CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY PIPELINE FOR AIR FORCE DISTRIBUTED COMMON GROUND SYSTEM (AF DCGS)

THESIS

Carolyn W Fuller, NH-04, DAF

AFIT-ENV-MS-20-D-042

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-20-D-042

CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY PIPELINE FOR AIR FORCE DISTRIBUTED COMMON GROUND SYSTEM (AF DCGS)

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Systems Engineering

Carolyn W Fuller, BS

NH-04, DAF

Dec 2020

AFIT-ENV-MS-20-D-042

CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY PIPELINE FOR AIR FORCE DISTRIBUTED COMMON GROUND SYSTEM (AF DCGS)

Carolyn W. Fuller, BS

NH-04, DAF

Committee Membership:

Dr. Brent Langhals
Chair

Dr. David Long
Member

Dr. Tom Ford
Member

AFIT-ENV-MS-20-D-042

Abstract

AF DCGS has a recognized need to improve speed of delivery for modification and sustainment of the weapon system. New advances in software development practices have focused on automated continuous integration and testing. Given that the program office implemented a Continuous Integration/Continuous Delivery (CI/CD) process for the sole purpose of delivering capability to the field faster, there is a need to measure and report the pipeline throughput. This research conducts an independent evaluation of this newly implemented pipeline within AF DCGS's existing integration and test laboratories. A comparison between the two concurrent integration and test processes actively in use by the program is conducted to determine if the CI/CD pipeline has improved the speed of delivery. The study provides further insight into the processes of the CI/CD pipeline by examining performance as the pipeline matured and the impact different attributes have on delivery timelines. Actual project data from the agile development work environments is studied and hypothesis tests are conducted to substantiate that the CI/CD pipeline improved the speed of delivery. The research definitively shows that the CI/CD pipeline improves speed of delivery for AF DCGS from a range of 22% to 119% depending on the type of work product. Lastly, from observation and detailed study of the processes and data, recommendations are made for standardization and automated metrics collection, with suggestions for additional research to further characterize the pipeline with the intent to create a predictive model for more accurate estimation of delivery timelines.

Acknowledgments

I would like to thank AFIT for the opportunity to perform this research. To my husband and family: Thank you for all your support and encouragement on this long journey I undertook. I would like to express my sincere appreciation to my faculty advisor, Dr. Brent Langhals, for his guidance and support throughout the course of this thesis effort. The insight and guidance were certainly appreciated. I am also grateful to Mr. James Bechtel and LtCol Gutierrez for allowing me access to AF DCGS integration labs and the DI2E development platform to conduct this research. I would also like to thank my sponsors, Col Maddox and Mr. Scott Mangrum, from the Air Force Life Cycle Management Command for the support provided to me in this endeavor.

Carolyn W Fuller

Table of Contents

List of Figures

List of Tables

CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY PIPELINE FOR AIR FORCE DISTRIBUTED COMMON GROUND SYSTEM (AF DCGS)

I. Introduction

Background

The near-real time delivery of actionable intelligence data to the warfighter--which enables information superiority--is a long-standing principal goal of the Intelligence Community. The operational measure of success focuses on how to interconnect sensors, decision makers, and shooters with this collected data to achieve shared awareness, increased speed of command, higher tempo of operations, greater lethality, increased survivability, and a degree of self-synchronization (Alberts, Garstka, & Stein, 1999). The Air Force Distributed Common Ground System (AF DCGS), or AN/GSQ-272 SENTINEL, is a multi-intelligence system of systems that provides this actionable intelligence to the warfighter through processing, exploitation, and dissemination of collected sensor data from multiple ISR platforms across the globe, 24 hours per day, 7 days per week through distributed (reach-back and deployed) and collaborative operations (U.S. Air Force, 2015). AF DCGS accomplishes this interconnect through a library of software applications, a communication and data relay network, and an infrastructure of enterprise services, operating systems, virtualization layer and hardware that military intelligence analysts use to produce intelligence products. The Original Equipment Manufacturers (OEMs) delivered proprietary software applications on individual proprietary infrastructures resulting in hardware, firmware, enterprise services and software applications completely owned by the OEMs.

Because of this closed architecture and vendor lock with each OEM maintaining control and proprietary rights to their portion of the weapon system, upgrades became so slow and costly that new capabilities were virtually obsolete the day they became operational. To address this shortcoming a redesign of the system started in 2014 with the assistance of Air Combat Command and the Air Force Research Lab (AFRL) in Rome, NY. The AFRL solution was a government owned single open architecture to replace multiple OEM owned single-mission capabilities. The existing design forced all software/hardware application upgrades and enhancements to go through the OEM system integrators, taking up to 84 months from time of contract award to delivery. The new Open Architecture DCGS (OA DCGS) was rebuilt with an open architecture and Commercial off the Shelf (COTS) hardware and software for the infrastructure and enterprise layers with an OEM virtualized software application layer (Haga, 2017).

During the development of OA DCGS, the legacy AF DCGS was on the oversight list of the Director of Operational Test and Evaluation (DOTE). In the FY16 DOTE annual report, it was noted "The Air Force is in the process of transitioning AF DCGS to an open architecture system via an agile acquisition strategy. This transition is expected to take several years" (Gilmore, 2016). The FY16 annual report highlighted the slow progress of establishing a rigorous software problem tracking and reporting mechanism which was first recommended in the FY15 annual report. Specifically, the FY15 recommendation was to develop a software change process to track software metrics for problems open and closed by severity and time (Gilmore, 2016). The AF DCGS program office started development of a Request for Change (RFC) process in 2017 to address this finding in the FY15/16 reports. The implementation of this process provides the needed metrics to analyze the efficiencies of the software development process. OA DCGS began deployment to the operational sites starting in FY17 creating the foundation for building a Continuous Integration Environment (CIE) capable of testing all functionality and cybersecurity elements of AF DCGS in an automated manner.

The AF DCGS CIE is a set of tools on a controlled test environment that is part of the larger CI/CD process for the weapon system (Wellspring, 2020). Continuous Integration (CI) is the process of taking features (requirements) from the program backlog and developing, testing, integrating and validating them in a staging environment where they are ready for deployment and release (Wellspring, 2020). Continuous Delivery (CD) is the process that takes the work of the CI process and readies the delivery for deployment. CD is a software strategy that enables organizations to deliver new features to users as fast and efficiently as possible (Wellspring, 2020). When these two processes are combined with automated tools in a continuous integration environment, it is referred to as a CI/CD pipeline. To summarize, the CIE is a set of tools in a lab environment that enables automation for the CI/CD processes.

Problem Statement

Air Combat Command has concerns the AF DCGS software delivery timelines are too slow and number of deficiencies discovered in fielded software is too high. The literature asserts that using a properly implemented continuous integration and delivery pipeline will increase the speed of delivery and quality of software products through automation and feedback loops (Zaydi & Nassereddine, 2019). According to the Phoenix

Project, deployment frequency to the release environment for different companies drastically increased through continuous integration as follows: Amazon at 23,000 per day, Google at 5,500 per day, Facebook one per day, Twitter 3 per week. This is game changing when compared to a standard deployment for a typical enterprise that does not use continuous integration or continuous delivery and is cited at one deployment every 9 months (Kim, Behr, & Spafford, 2014). Furthermore, Myrbakken states CI and CD enable the speed required for DevOps practices through automation of build, deployment, and testing and is important to achieve rapid development and deployment of software (Myrbakken & Colomo-Palacios, 2017). AF DCGS had already laid the groundwork for a CI/CD pipeline with an open architecture delivered with OA DCGS, tailoring the entire program office around Scaled Agile Framework (SAFe), and implementing agile practices for software development using Atlassian Jira.

AF DCGS implemented a continuous integration environment but metrics have not been gathered or analyzed on the current projects to determine if increased speed and quality are directly attributed to the CI/CD pipeline. Industry claims that a CI/CD pipeline improves deployment speed as stated in the previous paragraph but there is minimal research to validate this claim. One group of Oregon State researchers conducted a study on CI improvements and reported that Flickr increased deployment to production more than 10 times per day and a product group at Hewlett Packard reduced development costs by 78% (Hilton, Tunnell, Huang, Marinov, & Dig, 2016).

Additionally the research found that projects that use CI average 0.54 releases per month, while projects that do not use CI average 0.24 releases per month and projects that eventually added CI used to release at a rate of 0.34 releases per month, below the average 0.54 rate they now release using CI (Hilton, Tunnell, Huang, Marinov, & Dig, 2016). Interestingly, Hilton noted that despite the increasing attention to CI and touted successes there is very little attention from the research community. The program office effort to build a continuous integration environment will be of benefit even if the research does not demonstrate definitive improvements. The current structure of the AF DCGS lab environments, processes, tools, and architecture are not cohesive, have redundancy, competing objectives, and anything but seamless integration. The current processes, policies and culture within the AF DCGS Program Office, ACC, and the 480th ISR Wing increase the difficulty for rapid deployment of software updates to the fielded weapon system, regardless of integration and test methodology. The intent of this research is to provide data that demonstrates the improvements to speed of delivery for the CI/CD pipeline and recommend future research and actions to assist with a successful CI/CD pipeline. Of special note is that any acronyms not spelled out are due to classification concerns and only the acronym is used throughout this thesis.

Research Objectives and Questions

The objective of this research is to investigate whether the implementation of a CI/CD process can improve AF DCGS software delivery timelines and software quality. Using a CI/CD process combined with the automated tools would normally increase software delivery throughput and quality, but with outside constraints and program culture to overcome, the efficiencies may not be realized. This research will compare the AF DCGS software delivery process that uses a CI/CD pipeline to the AF DCGS Request for Change (RFC) software delivery process that uses an integrated test cycle. Answering the following research questions could provide possible improvements to the AF DCGS software delivery process.

1. Has the implementation of the CI/CD pipeline reduced the software delivery timelines?
    a. For agile software work types what is the difference in workdays between using the CI/CD process and the RFC process?
    b. What differences can be observed with respect to time? Are the changes completing faster or slower as the CI/CD pipeline matures? If they are completing slower, can one or more causes be identified?
2. What differences in speed of delivery exist per agile software work type?
    a. Are there differences in speed of delivery based on priority? Are the high priority agile software work types resolved faster than the medium and low?
    b. Are there differences in speed of delivery based on value streams?
    c. Are there differences in speed of delivery based on story points? Do the higher story points take longer to complete?

Methodology

This research will consist of studying the implementation of the environments, processes, and tools used by the AF DCGS program office to build a CI/CD pipeline and then performing analysis on the RFC integration and test process compared to the CI/CD pipeline. The interfaces between the RFC process and the CI/CD process will be studied and analyzed. Additionally, performance measures will be compared from work executed using the CI/CD process versus the RFC process. Finally, the standard agile performance measures will be examined for the CI/CD pipeline. The metrics focus on the AF DCGS software development process and the continuous integration/continuous delivery capability currently in development.

The data for the existing process and the CI/CD process will be retrieved from the DI2E DevTools collaborative area and the AF DCGS CM libraries on Intelink. The DI2E area contains a wealth of qualitative data from the entire team with information and notes on status, issues, problems and rework. Quantitative data for software integration and test activity timelines will be collected on all completed work from Jul 2018 to Oct 2020 to evaluate the effects of the continuous integration and continuous deployment pipeline. Metrics will be collected and analyzed to compare software implemented through the CI/CD pipeline and software implemented through the RFC process. Between-subjects and within-subjects studies on pipeline data will analyze five different types of changes and three different attributes to further explore the effectiveness of the pipeline.

Assumptions and Limitations

The CI/CD process and the RFC process both employ agile software development principles and use Confluence and Jira for project tracking. The CI/CD pipeline started development in July 2018 with formal sprints starting January 2019 and 41 sprints currently completed (Lambert, Arnold, Sylvester, Koyle, & Dent, 2020). The RFC process started using agile practices in 2015 with a gradual roll-in of each of the projects/value streams completed by 2017. Based on the CI/CD pipeline development timeline this research will only study data from July 2018 to Oct 2020.

The analysis between the CI/CD pipeline and the RFC process will include all data from July 2018 to October 2020. The CI/CD within-subjects study will only include three value streams due to lack of enough data from the other value streams. The RFC process will include only OA DCGS value streams and will not include any legacy work. The differences between the OA and legacy architecture and processes are significant enough that comparisons in the data would not be conclusive evidence of any improvements due to the CI/CD pipeline.

Thesis Preview

Chapter II focuses on exploring the literature, architecture, and design for the AF DCGS CI/CD pipeline and the CIE. The review will include background on the AF DCGS weapon system, the intelligence process to understand the DCGS mission, defining the concepts of a CI/CD pipeline, providing insight into the tools and platforms used to manage the software change process and investigating the current implementation of the AF DCGS CI/CD pipeline. The details of the methodology used to analyze the CI/CD pipeline, the selection criteria of metrics, data cleaning methods, and explanation of evaluation criteria used for side-by-side comparisons are found in Chapter III. Chapter IV provides the results of the performance analysis of the CI/CD pipeline and compares it to the RFC software development performance. Conclusions are summarized and presented in Chapter V along with significant findings within the data, lessons learned, and recommendations for future research.

II. Literature Review

Chapter Overview

The purpose of this chapter is to provide context on the importance of a fully functioning CI/CD pipeline for AF DCGS. First, the Joint Chiefs of Staff Joint Doctrine for Intelligence Operations is explained to establish the purpose for the AF DCGS mission, followed by a breakdown of AF DCGS from operational, logical, and architectural views and then the AF DCGS change control process is discussed to frame how the CI/CD pipeline fits into the sustainment/modification process. Next, the Defense Intelligence Information Enterprise (DI2E) Devtools environment used by the AF DCGS team for change control management and business intelligence is explained. Then the RFC software delivery process is described explaining how the Jira workflow relates to the change control process. The components of a CI/CD pipeline are described with an examination of the AF DCGS implementation and how the automated tools in the CIE streamlines the software delivery process. This chapter concludes with the research hypothesis and a summary of the literature review.

Joint and National Intelligence

The objective of joint intelligence operations is to provide accurate and timely intelligence to commanders that promotes information superiority throughout the operational environment (Scott, Weaver, Brown, & Browder, 2017). Joint intelligence doctrine describes the roles and relationships of intelligence organizations from the national level down to the subordinate joint force levels. The doctrine emphasizes, "The goal is to maximize intelligence support to military operations by increasing the efficiency of the intelligence process and the effectiveness of the intelligence organizations that support the Joint Force Commander" and "Agile intelligence processes and procedures must be understood and utilized across the intelligence enterprise" (Scott, Weaver, Brown, & Browder, 2017). AF DCGS is attempting to meet the goal to be a more effective intelligence organization through developing the CI/CD pipeline to reduce the time spent in the integration and test cycle and removing unnecessary wait times in the existing processes. As the Air Force's primary intelligence, surveillance and reconnaissance (ISR) planning and direction, collection, processing and exploitation, analysis and dissemination (PCPAD) weapon system, the DCGS program office is actively employing new technologies in the systems engineering acquisition lifecycle (U.S. Air Force, 2015).

The PCPAD process--**P**lanning and direction, **C**ollection, **P**rocessing and exploitation, **A**nalysis and production, and **D**issemination and integration—referenced in Figure 1 describes how the various types of intelligence are integrated to meet the

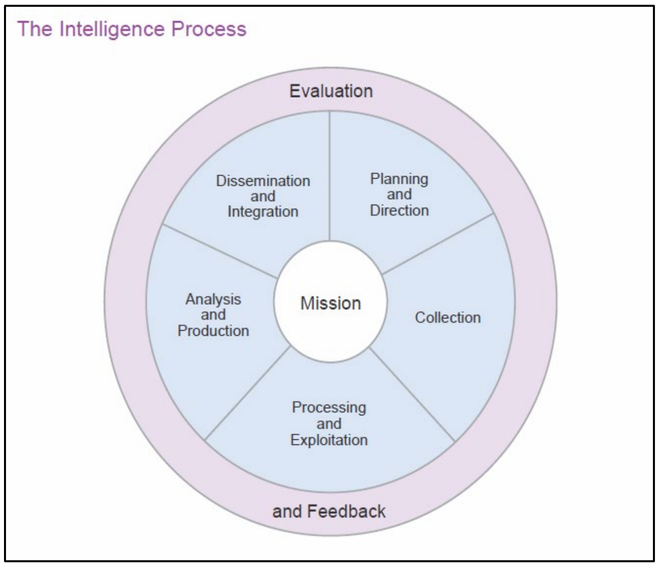


Figure 1. The Intelligence Process

commander's intelligence needs (Joint Chiefs of Staff, 2013). This process is not a rigid workflow requiring one-step to be accomplished before the next can begin or for all steps to be accomplished. For example, while electronic intelligence data is being processed and disseminated, there can be simultaneous cross-cueing of additional platforms for further intelligence collection. Likewise, information can be disseminated from the sensor on an unmanned aerial vehicle directly to the user without going through the analysis and production step (Scott, Weaver, Brown, & Browder, 2017).

Planning and Direction

Intelligence planning focuses on the optimal employment of assets, sensors, and PED systems across the full spectrum of joint operations in order to provide the commander with the data to achieve operational objectives. Defining intelligence requirements, developing the intelligence architecture and a collection plan, and preparing and then issuing the request for information to the information collection

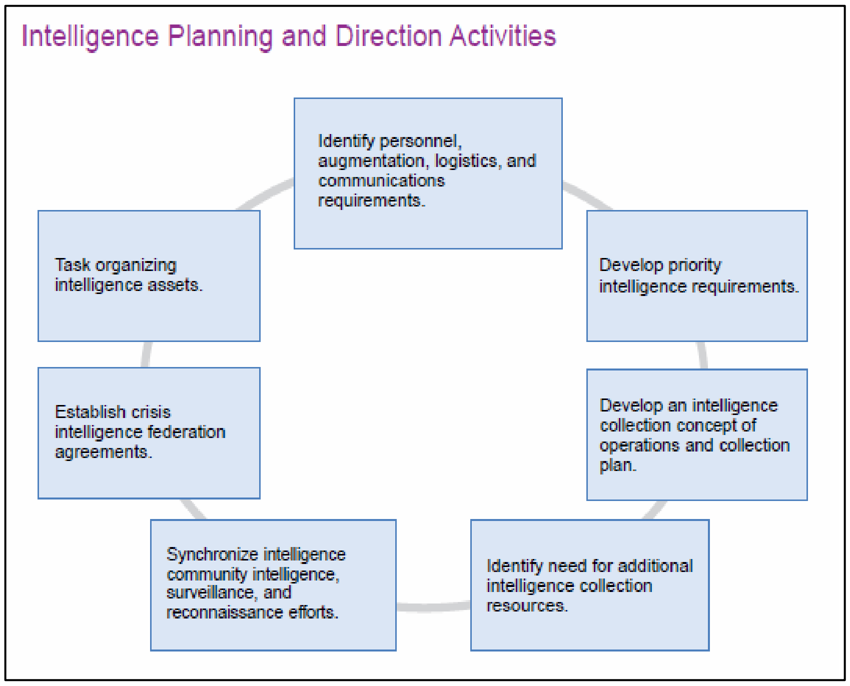

Intelligence Planning and Direction Activities
Identify personnel, augmentation, logistics, and communications requirements.
Develop priority intelligence requirements.
Develop an intelligence collection concept of operations and collection plan.
Identify need for additional intelligence collection resources.
Synchronize intelligence community intelligence, surveillance, and reconnaissance efforts.
Establish crisis intelligence federation agreements.
Task organizing intelligence assets.

Figure 2. Planning and Direction

agencies are the major activities for intelligence planning and direction as shown in Figure 2 and Figure 3 (Scott, Weaver, Brown, & Browder, 2017).

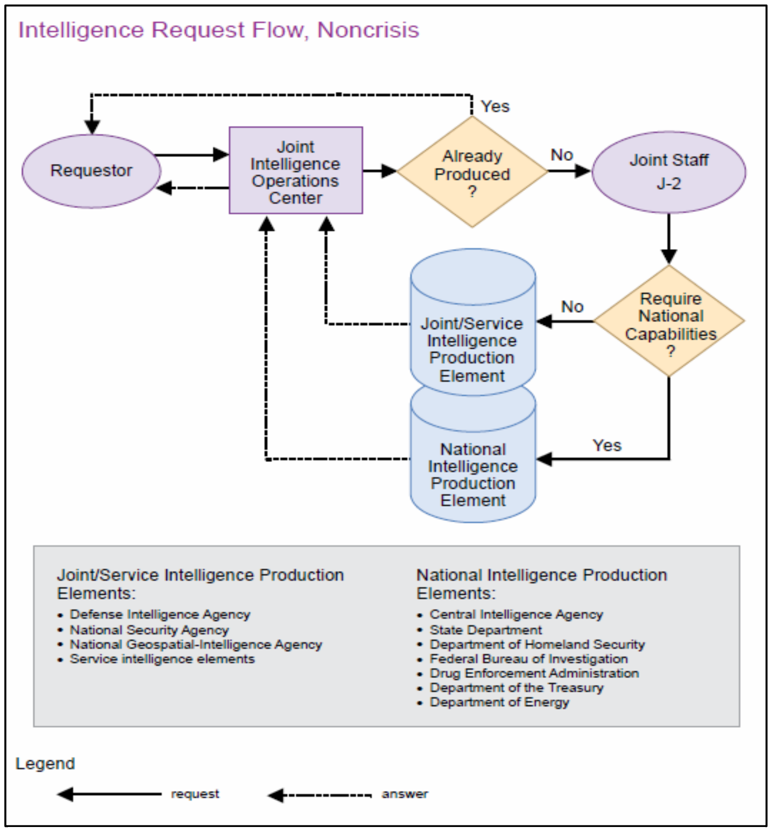

Intelligence Request Flow, Noncrisis
Yes
Requestor
Joint Intelligence Operations Center
Already Produced ?
No
Joint Staff J-2
Require National Capabilities ?
No
Joint/Service Intelligence Production Element
Yes
National Intelligence Production Element
Joint/Service Intelligence Production Elements:
• Defense Intelligence Agency
• National Security Agency
• National Geospatial-Intelligence Agency
• Service intelligence elements
National Intelligence Production Elements:
• Central Intelligence Agency
• State Department
• Department of Homeland Security
• Federal Bureau of Investigation
• Drug Enforcement Administration
• Department of the Treasury
• Department of Energy
Legend
request
answer

Figure 3. Intelligence Request Flow

Collection

Collection is the activity of acquiring data to satisfy the requirements specified in the collection strategy as shown in Figure 4 (Scott, Weaver, Brown, & Browder, 2017). Collection managers who select the most appropriate, available asset and then task the selected asset to conduct collection missions manage this activity. Collection managers may also direct dynamic cross-cueing of sensors to obtain higher confidence data. ISR visualization supports the collection activity by providing an easily comprehended

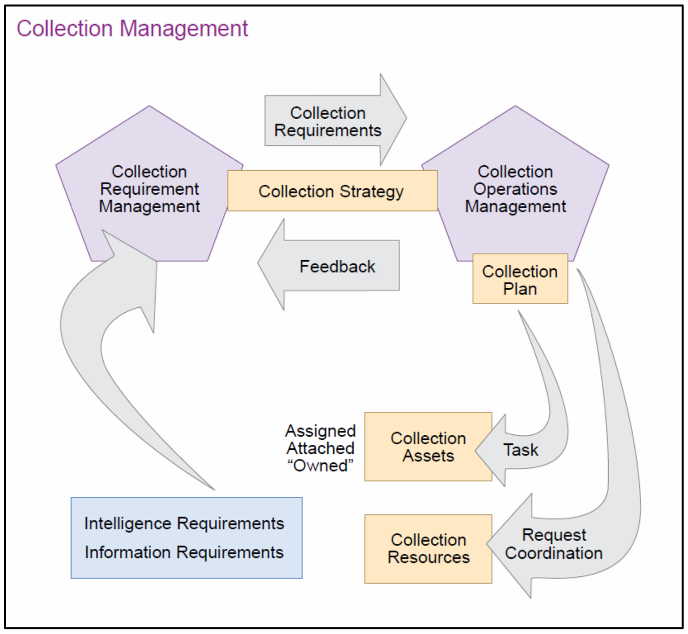


Figure 4. Collection Management

graphic display that depicts the current and future locations of collection assets, their capabilities, their field of regard, and their tasked targets. ISR visualization requires continuous feedback regarding the current and projected locations of all collection assets relative to their planned ground tracks as shown in Figure 5 (Scott, Weaver, Brown, & Browder, 2017). The ISR visualization display correlates in real time the collection status and location of all planned collection targets and the specific collection asset tasked to collect on each target. ISR visualization displays also depict the effects of the operating environment on the collection capabilities of individual airborne collection platforms as they progress along preplanned or ad hoc flight paths (e.g., the impact of terrain masking on sensor fields of regard at various altitudes). AF DCGS is the Air

Force's primary system that provides this integrated common operational picture (Scott, Weaver, Brown, & Browder, 2017).

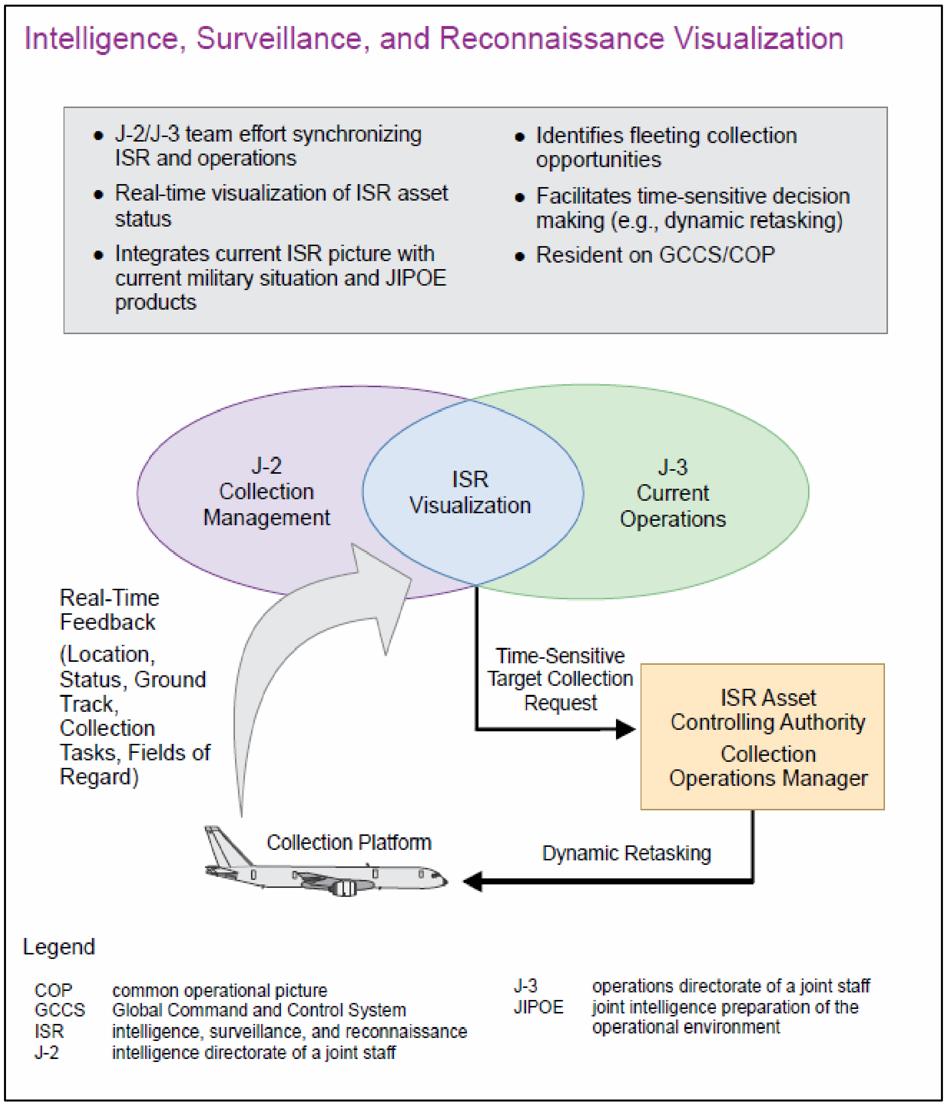

Intelligence, Surveillance, and Reconnaissance Visualization
• J-2/J-3 team effort synchronizing ISR and operations
• Real-time visualization of ISR asset status
• Integrates current ISR picture with current military situation and JIPOE products
• Identifies fleeting collection opportunities
• Facilitates time-sensitive decision making (e.g., dynamic retasking)
• Resident on GCCS/COP
J-2 Collection Management
ISR Visualization
J-3 Current Operations
Real-Time Feedback (Location, Status, Ground Track, Collection Tasks, Fields of Regard)
Time-Sensitive Target Collection Request
ISR Asset Controlling Authority Collection Operations Manager
Collection Platform
Dynamic Retasking
Legend
COP common operational picture
GCCS Global Command and Control System
ISR intelligence, surveillance, and reconnaissance
J-2 intelligence directorate of a joint staff
J-3 operations directorate of a joint staff
JIPOE joint intelligence preparation of the operational environment

Figure 5. ISR Visualization

Processing and Exploitation

Collected data is correlated and converted during the processing and exploitation activity into formats that can be analyzed and then turned into intelligence products as

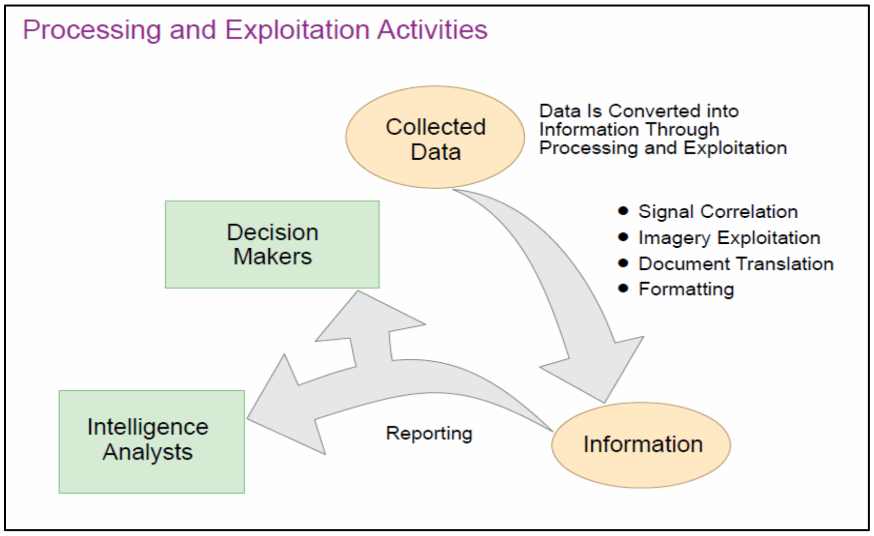


Figure 6. Processing and Exploitation

shown in Figure 6 (Scott, Weaver, Brown, & Browder, 2017). The AF DCGS systems engineering plan states processing and exploitation can be performed in a forward operating location or in a reachback capacity through the use of software applications referred to as mission apps for AF DCGS. Data is processed at the forward location when the environment is disconnected, interrupted or has low-bandwidth. Reachback processing is typically performed at a centralized or federated location, such as the AF DCGS sites (Priddy, AF DCGS Systems Engineering Plan, 2019). The type of processing and exploitation applied to the collected information depends on the mission and purpose for the collection and results in a particular category of intelligence product.

