the likeliest modules to have defects and allocate corrective resources effectively; (2) to provide an overview of the riskiest code modules to identify opportunities to re-architect the application; and (3) to understand the risk of deployment in production by an automated code complexity review.

Conclusions

The Rotunda Solutions exploration outlined the potential benefits of harnessing ML/AI throughout the DoD software acquisition lifecycle. These benefits include increased accuracy of budget predictions, comprehensive planning, mitigation of expensive defects, and transparency. Rotunda Solutions also identified many opportunities and applications that may improve DoD software development and estimation practices.

D.3 Implications of the Study Results for DoD

This ML study demonstrated promising results by creating models with publicly available software project data. It uncovered a promising approach (the APL Life-Cycle Prediction Model) that can be used to develop good predictions of software duration and effort in the early stages of software procurement and development. The study also uncovered another approach (the UC-Davis Forecasting Model) that can further improve project estimates once software development has been underway for 6 months or more. Finally, the Rotunda Solutions defect density model can highlight modules requiring additional resources and risk mitigation efforts.

The generalizability of these models to DoD software projects requires validation. For instance, a pilot study could be conducted with a small subset of DoD projects. Ultimately, strategies can be developed to enable DoD leadership to effectively leverage ML models.

One strategy could entail a strong centralized mandate for DoD software development teams to provide project data to DoD oversight personnel for evaluation with the APL and UC-Davis models.

A second, more streamlined and evolutionary strategy is to provide these models as tools for DoD software development teams to use as part of best practices to guide their development plans. This strategy would alleviate the exchange of data and would allow a more collaborative community effort to refine the models and resulting software development performance over time.

D.4 Caveats and Limitations

It is important to note that there are significant differences between the software repositories used in this work and important classes of software acquired by the DoD. For example, embedded software used in DoD weapons platforms is typically marked by high complexity, with low tolerance for reliability, availability, safety, and security issues. Although the testbeds on which the ML approaches were applied do contain some NASA software, only a small subset at best of the systems providing data are expected to have similar characteristics. As a result, it is important to view these results as showing a potential method that would be applicable to DoD programs and could learn characteristics of interest within that environment. While the method may be of interest, the specific results summarized may not directly carry over to some types of software present in the DoD environment.