DevOps for Federal Acquisition

Rick Cagle
The **MITRE** Corporation
Bedford MA, USA
rcagle@mitre.org

Michael Kristan
The **MITRE** Corporation
Bedford MA, USA
mkristan@mitre.org

Tim Rice
The **MITRE** Corporation
Bedford MA, USA
tbrice@mitre.org

Abstract— **Federal acquisitions are principally grounded in a traditional system development model with divisions of labor among development, independent test, and operations organizations. These silos are reinforced by the structure of current acquisition artifacts such as standard Contract Data Requirements List (CDRL) items and Data Item Definitions (DIDs).**

By contrast, in a DevOps ecosystem, development, test, and operations simply share responsibility for delivery of functioning services or products. Just as the movement toward a more Agile Development model within Federal acquisitions continues to struggle with adapting CDRL and DID artifacts, we anticipate programs seeking to inject DevOps approaches and proficiencies will have analogous challenges.

DevOps will continue to mature as a systems concept, and the government will be looking to adopt and adapt DevOps. This paper explores initial concepts of how a Request for Proposal (RFP) may be constructed to procure a system with DevOps principles, and proposes corresponding tailoring of acquisition artifacts.

Keywords—DevOps; Acquisition; CDRL, DID, RFP

I. Introduction

The process by which the government purchases products and services is referred to as Federal Acquisition. To guide evolving acquisition legislation, Congress applied administrative structure in the form of the Federal Acquisition Regulations (FAR), defining the procurement framework and mechanisms, and including contract-specific terminology. In many cases, individual agencies supplemented this core language with additional FAR Supplements to manage their unique regulatory requirements, which appear within the Code of Federal Regulations (CFR) volumes of the respective agencies.

Acquisitions have long followed a commonly referenced model for systems development consisting of divisions of labor among development, independent test, and operations organizations corresponding to like phases from the acquisition process. Conceptually, FAR guidance acknowledges phases that address Pre-Systems Acquisition, Systems Acquisition, and Sustainment. Such divisions of focus are reinforced by the structure of current artifacts of the process.

For example, a standard Contract Data Requirements List (CDRL) details items to be delivered as part of the acquisition. These items typically support design, development, testing, transition, and sustainment of the core contract deliverable, such as the software system or component. The CDRL is the standard artifact for identifying potential data requirements in a solicitation, and deliverable data requirements in a contract. It identifies products to be formally delivered by the supplier, and provides a standardized method of clearly and unambiguously delineating the minimum essential data needs.

Data requirements, format, delivery, and content can be further specified in sufficient detail to clearly identify delivery of the product of a sufficient quality to be accepted by the government and thereby satisfy the salient term in the Statement of Work (SOW). These Data Item Definitions (DIDs) may inherit from a default template, which can be found in the ASSIST Online Database [1], and is often further tailored to the needs of the specific acquisition.

II. Problem definition and its importance

A. DevOps

By contrast, in a DevOps ecosystem, development, test, and operations simply share responsibility for delivery of functioning services or products. The general principle of DevOps is based on the convergence of software development, quality assurance, and IT operations communities. Boundaries between phases are blurred in favor of stimulating collaborative responsibilities for delivery of product. Common components in a DevOps ecosystem include configuration management and infrastructure-as-code, virtual or cloud infrastructure, rich feedback loops, and process automation that drives continuous integration, packaging, auditing, and monitoring [2] [3]. An outcome of DevOps is a continuous delivery pipeline such that code changes can be automatically built, tested, and deployed into production using a well-defined, repeatable process [4].

A commercial software company makes money when they sell a product to a customer. The delivery mechanism used to be via shrink-wrapped boxes on retail store shelves. This inventory generates no revenue until purchased and excess inventory wastes space. The logistics chain introduces a delay from when the product is first created by developers to when the product is deployed in an operational environment. This delay causes update cycles to be larger and less frequent and creates opportunities for competitors to gain advantage by bringing a similar product to market faster. DevOps, when coupled with a rapid delivery system such as online deployment, allows software to be deployed to customers at a much quicker pace. A feedback loop allows customers to request and receive changes in smaller, incremental updates that can be quickly deployed to operations as shown in Figure 1.