Analysis and Production

Data received from sensors or other methods are in various forms depending on the collection asset used to gather the data. The raw input could be digitized data, unintelligible voice transmissions, large files of imagery, or spools of unprocessed wet film. Trained intelligence specialists convert the raw data into usable information and the resulting products are stored in intelligence databases. Virtual knowledge bases are integrated repositories of multiple databases, reference documents, and open-source material for extremely large and complex data. Analysts can easily access and update the information across the intelligence community with the data organized as shown in

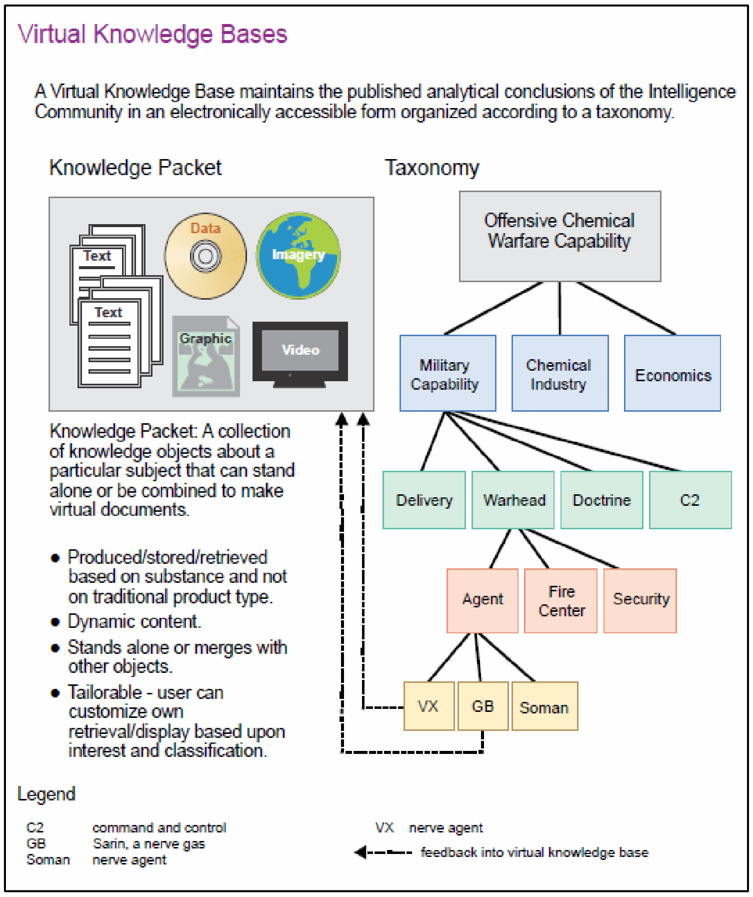

Virtual Knowledge Bases
A Virtual Knowledge Base maintains the published analytical conclusions of the Intelligence Community in an electronically accessible form organized according to a taxonomy.
Knowledge Packet
Taxonomy
Text
Data
Imagery
Text
Graphic
Video
Offensive Chemical Warfare Capability
Military Capability
Chemical Industry
Economics
Delivery
Warhead
Doctrine
C2
Agent
Fire Center
Security
VX
GB
Soman
Knowledge Packet: A collection of knowledge objects about a particular subject that can stand alone or be combined to make virtual documents.
• Produced/stored/retrieved based on substance and not on traditional product type.
• Dynamic content.
• Stands alone or merges with other objects.
• Tailorable - user can customize own retrieval/display based upon interest and classification.
Legend
C2 command and control
GB Sarin, a nerve gas
Soman nerve agent
VX nerve agent
feedback into virtual knowledge base

Figure 7. Virtual Knowledge Base

Figure 7 (Scott, Weaver, Brown, & Browder, 2017). AF DCGS is responsible for the processing, exploitation, storage, cataloguing, and retrieval for all Air Force processed intelligence data or knowledge packet as part of the wider DoD and National intelligence agencies according to the taxonomy shown above.

Dissemination and Integration

Timely dissemination of the finished intelligence products is critical to information dominance. Digital dissemination is the most predominant method that has improved the ability to search, retrieve, and store products across the many intelligence systems and multiple security levels (Scott, Weaver, Brown, & Browder, 2017). AF DCGS analysts post documents to servers such as Intelink, Intelink-S, or NIPRNET to deliver intelligence whenever and wherever required.

Air Force Intelligence

The Deputy Chief of Staff of the Air Force for Intelligence, Surveillance, and Reconnaissance (AF/A2) is responsible for policy formulation, planning, evaluation, oversight, and leadership of AF global integrated ISR capabilities and is directly responsible to the undersecretary of defense for intelligence (Joint Chiefs of Staff, 2013). The 25th AF is a subordinate to Air Combat Command (ACC) with responsibility for executing AF/A2's global integrated ISR. As such, they provide multisource ISR products, applications, capabilities and resources, to include cyberspace and geospatial forces and expertise. There are many 25th AF organizations, but the relevant ones for this research are the 480th and 70th ISR Wings that provide global distributed and reachback ISR. The 70th ISR Wing works closely with the NSA/CSS, leveraging the net-centric capabilities of a worldwide cryptologic enterprise and the 480th ISR Wing responsibilities include national cryptologic, IT, cyberspace ISR, tactical analysis, commander support for the joint force air component, and SIGINT integration (Scott, Weaver, Brown, & Browder, 2017). The 480th Wing executes these responsibilities through operational command of the Air Force Distributed Common Ground System weapon system.

AF DCGS

The Air Force Distributed Common Ground System is an intelligence enterprise system that is comprised of 27 geographically separated, networked sites including five core sites across the globe and is a component of the larger Department of Defense (DoD) DCGS Net-Centric Enterprise (U.S. Air Force, 2015). The weapon system has

Figure 8. AF DCGS Ops Floor

evolved from the first Deployable Ground Station-1 (DGS-1) supporting U-2 operations in July 1994 to a true distributed ISR operations network that interconnects platforms, sensors, and airman to provide critical intelligence to warfighters at the tactical level

(Dasovich, 2017). Intelligence analysts produce actionable, multi-discipline intelligence derived from multiple ISR platforms as shown in Figure 8 (U.S. Air Force, 2015).

AF DCGS performs this mission by supporting Combatant Commanders (COCOMs) and forces – primarily at the Joint Task Force (JTF) level and below – with actionable, decision-quality information. It operates with the full flexibility of the established intelligence process, as detailed in Joint Publication (JP) 2-01, Joint and National Intelligence Support to Military Operations, to make usable information immediately and simultaneously available to both engaged forces and intelligence analysts (Gutierrez, 2020). AF DCGS takes advantage of AF, sister services, national, and coalition sensors in the air, on land, in space, and at sea spanning Multiple Intelligence (Multi-INT) sources and provides tailored, correlated information as described in Figure 9 (Gutierrez, 2020).

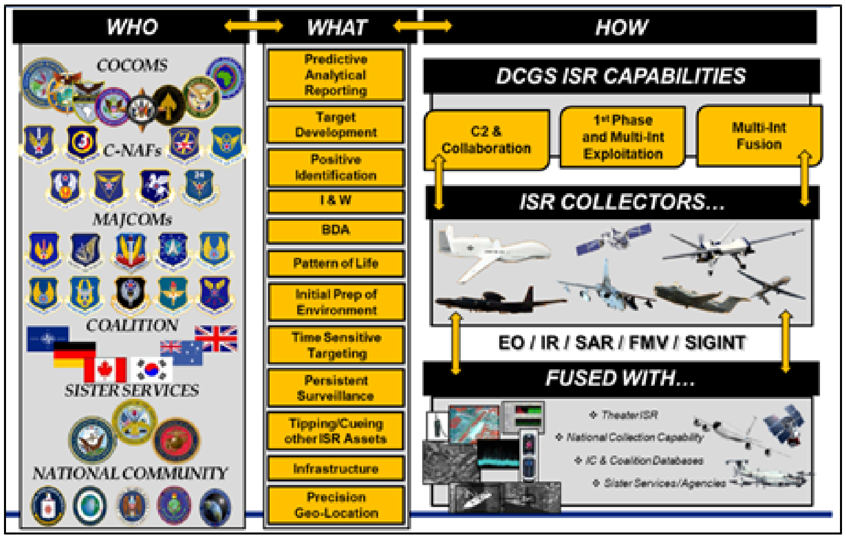


Figure 9. AF DCGS Global ISR Support

AF DCGS operations crew consists of Signals Intelligence (SIGINT) (to include Communications Intelligence (COMINT) and Electronic Intelligence (ELINT)), Geospatial Intelligence (GEOINT), Imagery Intelligence (IMINT), Measurement and Signatures Intelligence (MASINT), and mission management operators. The data collection and analysis results in a particular category of intelligence product. The Office of the Director of National Intelligence defines intelligence into the categories listed in Table 1 (ODNI, 2020) and the DCGS capabilities that support the intelligence process is shown in Figure 10 (Gutierrez, 2020).

Table 1. Categories of Intelligence

GEOINT	Geospatial Intelligence	Geospatial Intelligence is the analysis and visual representation of security related activities on the earth. It is produced through an integration of imagery, imagery intelligence, and geospatial information.
SIGINT	Signals Intelligence	Signals intelligence is derived from signal intercepts comprising -- however transmitted -- either individually or in combination: all communications intelligence (COMINT), electronic intelligence (ELINT) and foreign instrumentation signals intelligence (FISINT). The National Security Agency is responsible for collecting, processing, and reporting SIGINT. The National SIGINT Committee within NSA advises the Director, NSA, and the DNI on SIGINT policy issues and manages the SIGINT requirements system
MASINT	Measurement and Signature Intelligence	Measurement and Signature Intelligence is technically derived intelligence data other than imagery and SIGINT. The data results in intelligence that locates, identifies, or describes distinctive characteristics of targets. It employs a broad group of disciplines including nuclear, optical, radio frequency, acoustics, seismic, and materials sciences. Examples of this might be the distinctive radar signatures of specific aircraft systems or the chemical composition of air and water samples. The Directorate for MASINT and Technical Collection (DT), a component of the Defense Intelligence Agency, is the focus for all national and Department of Defense MASINT matters.
HUMINT	Human Intelligence	Human intelligence is derived from human sources. To the public, HUMINT remains synonymous with espionage and clandestine activities; however, most of HUMINT collection is performed by overt collectors such as strategic debriefers and military attaches. It is the oldest method for collecting information, and until the technical revolution of the mid- to late 20th century, it was the primary source of intelligence.
IMINT	Imagery Intelligence	Imagery Intelligence includes representations of objects reproduced electronically or by optical means on film, electronic display devices, or other media. Imagery can be derived from visual photography, radar sensors, and electro-optics. NGA is the manager for all imagery intelligence activities, both classified and unclassified, within the government, including requirements, collection, processing, exploitation, dissemination, archiving, and retrieval.

OSINT	Open-Source Intelligence	Open-Source Intelligence is publicly available information appearing in print or electronic form including radio, television, newspapers, journals, the Internet, commercial databases, and videos, graphics, and drawings. While open-source collection responsibilities are broadly distributed through the IC, the major collectors are the DNI's Open Source Center (OSC) and the National Air and Space Intelligence Center (NASIC).

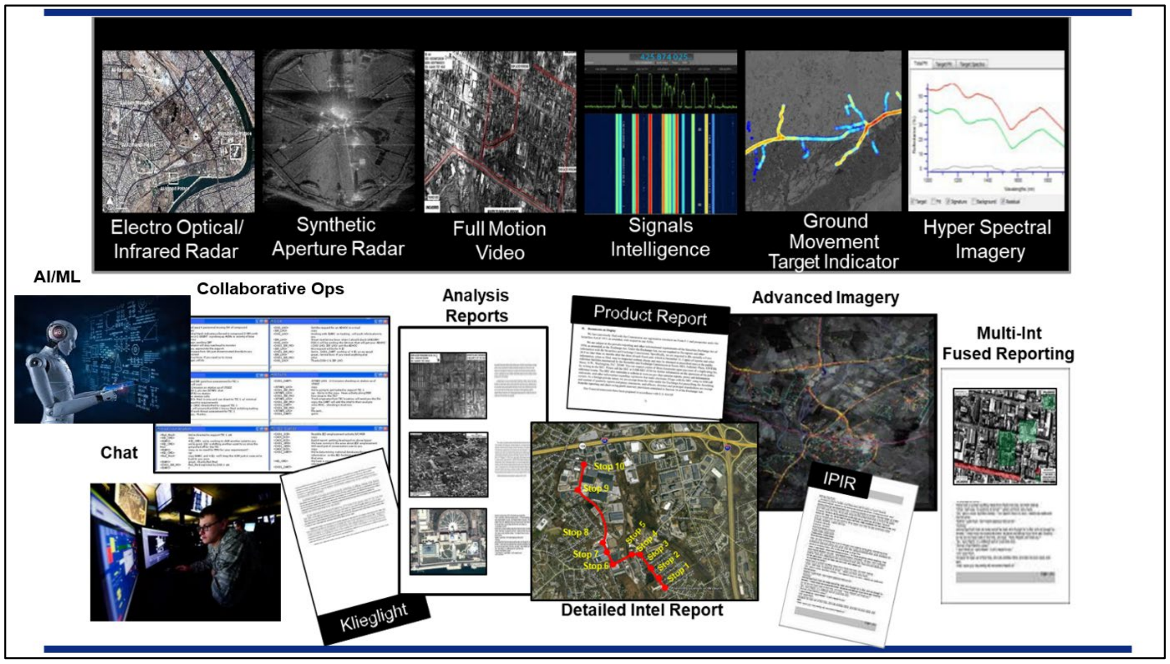


Figure 10. Supported AF DCGS Capabilities

The AF DCGS system is connected through the AF DCGS Campus Area Network/Wide Area Network (CAN/WAN) and employs a global communications architecture that connects multiple intelligence platforms and sensors. The mission and role AF DCGS performs to accomplish the related Joint Capability Areas is shown in Figure 11 (Gutierrez, 2020). AF DCGS has become a complex system over its history of

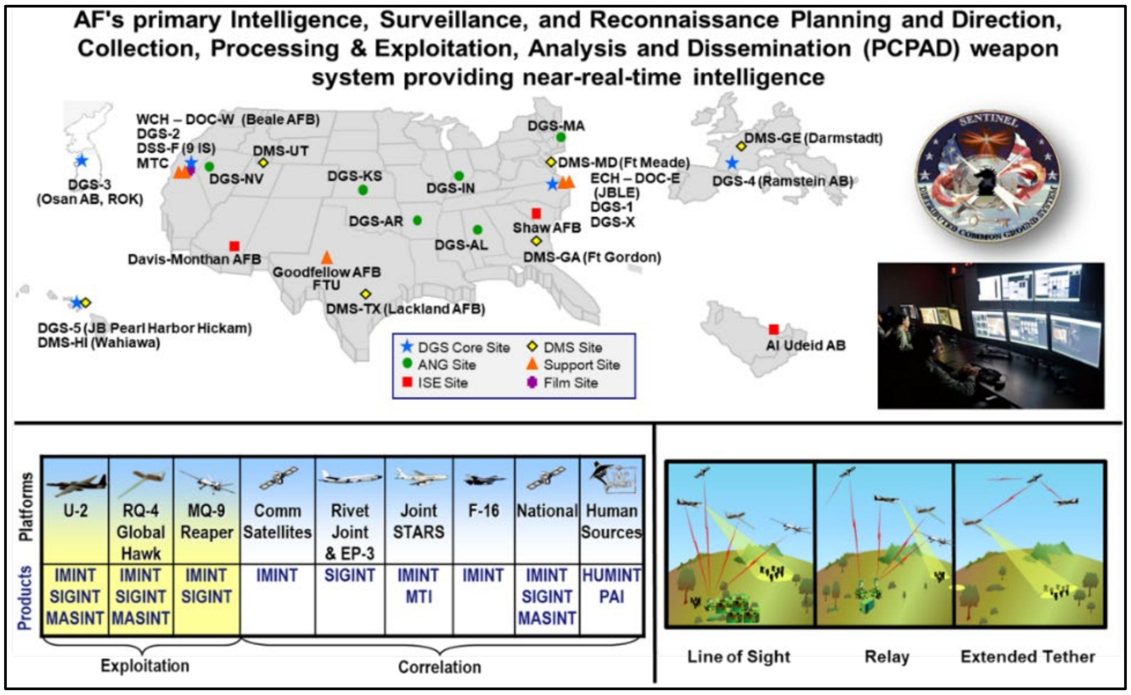


Figure 11. Joint Capability Areas

constant evolution, integration of quick reaction capabilities to meet operational needs in Iraq and Afghanistan and being a constant focal point for new technologies and capabilities within the Defense Intelligence Enterprise (DIE), the Intelligence Community (IC) and the Air Force ISR Enterprise. Addressing these emerging threats led to new sensor data streams and the associated software tools and hardware that were not fully integrated, documented, or tested. The result was a weapon system that became extremely difficult to sustain because of rapid acquisition practices and lack of strong systems engineering processes to ensure a cohesive system design and architecture. To resolve the increasing costs of sustainment and slow delivery of capabilities the AF DCGS program developed an aggressive transition plan and began the migration to an open architecture (Dasovich, 2017).

The OA DCGS migration plan was scheduled to be complete and fully installed at all the operational sites with the legacy system retired by 2020 (Bush, 2019). The OA DCGS hardware stacks and software applications are currently operational with some minor slips to the original timeline for retiring the legacy system (Jarnagin, 2020). Because of these schedule slips both OA and legacy DCGS systems are sustained in the field that becomes a forcing function for the teams to use multiple processes and environments. This is not as simple as maintaining two baselines for the same system. OA and legacy DCGS are entirely different environments, labs, test facilities, and processes that becomes a challenge for the program office with respect to the CI/CD pipeline innovation efforts (Sylvester C. , 2018).

The next sections first discuss AF DCGS from the operational perspectives for the As-Is (Legacy), Modernization (OA), and To-Be (Future Operating Environment) to give the reader an understanding of the AF DCGS mission. Next, the hierarchical services are explained to describe the system's capabilities as seen by the users (or analysts). Finally, the architectural design of OA DCGS is described for an understanding of how "as-a-service" is implemented to allow the rapid development and delivery of software. These sections highlight the ever-increasing system complexity driving the need for a CI/CD pipeline with the ability to rapidly deploy software, hardware, and firmware updates through automation, integrated cyber security, and feedback loops.

Legacy AF DCGS Operational View

An AF DCGS core site executes multi-intelligence PED activities to support time dominant intelligence needs for the warfighter working within Air Tasking Order (ATO) cycle timelines. The high-level legacy operational view in Figure 12 identifies the inability to exploit all sensor collection due to the data not flowing through the remote

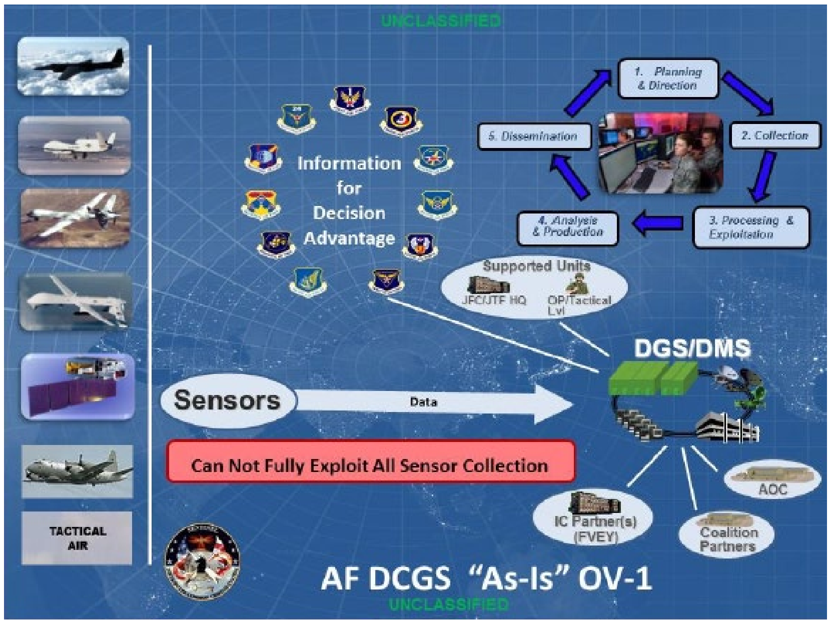


Figure 12. AF DCGS Legacy OV-1

ground stations/hubs, lack of metadata conditioning and standard formats, and lack of data flows out of the DGS to a common data repository (Dasovich, 2017). The raw data flows into the DGS sites and the Air National Guard DMS sites with no distribution to other organizations. Only the intel products that are generated by the DCGS analysts are sent to the AOC, COCOM, Intelligence Community and Coalition partners. This shortcoming is in part attributable to the AF DCGS system evolving from a series of disparate quick reaction capabilities bolted together to meet critical needs during operations in Iraq and Afghanistan.

Unlike traditional acquisitions for major weapon systems, the AF DCGS system did not go through the normal systems engineering requirements and design that result in a cohesive system. Additionally, the typical strategy of having one lead system integrator to seamlessly integrate all the subsystems into one system was not executed due to the urgent need from real-world activity. Contracts were awarded to multiple OEMs for a particular capability and they maintained control and proprietary rights to their portion of the weapon system. This resulted in multiple operating systems, different hardware server stacks, and duplication of enterprise services such as identity and asset management, domain name services, and network time. In addition, the system had aging equipment for cyber protection and timelines of 150 to 200 days to integrate, test, and install software patches and updates at the sites (Cazares, Request for Change, 2019). The result was a complex system of systems with a hardware and enterprise services architecture preventing the rapid delivery of software upgrades to the operational sites.

OA DCGS Operational View

The migration to an open architecture has removed the barriers that were preventing AF DCGS from rapidly integrating new and non-traditional ISR sensor data from 5th generation platforms (Dasovich, 2017). Locally stored data at DCGS sites, non-standard data formats, and proprietary interfaces are being replaced with centrally pooled operational sensor data that is virtually hosted on enterprise servers and cloud computing technologies. Figure 13 (Dasovich, 2017) illustrates the introduction of hub-based storage, ability to process all sensors through meta-data conditioning at the hubs, and shared applications and data in an Intelligence Community (IC) integrated

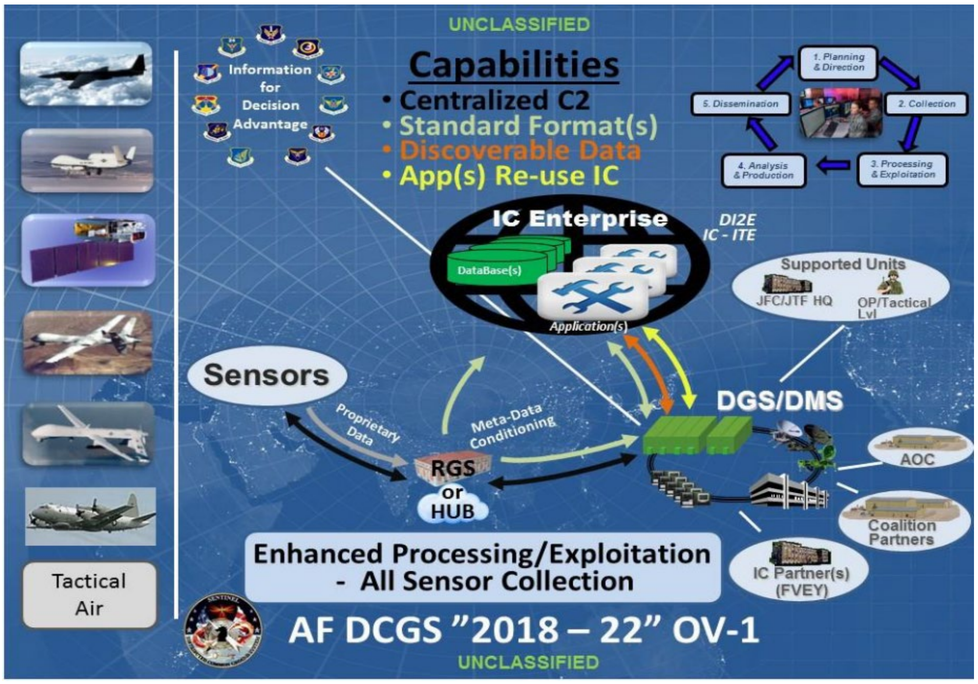


Figure 13. AF DCGS Modernization OV-1

environment. The raw sensor data is now shared across the community, not just the finished intel products produced by the DCGS analysts.

Updating the DCGS weapon system hardware and infrastructure provided the runway to enable rapid integration of new capabilities. The DCGS integration and test labs, systems engineering processes and waterfall software development processes also required modernization to support rapid delivery of software (Durante & Haga, 2015). The first iteration of process improvement consisted of an environment based on Scaled Agile Framework (SAFe) principles that enabled an agile software development process, program increments, and integrated test cycles (Priddy, AF DCGS Agile Execution

Guide, 2018). While it was expected these agile processes would field capabilities via an established 30-day release cadence, no analysis was performed to validate the assumption (Durante & Haga, 2015). The 30-day release cadence was never established and the CIE Tiger Team recommendation re-vectored efforts to establish a CI/CD pipeline using a set of tools on the Controlled Test Environment. The CI/CD pipeline and associated continuous integration environments will be discussed in detail in Chapter III.

AF DCGS Future Operating Environment Operational View

The Air Force continues to move aggressively towards true seamless operations in multiple information and security domains with an exponential growth in the ability to share data. The future of DCGS is to provide a platform that creates an environment for all data to be discoverable and accessible. Achieving the operational goal in Figure 14

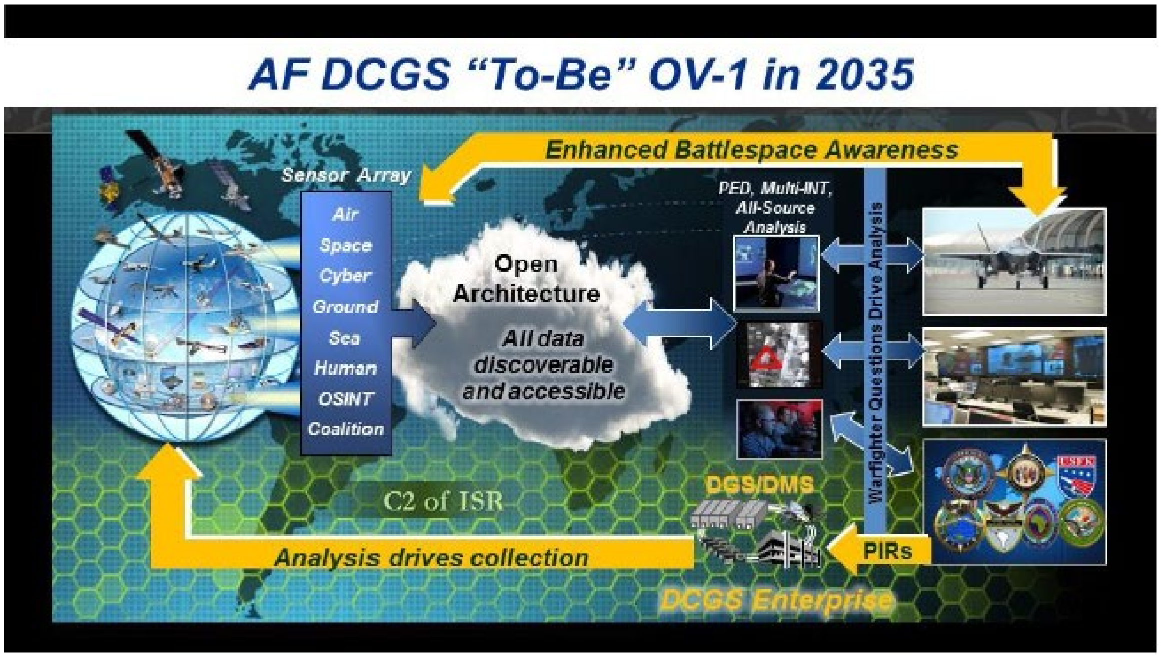

AF DCGS "To-Be" OV-1 in 2035
Enhanced Battlespace Awareness
Sensor Array
Air
Space
Cyber
Ground
Sea
Human
OSINT
Coalition
Open Architecture
All data discoverable and accessible
PED, Multi-INT, All-Source Analysis
Warfighter Questions Drive Analysis
DGS/DMS
C2 of ISR
Analysis drives collection
PIRs
DCGS Enterprise

Figure 14. AF DCGS Future Operating Environment OV-1

will require a system that can integrate changes in days, not months. While the ability to deploy mission applications in days is an operational goal, a system that will be extensively connected across a huge network will need the ability to deploy security patches and infrastructure updates in hours as a functional goal. The assumption is the CI/CD pipeline will enable this rapid integration and delivery as will be tested through the analysis described in Chapter III.

AF DCGS Logical View

The logical view is an architectural model that depicts a system's capabilities as seen by the users (Howard, 2014). AF DCGS provides three core software service layers that represent how analysts and operators interface with the system. The first layer is the user interface services that run on the common workstations and includes the analyst widgets, tools, and visualization services. The second layer contains the mission applications and core services that are designed to automate processing and workflow tasks. The third layer is the infrastructure services layer that contain common services required for managing and sustaining a large IT networked infrastructure. The logical view is shown in Figure 15 (Howard, 2014).

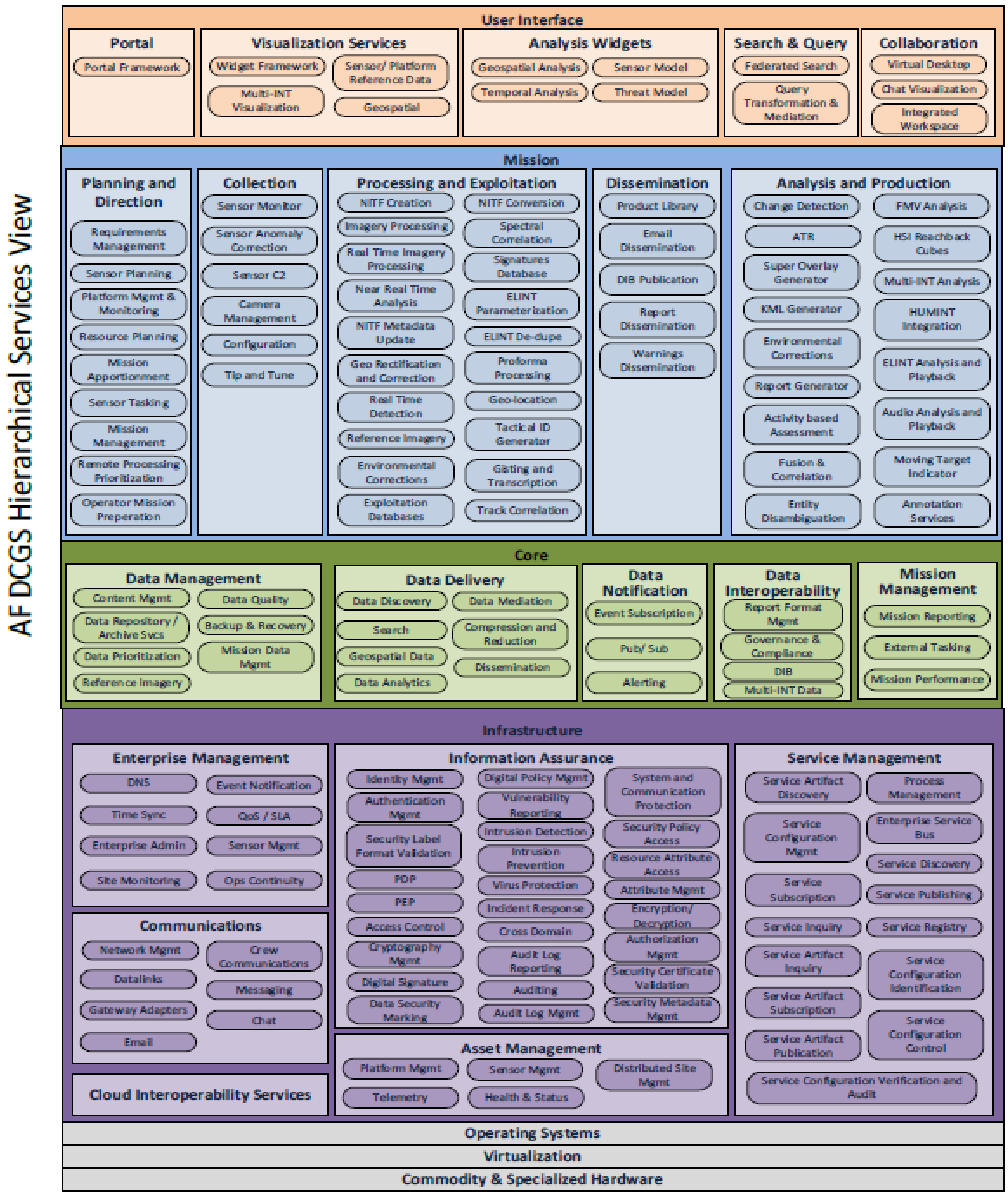


Figure 15. AF DCGS Logical View

The user interface layer contains four groups of services. The common widget/portal framework service contains basic visualization applications and the core visualization framework such as portal and widget libraries. This layer is designed to provide a common look and feel for every workstation and analyst across all the different intelligence areas. The exploitation utilities service contain intelligence sensor exploitation for all sensor types. The geospatial service provides a common map infrastructure and map query services. The collaboration service provides the ability to share information and chat with internal and external partners.

The mission and core services layer contains eight groups of services. These services are decomposed into the five PCPAD workflows and three core services required to execute the PCPAD mission. Table 2 describes at a high level the AF DCGS activity for each of these services (Howard, 2014).

Table 2. AF DCGS Mission Application Services

	Service	Description
Mission Layer	Planning and Direction	Services used for resource allocation, planning a mission, sensor planning, sensor tasking prior to mission, mission apportionment, and similar activities
	Collection	Includes sensor monitoring, corrections, and tasking during mission, as well as ingest of sensor collections
	Processing and Exploitation	Involves image processing, corrections, and correlation of collected data
	Analysis and Production	Includes manual tasks, such as imagery annotation, manual change detection, etc. required to produce products required to answer EEIs
	Dissemination	Includes distribution of exploited data and products to team members and stakeholders
Core Layer	Mission Data Management	Includes services required for managing data required for mission success, whether actual mission data storage (e.g. GMTI Store), mission reference data (e.g. modern modulations or MASINT Signatures DB), or reporting archives
	Data Delivery and Notification	Includes capabilities to deliver data to users; enables users to search through stored data; and notifies users of changes to data of interest
	Data Interoperability	Includes services required to format data to satisfy IC or mission partner data standards.

The Infrastructure services layer supports the execution and management of the other services. The infrastructure services shown in Table 3 includes the hardware, virtualization, operating systems and software for the mission services layer to run on (Howard, 2014).

Table 3. AF DCGS Infrastructure Services

	Service	Description
Infrastucture Layer	Enterprise Management	Includes services required to operate the DCGS infrastructure needed to conduct missions
	Asset Management	Includes services for monitoring and/or managing sites, platforms, sensors, and other assets
	Information Assurance	Includes services required to ensure information is available to DCGS operators and mission partners
	Service Management	Includes services required to manage and monitor the various services defined in this document
	Communication Services	Includes services required to enable communications across DCGS and with mission partners

AF DCGS Architectural View

AF DCGS has migrated to an open architecture where the hardware, virtualization and enterprise services are abstracted away from the mission application layer. The government owned hardware, virtualization layer, common operating system, and enterprise services are the items colored gray in Figure 16 (D'hara, 2020). The mission applications are in the green, blue, and purple areas and are OEM owned and controlled.

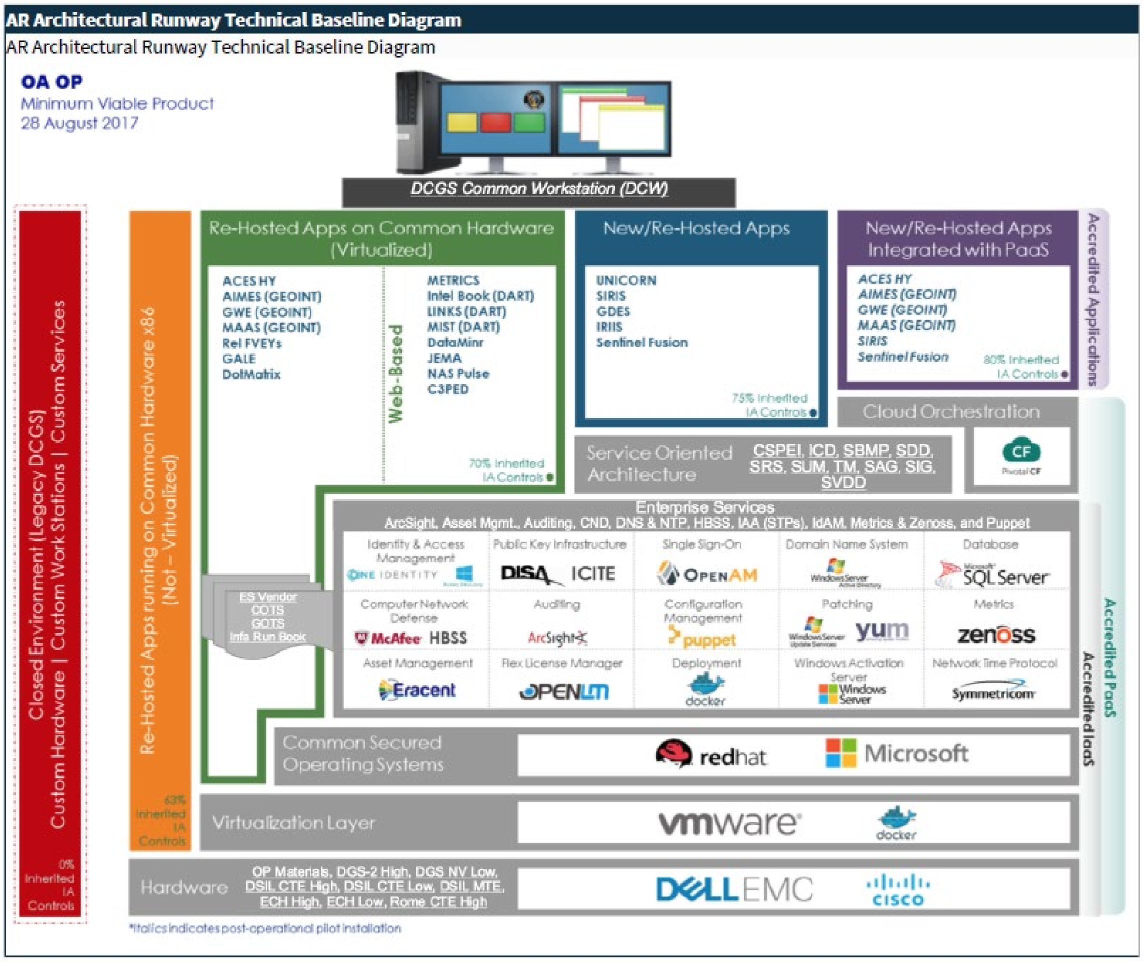

AR Architectural Runway Technical Baseline Diagram
AR Architectural Runway Technical Baseline Diagram
OA OP
Minimum Viable Product
28 August 2017
DCGS Common Workstation (DCW)
Closed Environment (Legacy DCGS)
Custom Hardware | Custom Work Stations | Custom Services
0% Inherited IA Controls
Re-Hosted Apps running on Common Hardware x86 (Not – Virtualized)
63% Inherited IA Controls
Re-Hosted Apps on Common Hardware (Virtualized)
ACES HY
AIMES (GEOINT)
GWE (GEOINT)
MAAS (GEOINT)
Rel FVEYs
GALE
DotMatrix
Web-Based
METRICS
Intel Book (DART)
LINKS (DART)
MIST (DART)
DataMinr
JEMA
NAS Pulse
C3PED
70% Inherited IA Controls
New/Re-Hosted Apps
UNICORN
SIRIS
GDES
IRIIS
Sentinel Fusion
75% Inherited IA Controls
New/Re-Hosted Apps Integrated with PaaS
ACES HY
AIMES (GEOINT)
GWE (GEOINT)
MAAS (GEOINT)
SIRIS
Sentinel Fusion
80% Inherited IA Controls
Accredited Applications
Cloud Orchestration
Service Oriented Architecture
CSPEI, ICD, SBMP, SDD, SRS, SUM, TM, SAG, SIG, SVDD
Pivotal CF
Enterprise Services
ArcSight, Asset Mgmt., Auditing, CND, DNS & NTP, HBSS, IAA (STPs), IdAM, Metrics & Zenoss, and Puppet
ES Vendor
COTS
GOTS
Infa Run Book
Identity & Access Management
Public Key Infrastructure
Single Sign-On
Domain Name System
Database
Computer Network Defense
Auditing
Configuration Management
Patching
Metrics
Asset Management
Flex License Manager
Deployment
Windows Activation Server
Network Time Protocol
Accredited PaaS
Accredited IaaS
Common Secured Operating Systems
Virtualization Layer
Hardware
OP Materials, DGS-2 High, DGS NV Low, DSIL CTE High, DSIL CTE Low, DSIL MTE, ECH High, ECH Low, Rome CTE High
*Italics indicates post-operational pilot installation

Figure 16. AF DCGS Architectural View

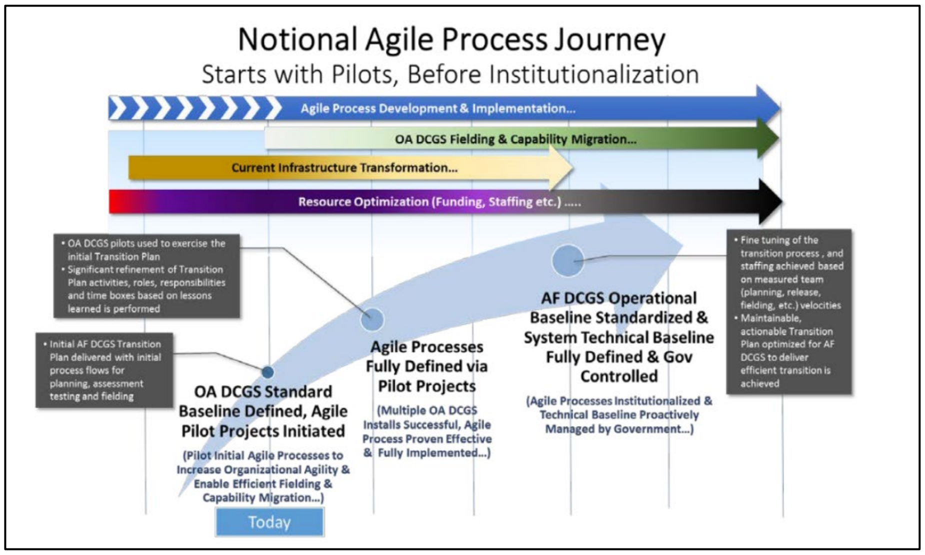
Developing a common infrastructure that integrates all segments of AF DCGS and gains government ownership for the sustainment of the software paved the way to implement a

CI/CD pipeline starting in Sep 2018. The CI/CD pipeline is running concurrently with the previous process using the RFC process. In addition to this new hardware and software architecture, new processes were developed as part of the integration environment to increase effectiveness in delivering capability to the warfighter.

RFC Delivery Process

AF DCGS Change Control Process

The OA DCGS AFRL team developed a change management process using a Jira ticketing system based on the information technology infrastructure library best practices for managing IT systems (Spinelli & Newton, 2016). The program office adopted this process and began implementing agile practices in 2016 using the transition plan shown in Figure 17. They created agile release trains for development teams, moved to program



Notional Agile Process Journey
Starts with Pilots, Before Institutionalization
Agile Process Development & Implementation…
OA DCGS Fielding & Capability Migration…
Current Infrastructure Transformation…
Resource Optimization (Funding, Staffing etc.) …..
• OA DCGS pilots used to exercise the initial Transition Plan • Significant refinement of Transition Plan activities, roles, responsibilities and time boxes based on lessons learned is performed
• Initial AF DCGS Transition Plan delivered with initial process flows for planning, assessment testing and fielding
OA DCGS Standard Baseline Defined, Agile Pilot Projects Initiated
(Pilot Initial Agile Processes to Increase Organizational Agility & Enable Efficient Fielding & Capability Migration…)
Today
Agile Processes Fully Defined via Pilot Projects
(Multiple OA DCGS Installs Successful, Agile Process Proven Effective & Fully Implemented…)
AF DCGS Operational Baseline Standardized & System Technical Baseline Fully Defined & Gov Controlled
(Agile Processes Institutionalized & Technical Baseline Proactively Managed by Government…)
• Fine tuning of the transition process , and staffing achieved based on measured team (planning, release, fielding, etc.) velocities • Maintainable, actionable Transition Plan optimized for AF DCGS to deliver efficient transition is achieved

Figure 17. Agile Process Transition Plan

increments for decomposition of requirements into epics, features and user stories, and an integrated test cycle to develop cadence driven test (Durante & Haga, 2015). However, the existing program office change management process continued to use forms completed by hand centered around traditional aircraft configuration control boards with gates and approvals at senior leader levels and did not use the automated Jira work in progress features (Williams & Clark, DCGS RFC Form Rev 8 Instruction Guide, 2019). The configuration management team could not quickly implement process improvements and metrics collection and analysis was essentially non-existent.

The configuration management team recognized the configuration management processes needed to improve from the current method of email, face-to-face, and shared files on Intelink. The result was a change control process declared operable in September 2019 using the same Jira ticketing system on the DI2E digital collaborative environment that the agile software development teams had been using since 2016 (Cazares & Hamilton, RFC Transition Plan, 2020). The program office fully adopted Confluence and Jira for change management and mandated the Request for Change process on 21 Jan 2020 (Cazares & Hamilton, RFC Transition Plan, 2020). The teams use Confluence and Jira for accurately tracking every request for change to both the legacy and OA versions of the weapon system through the life cycle from initiation to completion.

Request for Change Types

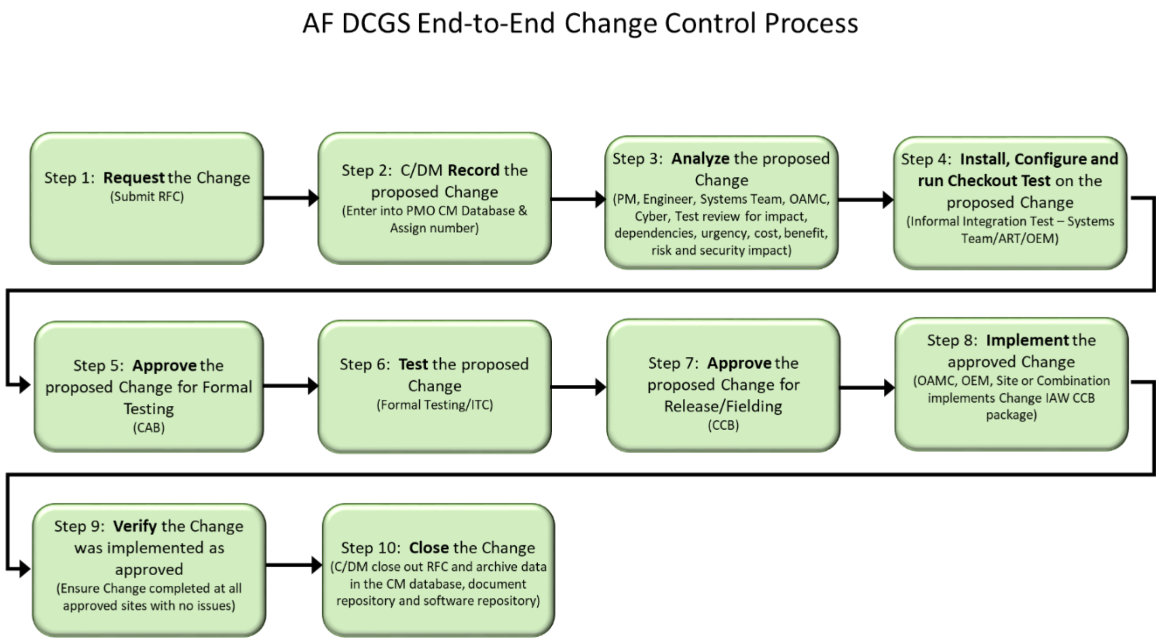
There are four change types: Emergency, Normal, Standard, and Pre-approved (Hamilton, Request for Change Form Instructions, 2020). Emergency changes have a critical impact to mission failure and must be implemented in under 48 hours. Standard changes are low risk, occur frequently, do not change the executable software and are not required to go through any formal approvals (Priddy, AF DCGS Standard Change

Approval Memo for Record, 2019). Pre-approved changes are requests to install a previously approved RFC at an additional site exactly as the original. Normal changes are any change to the baseline that does not fit one of the previously mentioned types and follows the normal process.

In addition, simple bug fixes, security patches, configuration file updates, scripts, and small software changes do not go through the full twenty-eight steps and only require approval at the change advisory board and not the additional configuration control board (Hamilton, Request for Change Form Instructions, 2020). These type of changes are primarily for applications in the infrastructure and user interface services layer. The majority of the mission application changes from the mission services layer are still required to go through the full process due to the broad scope of operational impact and direct effects in the kill-chain. Reference Figure 15 for the details of these layers.

Change Control Process

To understand how the change control process and Jira ticketing workflow process work together, first we examine the change control process shown in Figure 18 (Williams, Hamilton, Cazares, & Noreen, 2020).



AF DCGS End-to-End Change Control Process
Step 1: Request the Change (Submit RFC)
Step 2: C/DM Record the proposed Change (Enter into PMO CM Database & Assign number)
Step 3: Analyze the proposed Change (PM, Engineer, Systems Team, OAMC, Cyber, Test review for impact, dependencies, urgency, cost, benefit, risk and security impact)
Step 4: Install, Configure and run Checkout Test on the proposed Change (Informal Integration Test – Systems Team/ART/OEM)
Step 5: Approve the proposed Change for Formal Testing (CAB)
Step 6: Test the proposed Change (Formal Testing/ITC)
Step 7: Approve the proposed Change for Release/Fielding (CCB)
Step 8: Implement the approved Change (OAMC, OEM, Site or Combination implements Change IAW CCB package)
Step 9: Verify the Change was implemented as approved (Ensure Change completed at all approved sites with no issues)
Step 10: Close the Change (C/DM close out RFC and archive data in the CM database, document repository and software repository)

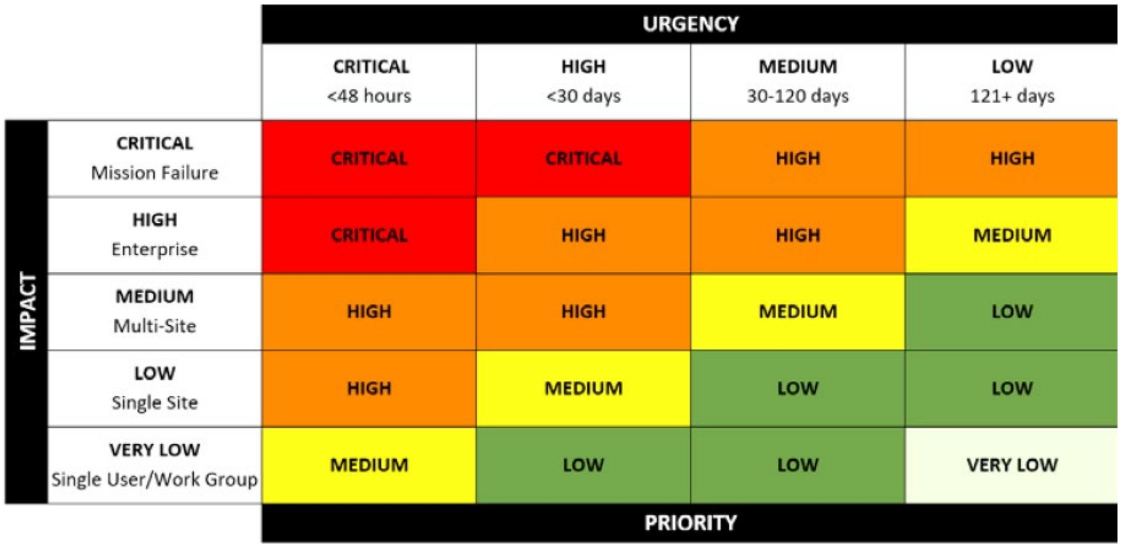
Figure 18. DCGS Change Control Process

Step 1: Request the change. The process begins with the initiation of a request for change by completing the RFC form and submitting it to configuration management. These change requests are for end items (mission app custom software, enterprise services, core shared services, hardware, middleware, or other) that have already been developed by the provider in their factory environment. The initial request contains information such as the source of change, system, system segment, detailed change description with requirement and funding type, need date, and configuration items affected (Williams, Hamilton, Cazares, & Noreen, 2020). The product owner opens a Jira ticket in the RFC tracking system hosted on the DI2E DevTools workspace, which is explained in the RFC Management section.

Step 2: C/DM record the proposed change. C/DM team assigns RFC number.

Step 3: Analyze the proposed change. The entire team determines impacts to operational safety, suitability, and effectiveness, the mission impact and level of urgency based on the priority matrix in Table 4 (Williams, Hamilton, Cazares, & Noreen, 2020),

Table 4. RFC Priority Matrix



		URGENCY			
		CRITICAL <48 hours	HIGH <30 days	MEDIUM 30-120 days	LOW 121+ days
IMPACT	CRITICAL Mission Failure	CRITICAL	CRITICAL	HIGH	HIGH
	HIGH Enterprise	CRITICAL	HIGH	HIGH	MEDIUM
	MEDIUM Multi-Site	HIGH	HIGH	MEDIUM	LOW
	LOW Single Site	HIGH	MEDIUM	LOW	LOW
	VERY LOW Single User/Work Group	MEDIUM	LOW	LOW	VERY LOW
		PRIORITY			

the overall initial risk assessment based on the standard 5 x 5 risk matrix, items impacted, network and port services required, interdependencies, and documentation impacts. The change type is determined and systems engineering reviews all applicable documentation for completeness (Williams, Hamilton, Cazares, & Noreen, 2020).

Step 4: Install, configure and run checkout test. The lab team members prepare the environment to test the change in the integration environment. The test team integrates and tests the changes in the managed test environment and/or the controlled test environment and generates an evaluation report documenting the results and recommendations (Williams, Hamilton, Cazares, & Noreen, 2020).

Step 5: Approve the proposed change for formal test. The product owner submits all applicable documentation to C/DM and requests a Change Advisory Board (CAB). At this point the change has 2 possible paths it can go--approved for release or approved for formal test. If the CAB is authorized to approve the RFC for fielding based on the change type, then this step verifies the package is complete and the CAB approves for fielding. If the CAB is not authorized to approve for fielding, the RFC moves into formal test in the controlled test environment once approved by the CAB (Williams, Hamilton, Cazares, & Noreen, 2020).

Step 6: Test the proposed change. The RFC enters the integrated test cycle and the lab, systems, and test teams execute an integration acceptance test and developmental test to ensure the RFC works in the operationally representative environment and that the functionality meets the user requirements (Williams, Hamilton, Cazares, & Noreen, 2020).

Step 7: Approve the proposed for release/fielding. The integrated team formally presents and briefs the RFC to the CCB using a standard template that ensures all requirements and activity have been accomplished (Williams, Hamilton, Cazares, & Noreen, 2020).

Step 8: Implement the approved change. For changes to OA the OA management center coordinates with the 480th ISR Wing and remotely installs and deploys the change to all approved sites after coordination with each affected site. For legacy changes the program manager coordinates with the 480th ISR Wing for scheduling installation and deployment. The sites have up to 270 days to install the changes (Williams, Hamilton, Cazares, & Noreen, 2020).

Step 9: Verify the change was implemented as approved. C/DM audits the system baseline at all affected labs and sites and verifies the change was implemented as approved (Williams, Hamilton, Cazares, & Noreen, 2020).

Step 10: Close the change. C/DM updates the RFC status to indicate done, withdrawn or not approved (Williams, Hamilton, Cazares, & Noreen, 2020).

Jira Workflow

A standard workflow was established using the Jira work management tool for agile teams to provide and track the activities required to accomplish the change control process that automated the workflow and incorporated security and operations in the existing AF DCGS labs and configuration management processes. The Jira workflow established the standard statuses and definitions that are essential to track progress of the changes through the process. The team defined twenty-eight steps to track the progress of a RFC and provide metrics to determine bottlenecks or issues with a particular activity. A group of subject matter experts from the AF DCGS program office, the OA management center, contractor experts, and AFRL determined expected timelines for each step in the process as shown in Table 5 (Cazares & Hamilton, RFC Transition Plan, 2020). No evidence could be found that these expected timelines are used to manage the projects. A thorough search of the DCGS Confluence pages did not reveal any reporting to these timelines with analysis or corrective action if not met. They are simply displayed on the RFC Leadership Board. These twenty-eight activity steps were divided into swim lanes by area of responsibility for each team. The four teams are the program office team, lab team, test team, and deployment team as shown in Figure 19 that depicts the entire workflow from receipt of a change to fielding (Williams, Hamilton, Cazares, &

Table 5. RFC Expected Timelines

	Status	**Workdays in Phase**		**Status**	**Workdays in Phase**
SPO AOR	Submitted	1	**Test AOR**	TRR, TRR Complete	1
	Systems Engineering Review	3		Ready for IAT (C-IAT) / Ready for ITC	1
	CM in Progress	2		Integration (Systems Integration)	10
	Cyber in Review	3		Integration Test (IAT)	10
Lab AOR	Integration Ready	2		Dev Test (DT)	10
	Ready for MTE and MTE in Progress	5		Needs Review/Decision	1
	Staging Review	3	**SPO AOR**	Prep for CCB	6
	Ready for CTE and CTE in Progress	5		CCB Ready	3
SPO AOR	Prep for CAB	4		CCB Review (Change CCB) / Fielding CCB	1
	Ready for CAB	1		Awaiting Signatures	1
	CAB for ITC / CAB for Field	1	**Deployment AOR**	Ready for Release	1
	Awaiting Signature and Signed	1		Pending 480th	10
				Ready for Deployment	4
				Deployment in Progress	10-90
				OUE (when required)	15
				Implemented	1

Noreen, 2020).