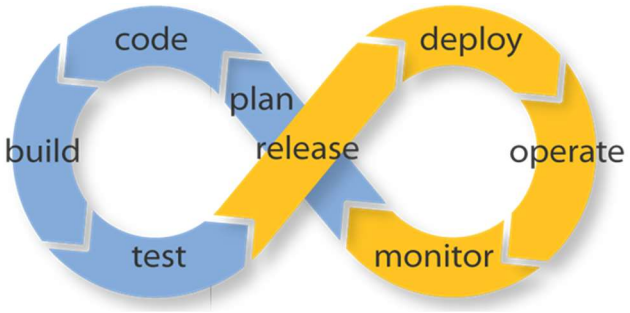

Fig. 1. The continuous cycle of DevOps in software development [5]

By adopting this methodology for all components of a software system, it transforms software development into a just-in-time manufacturing model [6].

B. DevOps and the Acquisition Process

The Defense Acquisition Guidebook [7] details the complex process of Government acquisition, with phases characterized in Figure 2.

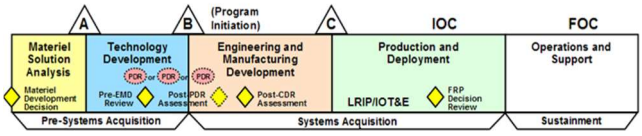

Fig. 2. Acquisition Phases

The government initiates an acquisition with a "Materiel Development Decision" identifying a need to be filled. While acquisitions can begin in any of the phases depicted, this paper focuses on those originating in the "Materiel Solution Analysis" phase. In this phase, the government determines the set of available solutions to satisfy that need. When one or more of those requires transition to a "Technology Development" phase to be performed under contract by external entities, a Request for Proposal (RFP) is drafted to govern the selection.

An RFP is a formal invitation to propose a solution to a stated need or problem. In the government's case, it often involves a competitive bidding process, where the agency interested in procuring a capability solicits potential suppliers to submit proposals. The RFP often details parameters, circumstances, constraints, and even an explicit framework to which any potential response must adhere. To the assessment team who will be responsible to evaluate and select from among prospective suppliers of the solution, it provides for structured evaluation and selection, and supports elements of impartiality in the procurement. These constraints often include a detailed list of deliverable items, as well as guidelines on how each RFP response will be evaluated, clarifying the scope of potential offers, as well as providing an objective framework for government assessment. This process is shown in Figure 3 with typical timelines.

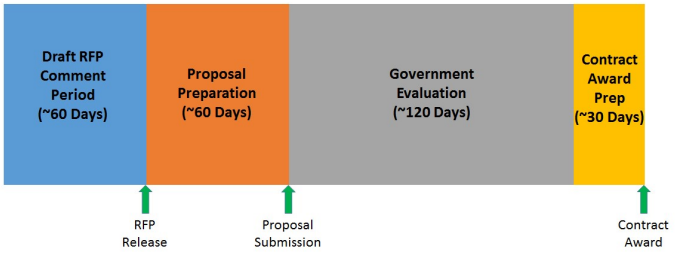

Fig. 3. Source selection phase of acquisition process [8]

During the "Comment Period", prospective respondents or "offerors" may influence the framework of the RFP. Then, once "Government Evaluation" begins, the government assesses each proposal, with respondents competing on the basis of this established framework. Following "Contract Award", the combination of the RFP and the awarded proposal evolve into the contract between the government and the organization awarded the contract.

An RFP is divided into sections A–M, where sections A–J primarily involve contract documents, except for section C, which is the Statement of Objective (SOO) or SOW. Section K contains attachments like the Requirements Document (TRD) or SRD. The two sections over which an acquisition can have the earliest influence in the process are sections L and M. Section L is the "Instructions for Proposal Preparation", and section M is the "Evaluation Criteria". For section M, technical criteria need to be included and need to address areas of technical risk and complexity.

Section L for an RFP developed in accordance with the FAR consists of any solicitation provisions and other information and instructions to guide respondents in preparing proposals or responses to requests for information. Prospective respondents may be instructed to submit proposals or information to facilitate evaluation. The instructions may specify parts, such as administrative, management, technical, and past performance. Several artifacts impacted by changes to section L are considered in our analysis and results.

Likewise, section M contains the standard against which the proposal will be evaluated. It forms the basis for selection, identifying all significant factors to be considered in awarding the contract and their relative importance. Elements of section M are often mapped to supporting DIDs in the CDRL. Discriminating factors may include: program risks, key performance indicators, costs, etc. It is in this section that a procurement team must establish ratings for factors, such as "Unacceptable", "Marginal", "Acceptable", and "Exceptional" or "Low", "Moderate", "High", and "Unacceptable". Proposed

language and modifications to specific artifacts are also discussed below.