The program office begins the process by receiving a request for change, putting the RFC under configuration control, performing systems engineering and cyber reviews,

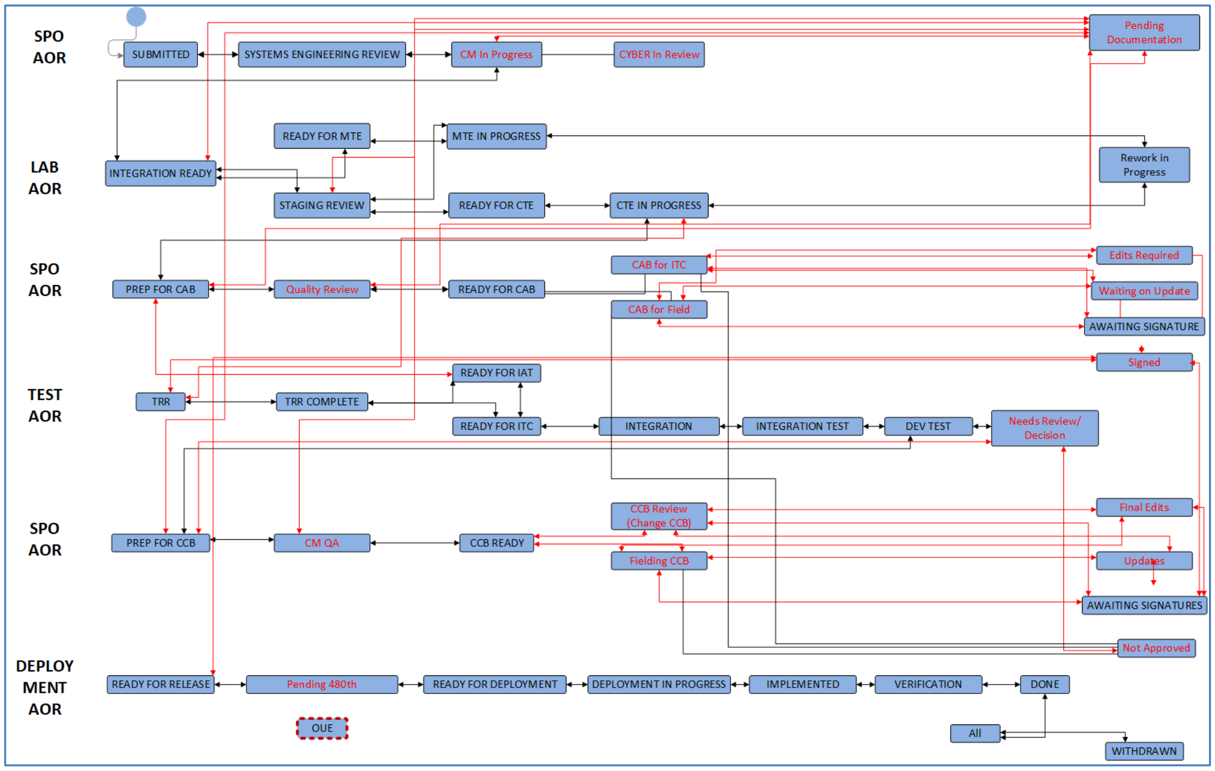


Figure 19. Standard Jira Workflow

and then passing the RFC to the lab team. The lab team takes responsibility and then reviews the change to determine if it is ready for integration and meets the entry criteria. Once the entry criteria is satisfied, the lab team tests the change in the managed and controlled test environments. Once the tests pass and test results documented, responsibility transfers back to the program office. The program office prepares for the CAB, performs a quality check, and then holds either a CAB for fielding or a CAB for the integrated test cycle. Once the CAB approves the change, responsibility goes to the test team for additional testing or to the deployment team for fielding.

If the change went to the test team for the integrated test cycle a Test Readiness Review (TRR) is conducted and once all test entry criteria is met, the change is queued up to wait on the next test cycle. Depending on the timing, the delivery timeline is extended anywhere from 1 day to 6 weeks. Once the integrated test cycle starts, the change goes through Integration Acceptance Test (IAT) and then Development Test (DT). Once the test team successfully completes both IAT and DT, responsibility goes back to the program office team. The program office prepares the boarding package for the change to go through a formal Configuration Control Board (CCB). The team builds the CCB slide deck, performs quality checks, and conducts a formal briefing at the CCB to senior leadership. Once the CCB approves the change, responsibility transitions to the deployment team. This RFC process has remnants of a waterfall software development process with activities like a TRR, CCB, IATs, formal briefings and approvals merged with the new agile development process introduced by the AFRL OA team. This is completely counter to agile development methodology and represents a second layer of testing and additional gates and approvals. The result is much lengthier timelines and many more steps and approval gates than a true agile development process.

Confluence

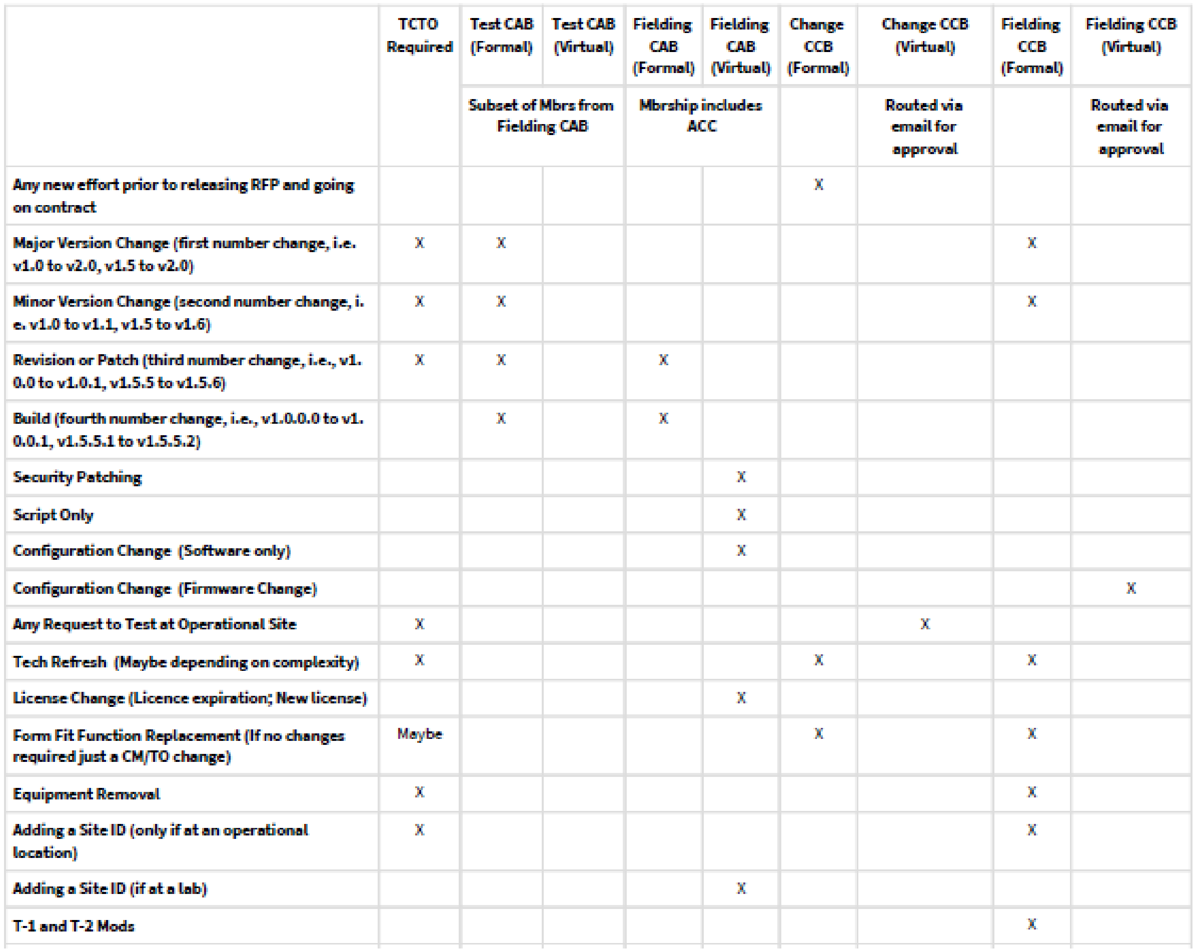
Confluence is a tool hosted on the DI2E DevTools site used for team workspace to share and collaborate on projects with built in functions to accelerate the startup time for building the workspace. Confluence provides wiki, documentation, templates, structures, designs, reports, policies and procedures for teams to create, share and discuss files, ideas, minutes, specs, mockups, diagrams, and projects (DI2E, 2020).

Business Rules

With any system and process that is going to be used by more than one person or team business rules need to be established to ensure a common understanding of roles, responsibilities, permissions, standardization requirements and policies. The program office started creating business rules in Sep 2019 initially focused on the change control process, rules using Jira and Confluence, expected timelines, and required documentation (Sawyer & Smith, 2019). These business rules have had minimal updates since they were established and no visible expansion to include standardization of status, priority, and issue type or focus on identifying attributes that will provide insight into the speed of delivery.

RFC Path Matrix

The RFCs have multiple workflow paths and multiple methods for approval based on the type of change, complicating the process flow even further. For instance, a major version change requires a formal test CAB and formal fielding CCB while a firmware change is the only type that is allowed to be approved through a virtual fielding CCB. Changes such as security patches, scripts or software configuration changes that are lower risk and improve the cyber resiliency of the system have an accelerated workflow and are approved at a virtual fielding CAB. A formal CCB or CAB requires an in-person briefing of the complete CCB package. Configuration management routes the completed package electronically for a virtual CCB or CAB to the core board members for input/recommendation and then to the CCB Chair for approval. Entrance requirements are the same for virtual or formal CCB or CAB. The mapping of change type to approval authority in Table 6 and the tailoring by specific change type is clearly evident (Williams,



	TCTO Required	Test CAB (Formal)	Test CAB (Virtual)	Fielding CAB (Formal)	Fielding CAB (Virtual)	Change CCB (Formal)	Change CCB (Virtual)	Fielding CCB (Formal)	Fielding CCB (Virtual)
		Subset of Mbrs from Fielding CAB		Mbrship includes ACC			Routed via email for approval		Routed via email for approval
Any new effort prior to releasing RFP and going on contract						X			
Major Version Change (first number change, i.e. v1.0 to v2.0, v1.5 to v2.0)	X	X						X	
Minor Version Change (second number change, i.e. v1.0 to v1.1, v1.5 to v1.6)	X	X						X	
Revision or Patch (third number change, i.e., v1.0.0 to v1.0.1, v1.5.5 to v1.5.6)	X	X		X					
Build (fourth number change, i.e., v1.0.0.0 to v1.0.0.1, v1.5.5.1 to v1.5.5.2)		X		X					
Security Patching					X				
Script Only					X				
Configuration Change (Software only)					X				
Configuration Change (Firmware Change)									X
Any Request to Test at Operational Site	X						X		
Tech Refresh (Maybe depending on complexity)	X					X		X	
License Change (Licence expiration; New license)					X				
Form Fit Function Replacement (If no changes required just a CM/TO change)	Maybe					X		X	
Equipment Removal	X							X	
Adding a Site ID (only if at an operational location)	X							X	
Adding a Site ID (if at a lab)					X				
T-1 and T-2 Mods								X	

Table 6. RFC Path Matrix

Hamilton, Cazares, & Noreen, 2020).

RFC Management

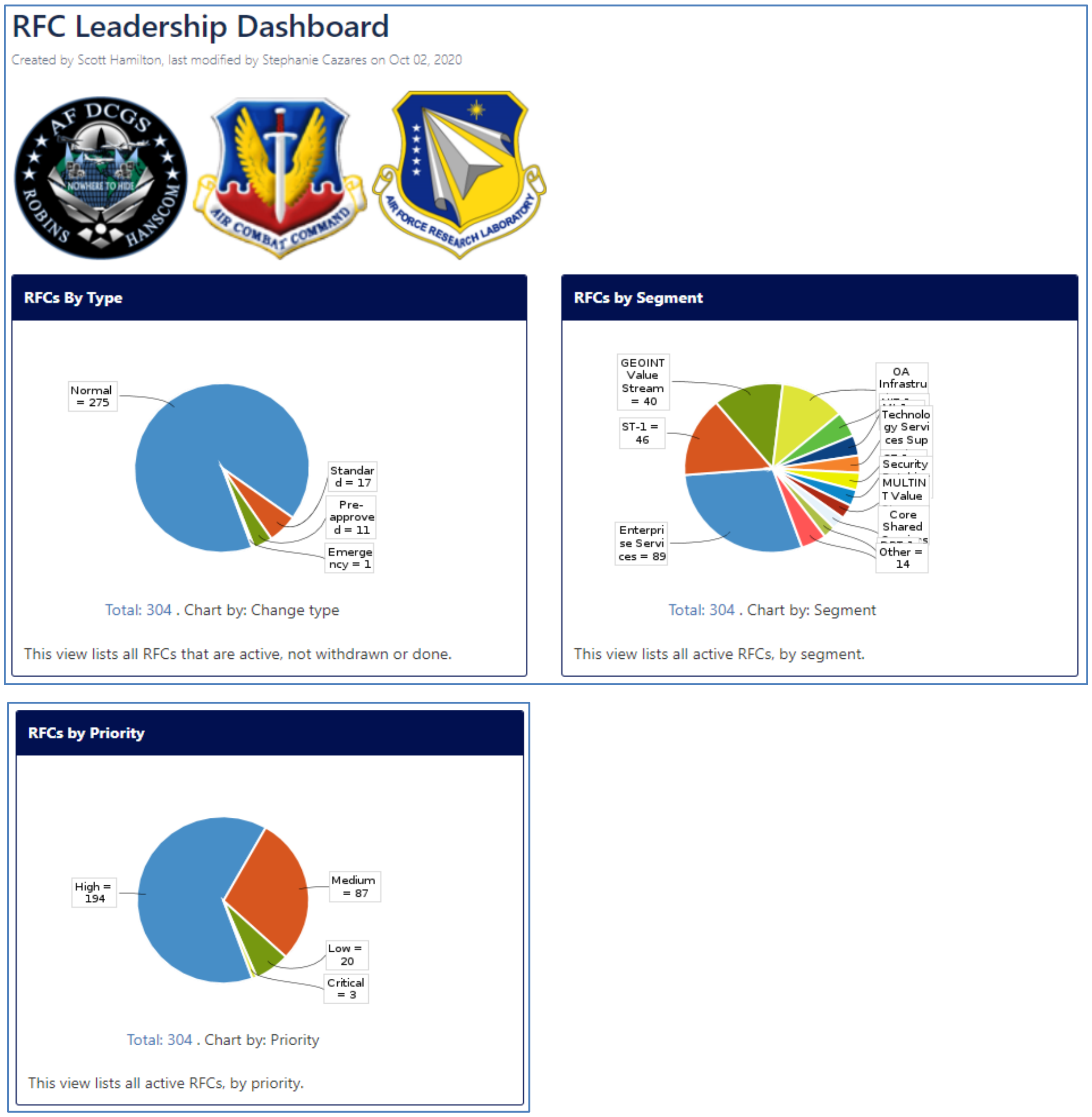
RFC Management Tool

The RFC management tool (RMT) was developed to meet the need to improve the existing RFC process through automation. Prior to the creation of the RMT each team had created their own tool to track RFCs for changes to the subsystems they have responsibility for sustaining. The implementations had so much variation that it caused confusion and inconsistencies and prevented a holistic and uniform approach to change management. The project data was not connected making it challenging to understand or visualize the status of RFCs flowing through the process at the enterprise level. With that in mind, the RMT capability was developed to provide visibility of all RFCs as they progress through the activities from initiation to deployment. The RMT development team identified three key features that would be essential to delivering this capability: 1) Jira and Confluence Entry Forms, 2) Jira Workflow, and 3) Leadership Dashboards. Using these collaboration tools ensures standardization, gives visibility across the teams, automates metric generation, and provides business intelligence for leadership (Cazares & Hamilton, RFC Transition Plan, 2020).

After five development sprints to build these three features the RMT was deployed with a deadline for all teams to transition by 21 Jan 2020. Business rules were established to ensure standardization across the multiple teams, Kanban boards built to facilitate visibility of work in progress across the IPTs, and RFC Management dashboards to provide insight into RFC status/schedule/metrics with views based on roles and teams (Cazares & Hamilton, RFC Transition Plan, 2020).

RFC Leadership Board

The RFC Leadership Board provides overview metrics and status for the AF DCGS leadership team to gain quick insight into the health and progress across the entire portfolio. Figure 20 uses pie charts to show the active RFCs to quickly identify the weighting of a particular type, segment or priority (Hamilton, RFC Leadership Dashboard, 2020). This view gives leadership visibility of the total number of open changes currently in work by the team broken out by the change type and highlights any emergency changes. It also identifies the workload balance between the segments and


RFC Leadership Dashboard

Created by Scott Hamilton, last modified by Stephanie Cazares on Oct 02, 2020

RFCs By Type

Total: 304 . Chart by: Change type

This view lists all RFCs that are active, not withdrawn or done.

RFCs by Segment

Total: 304 . Chart by: Segment

This view lists all active RFCs, by segment.

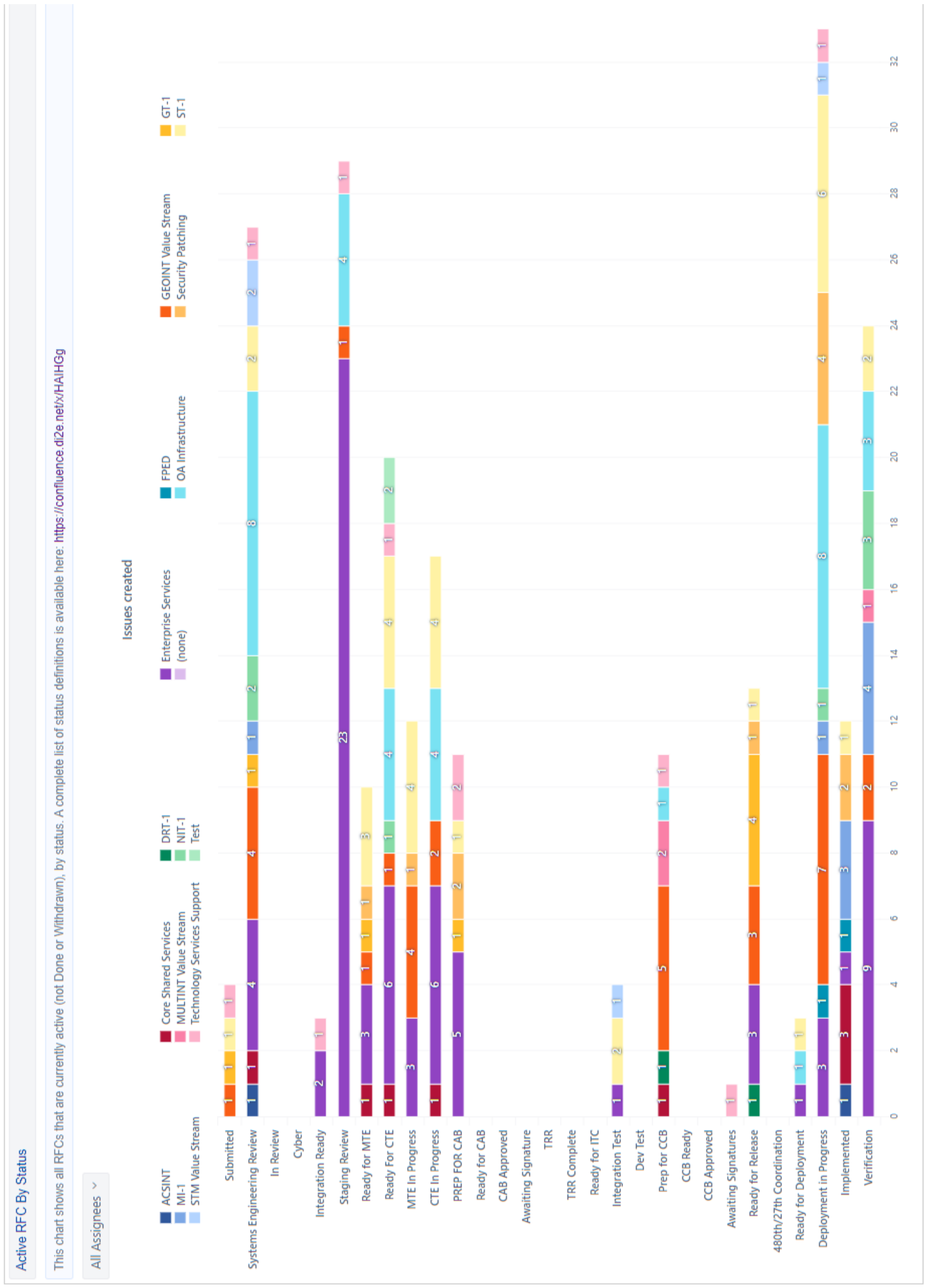
RFCs by Priority

Total: 304 . Chart by: Priority

This view lists all active RFCs, by priority.

Figure 20. RFC Dashboard

shows the distribution by priority. The next view shown in Figure 21 displays the


Active RFC By Status

This chart shows all RFCs that are currently active (not Done or Withdrawn), by status. A complete list of status definitions is available here: https://confluence.di2e.net/x/HAIHGg

Figure 21. Sample RFC Status View

number of RFCs from the perspective of the workflow steps (Hamilton, RFC Leadership Dashboard, 2020). This view provides insight into the steps that currently have the most RFCs indicating possible bottlenecks if the assigned work is greater than the capacity of the teams. For instance, in Figure 21 we observe the steps “deployment in progress”, “staging review” and “systems engineering review” have the most work in progress while “awaiting signatures” has one RFC and “cyber” has no work in progress. Investigation into the systems engineering, staging, and deployment activities would probably be a good idea because there is obviously a backlog of work for certain workflow steps. Figure 22 is a detailed view of the current RFCs from the perspective of the amount of

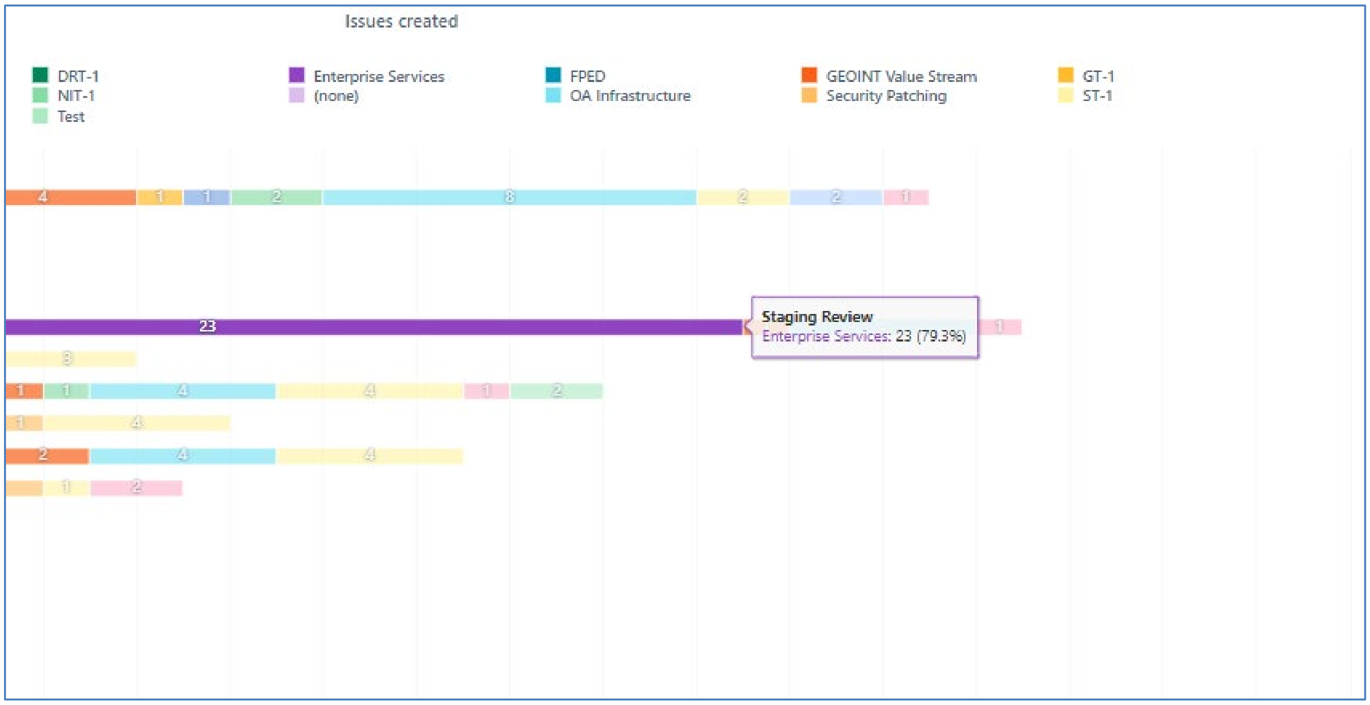
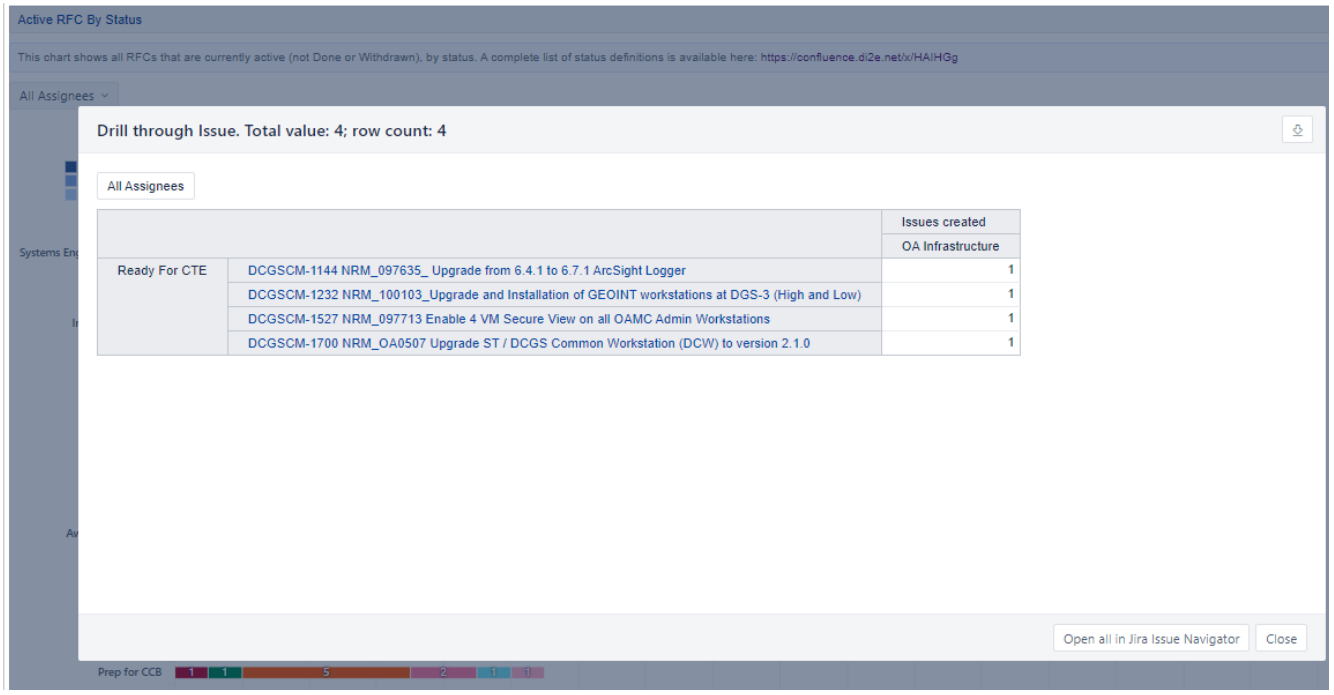


Figure 22. Sample RFC Detail View

work being performed on each of the weapon system components. For this example Enterprise Services has 23 RFCs currently in the “staging review” step and they account for 79% of all work in this step. Considering there are 16 other components, this may indicate possible issues (Hamilton, RFC Leadership Dashboard, 2020).   Finally, Figure 23 is from the perspective of viewing the details of each of the RFCs for a particular component and activity step. This example for OA Infrastructure lists all the active RFCs for that segment that is in the "ready for CTE" step (Hamilton, RFC Leadership Dashboard, 2020). Further details can be viewed on the Confluence page if any of the RFCs warrant more investigation.