In tailoring the process, by influencing development of artifacts such as the RFP, the CDRL items, or the corresponding individual DID governing each of those items, it is easier to influence the outcome as far "up-stream" in the acquisition as possible [8]. The government must drive to impact criteria and language at the proposal stage – and thereby influence selection of a winning respondent, and manage the resulting contract.

The Federal government is aware [9] of the need to overcome outdated practices. Play 7 of the US Digital Services Playbook [10] addressing experienced teams includes specific reference to familiarity with DevOps techniques. In September 2015, "18F", the Digital Services Agency within the General Services Administration (GSA), awarded their Agile Blanket Purchase Agreement (BPA), introducing over a dozen vendors into the available pool of Federal Contracting. Whether the Federal Acquisition process is similarly equipped with teams of acquisition assessment resources prepared for DevOps tempos is about to be tested. This paper examines areas of the CDRL and associated DIDs ripe for review and tailoring to accommodate accelerated deliveries.

Just as the movement toward a more Agile Development model within Federal acquisitions continues to struggle with adapting CDRL and DID artifacts [11] [12] [13], we anticipate programs seeking to inject DevOps approaches and proficiencies will have analogous challenges.

C. *Restructuring Traditional Development and Operations for DevOps*

The success of DevOps depends on a foundation of culture and tools. Traditional silos need to be broken down to foster open communication and debate instead of interfacing through ticketing systems and set resource allocations. Being a stakeholder in all steps results in individuals with knowledge and experience in development, test, and operations [14]. In the case where different contracts, and potentially different companies govern different stages of software development, operations, and maintenance, language must be introduced to mandate open communication between all parties.

Many open source and commercial tools exist with the purpose of fulfilling and automating the steps in a DevOps model. These tools will monitor code commits, build software products, execute test plans, and deploy the changes to development, testing, and production environments with minimal human interaction. The result is a deployment that can take a matter of minutes instead of hours or days of coordination, scheduling, and manual processes.

Challenges arise when the government funding model is considered. Historically, there is a clear delineation between the systems acquisition phase and sustainment phase. Each of those phases comes with its own budget. DevOps blurs the line between development and operation which results in the question of what funding vehicle pays for incremental changes that make their way automatically into operations.

Additional considerations to ensure the success of DevOps in a government acquisition include making changes to the FAR to provision for commodity cloud services as utilities [15] and expediting the security review and accreditation process for software, particularly for small-but-frequent changes [16].

III. Potential solution(s) tried to address the problem

The MITRE Corporation adopted DevOps for internal development in 2011. MITRE started with transitioning a simple Java application with an Oracle backend to a DevOps methodology. The project failed at first due to lack of sufficient training and expertise but was able to succeed and scale up by 2012 [17]. While lacking an RFP component, lessons learned from this and other pilots were applied to projects within MITRE that involved formal acquisition methodologies.

IV. Analysis of the results / lessons learned

One of the larger points to make here is the incorporation of the operational environment into the concern-set of the Development side of the DevOps partnership. It is also easily arguable that the current government model of delivering mission capabilities needs to be reconsidered.

One such consideration is to rework the acquisition boundaries to go from traditional government product lifecycle (vertical organizational factoring) to potential boundaries that deliver based on environment separately from mission with required dependencies (vertical organizational factoring).

The following sections provide an analysis of current set of MIL-STD-498 and related documents that the authors expect to be impacted by a DevOps procurement. Although MIL-STD-498 is a catalogue of DoD standard CDRLs and resulting DIDs, we expect that most government procurements with significant software–based capability will have a similar set of CDRLs.

A. *Planning Phase Documentation*

Software Development Plan (SDP) – The Software Development Plan (SDP) describes the plans for conducting a software development effort, including new development, modification, reuse, re-engineering, maintenance, and all other activities resulting in software products (to be delivered). Further this plan provides insight into the processes to be followed, the methods to be used, the approach to be followed for each activity, and project schedules, organization, and resources.

Typical SDP's are written by development organizations that are delivering to software product to operations. Considerations for tailoring content sections of the ASSIST standard SDP to a DevOps style procurement include the following sections:

SDP Section 3.0 - OVERVIEW OF THE REQUIRED WORK (including subsections on Scope, Constraints): extend to include software developed to support operations. Subsection on technology Scope should also include products and techniques to automate the delivery process to operations, as well as automating the feedback loops from mission operations to the development process.