Active RFC By Status

This chart shows all RFCs that are currently active (not Done or Withdrawn), by status. A complete list of status definitions is available here: https://confluence.di2e.net/x/HAIHGg

Drill through Issue. Total value: 4; row count: 4

		Issues created
		OA Infrastructure
Ready For CTE	DCGSCM-1144 NRM_097635_ Upgrade from 6.4.1 to 6.7.1 ArcSight Logger	1
	DCGSCM-1232 NRM_100103_Upgrade and Installation of GEOINT workstations at DGS-3 (High and Low)	1
	DCGSCM-1527 NRM_097713 Enable 4 VM Secure View on all OAMC Admin Workstations	1
	DCGSCM-1700 NRM_OA0507 Upgrade ST / DCGS Common Workstation (DCW) to version 2.1.0	1

Figure 23. Sample RFC Issue List

Value Streams

The AF DCGS program portfolio is organized based on Scaled Agile Framework (SAFe) principles for agile software development. The value streams are mapped to the areas of intelligence that DCGS processes, exploits, and disseminates. The four value streams shown in Figure 24 are GEOINT, ST (SIGINT), MULTIINT, and Infrastructure as a Service (IAAS) (Russell & Hamilton, 2020). Under each value stream Agile Release Trains (ARTs) are established based on requirements received from ACC. The Value Streams are a permanent part of the portfolio, while the ARTs have a defined beginning

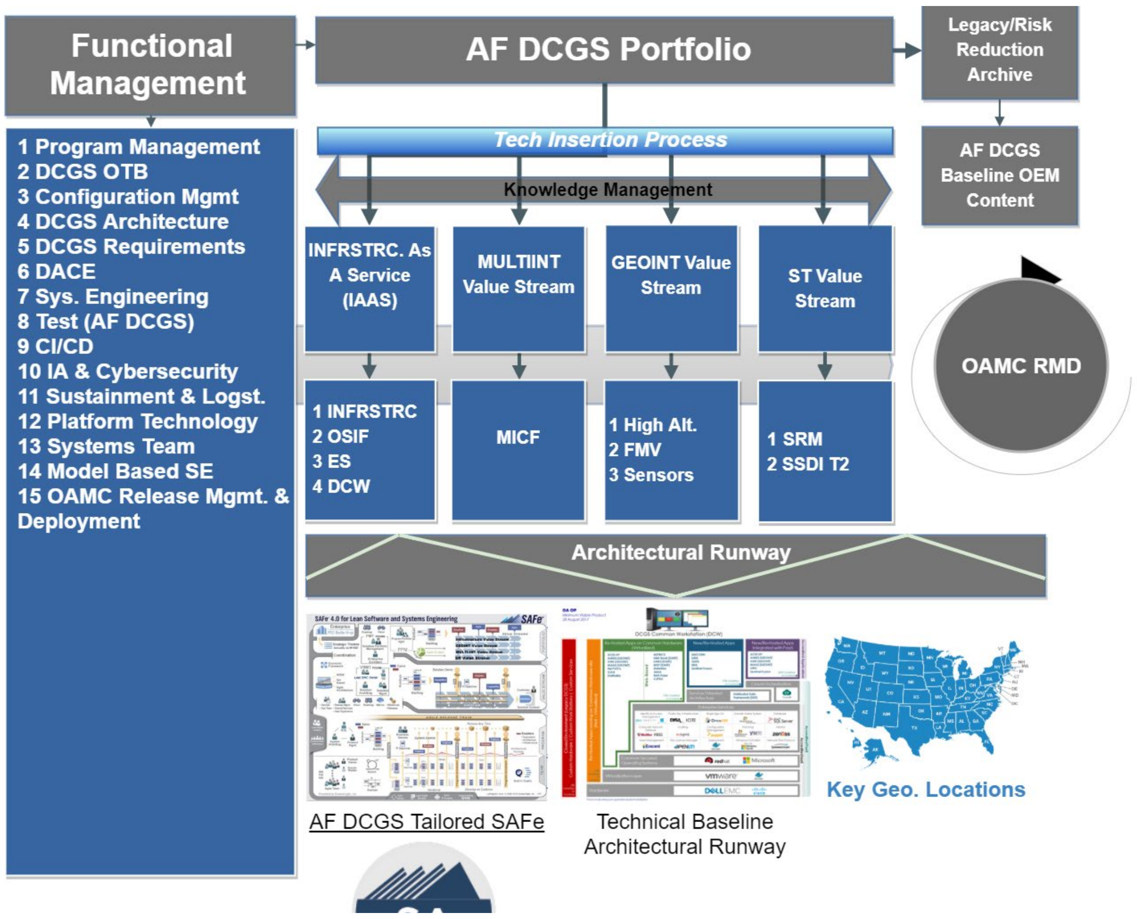
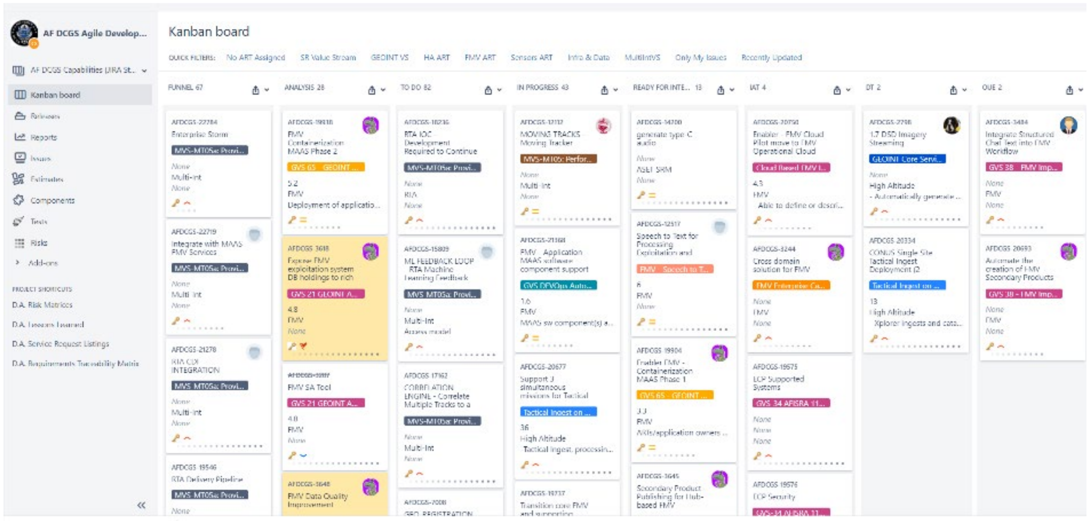


Figure 24. AF DCGS Program Portfolio

and end that may last from 6 months to 3+ years depending on the scope of the associated requirements. The DCGS Agile Execution Guide documents the program office implementation of a tailored SAFe model to fit the needs of the program (Billings, 2019).

RFC Kanban Board

A Kanban board is an agile project management tool used to visualize work, track the work-in-progress, and maximize the flow of work. Flow and bottlenecks are usually the main issues addressed in daily meetings and identify improvement opportunities that otherwise may not be identified (dos Santos, Beltrao, de Souza, & Travassos, 2018). The research performed by dos Santos validated that work visibility, control of project activities and tasks, workflow, communication and motivation were positive effects while internal quality, and team cohesion were negative effects (dos Santos, Beltrao, de Souza, & Travassos, 2018). Kanban boards exist for each of the value streams, agile release trains, and functional area to view assigned RFCs and the associated work in progress. A sample board for the entire portfolio is shown in Figure 25. The columns are organized


AF DCGS Agile Develop...
AF DCGS Capabilities JIRA St...
Kanban board
Releases
Reports
Issues
Estimates
Components
Tests
Risks
Add-ons
PROJECT SHORTCUTS
D.A. Risk Matrices
D.A. Lessons Learned
D.A. Service Request Listings
D.A. Requirements Traceability Matrix

Kanban board

QUICK FILTERS: No ART Assigned | SR Value Stream | GEOINT VS | HA ART | FMV ART | Sensors ART | Info & Data | MultiIntVS | Only My Issues | Recently Updated

FUNNEL 67
AFDCGS-22784 Enterprise Storm; MVS-MT05a: Provi...; None; Multi-Int; None
AFDCGS-22719 Integrate with MAAS FMV Services; MVS-MT05a: Provi...; None; Multi-Int; None
AFDCGS-21278 RIA CDI INTEGRATION; MVS-MT05a: Provi...; None; Multi-Int; None
AFDCGS-19546 RTA Delivery Pipeline; MVS-MT05a: Provi...; None

ANALYSIS 28
AFDCGS-19818 FMV Containerization MAAS Phase 2; GVS 65 - GEOINT ...; 5.2; FMV; Deployment of applicatio...
AFDCGS-3618 Expose FMV exploitation system DB holdings to rich; GVS 21 GEOINT A...; 4.8; FMV; None
AFDCGS-3817 FMV SA Tool; GVS 21 GEOINT A...; 4.0; FMV; None
AFDCGS-3648 FMV Data Quality Improvement

TO DO 82
AFDCGS-18236 RTA IOC Development Required to Continue; MVS-MT05a: Provi...; None; RIA; None
AFDCGS-15809 ML FEEDBACK LOOP RTA Machine Learning Feedback; MVS-MT05a: Provi...; None; Multi-Int; Access model
AFDCGS-17162 CORRELATION ENGINE - Correlate Multiple Tracks to a; MVS-MT05a: Provi...; None; Multi-Int; None
AFDCGS-7008 GEO REGISTRATION

IN PROGRESS 43
AFDCGS-12112 MOVING TRACKS Moving Tracker; MVS-MT05: Perfor...; None; Multi-Int; None
AFDCGS-21368 FMV - Application MAAS software component support; GVS DEVOps Auto...; 1.6; FMV; MAAS sw component(s) a...
AFDCGS-20677 Support 3 simultaneous missions for Tactical; Tactical Ingest on ...; 36; High Altitude; Tactical ingest, processin...
AFDCGS-19737 Transition core FMV and supporting

READY FOR INTE... 13
AFDCGS-14700 generate type C audio; None; ASET SRM; None
AFDCGS-12517 Speech to Text for Processing Exploitation and; FMV - Speech to T...; 6; FMV; None
AFDCGS-19904 Enabler FMV - Containerization MAAS Phase 1; GVS 65 - GEOINT ...; 3.3; FMV; ARIs/application owners ...
AFDCGS-3645 Secondary Product Publishing for Hub-based FMV

IAT 4
AFDCGS-20750 Enabler - FMV Cloud Pilot move to FMV Operational Cloud; Cloud Based FMV I...; 4.3; FMV; Able to define or descri...
AFDCGS-3244 Cross domain solution for FMV; FMV Enterprise Ca...; None; FMV; None
AFDCGS-19575 LCP Supported Systems; GVS 34 AFISRA 11...; None; None; None
AFDCGS-19576 LCP Security; GVS-34 AFISRA 11...

DT 2
AFDCGS-2798 1.7 DSD Imagery Streaming; GEOINT Core Servi...; None; High Altitude; - Automatically generate ...
AFDCGS-20334 CONUS Single Site Tactical Ingest Deployment (2; Tactical Ingest on ...; 13; High Altitude; Xplorer ingests and cata...

OUE 2
AFDCGS-3484 Integrate Structured Chat Text into FMV Workflow; GVS 38 - FMV Imp...; None; FMV; None
AFDCGS-20693 Automate the creation of FMV Secondary Products; GVS 38 - FMV Imp...; None; FMV; None

Figure 25. Agile Development Kanban Board

by workflow steps from left to right, with the far left typically reserved for “Backlog” and the far right for “Done”. Each task is displayed on a card with key information and as work is accomplished the card progresses from left to right until it is done. This board is a powerful tool for tracking work in progress with the flexibility to display any of the data in whatever format the user chooses.

DI2E DevTools

The DI2E Developer Tools provide an open development environment for the defense and intelligence community providing a suite of popular, widely used development tools. The tools align with the Office of Acquisition, Technology, and Logistics (A&TL) open systems architecture guidance supporting development, integration and test needs (DI2E, 2020). In addition to using DI2E for the Confluence and Jira tools, the program office uses DI2E for the development portion of the CI/CD pipeline. To ensure compliance a software delivery policy was issued to instruct all software for testing and/or deployment be delivered via the Nexus repository on DI2E (Williams & Anderson, Software Delivery Guide, 2020). The C/DM team then stores all software for formal testing (i.e. Integrated Test Cycle) and release in the Nexus repository at the appropriate classification level in the Controlled Test Environment (CTE) aggregated across three facilities at Robins AFB, Langley AFB, and Rome AFRL (Williams & Anderson, Software Delivery Guide, 2020).

CI/CD Delivery Process

As discussed in the RFC Delivery Process section, AF DCGS adopted an agile approach, based on commercial best practices and an adaptation of the Scaled Agile

Framework to a Government acquisition program. Developers work in agile development teams on Agile Release Trains (ARTs) and Value Streams (VSs), with short, time-boxed cadences for their development cycles. The delivery pipeline started to suffer from bottlenecks and unnecessary delays, as the ARTs began to deliver capabilities to integration and test environments in weeks or even days compared to the previous timeline of months and sometimes years. As an example, the formal testing cycle at the program level originally had a four week Integrated Test Cycle (ITC) resulting in a delay of one day to four weeks while the changes waited in the queue for the next test cycle. This delay in integration and test had a ripple effect of delaying feedback to the development teams. Three ITCs in a row failed to successfully demonstrate a completed capability ready for fielding (Sylvester C. , 2018). Due to these failures the ITC was extended to six weeks, resulting in even longer feedback loops to the developers, and more importantly, users now have a longer wait for upgraded products to be available.

The processes, teams, and tools for the RFC process are not optimized to provide capabilities to the OA DCGS operators on the desired, short time scales necessary of a modern IT-based system. The AF DCGS Chief Engineer directed the OA DCGS Continuous Integration Environment (CIE) team in March 2018 to establish a true Continuous Integration/Continuous Delivery (CI/CD) process within a Continuous Integration Environment (CIE). The goal was to streamline the delivery process and build a CI/CD pipeline that would "once and for all" address the convoluted existing RFC process (Sylvester C. , 2018).

Continuous Integration Environment

The basic definition of a Continuous Integration Environment (CIE) is an Information Technology environment where:

1) Developers upload each potential change to the baseline (feature, patch, new capability, enhanced service, etc.), as soon as the change is committed;
2) The uploaded change is a module (or “ingredient”: “wrapped” executable code, including installation script, configuration file, changes to documentation, etc.) deposited to a common repository under configuration control;
3) A master integration script (“recipe”) automatically integrates modules into a new candidate baseline on a frequent (i.e., “continuous”) basis (using, for example, the latest version of each module);
4) The master integration script then initiates automated testing of the new candidate baseline;
5) Metrics and test results are automatically compiled and posted/delivered to the responsible development community; and
6) The candidate baseline can be identically and automatically recreated from the repository at will for formal testing (DT/OT), experimentation, troubleshooting, or any other purpose (Sylvester, Tschuor, & Dent, 2018).

The problem is to create this environment with the following axioms as overarching goals:

1) Feedback must get back to the source as early as possible (issues from test, integration, and operations) in less than 6 hours from failure or test success to source.

2) The CIE must allow for the timeline from software commit to operational deployment to be less than 24 hours.

3) Automate, automate, automate with greater than 50% of each team's CIE actions automated (Sylvester, Tschuor, & Dent, 2018).

The environment includes a hardware infrastructure, network connections and protocols, and the software tools for creating a representative production environment, automated tools for installation and test, and monitoring and measuring products to monitor the health and status (Sylvester, Tschuor, & Dent, 2018). The AF DCGS CIE is a set of tools that reside on the DCGS controlled test environment that is part of the test and integration labs (Stocum, 2019). The software changes for each RFC use the tools

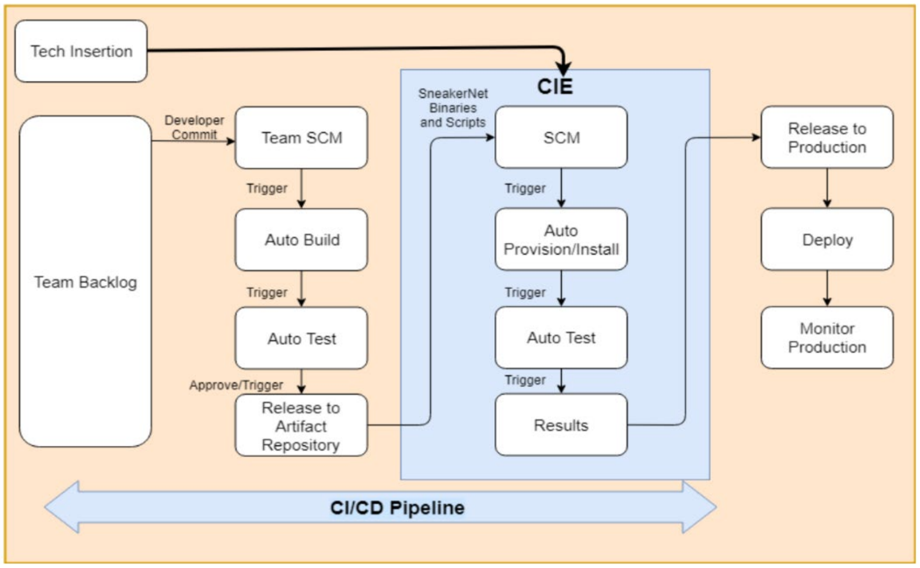


Figure 26. AF DCGS CIE High-level Workflow

and processes in the CIE to rapidly integrate, test, and prepare for deployment to operations as shown in Figure 26 (Stocum, 2019).

Continuous Integration/Continuous Delivery Pipeline

An element of the DevOps philosophy, continuous integration and delivery is focused on getting code into production as quickly as possible, in contrast with earlier approaches to developing software, including waterfall, which produce larger chunks of code over longer periods, making testing more time consuming and less reliable according to Vladysalv Gram (Clark, 2019). CI/CD is also known as a pipeline because once a new software or configuration enters the pipeline it either completes successfully or terminates in failure. There is no modifying of the configuration or software once it has entered the pipeline (Sylvester C. , 2018). Continuous integration is a software engineering practice that strives to integrate code at least daily and at best hourly (Steven, 2018). Continuous delivery is a software strategy that enables organizations to deliver new features to users as fast and efficiently as possible (Phillips, 2014). Phillips states the core idea of CD is to create a repeatable, reliable and incrementally improving process for taking software from concept to customer.

The three concepts of DevOps, CI/CD and Agile work together to create a CI/CD pipeline as shown in Figure 27 (Steven, 2018). Bridgwater agrees with Steven and states, “CI/CD is a build, test, and release automated process that complements an agile development process” (Bridgwater, 2019). Agile focuses on the processes, CI/CD focuses on software-defined lifecycles, and DevOps focuses on culture. The program office has integrated all three of these concepts into their foundational acquisition practices creating a transformational culture.

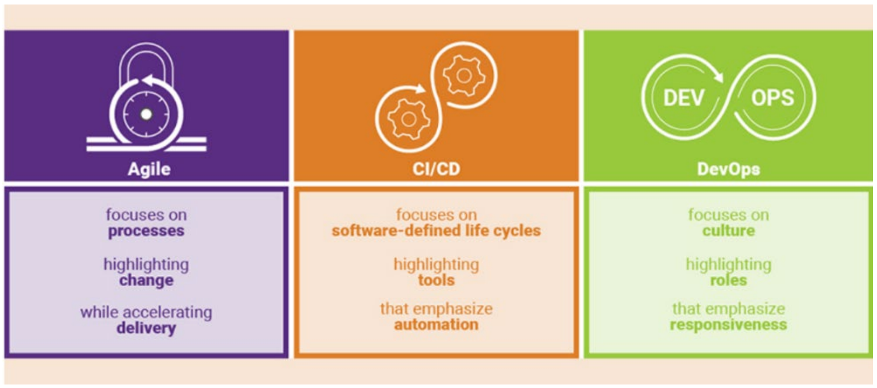

Agile
CI/CD
DEV
OPS
DevOps
focuses on processes
highlighting change
while accelerating delivery
focuses on software-defined life cycles
highlighting tools
that emphasize automation
focuses on culture
highlighting roles
that emphasize responsiveness

Figure 27. Building the CI/CD pipeline

The program office built a CIE to facilitate DCGS applications being fully tested and fully integrated as they rapidly move from development into production (Wellspring, 2020). Wellspring stated "The key values were to have a 1) government owned, controlled and managed integration environment, 2) built using the open architecture hardware, 3) code checked in and integrated several times a day, 4) repeatable code base across any OA stack, 5) fully automated test and integration, and 6) quick feedback to developers on test success/failures" (Wellspring, 2020). The CI/CD pipeline structure developed by Jez Humble and David Farley in Figure 28 was the basis for the design of the AF DCGS CI/CD pipeline (Babbitt, 2019). Humble articulates the different potential stages in the CI/CD pipeline in five simplistic statements: 1) The CIE is triggered by a new drop of software into version control, 2) The software is built, configured and tested, 3) After successful testing, the software may be released, which is part of the Release and

Deployment process, 4) The CI/CD process could fail at any step in the delivery, providing feedback to the team, and 5) This process is iterated until success is achieved (Sylvester, Tschuor, & Dent, 2018).

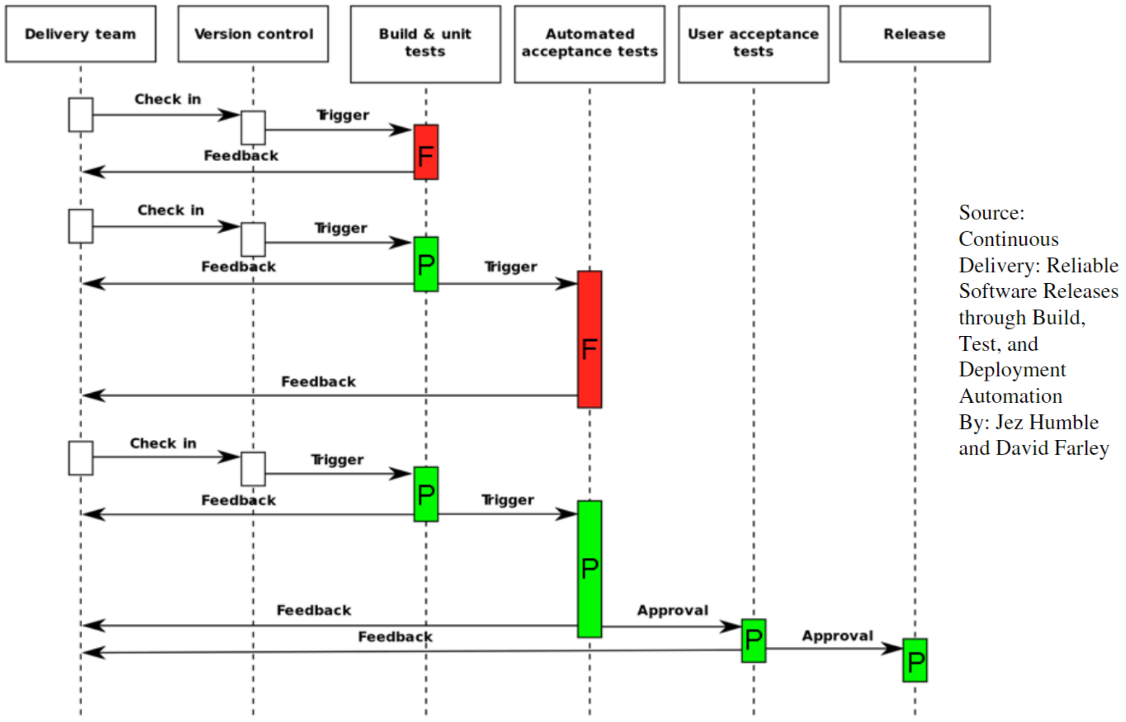


Figure 28. Humble and Farley CI/CD Pipeline Structure

The AF DCGS CI/CD pipeline consists of a development portion hosted on DI2E and an integration and test portion hosted on the Controlled Test Environment (CTE) as shown in Figure 29 and Figure 30 (Wellspring, 2020). The development portion provides the environment for code to be pulled from the backlog, run a static code analysis, perform unit test, execute the build, run functional tests, and release for integration. The integration and test portion provides the environment for a build infrastructure, install/configure of the application, integration, end-to-end testing, generation of final reports and release for CCB/Fielding (Wellspring, 2020).

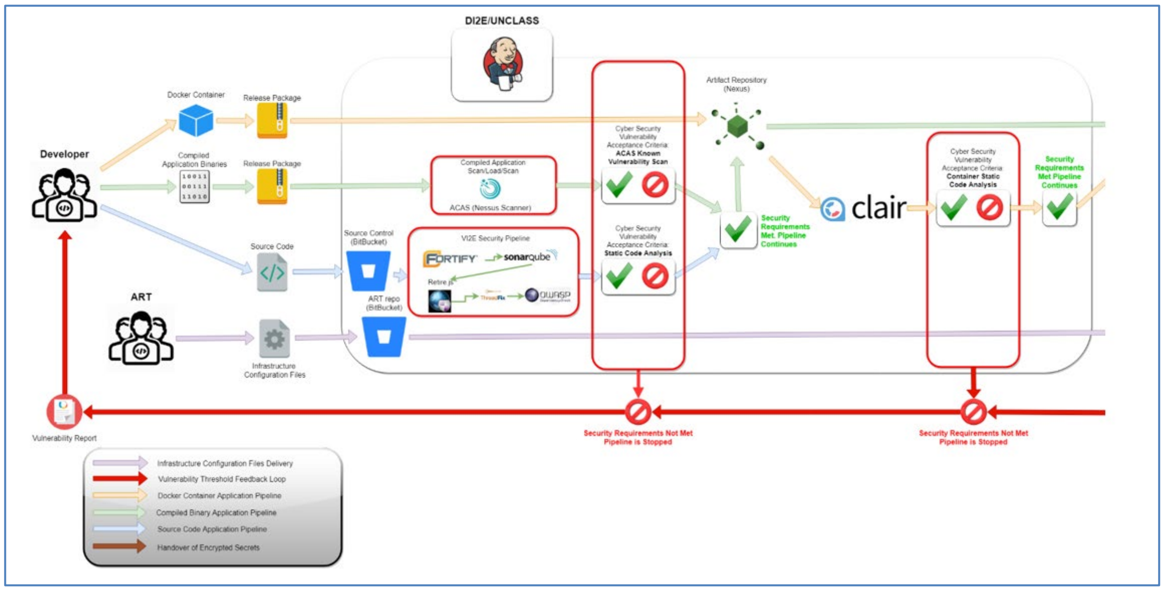

DI2E/UNCLASS
Developer
Docker Container
Release Package
Compiled Application Binaries
Source Code
ART
Infrastructure Configuration Files
Source Control (BitBucket)
ART repo (BitBucket)
Compiled Application Scan/Load/Scan
ACAS (Nessus Scanner)
VI2E Security Pipeline
Cyber Security Vulnerability Acceptance Criteria: ACAS Known Vulnerability Scan
Cyber Security Vulnerability Acceptance Criteria: Static Code Analysis
Artifact Repository (Nexus)
Security Requirements Met. Pipeline Continues
clair
Cyber Security Vulnerability Acceptance Criteria: Container Static Code Analysis
Security Requirements Met Pipeline Continues
Security Requirements Not Met Pipeline is Stopped
Vulnerability Report
Infrastructure Configuration Files Delivery
Vulnerability Threshold Feedback Loop
Docker Container Application Pipeline
Compiled Binary Application Pipeline
Source Code Application Pipeline
Handover of Encrypted Secrets

Figure 29. CI/CD pipeline – Development

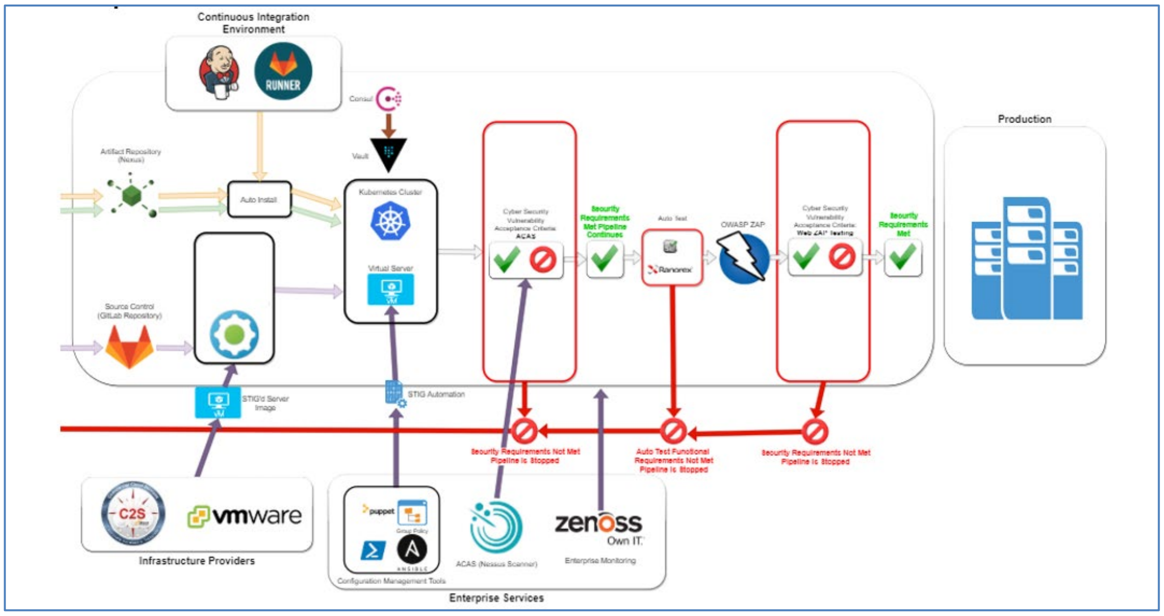

Continuous Integration Environment
RUNNER
Consul
Vault
Artifact Repository (Nexus)
Auto Install
Kubernetes Cluster
Virtual Server
Source Control (GitLab Repository)
Cyber Security Vulnerability Acceptance Criteria: ACAS
Security Requirements Met Pipeline Continues
Auto Test
Ranorex
OWASP ZAP
Cyber Security Vulnerability Acceptance Criteria: Web ZAP testing
Security Requirements Met
Production
STIG'd Server Image
STIG Automation
Security Requirements Not Met Pipeline is Stopped
Auto Test Functional Requirements Not Met Pipeline is Stopped
C2S
vmware
Infrastructure Providers
puppet
Group Policy
Configuration Management Tools
ACAS (Nessus Scanner)
zenoss Own IT.
Enterprise Monitoring
Enterprise Services

Figure 30. CI/CD pipeline – Integration and Test

The DI2E development environment provides automated tools such as Bitbucket for source code control, Fortify and SonarCube for static code analysis, ACAS to perform scan/load/scan of compiled applications, Clair to perform docker container vulnerability analysis, and Nexus for the artifact repository (Castellon & Hurst, 2019). The CTE integration and test environment provides the Jenkins application that is a continuous integration and build server, essentially the orchestrator of the CD/DevOps ecosystem. A Jenkins job pulls source from the configuration library, builds infrastructure as code, configures the application, quality checks the code, compiles the source code, executes unit tests, creates the release package, and posts to the artifact repository. To accomplish this Jenkins interfaces with other applications such as IBM Rational, Nexus, puppet, sonar, Serena, and uDeploy to perform each step of integration and test. These tools are the heart of the CI/CD pipeline and provide the ability to streamline the delivery process. The research hypothesis in the following section outlines the studies that will be conducted to examine how well the CI/CD pipeline streamlined the delivery process.