SDP Section 4.0 - PLANS FOR PERFORMING GENERAL SOFTWARE DEVELOPMENT ACTIVITIES: extend to include explicit feedback loop processes between operations, test and development activities. Primary focus will be in the SW Methods subsections. The subsection on handling of critical requirements typically provides plans and methods for working against Key Performance Indicators (KPI)s as well as information assurance (IA), privacy, safety and other domain critical qualities. This subsection needs to set the stage for automating to the greatest extent possible the assurance component of the repeatable activities.

SDP Section 5.0 - PLANS FOR PERFORMING DETAILED SOFTWARE DEVELOPMENT ACTIVITIES will detail the plans for establishing development and test environments, including people processes and technologies. We also recommend that a staging environment be included, as well as the explicit connection to the operational environment(s). Any tooling and processes for connecting the people in each of the constituent DevOps environmental components must be addressed. Subsections on software testing should be focused on developing automated testing in keeping with continuous delivery models to include developer test, unit test, integration test, component test, and qualification testing.

This section also provides for planning and developing guides for delivery, installation, and/or transition of artifacts to operations. We will address these topics in the Software Installation Plan and Software Transition Plan.

Software Installation Plan (SIP) - A plan for installing the software at user sites, including preparations, training, and planning of upgrades to legacy. The details of a SIP maybe included within the SDP. Critical components for the SIP include sections on installation procedures, tasks, and site-specific information and instructions, as well as the software inventory, and upgrade instructions for existing legacy software and data systems.

Details for employing Continuous Delivery/Deployment practices and/or mechanisms will be described in the SIP including plans for automation support to perform upgrades and installation of software releases. Also of note, the site-specific as well as environment variants become planning requirements on the staging environment prior to releasing products. In addition, the SIP will capture support planning for feedback from the user and operational community to the development community.

Software Transition Plan (STrP) - A plan for transitioning to the support agency. A DevOps procurement could take the approach that such a plan is obsolete, or overtaken by another artifact such as the SDP, describing the automation of deployment. Alternatively, if considerations for the regular transfer retain complexity or sensitivity, the STrP could be adapted to assume a recurring operation of transition. It would also then need to detail the feedback that Operations would be entitled to receive from, and be expected to give to, Development.

B. Concept/requirements Phase Documentation

Operational Concept Description (OCD) - The operational concept for the system. The OCD serves to generate consensus among the government, contractor, operations, and end-users. The OCD does not dictate how the system shall be developed and implemented. This is the first opportunity in the software acquisition lifecycle to socialize and introduce DevOps concepts. DevOps plays a role in the operating concept by describing the types of users and their roles in the system. In this case developers are a type of user, and interact in the same way as an end-user or operator. Furthermore the OCD requires a description of the major system components and interconnections in the new system. A development environment with a delivery pipeline should be treated as one of these connections. Lastly, DevOps has an influence on operational impacts, organizational impacts, and developmental impacts. With operational impacts, DevOps will require a change in procedures for deployment, but not necessarily use. With organization impacts, the research and development organizations will likely merge with the operations and support organizations and result in shared responsibilities. Lastly, the impacts to development in a DevOps acquisition require a plan for testing, running versions of systems in parallel, and rollback in the event of failure. The development impacts should also mention the need for adequate training across all teams so that they are proficient in support, development, and operations instead of being stove-piped in one category. Given the rapidly evolving technological landscape, the expectation is that all members of the team will be involved in continuous learning, even after the initial ramp up on tools and processes. Learning can be applied via incremental improvements to the end product. [18]

DevOps approaches often also embrace a “Minimum Viable Product” definition, adopted by the Lean Startup methodology from earlier examinations. Common CDRL items, including:

- System/Subsystem Specification (SSS) - The requirements to be met by the system
- Software Requirements Specification (SRS) - The requirements to be met by a Computer Software Configuration Item (CSCI)
- Interface Requirements Specification (IRS) - The requirements for one or more interfaces

These will likely be impacted similarly, as requirements for delivered systems can be expected to undergo repeated changes through evolution of the product or system – as the definition of “minimum” grows with each DevOps delivery.

C. Design Phase Documentation

System/Subsystem Design Description (SSDD) - The design of the system.

Software Design Description (SDD) - The design of a CSCI.

Database Design Description (DBDD) - The design of a database.

Interface Design Description (IDD) - The design of one or more interfaces.

System Architecture Plan (SAP) – The capture of architectural principles of the system and the details of how the system should be designed to support the architectural qualities desired.

Designs to support DevOps principles will necessarily embrace the merging of increasing infrastructure components into code artifacts. The SAP should address qualities including *reliability* and *resiliency*, integrating with artifacts such as the SSDD to capture how the system is designed for graceful degradation.