Research Hypothesis

This research will examine the effectiveness of the CI/CD pipeline by conducting an observational study on the project data for the CI/CD process and the RFC process as described earlier. The two specific types of studies that will be performed are a between-subjects study and a within-subjects study. A between-subjects study will be conducted to examine the deployment timelines between using the CI/CD pipeline and the RFC delivery process. A within-subjects study will be conducted on the CI/CD pipeline to characterize the process based on selected attributes. The hypotheses that will be tested are summarized in Table 7.

Table 7. Research Hypotheses

Between-Subjects Study: CI/CD, RFC			
Data Type	Attribute	Null Hypothesis	Alternate Hypothesis
Feature	Workdays	H_0 = There is no difference in the delivery time for features between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for features
Story	Workdays	H_0 = There is no difference in the delivery time for stories between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for stories
Task	Workdays	H_0 = There is no difference in the delivery time for tasks between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for tasks
Bug	Workdays	H_0 = There is no difference in the delivery time for bugs between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for bugs
Spike	Workdays	H_0 = There is no difference in the delivery time for spikes between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for spikes
Within-Subjects Study: Value Stream, Priority, Story Points			
Feature	Value Stream	H_0 = There is no difference in the delivery time for features between value streams	H_a = The features have different delivery times based on value streams
	Priority	H_0 = There is no difference in the delivery time for features based on priority	H_a = The features have decreasing delivery times as the priority increases
Story	Value Stream	H_0 = There is no difference in the delivery time for stories between value streams	H_a = The stories have different delivery times based on value streams
	Priority	H_0 = There is no difference in the delivery time for stories based on priority	H_a = The stories have decreasing delivery times as the priority increases
	Story Points	H_0 = There is no difference in the delivery time for stories based on story points	H_a = The stories have increasing delivery times as the story points increase
Task	Value Stream	H_0 = There is no difference in the delivery time for tasks between value streams	H_a = The tasks have different delivery times based on value streams

	Priority	H_0 = There is no difference in the delivery time for tasks based on priority	H_a = The tasks have decreasing delivery times as the priority increases
	Story Points	H_0 = There is no difference in the delivery time for tasks based on story points	H_a = The tasks have increasing delivery times as the story points increase
Bug	Value Stream	H_0 = There is no difference in the delivery time for bugs between value streams	H_a = The bugs have different delivery times based on value streams
	Priority	H_0 = There is no difference in the delivery time for bugs based on priority	H_a = The bugs have decreasing delivery times as the priority increases
Spike	Value Stream	H_0 = There is no difference in the delivery time for spikes between value streams	H_a = The spikes have different delivery times based on value streams
	Priority	H_0 = There is no difference in the delivery time for spikes based on priority	H_a = The spikes have decreasing delivery times as the priority increases

Summary

While the program office has developed kanban boards to give situational awareness into progress of the teams, none of the available boards or views provide data to determine if the CI/CD pipeline is actually improving the overall timeline of the RFC process. There are no details on timelines by priority, value stream, story points or issue type for either the RFC process or the CI/CD process. The pie charts simply display the current number of RFCs by segment (value steam), priority, and RFC type (not the same attribute as issue type). The bar charts also display the number of RFCs in each step of the workflow but have no associated delivery timeline. This may be interesting data but does not give an indication of the speed through the workflow. Given that the program office moved to a CI/CD process for the weapon system for the sole purpose of delivering capability to the field faster, it is evident the current RFC management boards do not have the automated reporting to continuously measure and report the throughput.

The program office has not analyzed the data to determine how much, if any, the CIE and CI/CD pipeline improved the delivery timeline. This research will perform the analysis and provide the results to the AF DCGS program office, with recommendations for further action and research. The details of the methodology for the study are discussed in detail in Chapter III.

III. Methodology

Chapter Overview

This chapter explains the methods used to examine the performance measures of the AF DCGS CI/CD delivery process and the RFC delivery process. This chapter begins with a description of the study, data sets and sources, and data types. Next, data standardization and data cleaning is covered, followed by measures and assumptions. This chapter concludes with statistical methods and a summary.

Observational Study

This research was an observational study due to the constraints that prevented conducting an experimental study. An observational study is one that records information concerning the subject under study without any interference with the process that is generating the data (Ott & Longnecker, 2018). The information analyzed was selected from data already generated and stored in the Jira databases for software projects that have completed the integration and test phase. A between-subjects and within-subjects observational study were conducted to determine the effectiveness of the CI/CD pipeline with respect to speed of delivery.

The first study was a between-subjects study to examine the effectiveness of the AF DCGS integration and test process for the two current delivery methods, CI/CD process and RFC process, by comparing workflow days. The first delivery method is the established process of using the existing integration labs with zero automation, multiple testing events with numerous gates, and disconnected labs and environments. The second delivery method is the newer process of using automated tools, streamlined testing, and fully integrated labs using a continuous integration environment. The second study was a within-subjects study to characterize the CI/CD process for five different data types and three different attributes. This is an observational study with the data sampled from a population that the factors are already present and we are comparing samples with respect to the factors of interest.

Description of Data Set and Sources

Data Platform

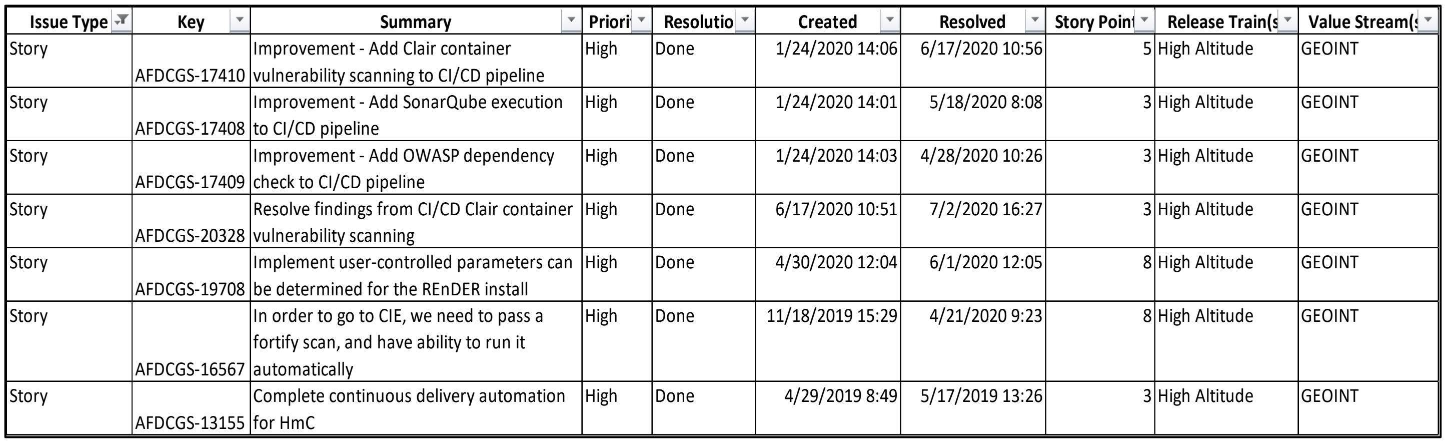
Data was collected from the DI2E DevTools AF DCGS collaboration area that has RFC and CI/CD process data for software integration, test and delivery. All personnel from the program office, OEMs, test organizations, and ACC use the Confluence work area to manage and track progress on all changes to the configuration baseline. Detailed activity and statuses are updated daily by the team through the use of Kanban boards and a standard Jira workflow.

Data Collection

Data was collected from the DI2E cloud based environment that can be accessed with a government laptop and Common Access Card (CAC). The data was not centrally located in one project and required access to the AFDCGS, AFDCGSCICD, CIEService, and DCGSCIE projects and boards. Collection was a very time consuming, manual effort that required many hours simply to figure out what data was available for analysis and where it was stored. Additionally, sorting the data between the CI/CD process and the RFC process was very challenging because the data is stored by project, not by which process it used. Discussions with the CIE team revealed that no progress had been made on setting up automated metrics reporting. Data was very limited in some cases and the analysis constrained by the available data.

The data fields that were selected for the analysis included Issue Type, Key, Summary, Priority, Status, Resolution, Created, Resolved, Story Point, Release Train and Value Stream. The data was downloaded to Excel from each of the projects and the eazyBI tool was also used to collect detailed data on transitions and associated transition times for each issue from Jun 2018 to Sep 2020. Sample cleaned data from the CI/CD process for agile stories is shown in Table 8. The data was extracted from the AFDCGS

Table 8. Sample data for CI/CD Story



Issue Type	Key	Summary	Priori	Resolutio	Created	Resolved	Story Poin	Release Train(s	Value Stream(s
Story	AFDCGS-17410	Improvement - Add Clair container vulnerability scanning to CI/CD pipeline	High	Done	1/24/2020 14:06	6/17/2020 10:56	5	High Altitude	GEOINT
Story	AFDCGS-17408	Improvement - Add SonarQube execution to CI/CD pipeline	High	Done	1/24/2020 14:01	5/18/2020 8:08	3	High Altitude	GEOINT
Story	AFDCGS-17409	Improvement - Add OWASP dependency check to CI/CD pipeline	High	Done	1/24/2020 14:03	4/28/2020 10:26	3	High Altitude	GEOINT
Story	AFDCGS-20328	Resolve findings from CI/CD Clair container vulnerability scanning	High	Done	6/17/2020 10:51	7/2/2020 16:27	3	High Altitude	GEOINT
Story	AFDCGS-19708	Implement user-controlled parameters can be determined for the REnDER install	High	Done	4/30/2020 12:04	6/1/2020 12:05	8	High Altitude	GEOINT
Story	AFDCGS-16567	In order to go to CIE, we need to pass a fortify scan, and have ability to run it automatically	High	Done	11/18/2019 15:29	4/21/2020 9:23	8	High Altitude	GEOINT
Story	AFDCGS-13155	Complete continuous delivery automation for HmC	High	Done	4/29/2019 8:49	5/17/2019 13:26	3	High Altitude	GEOINT

Kanban board for stories filtered for the CI/CD pipeline. Next, Table 9 is an example of data extracted from the DCGSCIE Kanban board for bugs that were executed through the CI/CD pipeline. This board included all the changes to the OA DCGS infrastructure, enterprise services, hardware stacks, workstations, continuous integration environment updates and security patches. The final example is Table 10 extracted from the AFDCGS

Table 9. Sample data for CI/CD Bug

Issue Type	Key	Summary	Priority	Resolution	Created	Release Train(s)	Value Stream(s)
New Feature	AFDCGS-10585	Resolve or turn off MEPI capability	Medium	Done	11/19/2018 14:46	CETS SRM	SR
New Feature	AFDCGS-10595	Remove CETS fuse or choose geolocation logic	Medium	Done	11/19/2018 15:05	CETS SRM	SR
New Feature	AFDCGS-10681	Automate FR 3.2 POD	Medium	Done	11/27/2018 14:09	CETS SRM	SR
New Feature	AFDCGS-10773	CDA NETWORK ANALYSIS CURRENT - Integrate with Unicorn data source	High	Done	11/30/2018 7:40	Multi-Int	MULTIINT
New Feature	AFDCGS-10791	CyAN Feature SolarWinds Log Event Management Syslog data to Cyber Dashboard tool	High	Done	12/3/2018 8:19	CyAN	Infrastructure
New Feature	AFDCGS-10792	CyAN Feature SolarWinds NPM Network Data to Cyber Dashboard tool	High	Done	12/3/2018 8:28	CyAN	Infrastructure
New Feature	AFDCGS-10798	CyAN Feature Display SolarWinds Security Information Event Management widgets	High	Done	12/3/2018 10:06		Infrastructure
New Feature	AFDCGS-10949	CyAN Feature Visualize Data Geospatially	High	Done	12/10/2018 14:46	CyAN	Infrastructure
New Feature	AFDCGS-11159	(BCR) Map updates when country code is changed	Medium	Done	1/2/2019 7:04	High Altitude	GEOINT
New Feature	AFDCGS-11163	(BCR) FIVE EYES-Change Name	Medium	Done	1/2/2019 7:16	High Altitude	GEOINT
New Feature	AFDCGS-11164	(BCR) REnDER Multi-Select Capability for approved/prohibited lists	Medium	Done	1/2/2019 7:17	High Altitude	GEOINT
New Feature	AFDCGS-11168	(BCR) Need method for adding tetragraphs	Medium	Done	1/2/2019 7:24	High Altitude	GEOINT
New Feature	AFDCGS-11817	Unicorn plugin to configure (Internal GDMS - 640)	High	Done	2/13/2019 11:26	FMV	GEOINT
New Feature	AFDCGS-11992	Artificial Intelligence (AI) Harness to send 1st AI Model metadata to a visualization tool	High	Done	2/21/2019 10:47	Multi-Int	MULTIINT
New Feature	AFDCGS-12175	Audio Playback of conversations	Medium	Done	3/7/2019 11:28	ASET SRM	SR

Table 10. Sample data for Feature not using CI/CD

Issue Type	Key	Summary	Priority	Status	Resolution	Created	Resolved
IT Help	AFDCGSCIE-26	Disk Utilization scans indicate CE23 Var/log is at 100% and CE 07 CPU is at 100%	Medium	Closed	Closed	11/8/2019 14:35	11/21/2019 10:23
IT Help	AFDCGSCIE-51	Jenkins fortify scan failing due to permission error	Blocker	Resolved	Known Error	3/4/2020 10:53	3/4/2020 13:57
IT Help	AFDCGSCIE-69	Windows Vagrant box images have the AWS transit proxy hardcoded. This needs to be removed in future builds. This bug denies access to the internet on the image. Discovered by Jason Weitzel (ESS)	Low	Resolved	Fixed	4/28/2020 14:48	5/18/2020 8:13
Service Request	AFDCGSCIE-36	ce23 on CTE-L is nearing capacity -- /var/log	Medium	Resolved	Fixed	1/17/2020 12:22	1/22/2020 14:09
Service Request	AFDCGSCIE-37	ce07 is throwing errors due to high CPU	Medium	Resolved	Fixed	1/17/2020 12:22	1/22/2020 13:59
Service Request	AFDCGSCIE-40	I am not able to delete artifacts from the Nexus Repository	Medium	Resolved	Fixed	1/27/2020 15:57	2/5/2020 7:52
Service Request	AFDCGSCIE-46	Watchman 4 Defense requests the updates made on ce25 get pushed into GitLab on CTE-H	Medium	Resolved	Fixed	2/27/2020 9:40	2/27/2020 10:07

Agile Kanban board for features that went through the RFC process. This data was the easiest to sort and parse out the relevant data entries.

Data Types

AF DCGS 1067 Decomposition

ACC submits new operational requirements to the DCGS program office using the AF Form 1067 as part of the requirements management process. The process flow from a 1067 to features, stories and tasks is shown in Figure 31 (Priddy, AF DCGS Agile Execution Guide, 2018). Mission experts from the 480th Wing and ACC/A5/2D are the

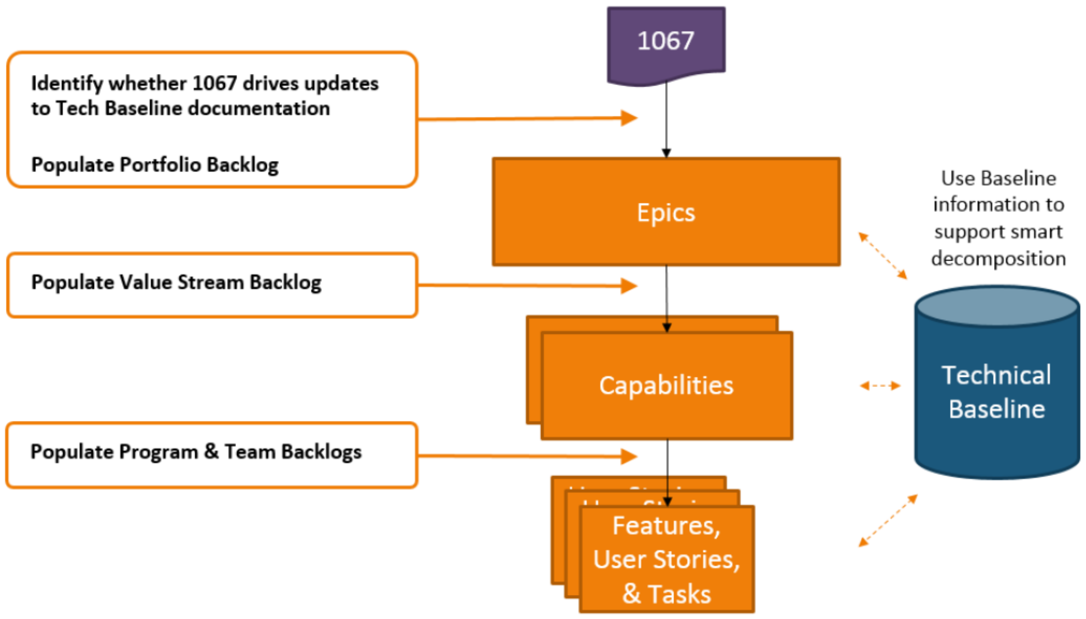

1067
Identify whether 1067 drives updates to Tech Baseline documentation
Populate Portfolio Backlog
Epics
Use Baseline information to support smart decomposition
Populate Value Stream Backlog
Capabilities
Technical Baseline
Populate Program & Team Backlogs
Features, User Stories, & Tasks

Figure 31. 1067 Decomposition

epic owners and shepherd the requirement from the 1067 through the value stream planning phase. The team analyzes each 1067 requirement and the candidate solution approach documented in an epic value statement. The epics are assigned to one of four value streams (Infrastructure, GEOINT, MULTIINT, or SIGINT) organized around the intelligence categories mentioned in Table 1.

Value Streams

The infrastructure value stream entails the services, infrastructure, enterprise services, workstations and architecture to create and field DCGS systems at on-site locations. The MULTIINT value stream consists of capabilities that support data fusion from multiple intelligence domains and the tools that support analysis and reporting. The GEOINT value stream consists on capabilities that support full motion video, imagery, Global Hawk Block 30/40 exploitation, and other high-altitude platforms. The SIGINT value stream supports various capabilities (Priddy, AF DCGS Agile Execution Guide, 2018).

Epics

The epics are decomposed into capabilities and features by the value stream teams and the development team writes the user stories and tasks that are required to create the features and capabilities. Figure 32. Epic Decomposition shows the linkage between these different issue types (Oligmueller & Smith, 2020). The epics are too broad and complex to perform meaningful research on the CI/CD pipeline effectiveness. An epic could take up to two or three years to be fully implemented and represent thousands of hours of work. Under agile principles delivery is not accomplished at the epic level, instead it is at the smallest level possible to push updates and changes out to the user on the most frequent tempo possible. Therefore, to gain an understanding on the CI/CD pipeline performance, this research focuses on the features, improvements, stories, tasks, deficiencies, bugs and spikes that are the smallest increments of work packages deployed to the production environment.

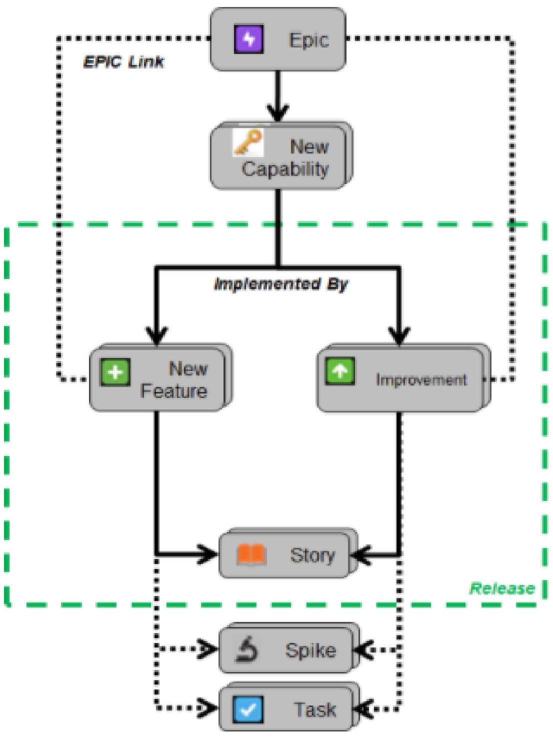


Figure 32. Epic Decomposition

Feature

AF DCGS defines features as a service that fulfills a stakeholder need and is sized to be delivered by a single agile release train in a single program increment. An improvement is a fundamental component of software development that enhances existing functionality for the operator, with mission value (Oligmueller & Smith, 2020).

Story

A story is a software development effort that represents the code changes necessary to meeting the minimum viable product of the linked feature or improvement (Oligmueller & Smith, 2020).

Task

A task is a non-software development related effort such as documentation, certificate to field, drawings, tech orders, and other required activity to complete the delivery package (Oligmueller & Smith, 2020).

Bug

A bug is a problem with code identified outside of formal integration test and does not have to be linked to a feature. A bug is part of a release as a fix and the assigned priority is important (Oligmueller & Smith, 2020).

Spike

A spike is used for research and is the first step in a new design of a feature. Stories, tasks and spikes must always be linked to a feature or improvement (Oligmueller & Smith, 2020).

Data Standardization

Analysis of the CI/CD pipeline data set identified several categories with very few data points probably due to the fact it is a relatively new environment and not all the work is flowing through the pipeline yet. The RFC dataset had a satisfactory number of data points for all categories due to the volume of work flowing through the RFC process. Work products were grouped to have enough data points to adequately study all five data types. Standard agile definitions were used to ensure similar work products were grouped for both the CI/CD process and the RFC process as shown in Table 11.

Table 11. Data Type Grouping

Data Type	Jira Symbol	Includes Issue Types...
Feature	➕	New Feature
	➕	Improvement
Story	📖	Story
	📖	User Story (no longer used)
Task	☑	Task
Bug	🔴	Bug
	🐞	DR
	🐞	TPR
Spike	🔬	Spike
	⚙	Non-Functional Requirement/Technical Debt

Improvement and feature were grouped together as feature, Deficiency Report (DR) and Test Performance Report (TPR) were added to bug, non-functional requirement was added to spike, user story was added to story, and task was not combined with any other data type. Improvements and features are both implementations for a new capability and there was no evidence of differences between them or standards for selecting one over the other. They appeared to be interchangeable terms and dependent on the person creating the issue type. DR and TPR are bug types discovered during testing or fielding activities and there was no observable differences between these data types and the bug data type. The non-functional requirement identifies work that the team performs that is necessary for the design of a capability but is not a direct requirement from the customer. For example, the MULTIINT value steam had a non-functional requirement for data storage that included defining the storage capacity necessary for the data lifecycle as well as the amount of time for data retrieval. Technical debt is work that is identified by the team during implementation of a capability that does not meet internal requirements for items such as scalability and reliability. A spike was work that also investigated and identified items such as the bare minimum to have running for the SOA ESB testing to test horizontal scalability. All three of these are similar in that they are derived requirements for a capability.

Studying the structure within Jira showed that none of these five data types were subordinate to each other. The only mapping of these data types was back to the epic or new capability they supported. Therefore, we concluded that the time spent on each of these data types was self-contained and we would not have any issues with our analysis by the possibility of cross contamination with regards to work days between data types.

Standards were not the same across the projects for priority. Some projects used blocker or critical where a blocker prevented completion of other work and a critical was an issue that needed immediate correction. No differences between these two statuses was evident. Since we limited our issues to ones that were completely done, the statuses of blocker or critical were not relevant so they were replaced with a priority of high. Low and medium priorities required no changes.

Data Cleaning

The data for the existing process was from 2016 through Sep 2020 while the CIE was from 2018 through Sep 2020. The data prior to 2018 was removed so that the analysis would be based on the same time period. Since we only examined issues that had completed the CI/CD pipeline, only entries that had a resolution status of "Done" were kept. The Jira issue types and statuses that documented the current state for a particular software change were not consistent between the two processes. For instance, in Table 12, Key ID: AFDCGSCIE-51 had a resolution of known error and status was resolved. The work history stored in Confluence for this bug was studied and we observed it was worked and resolved within 3 hours and delivered to the CIE production environment. Therefore, it was kept as a valid data point.

Table 12. Data Cleaning Sample 1

Issue Type	Key	Summary	Priority	Status	Resolution	Created	Resolved	Workdays
IT Help	AFDCGSCIE-51	Jenkins fortify scan failing due to permission error	Blocker	Resolved	Known Error	3/4/2020 10:53	3/4/2020 13:57	0.128

Another example was Key ID: AFDCGSCIE-94 shown in Table 13 that had a resolution of fixed and status of done. The work history for this bug revealed it was completed and delivered to the CIE production environment. Therefore, it was kept as a valid data point.

Table 13. Data Cleaning Sample 2

Issue Type	Key	Summary	Priority	Status	Resolution	Created	Resolved	Workdays
Task	AFDCGSCIE-94	Put Fortify on render VM	Low	Done	Fixed	7/8/2020 11:45	9/11/2020 8:19	64.857

Another data cleaning example that was more difficult was how to handle missing data. Most of theses cases required subject matter expert knowledge of the software projects and AF DCGS to provide the missing values. Table 14 is an example of missing value streams that knowledge of the ARTs and system segments was required to determine the value stream. SSDI T2 and CICS SRM are both SIGINT release trains while Full Motion Video (FMV) is a GEOINT value stream.

Table 14. Data Cleaning Sample 3

Issue Type	Key	Summary	Priority	Status	Resolution	Created	Resolved	Release Train(s)	Value Stream(s)
Spike	AFDCGS-18591	Prepare for ONEROOF Enterprise Usage	High	Done	Done	4/1/2020 13:03	7/8/2020 10:06	SSDI T2	
Spike	AFDCGS-20293	AGS Spike Report: Audio Compatibility with ONEROOF	High	Ready for ITC	Done	6/12/2020 10:01	9/15/2020 13:27	CICS SRM	
Spike	AFDCGS-21644	Research how SOA ESB fits into the cloud	High	Done	Done	9/4/2020 8:31	9/25/2020 7:43	FMV	
Spike	AFDCGS-18775	Processing other Hdet radio types, Serena: 100092133	High	Done	Done	4/6/2020 13:47	7/20/2020 13:19	CICS SRM	
Spike	AFDCGS-19569	Limited Number of LOBs sent to CEGS, Serena: 100097095	High	Ready to close	Done	4/27/2020 18:46	6/22/2020 10:41	CICS SRM	

The final example shown in Table 15 for Key ID: AFDCGSCIE-3 shows a data point that did not actually go through the CIE. Further investigation on the Confluence page revealed this was a trial test case early in the initial CIE standup that was not run through the CI/CD process. This type of data point was not kept as a valid data point.

Table 15. Data Cleaning Sample 3

Jira

DI2E Framework Jira

Issue Type	Key	Summary	Priority	Status	Resolution	Created	Resolved
Task	AFDCGSCIE-3	CIE = awesome	Medium	Done	Closed	8/16/2019 13:19	8/16/2019 14:18

Measures

Measures were collected for all of the issue types to gain an understanding of the projects and the process, and to identify what was available for the speed of delivery analysis. The initial data set included feature, story, spike, bug, task, backlog, tech debt, improvement, new capability, enabler, DR, TPR, user story, and non-functional requirement for the issue types. The attributes collected were key, title, resolution, issue initiation date, issue resolved date, issue status, issue type, priority, value stream story points, agile release train, transition to status, transition from status, days in transition status. After studying the data, the issue types selected for analysis were feature, story, spike, task, and bug. The attributes selected for analysis were priority, story points, value stream, and calculated workdays.

Assumptions

The data collected was from actual work that was conducted in the AF DCGS program office over the past several years. The data was not from controlled experiments and was assumed to contain human error due to manual inputs. Data cleaning and standardization was required to generate appropriate datasets for statistical analysis. The Confluence environment contained completed and on-going projects so proper filtering to only extract projects that were completely done was necessary. A review of the history logs revealed issues such as duplicate entries and incomplete status updates. Datasets with less than 30 data points were removed from the analysis. The datasets did not have the same number of data points and ones that were extreme such as 15 points compared to 375 points were not analyzed.

Statistical Methods

Statistical hypothesis tests were executed to examine the mean and standard deviation for each dataset. Histograms were generated using Sturge's Rule to calculate the bin size and then examined to determine if the data set was normal with a standard distribution. T-tests were conducted between the CI/CD process and RFC process datasets using the Two-Sample assuming unequal variances. Analysis of variance was performed on both datasets for each issue type and set of attributes.

Summary

This chapter reviewed the methodology used to extract the data and develop the performance values needed to compare the CI/CD and RFC process. It covered the methodology that was used to perform the observational study for both the between-subjects study and within-subjects study. It also provided justification on values used in the performance evaluation, data cleaning methods, and data grouping. The following chapter will discuss the results of the statistical analysis.

IV. Analysis and Results

Chapter Overview

This chapter will show the results from the between-subjects study and the within-subjects study. The between-subjects study was performed to see if the CIE automated tools within the CI/CD pipeline resulted in significantly shorter deployment timelines for each data type. The within-subjects study was performed to see what affect each selected attribute had on each data type. The data types selected were feature, story, task, bug, and spike. The attributes selected were workdays, priority, value stream and story point. See Appendix A for definitions of these data types and attributes. Each section will discuss the metric and give explanations on the observations. It will also discuss how useful the data types and attributes were in determining if the CIE automated tools improved deployment timelines.

Method 1: Between-Subjects Study

For this study, we were interested in knowing if the CI/CD pipeline improved the delivery timeline for different issue types. The next sections detail the results from the analysis of each issue type for the hypothesis in

Table 16.