Overall design must additionally account for the identification of feedback opportunities (from testing, operations, end-users, etc.) Furthermore, once identified, such feedback loops should be regularly examined, refined, and made more robust. Fundamentally, the system and the work plan must account for design of "Continuous Learning" to improve the system and the functioning of the constituent actors.

D. Qualification/Test Products Documentation

Software Test Plan (STP) - A plan for conducting qualification testing.

Software Test Description (STD) - Test cases/procedures for qualification testing.

Software Test Report (STR) - Test results of qualification testing.

To produce the best conditions for an efficient flow of work and rapid exploitation of feedback, the STP should maximize automation in the application lifecycle. Automation of tests to verify all direct functional requirements should be a paramount goal. Further automation at the level of unit and integration testing, as well as invocation of such tests at the appropriate point in development is evidence of a maturing DevOps process. For each of the potentially automatable test procedures, the STP should provide a comprehensive description.

The extent to which testing activities can support quality objectives is governed by how the STP and STD can align with, amplify, and nourish quality improvement throughout the application lifecycle. Similarly, the STR must effectively and efficiently communicate this feedback, and the SDP must accommodate analysis, prioritization, and re-injection of deficiencies into the development lifecycle, and support the regular, smooth flow of deliverable work product.

E. User/operator manuals

Software User Manual (SUM) - Instructions for hands-on users of the software.

Software Input/Output Manual (SIOM) - Instructions for users of a batch or interactive software system that is installed in a computer center.

Software Center Operator Manual (SCOM) - Instructions for operators of a batch or interactive software system that is installed in a computer center.

Computer Operation Manual (COM) - Instructions for operating a computer.

Application Monitoring becomes an important component in the overall DevOps strategies of identifying and amplifying feedback loops, and of continuous learning. User/operator manuals can be productively augmented with both platform and application monitoring features. Instructions for configuring, tuning, analyzing, and reporting feedback from instrumentation and dashboard functionality are useful enhancements to these DIDs.

F. Support manuals

Computer Programming Manual (CPM) - Instructions for programming a computer.

Firmware Support Manual (FSM) - Instructions for programming firmware devices.

These documents could contain instructions for how to instrument components, how to configure, and account for how to scale. Programs may want to add automating testing (unit, integration, CI/CD, etc.)

G. Software Product Documentation

Software Product Specification (SPS) - The executable software, the source files, and information to be used for support.

Software Version Description (SVD) - A list of delivered files and related information.

In DevOps, virtualization has led to the evolution of infrastructure – the operating system, supporting libraries and applications, and any configuration required – as a scriptable mechanism, ensuring consistent, repeatable, and scalable capabilities on-demand. This *infrastructure-as-code* enables description of the requisite framework to support the software for an application in identical language and artifacts as the software source code itself. The resulting infrastructure artifacts can then be included in the SPS, and depicted analogously to the application. Although cloud and virtualization can be considered to be an enabler of DevOps through rapid elasticity of infrastructure, they are by no means requirements for DevOps.

The version-controlled software library detailed in the SVD will also include infrastructure artifacts. The SVD scope should expand, becoming part of the collection that ensures all system artifacts – including the software defined infrastructure – are well-defined, consistently shared, and up to date across the release lifecycle. All DevOps organizational units, from development, through testing, quality assurance, and operations must have access to the same version-control system for these artifacts. From such a system, they likewise should be able to generate any version, including the applicable source code, the infrastructure, and all configuration parameters and supporting data.

H. Recommendations for "Instructions for Proposal Preparation" (Section L)

This section provides guidance on the specific information requested as part of the proposal. This section must communicate to offerors the elements and qualities of DevOps desired by the project, and it should be modified further to suit the specific acquisition strategy. The set below is adapted from standard guidance, and intended here as an example to facilitate RFP writing.

1) The offeror's proposal shall include a proposed Software Development Plan (SDP) which describes their approach to software development, to include the tools, techniques and standards to be used for development, unit testing and

component testing, and system deployment; integration tools and techniques (including configuration management) used to ensure the integrity of system builds; the number and type of reviews that are part of the development process; and the methods and tools used to manage defect reports and analysis, including root cause analysis as necessary. The proposed SDP will form the basis for a completed SDP to be available after contract award as a Contract Deliverable Requirements List (CDRL) item, subject to government review and approval.

2) The offeror shall describe the extent to which the software development process is automated.

3) The offeror shall provide a Performance Work Statement (PWS) in response to the Statement of Objectives and this RFP. The proposed solution shall include an explanation of how project and contract management, communication and collaboration among development, testing, quality assurance, and operations will function in the context of the proposed DevOps structure.

4) The offeror shall propose a DevOps Product Development Roadmap, which correlates how the stated objective aligns with the timeframe for implementation and the offeror's proposed DevOps approach. The Product Development Roadmap shall demonstrate communications, tooling and other collaboration enablers among development, testing, quality assurance, security, and operations.

5) The offeror shall propose a Software Product Specification that defines the extent to which the proposal will employ Infrastructure-as-Code, including tools for creating and deploying instances, scripts, and autonomic scaling of the software for the system.

6) The offeror shall propose a Software Version Description that includes any Infrastructure-as-Code source, executables, scripts, etc, as well as how all aspects of the application lifecycle, from Development to Operations, can access, communicate, and inject feedback into subsequent versions.

7) The offeror shall detail the plan for employing any Continuous Integration practices and/or mechanisms in the Software Development Plan (SDP), to include any automation for building, testing, and reporting of success or failure of the integration.

8) The offeror shall detail the plan for employing any Continuous Delivery/Deployment practices and/or mechanisms in the Software Installation Plan (SIP), including installation procedures, tasks, and site-specific configuration information and instructions, as well as the software inventory, and upgrade instructions for existing legacy software and data systems. The plan shall also include any automation for upgrades and installation of software releases. In addition, the plan shall detail support for feedback from the user and operational community to the development community.

9) The offeror shall propose language in the Product Development Roadmap, which correlates how the stated objective aligns with the timeframe for implementation and the offeror's proposed DevOps approach. A Product Development Roadmap shall demonstrate communications, tooling and other collaboration enablers among development, testing, quality assurance, security, and operations.

I. Recommendations for "Evaluation Criteria" (Section M)

When evaluating proposals constructed around enabling DevOps, the key elements to consider include *feedback loops, automation,* and *communications* across the application lifecycle. As the domains of infrastructure and source code become entangled, definitions of software developed artifacts expand along the lines of *Infrastructure-as-Code*. Evaluation criteria related to the SDP should therefore include the following, with "Low" to "High" ratings applied according to the tailored need for each DevOps quality:

- The numbers of and overall plan for virtual machine deployments across the infrastructure, to mitigate risks to software and functional requirements from potential divergences of instance configuration
- The use of tools to ensure sufficient uniformity of infrastructure in virtual machines employed
- The use of tools to automate configuration, ensuring machines across the infrastructure have uniform and predictable software configuration
- Detailed description of automated monitoring capabilities to properly manage the infrastructure of system clusters to be employed
- The degree to which DevOps application and infrastructure instrumentation supports self-awareness and self-correction in the system. A "dumb" system has no awareness of how it's performing; next would be "self-aware" systems that report on components and identify choke-points, error-handling, graceful failover and degradation; and above that are systems that can act autonomically to correct issues within pre-established parameters.

J. Recommendations for Scope, Technical Evaluation Factors, and Functional Area for Statement of Objective

The following section includes some key existing portions of the GSA Blanket Purchase Agreement (BPA) [9] pertaining to an Agile acquisition, which are also applicable to DevOps. Additionally, we are suggesting expansion of relevant language particular to DevOps. Specific BPA sections are shown with modifications to existing language in bold. These updates to the BPA would support the acquisition team in source, but if necessary, could be addressed using the CDRL and DID modifications above, even in the context of the existing BPA.

Scope Language for BPA**: An additional goal is to implement a DevOps process that achieves results through automation, continuous capability enhancements, and robust feedback loops.** The scope of the BPA encompasses requirements for [insert certification standard] expert companies to provide the business analysis, development, implementation, enhancements, and maintenance services required to successfully **implement and sustain the applications in the context of a DevOps environment**. The application lifecycle environment will allow the Government to provide an environment to automate building, testing, and deploying business applications. **Mechanisms will be included to**

support reinjection of end-user feedback on the applications to the DevOps team, and priorities set by the team for addressing feedback in the same context as existing and planned requirements. The Government wants to provide the option of a shared, collaborative environment among the development, testing, quality assurance, information assurance, and operations teams. **DevOps task orders can be placed under most if not all of the Functional Areas:** (1) Business Analysis, Development Integration: creation of a technical architecture to establish business application, including development, integration with Agency's existing systems; (2) Data Management and Securitization: data management may include data migration efforts, securitization with a Government provided third party Encryption tool, and creation of policy surrounding data implementation; (3) Program Management Support: additional support to assist in overall process; (4) Post-Implementation Maintenance Support: for production applications on the collaboration platform; (5) Post-Implementation Development: expansion or updates of production applications to meet ongoing unique objectives and requirements of specific Agency components; (6) Support: ongoing issue tracking and support desk functions for software development; and (7) Training: end use, administrator, knowledge management support for application.