Table 16. Between-Subjects Study Hypothesis

Between-Subjects Study: (CI/CD, RFC)			
Issue Type	Attribute	*Null Hypothesis*	*Alternative Hypothesis*
Feature	Workdays	H_0 = There is no difference in the delivery time for features between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for features than not using the CI/CD pipeline

Story	Workdays	H_0 = There is no difference in the delivery time for stories between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for stories than not using the CI/CD pipeline
Task	Workdays	H_0 = There is no difference in the delivery time for tasks between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for tasks than not using the CI/CD pipeline
Bug	Workdays	H_0 = There is no difference in the delivery time for bugs between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for bugs than not using the CI/CD pipeline
Spike	Workdays	H_0 = There is no difference in the delivery time for spikes between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for spikes than not using the CI/CD pipeline

Feature

Examining Figure 33, we observed that 90% of the features using the CI/CD pipeline are less than 300 workdays and 75% are less than 175 workdays. The data points greater than 450 workdays represent 2% of the sample. These were examined as possible outliers and it was concluded they were valid data points. Next we examined the RFC features using the histogram in Figure 34. We observe that 90% of the data points are less than 450 workdays and 75% are less than 275 workdays. The data points greater

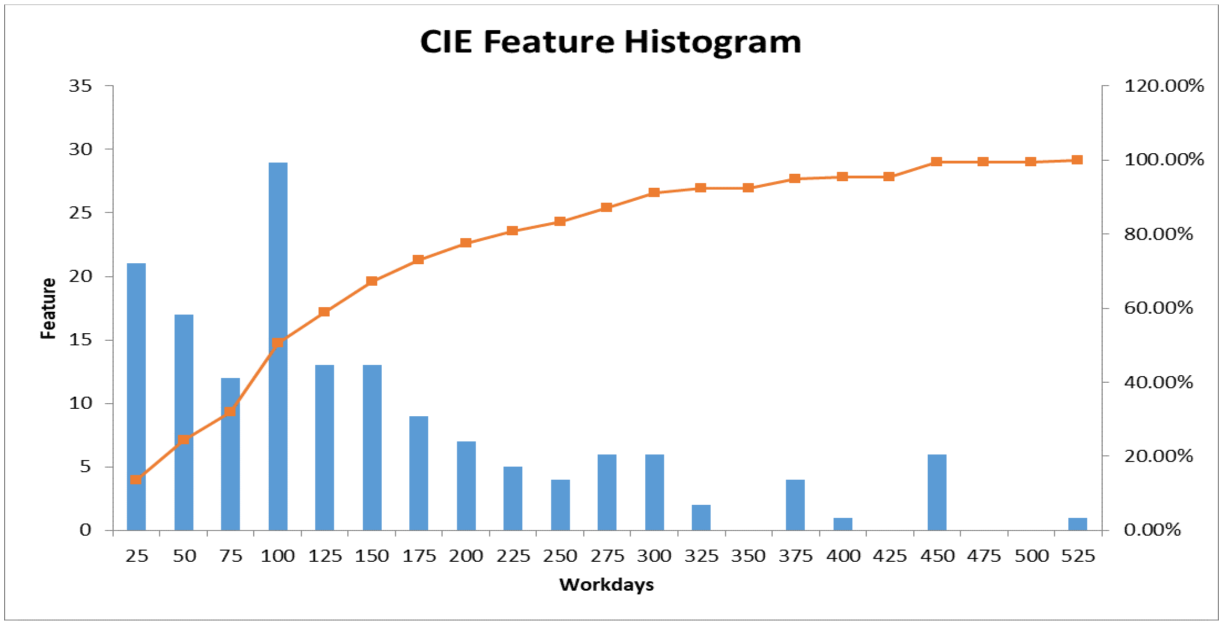


Figure 33. CIE Feature Histogram

than 600 workdays represent 2% of the sample. The data was examined for possible outliers and it was concluded they were all valid data points. Comparing these two histograms we observed the delivery time for features was reduced by 150 workdays for 90% of the data points. Additionally we observed the maximum workdays for features that did not use the CI/CD pipeline is significantly higher at 800 workdays compared to 525 workdays for features that did use the CI/CD pipeline.

The t-Test for the CI/CD pipeline and RFC feature workdays was significantly different at (t = -3.174, $p < 0.001$) and we rejected the null hypothesis that there is no difference in the delivery time for features between using the CI/CD pipeline and not using the CI/CD pipeline. There is evidence that the CI/CD pipeline improved the delivery time for features with an average of 135 workdays compared to an average of

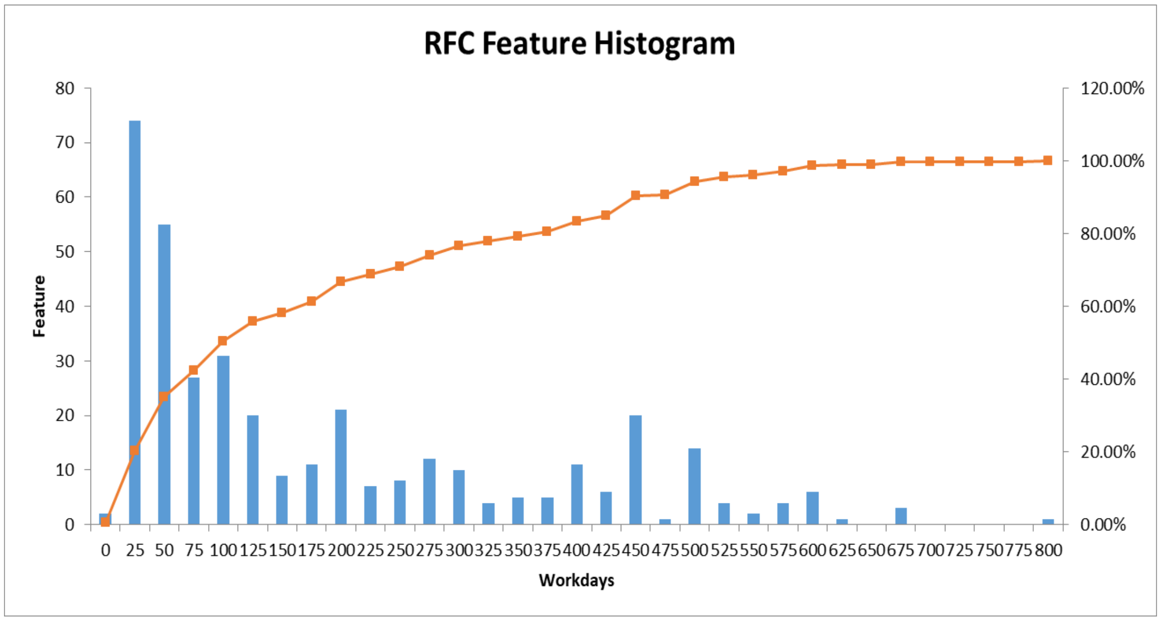


Figure 34. RFC Feature Histogram

177 workdays not using the CI/CD pipeline. The results of the t-Test are in Table 17.

Table 17. Two-sample t-Test Feature Workdays

t-Test: Two-Sample Assuming Unequal Variances		
	CIE Feature Workdays	*RFC Feature Workdays*
Mean	135.818	176.836
Variance	12824.787	31726.081
Observations	156.000	374.000
Hypothesized Mean Difference	0.000	
df	444.000	
t Stat	-3.174	
P(T<=t) one-tail	0.001	
t Critical one-tail	1.648	

Finally, we wanted to know if the delivery time shortened for features over time as the CI/CD process matured. We expected delivery times to shorten as the teams became more proficient with the new process and the automation improved. The average delivery time decreased from 1Q19 to 3Q20 from 99 workdays to 65 workdays. There was an initial spike up to 250 workdays for 2Q19 with an overall downward trend as shown in Figure 35. Examining the details of the data, we discovered 20 SIGINT features that entered the CI/CD pipeline as part of SIGINT Program Increment (PI)-7. These were the first features SIGINT ran through the pipeline for the OA - Airborne Sensor Emulator and Trainer that was a large and complex new development effort. They documented many issues such as working in docker containers for the first time (related to the CIE), but most were not due to the CIE such as getting security documentation ready for certificate to field and ASET failing functional and specification tests. These features were started through the pipeline over the next three to four quarters and the delivery speeds continued to improve. Features for several ARTs and value streams have been regularly flowing through the pipeline over the past three quarters. The evidence supports that the CI/CD speed of delivery is improving over time for features

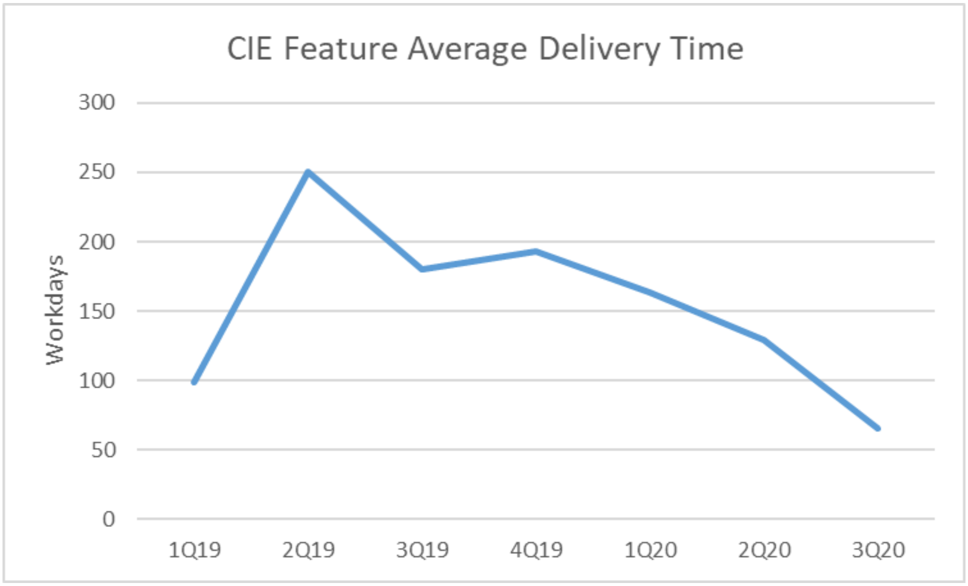


Figure 35. CIE Feature Delivery Speed by Qtr

Story

Examining Figure 36 we observed that 90% of the data points for stories using the CI/CD pipeline are less than 120 workdays and 70% are less than 50 workdays. The data points greater than 200 workdays represent 2% of the sample. The data was studied for possible outliers and it was concluded they were all valid data points. Next, we examined the

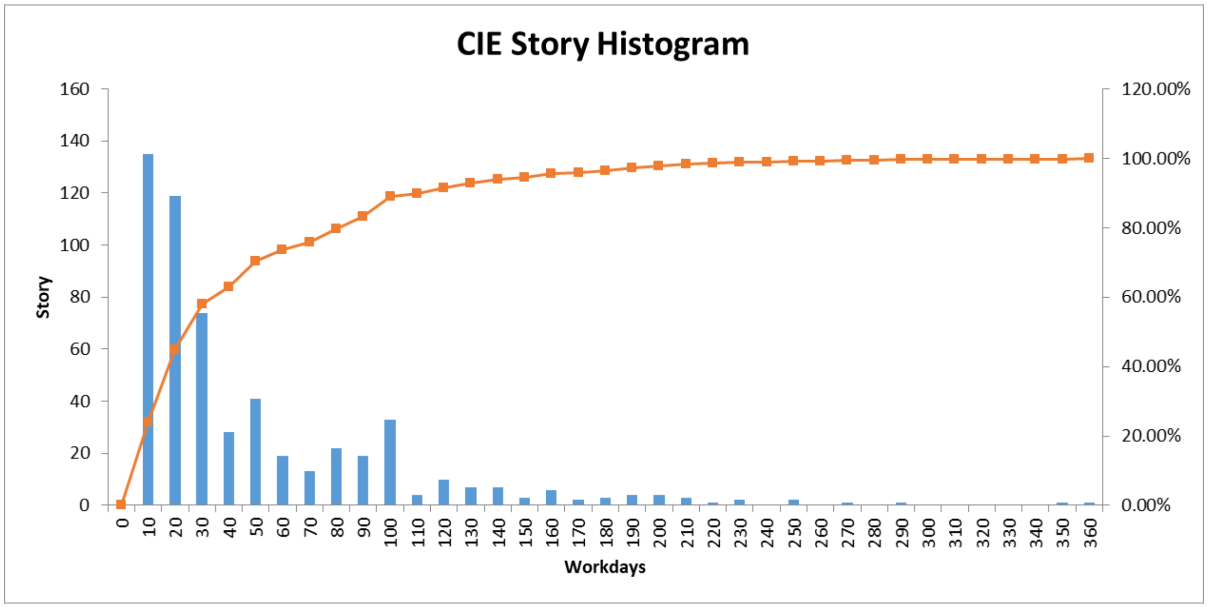


Figure 36. CIE Stories Histogram

RFC Stories using the histogram in Figure 37. We observed that 90% of the data points are less than 225 workdays and 70% are less than 75 workdays. The data points greater than 525 workdays represent 2% of the sample. The data was examined for possible outliers and it was concluded they were all valid data points. Comparing these two histograms we observed the delivery time for stories was reduced by 105 workdays for 90% of the data points. Additionally we observed the maximum workdays for stories that did not use the CI/CD pipeline is significantly higher at 775 workdays compared to 360 workdays for stories that did use the CI/CD pipeline.

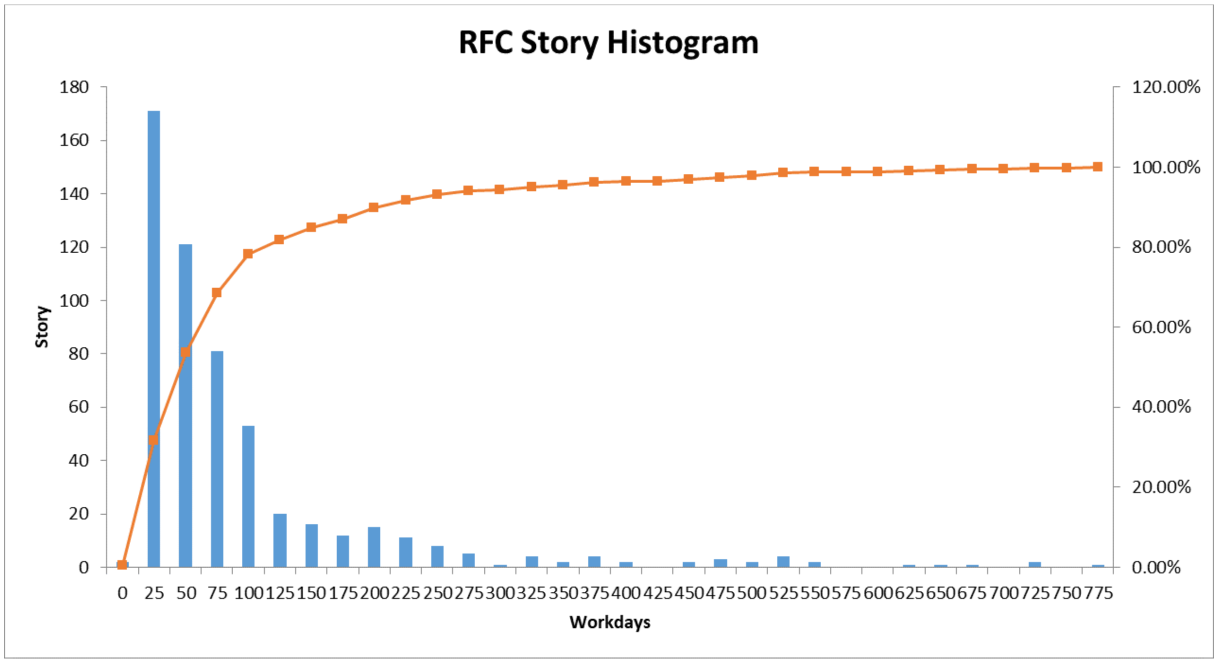


Figure 37. RFC Stories Histogram

The t-Test for the CI/CD pipeline and RFC story workdays was significantly different at (t = -7.137, $p < 0.000$) and we rejected the null hypothesis that there is no difference in the delivery time for stories between using the CI/CD pipeline and not using the CI/CD pipeline. The evidence supports that the CI/CD pipeline improved the delivery time for stories with an average of 45 workdays compared to an average of 84 workdays not using the CI/CD pipeline. The results of the t-Test are in Table 18.

Table 18. Two-sample t-Test Story Workdays

t-Test: Two-Sample Assuming Unequal Variances		
	CIE Story Workdays	*RFC Story Workdays*
Mean	45.146	83.740
Variance	2759.095	13324.710
Observations	565.000	547.000
Hypothesized Mean Difference	0.000	
df	757.000	
t Stat	-7.137	
P(T<=t) one-tail	0.000	
t Critical one-tail	1.647	

Finally, we wanted to know if the delivery time shortened for stories over time as the CI/CD process matured. We expected delivery times to shorten as the teams became more proficient with the new process and the automation improved. The average delivery time increased from 31 workdays 3Q18 to 44 workdays 3Q20 as shown in

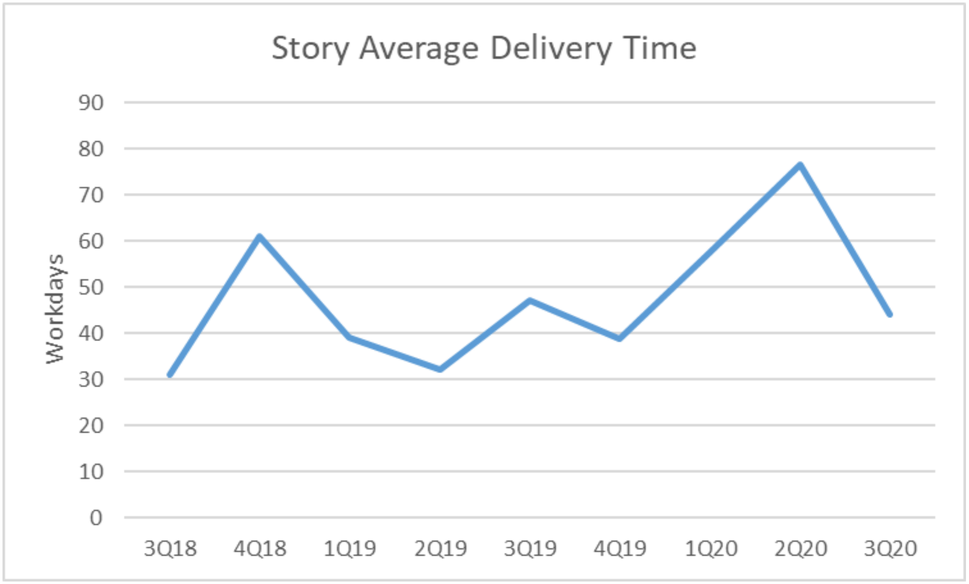


Figure 38. CIE Story Delivery Speed by Qtr

Figure 38. There was not a constant improvement trend observed and at the end of FY19 delivery times increased higher until 2Q20 when it started to decrease again. The data does not support a steady increasing or decreasing trend. Examining the details of the data, we discovered 26 GEOINT stories that entered the CI/CD 3Q18 for the MS-177 sensor integration 5+ year development effort. These were the first stories GEOINT ran through the pipeline for the High Altitude (HA) ART that was a large and complex new air/ground integration effort. The increase for delivery times starting in 4Q19 was due to additional MS-177 stories, SIGINT stories associated with the ASET features, and improvements made to the CIE automated tools and environment. Stories for automated virtual machines, Amazon Web Services (AWS) refactor, Kubernetes integration and Watchman improvements equally accounted for the rise in delivery times. There is not conclusive evidence to support faster delivery times using the CI/CD pipeline over time because the stories that slowed down the delivery time were typical work that is expected to flow through the pipeline. We concluded the more complex stories entering the pipeline as the ARTs spin up on the CI/CD process is causing the slower delivery speed.

Task

Examining Figure 39 we observed that 90% of the data points for tasks using the CI/CD pipeline are less than 125 workdays and 70% are less than 50 workdays. We also observed that 50% of the data is less than 25 workdays. The data points greater than 375 workdays represent 2% of the sample. The data was examined for possible outliers and it was concluded they were all valid data points. Next, we examined the RFC Tasks using

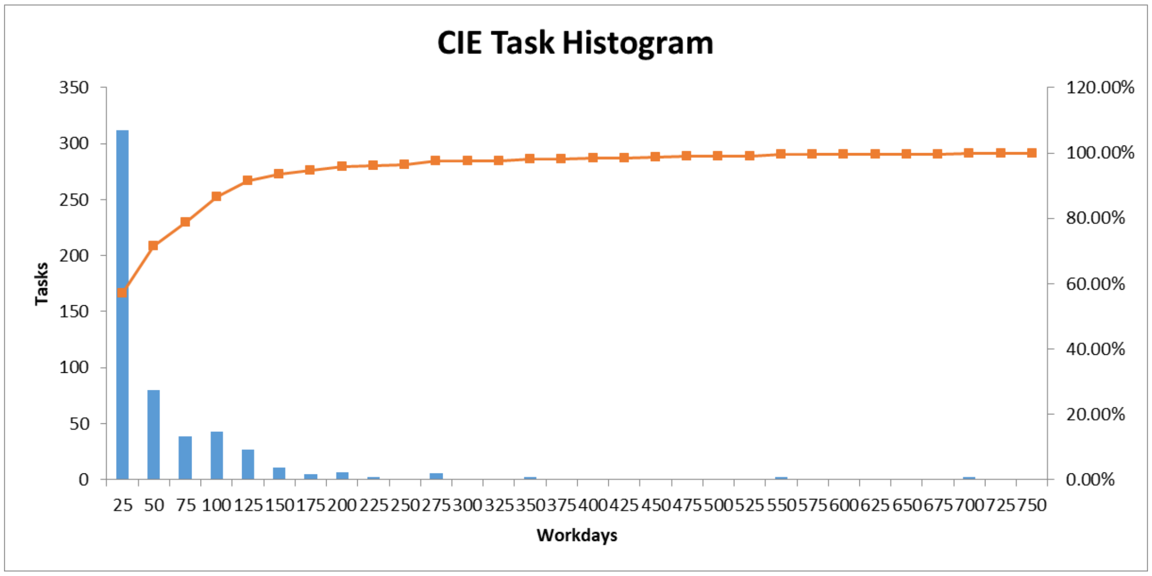


Figure 39. CIE Task Histogram

the histogram in Figure 40. We observed that 90% of the data points are less than 175 workdays and 70% are less than 62 workdays. The data points greater than 375

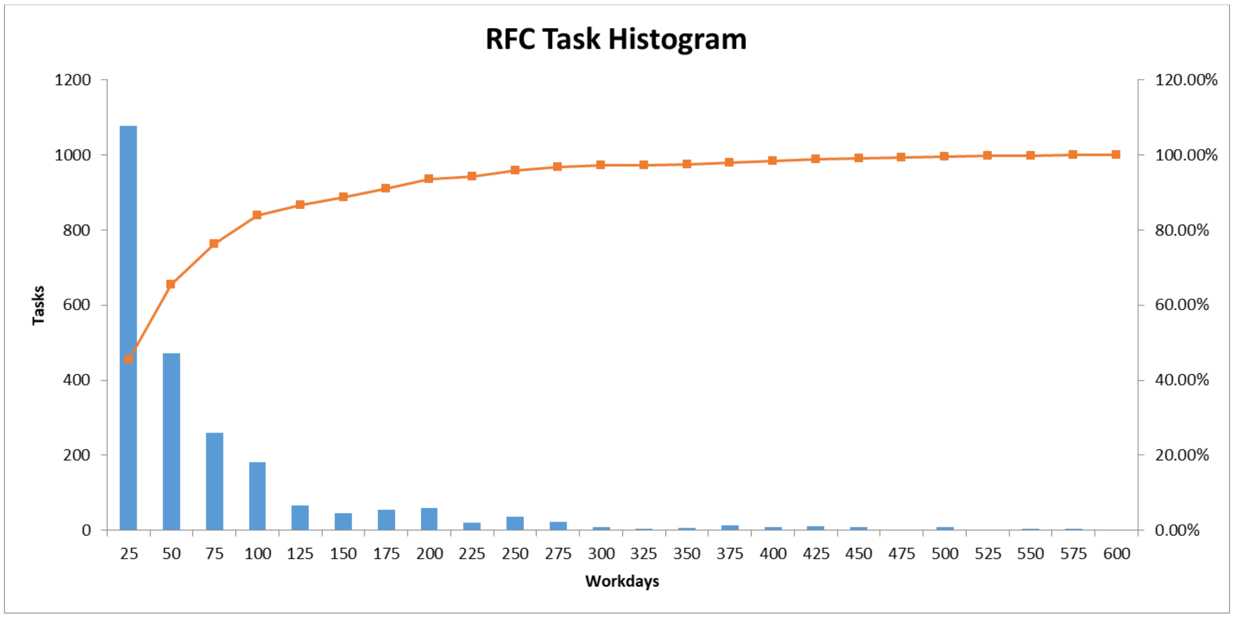


Figure 40. RFC Task Histogram

workdays represent 2% of the sample. The data was examined for possible outliers and it was concluded they were all valid data points. Comparing these two histograms we observed the delivery time for tasks was reduced by 50 workdays for 90% of the data points. Additionally we observed the maximum workdays for tasks that did use the CI/CD pipeline is slightly higher at 750 workdays compared to 600 workdays for tasks that did not use the CI/CD pipeline. There were six CI/CD pipeline tasks more than 500 workdays and 18 RFC tasks more than 500 days. We also observed only three CI/CD pipeline tasks were greater than the highest RFC task of 600 days.

The t-Test was run on the CI/CD pipeline and RFC task workdays and it was significantly different at (t = -2.948, p < 0.002). Therefore, we rejected the null hypothesis there is no difference in the delivery time for tasks between using the CI/CD pipeline and not using the CI/CD pipeline. The evidence supports the CI/CD pipeline improved the delivery time for tasks with an average of 49 workdays compared to an average of 62 workdays not using the CI/CD pipeline. The results of the t-Test are in Table 19.

Table 19. Two-sample t-Test Task Workdays

t-Test: Two-Sample Assuming Equal Variances		
	CIE Task Workdays	*RFC Task Workdays*
Mean	49.325	61.565
Variance	7776.702	7167.775
Observations	547.000	2368.000
Pooled Variance	7281.910	
Hypothesized Mean Difference	0.000	
df	2913.000	
t Stat	-3.024	
P(T<=t) one-tail	0.001	
t Critical one-tail	1.645	

Finally, we wanted to know if the delivery time shortened for tasks over time as the CI/CD process matured. We expected delivery times to shorten as the teams became more proficient with the new process and the automation improved. The average delivery time decreased from 267 workdays 1Q19 to 48 workdays 3Q20. The delivery times dropped to 32 workdays in 4Q18 and increased slightly holding stable through 3Q20 around 50 workdays. Examining the details of the data, we discovered 17 CI/CD pipeline tasks that entered the CI/CD 3Q18. These were some of the first tasks for the CIE to integrate enterprise services, infrastructure upgrades, and increase automation of the CI/CD pipeline. After the initial spin up in the first quarter, the remainder of the tasks were of similar scope and complexity accounting for the steady delivery time with fluctuations between 30 and 50 days as shown in Figure 41. There is not conclusive evidence to support faster delivery times for tasks using the CI/CD pipeline over time because the tasks for the initial standup give an artificially drastic improvement in the CI/CD delivery speed. A more accurate comparison is from 4Q18 at 32 workdays to 3Q20 at 48 workdays showing evidence that task delivery speed is slightly increasing over time.

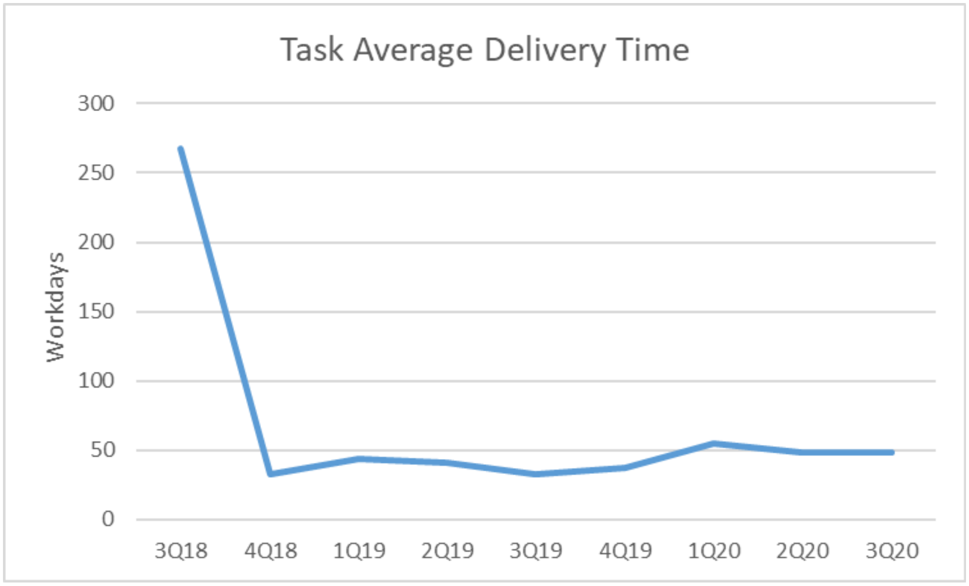


Figure 41. CIE Task Delivery Speed by Qtr

Examining throughput as a possible factor for the increase was not conclusive. The data in Table 20 shows higher throughput for some of the faster quarters. For example, 3Q19 had one of the faster delivery speeds of 32.400 days with 67 tasks completed, while 4Q19 has delivery speed of 36.918 days and only 39 tasks completed.

Quarter	Task	
	Workdays	#Points
3Q18	267.641	19
4Q18	32.262	65
1Q19	44.035	100
2Q19	40.891	63
3Q19	32.400	67
4Q19	36.918	39
1Q20	54.432	31
2Q20	48.620	56
3Q20	48.485	96

Table 20. CIE Task Throughput

Additionally 2Q20 and 3Q20 had speeds of 48.620 and 48.485 days yet throughput was 56 and 96 tasks respectively. There is no evidence that the pipeline slows down due to more tasks flowing through.