Technical Factors for BPA**:** Technical submission should include the following: Factor 1 – **Technical Approach to DevOps environment (overview of performance-based solution and quality control and performance measurement approach)**; Factor 2 – Method for planning and sizing of work to be performed; Factor 3 – Past Performance Information; Factor 4 – Real Technical Factors (**automated unit testing, test-driven development, monitoring and instrumenting the application lifecycle environment, and scaling strategies**).

Functional Area for Statement of Objective**:** Functional Area 1: Business Analysis, Development, Integration - Creation of a technical architecture to establish business applications, including development, and integration with Agency's existing systems.

- **The contractor shall provide subject matter expertise for DevOps practices and tooling.**
- **The contractor shall develop or configure, test, stage, and release business applications by applying iterative processes utilizing the proposed DevOps environment and supporting tooling, and a frequent release cycle.**
- The contractor shall provide customer-friendly open source solutions that provide ease of use for non-technical Government users.
- **The contractor shall provide comprehensive documentation and information necessary to analyze DevOps processes, procedures, and/or policies that were implemented in the creation of the applications.**
- **The contractor shall provide business process analysis expertise with regard to optimizing the workflow, automation, manual processes (where necessary), and feedback loops across the DevOps environment.**
- The contractor shall develop system configuration in such a manner as to leverage maximum re-use and sharing across the platform by other federal agencies.
- The contractor shall provide incremental documentation that results in full technical and end-user documentation or configuration for all software development efforts and product releases with all information necessary to document processes, procedures, code artifacts, and/or policies that were implemented in the creation of the development work.

Key Personnel**:** The contractor shall identify key personnel and provide statements of qualifications for these individuals. Key personnel shall be current full time employees, contingent hires will not be accepted as key personnel submissions. The contractor shall identify key personnel who shall be the management lead and the technical lead for the task order as a whole. **These individuals must have expertise in DevOps processes, tooling, and environments**.

K. Recommendations for Contract Performance Metrics

Contract performance metrics should be relevant to the proposed methodology and should be tailored to the **DevOps** environment. "Offerors shall describe the Quality Control and Performance Measurement approach, including how proposed performance standards will be monitored, evaluated, and reported. The purpose of the notional Quality Control Plan is to provide evaluators with an understanding of how measures and metrics will be applied based on the proposed technical solution. These metrics shall cover planning, inspecting, and understanding progress under time. The metrics shall correspond with the "definition of done" as proposed in the PWS. **These may include such measures as extent of automation compared to manual processes, release success rates, defect resolutions, time to market, end user satisfaction, robustness of feedback mechanisms, etc.**"

L. Other Recommendations

Language for Enterprise Engineering Support: "The contractor shall provide capabilities engineering design to improve data interoperability, integrity, and quality for new systems and initiatives. The scope of this contract focuses more on integrating new systems and initiatives into the enterprise. The contractor shall evaluate commercial and Government software, freeware, shareware, tools, techniques, processes and standards. The contractor shall deliver design specifications, technical papers, reports, analyses, recommendations, and Service Level Agreements. **The work effort will be structured as a DevOps environment, with work deployed iteratively, periodic reviews, automation performance indicators captured, and planning meetings and product deliverables planned as releases."**

Language for Maintenance of Software: "The contractor shall maintain the software to include fixing defects, application software, tools, capabilities, and databases for the software applications, and related functionality in support of the user

community and the Program Management Office. Applications shall include clear, easy to use feedback mechanisms for reporting performance issues, software defects, and new and enhanced requirements. **The contractor shall operate as a DevOps environment in order to provide timely capabilities, scaling on-demand, and robust feedback mechanisms to the user community.** The maintenance tasks also include maintaining system/software engineering, integration activities, system security, program lifecycle documentation, application documentation, and database documentation required for continued software support and requirements management. Target release timeframes will be conducted in 2-week iterations with releases to production at least once every two months."

V. Open research questions to be tackled or recommended follow-on activities

This paper has examined areas of the CDRL and associated DIDs ripe for review and tailoring to accommodate accelerated deliveries. As Federal Agencies consider DevOps approaches to projects with newly awarded vendors under the GSA 18F BPA, further work to model tailoring of those DIDs and to develop templates for DevOps Acquisition should be considered. There should be a detailed examination of the extent to which the government can modify existing templates, merge where necessary, and add DevOps specific language or documents where expedient. We are also considering addressing specific RFP language for sections L & M.

Other open questions precipitate out of presumed initial successful introduction of DevOps principles to any acquisition. How do we amplify DevOps feedback loops in the context of the Acquisition Process? What are the cost versus benefit points involved in DevOps acquisitions? What are the relationships with other newer methodologies such as Agile, virtualization, and cloud when tailoring acquisitions?

In the long run, impacts from the evolution of software development in the commercial sector must be absorbed by one of the largest consumers – the Federal government. Existing processes for government procurement are not transparently compatible with DevOps and other emerging software lifecycle processes. Do we need to take more dramatic steps, and explore alternative acquisition and organizational models that will support DevOps? The question of whether Federal Acquisitions teams are sufficiently equipped with resources to manage DevOps tempos remains open.

Acknowledgments

We are grateful for the conversations and input provided by Mr. Rick Spiewak, and his work on injecting quality into the acquisition process.

References

[1] "ASSIST Online - The Official Source of DoD Standards and Specifications," [Online]. Available: https://assist.dla.mil/.

[2] M. Loukides, "What is DevOps?," 7 June 2012. [Online]. Available: http://radar.oreilly.com/2012/06/what-is-devops.html. [Accessed 1 8 2015].

[3] D. Edwards, "What is DevOps?," [Online]. Available: http://dev2ops.org/blog/2010/2/22/what-is-devops.html.

[4] J. Roche, "Adopting DevOps practices in quality assurance," *Communications of the ACM,* vol. 56, no. 11, pp. 38-43, November 2013.

[5] K. Hong, "DevOps (Development & Operations) 2015," Bogotobogo, 2015. [Online]. Available: http://www.bogotobogo.com/DevOps/DevOps_Jenkins_Chef_Puppet_Graphite_Logstash.php. [Accessed 8 August 2015].

[6] D. W. Karolak and N. Karolak, Software Engineering Risk Management: A Just-in-Time Approach, Los Alamitos, CA: IEEE Computer Society Press, 1995.

[7] Defense Acquisition University, Defense Acquisition Guidebook, Huntsville, AL: Defense Acquisition University, 2013.

[8] S. Bygren, G. Carrier, T. Maher, P. Maurer, D. Smiley, R. Spiewak and C. Sweed, "Applying the Fundamentals of Quality to Software Acquisition," in *Systems Conference (SysCon)*, Vancouver, 2012.

[9] Whitehouse, "TechFAR," [Online]. Available: https://github.com/WhiteHouse/playbook/blob/gh-pages/_includes/techfar-online.md.

[10] U. S. CIO, "U.S. Digital Services Playbook," [Online]. Available: http://playbook.cio.gov/.

[11] M. Lapham, R. Williams, C. Hammons, D. Burton and A. Schenker, "Considerations for Using Agile in DoD," Software Engineering Institute, Pittsburgh, PA, 2010.

[12] M. Lapham, S. Miller, L. Adams, N. Brown, B. Hackemack, C. Hammons, L. Levine and A. Schenker, "Agile Methods: Selected DoD Management and Acquisition Concerns," Software Engineering Institute, Pittsburgh, PA, 2011.

[13] R. Cagle, "Enterprise Architecture Facilitates Adopting Agile Development Methodologies into a DoD Acquisition," IEEE, Vancouver, 2012.

[14] M. Walls, Building a DevOps culture, First Edition ed., Sebastopol, California: O'Reilly Media Inc, 2013.

[15] K. Caraway, N. Gong, M. Kristan, N. Ross, J. Brunelle and T. Suder, "January 2015 Federal Cloud Computing Summit Summary," The MITRE Corporation, McLean, 2015.

[16] A. S. Voultepsis, J. Hess and K. A. Guisewite, Interviewees, *Intelligence Community Customer Voice.* [Interview]. 26 June 2015.

[17] A. Bechtle and B. Donaldson, "How a Not-for-profit Corporation Adopted DevOps," The MITRE Corporation, McLean, 2014.

[18] Defense Acquisition University, "Data Item Description Operational Concept Description (OCD)," 5 December 1994. [Online]. Available: https://acc.dau.mil/CommunityBrowser.aspx?id=59121. [Accessed 8 August 2015].