Bug

Examining Figure 42 we observed that 90% of the bugs using the CI/CD pipeline are less than 40 workdays and 70% are less than 20 workdays. The data points greater than 110 workdays represent 2% of the sample. The data was examined for possible outliers and it

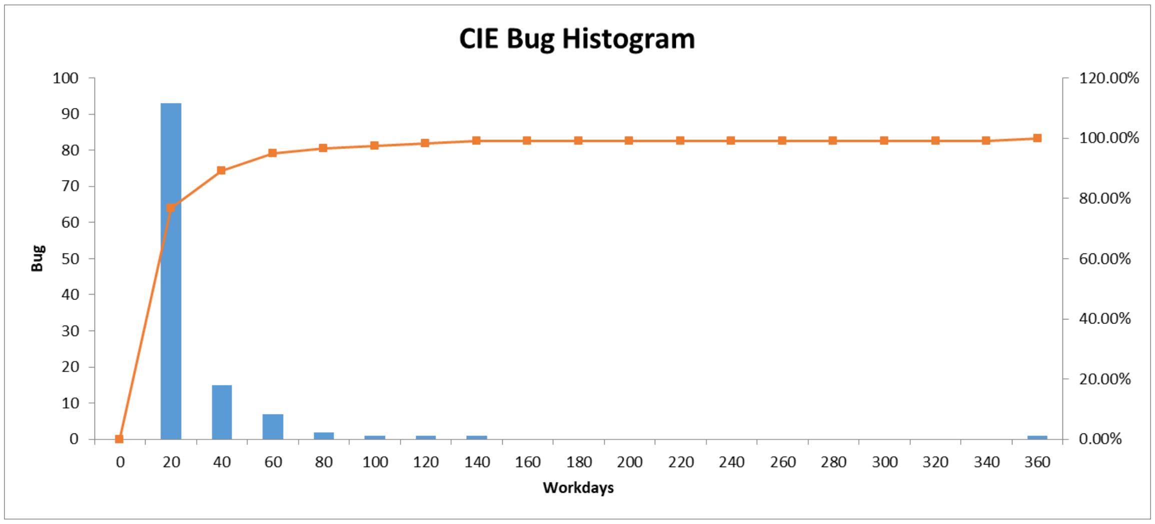


Figure 42. CIE Bug Histogram

was concluded they were all valid data points. Next, we examined the RFC bugs using the histogram in Figure 43. We observed that 90% of the data points are less than 175 workdays and 70% are less than 50 workdays. The data points greater than 475 workdays represent 2% of the sample. The data was examined for possible outliers and it was concluded they were all valid data points. Comparing these two histograms we observed the delivery time for bugs was reduced by 135 workdays for 90% of the data points. Additionally we observed the maximum workdays for bugs that did not use the

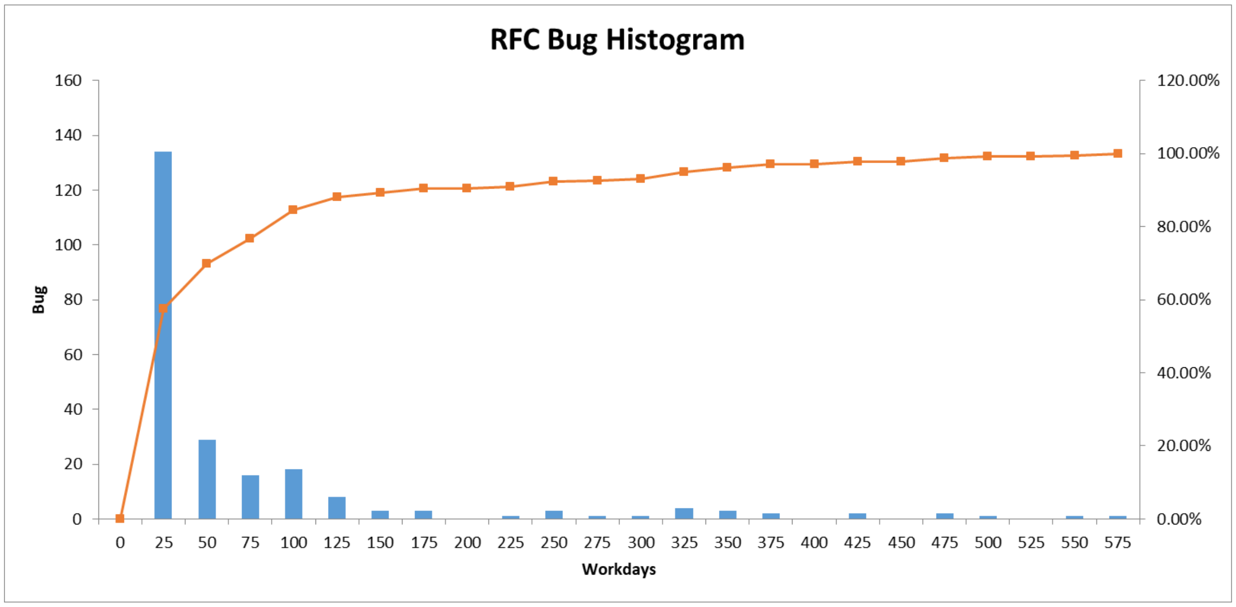


Figure 43. RFC Bug Histogram

CI/CD pipeline is significantly higher at 575 workdays compared to 360 workdays for stories that did use the CI/CD pipeline.

The t-Test was run on the CI/CD pipeline and RFC bug workdays and it was significantly different at (t = -6.125, p < 0.000). We observed that the CI/CD pipeline improved the delivery time for bugs with an average of 16.013 workdays compared to an average of 63.14 workdays not using the CI/CD pipeline. Therefore, we rejected the null hypothesis that there is no difference in the delivery time for bugs between using the CI/CD pipeline and not using the CI/CD pipeline and concluded that the CI/CD pipeline improved the delivery time for bugs. The results of the t-Test are in Table 21.

Table 21. Two-Sample t-Test Bug Workdays

t-Test: Two-Sample Assuming Unequal Variances		
	CIE Bug Workdays	*RFC Bug Workdays*
Mean	16.013	63.140
Variance	1355.618	11184.013
Observations	121.000	233.000
Hypothesized Mean Difference	0.000	
df	319.000	
t Stat	-6.125	
P(T<=t) one-tail	0.000	
t Critical one-tail	1.650	

Finally, we wanted to know if the delivery time shortened for bugs over time as the CI/CD process matured. We expected delivery times to shorten as the teams became more proficient with the new process and the automation improved. The average

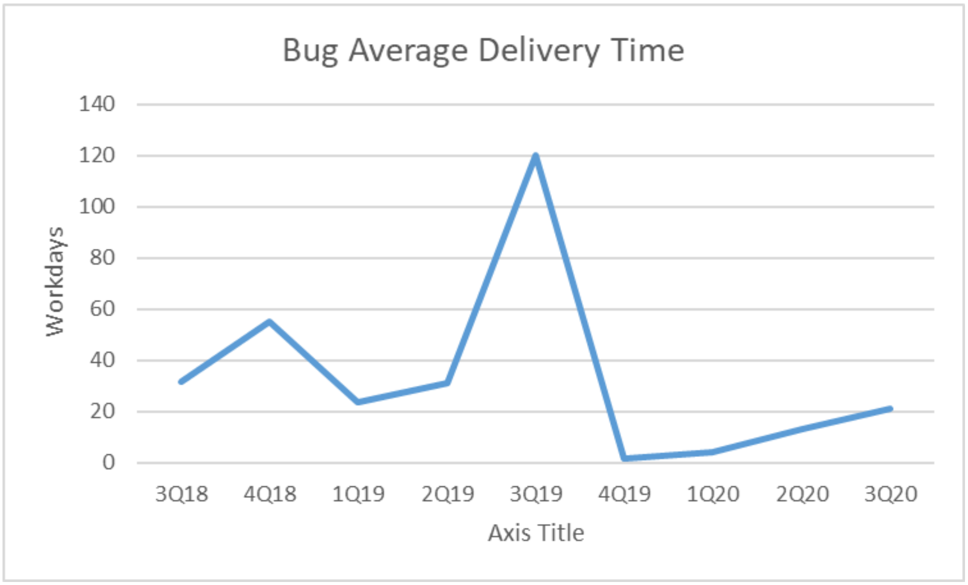


Figure 44. CIE Bug Delivery Speed by Qtr

delivery time decreased from 32 workdays 3Q18 to 21 workdays 3Q20 as shown in Figure 44. The delivery times had a large spike 3Q19 at 120 workdays, dropped to 1.5 workdays while gradually increasing up to 21 workdays. Examining the details of the data, we discovered one CI/CD pipeline task that entered the CI/CD pipeline 3Q19 for enterprise services. This bug was to make corrections to the Service Oriented Architecture Enterprise Service Bus that was having many issues with integration so it sat for months with no action. Examining throughput we note the earlier quarters delivered 2 to 3 bugs and the later quarters ranged from 20 to 37. There is not conclusive evidence to support faster delivery times for bugs over time.

Spike

Examining Figure 45 we observed that 90% of the spikes using the CI/CD pipeline are less than 100 workdays and 70% are less than 30 workdays. The data points greater than 220 workdays represent 2% of the sample. The data was examined for possible outliers

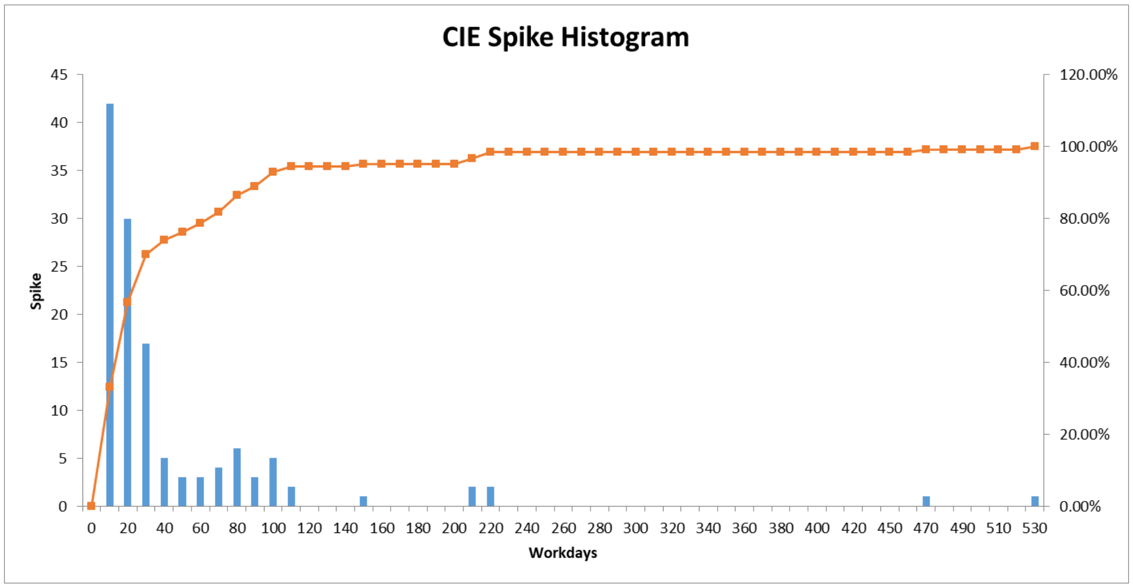


Figure 45. CIE Spike Histogram

and it was concluded they were all valid data points. Next, we examined the RFC Spikes using the histogram in Figure 47. We observed that 90% of the data points are less than 125 workdays and 70% are less than 62 workdays. The data points greater than 375 workdays represent 2% of the sample. The data was examined for possible outliers and it was concluded they were all valid data points. Comparing these two histograms we observed the delivery time for spikes was reduced by 25 workdays for 90% of the data points. Additionally we observed the maximum workdays for spikes that did not use the CI/CD pipeline is slightly higher at 575 workdays compared to 530 workdays for spikes that did use the CI/CD pipeline.

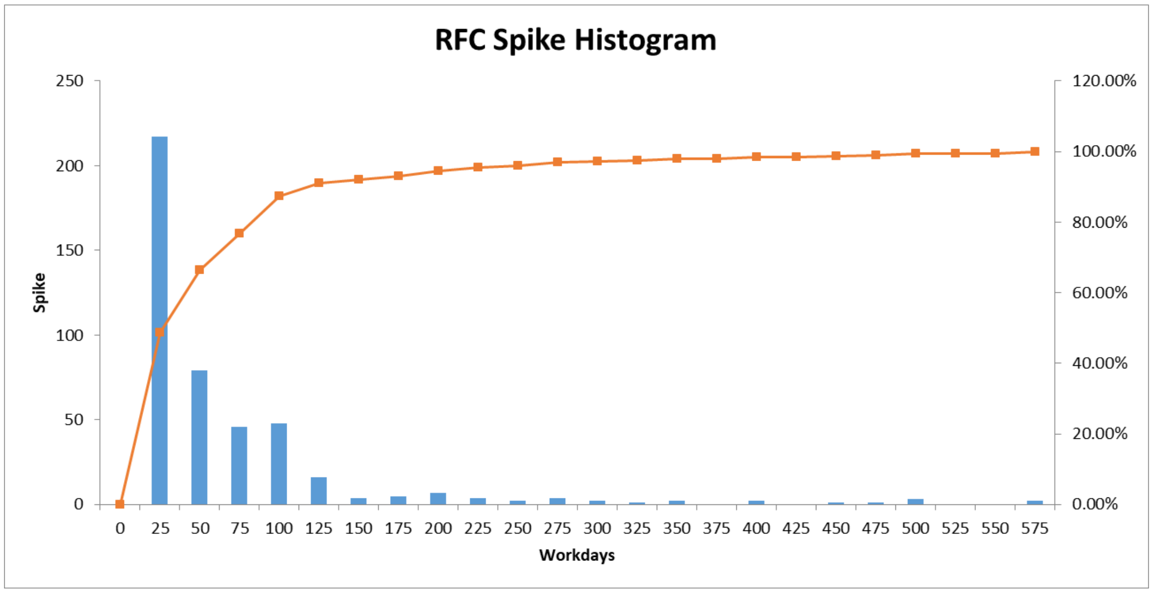


Figure 46. RFC Spike Histogram

The t-Test was run on the CI/CD pipeline and RFC spike workdays and it was significantly different at (t = -2.361, p < 0.010). Therefore, we rejected the null

Table 22. Two-Sample t-Test Spike Workdays

t-Test: Two-Sample Assuming Unequal Variances		
	CIE Spike Workdays	*RFC Spike Workdays*
Mean	40.680	58.354
Variance	5207.587	6713.440
Observations	127.000	446.000
Hypothesized Mean Difference	0.000	
df	227.000	
t Stat	-2.361	
P(T<=t) one-tail	0.010	
t Critical one-tail	1.652	

hypothesis there is no difference in the delivery time for spikes between using the CI/CD pipeline and not using the CI/CD pipeline. The results provide support for our hypothesis that the CI/CD pipeline improved the delivery time for spikes with an average of 41 workdays compared to an average of 58 workdays not using the CI/CD pipeline. The results of the t-Test are in Table 22.

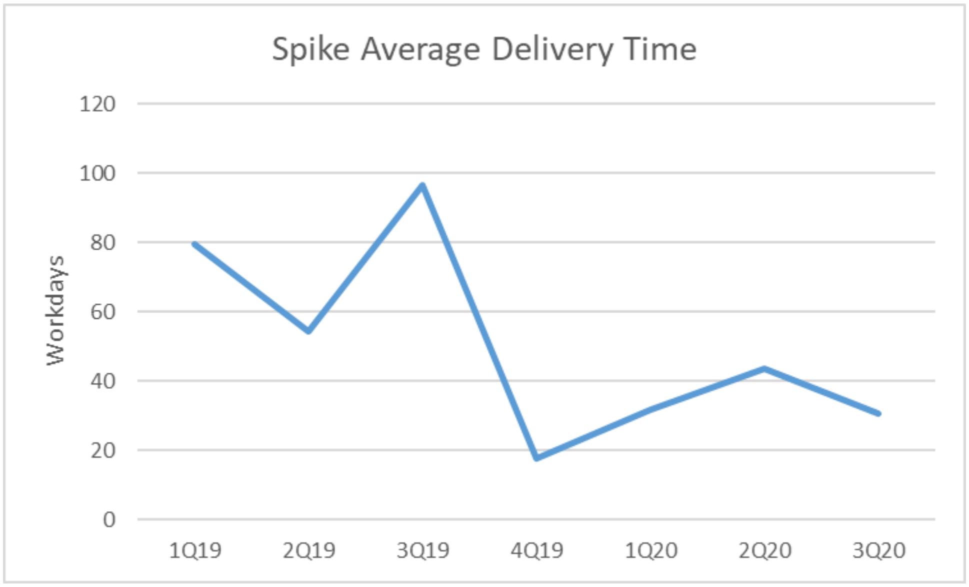


Figure 48. CIE Spike Delivery Speed by Qtr

Finally, we wanted to know if the delivery time shortened for spikes over time as the CI/CD process matured. We expected delivery times to shorten as the teams became more proficient with the new process and the automation improved. The average delivery time decreased from 1Q19 to 3Q20 from 80 workdays to 31 workdays. There was one increase of 97 workdays for 3Q19 but an overall downward trend as shown in Figure 48. Examining the details of the data, we discovered a few spikes for the CI/CD pipeline Infrastructure that attributed to the slower delivery speed for 3Q19 but the associated work was typical and not associated with the initial standup of the CIE automated tools. Examining the rest of the data points did not identify any spikes that were unusual. Therefore the evidence supports that the CI/CD speed of delivery is improving over time for spikes.

Results of Between-Subjects Study

Our analysis found that all issue types had shorter deployment times as summarized in Table 23. We observed that the CI/CD pipeline has made an improvement in speed of delivery for all issue types we examined and conclude that the pipeline is an improvement to the overall process and is of benefit to the AF DCGS program office.

Table 23. Results of Between-Subjects Study

Between-Subjects Study: Deployment timeline Results				
Issue Type	Attribute	*Null Hypothesis*	*Alternative Hypothesis*	*Result*
Feature	Workdays	H_0 = There is no difference in the delivery time for features between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for features than not using the CI/CD pipeline	🟩 *Rejected the null.* *The CI/CD pipeline resulted in shorter deployment times.*
Story	Workdays	H_0 = There is no difference in the delivery time for stories between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for stories than not using the CI/CD pipeline	🟩 *Rejected the null.* *The CI/CD pipeline resulted in shorter deployment times.*
Task	Workdays	H_0 = There is no difference in the delivery time for tasks between using the CI/CD pipeline and not using the CI/CD pipeline	H_a = The CI/CD pipeline will result in significantly shorter deployment timelines for tasks than not using the CI/CD pipeline	🟩 *Rejected the null.* *The CI/CD pipeline resulted in shorter deployment times.*

Legend:

🟩 CI/CD improved delivery timeline	🟥 CI/CD did not improve delivery timeline
🟨 Mixed results	⬜ Not enough data

Between-Subjects Study: Deployment timeline Results				
Issue Type	Attribute	*Null Hypothesis*	*Alternative Hypothesis*	*Result*
Bug	Workdays	*H_0* = There is no difference in the delivery time for bugs between using the CI/CD pipeline and not using the CI/CD pipeline	*H_a* = The CI/CD pipeline will result in significantly shorter deployment timelines for bugs than not using the CI/CD pipeline	🟩 *Rejected the null.* *The CI/CD pipeline resulted in shorter deployment times.*
Spike	Workdays	*H_0* = There is no difference in the delivery time for spikes between using the CI/CD pipeline and not using the CI/CD pipeline	*H_a* = The CI/CD pipeline will result in significantly shorter deployment timelines for spikes than not using the CI/CD pipeline	🟩 *Rejected the null.* *The CI/CD pipeline resulted in shorter deployment times.*

Legend:

🟩 CI/CD improved delivery timeline	🟥 CI/CD did not improve delivery timeline
🟨 Mixed results	⬜ Not enough data

Method 2: Within-Subjects Study

For this study, we examined data on specific attributes for each type of issue to determine if they have a significant effect on the delivery time. We examined each issue to see if the attributes of priority, value stream and story points have an effect on the delivery time. We expected higher priority issues to deliver faster regardless of complexity and appropriate resources applied to ensure quick delivery. We expected issues with higher story point value to deliver slower due to the work requiring more effort. A story point is a measure of effort. We have no basis to expect a particular value stream to be faster or slower than another so want to know if there are any trends based on the value streams. See Appendix A for definitions. The next sections detail the results for the studies in Table 24.

Table 24. Within-Subjects study hypotheses

Descriptive Study: Characterization by attributes			
Issue Type	Attribute	*Null Hypothesis*	*Alternative Hypothesis*
Feature	Value Stream	H_0 = There is no difference in the delivery time for features between value streams	H_a = The features have different delivery times based on value streams
	Priority	H_0 = There is no difference in the delivery time for features based on priority	H_a = The features have decreasing delivery times as the priority increases
Story	Value Stream	H_0 = There is no difference in the delivery time for stories between value streams	H_a = The stories have different delivery times based on value streams
	Priority	H_0 = There is no difference in the delivery time for stories based on priority	H_a = The stories have decreasing delivery times as the priority increases
	Story Points	H_0 = There is no difference in the delivery time for stories based on story points	H_a = The stories have increasing delivery times as the story points increase
Task	Value Stream	H_0 = There is no difference in the delivery time for tasks between value streams	H_a = The tasks have different delivery times based on value streams
	Priority	H_0 = There is no difference in the delivery time for tasks based on priority	H_a = The tasks have decreasing delivery times as the priority increases
	Story Points	H_0 = There is no difference in the delivery time for tasks based on story points	H_a = The tasks have increasing delivery times as the story points increase
Bug	Value Stream	H_0 = There is no difference in the delivery time for bugs between value streams	H_a = The bugs have different delivery times based on value streams
	Priority	H_0 = There is no difference in the delivery time for bugs based on priority	H_a = The bugs have decreasing delivery times as the priority increases
Spike	Value Stream	H_0 = There is no difference in the delivery time for spikes between value streams	H_a = The spikes have different delivery times based on value streams
	Priority	H_0 = There is no difference in the delivery time for spikes based on priority	H_a = The spikes have decreasing delivery times as the priority increases

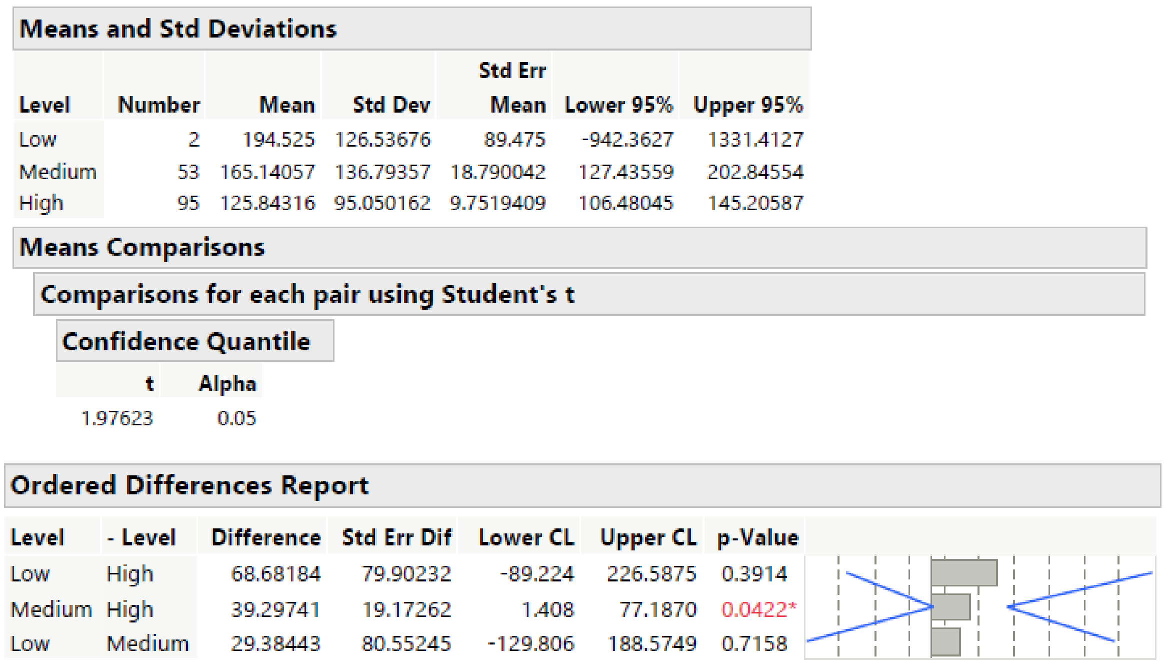
Feature

Priority

We ran a one-way analysis of variance for the feature workdays by the three priorities (Low, Medium and High). Examining Table 25 we observed there are only 2 data points for low priority and determine the tests comparing low to medium and high priority are inconclusive. We observed that there is a significant difference at (t = 1.976,

$p < 0.042$) between medium and high priority and rejected the null hypothesis. Our results provide support for decreased delivery times for high priority features as compared to medium priority features.

Table 25. Anova of Feature Workdays by Priority



Means and Std Deviations

Level	Number	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%
Low	2	194.525	126.53676	89.475	-942.3627	1331.4127
Medium	53	165.14057	136.79357	18.790042	127.43559	202.84554
High	95	125.84316	95.050162	9.7519409	106.48045	145.20587

Means Comparisons

Comparisons for each pair using Student's t

Confidence Quantile

t	Alpha
1.97623	0.05

Ordered Differences Report

Level	- Level	Difference	Std Err Dif	Lower CL	Upper CL	p-Value
Low	High	68.68184	79.90232	-89.224	226.5875	0.3914
Medium	High	39.29741	19.17262	1.408	77.1870	0.0422*
Low	Medium	29.38443	80.55245	-129.806	188.5749	0.7158

Value Stream

Three t-Tests assuming unequal variances were conducted for features between GEOINT and SIGINT, GEOINT and Infrastructure, and SIGINT and Infrastructure. We observed that there is a significant difference at ($t = -2.489$, $p < 0.008$) between the GEOINT and SIGINT value stream, at ($t = 2.675$, $p < 0.004$) between GEOINT and Infrastructure, and at ($t = -4.746$, $p < 0.000$) between SIGINT and Infrastructure. Our results provide support for our hypothesis that delivery times are different based on value streams and rejected the null hypothesis. Additionally there is support that Infrastructure has the fastest delivery times, then GEOINT, with SIGINT having the longest delivery times. The results of the t-Test are shown in Table 26.

Table 26. Two-sample t-Test Feature Value Stream

t-Test: Two-Sample Assuming Unequal Variances

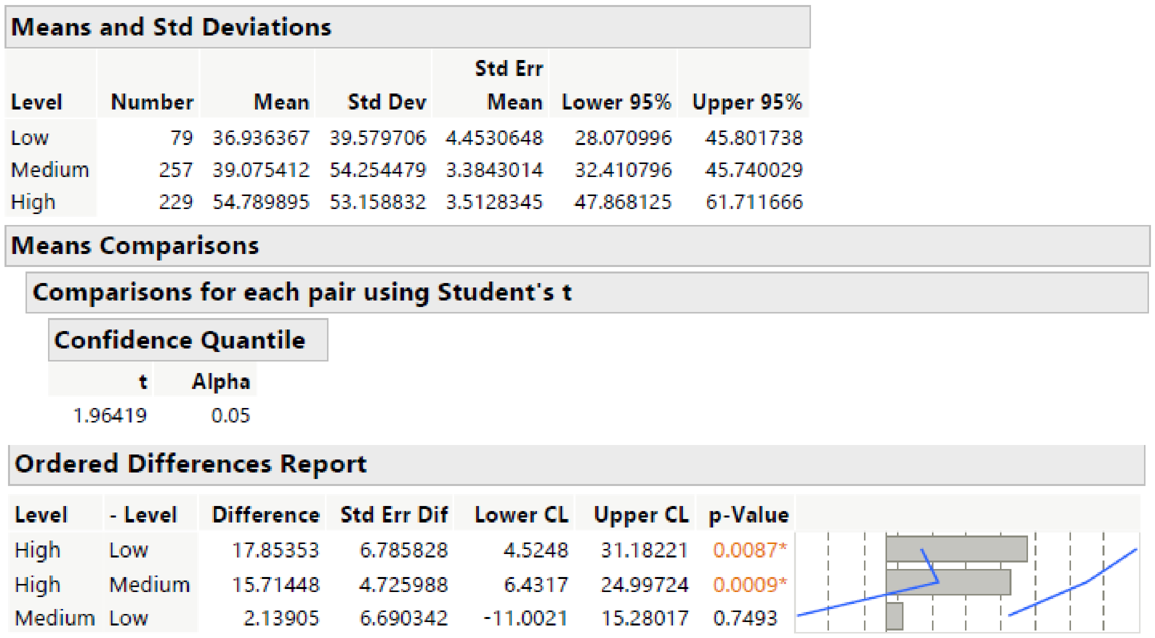
Issue Type: Feature	*GEOINT*	*SIGINT*	*GEOINT*	*Infrastructure*	*Infrastructure*	*SIGINT*
Mean	135.227	191.625	135.227	86.054	86.054	191.625
Variance	14800.426	11393.660	14800.426	5746.427	5746.427	11393.660
Observations	83.000	34.000	83.000	36.000	36.000	34.000
df	70.000		102.000		59.000	
t Stat	-2.489		2.675		-4.746	
P(T<=t) one-tail	0.008		0.004		0.000	
t Critical one-tail	1.667		1.660		1.671	

Story

Priority

We ran a one-way analysis of variance for the story workdays by the three priorities (Low, Medium and High). Examining Table 27 we observed that there is a significant difference at (t = 1.964, p < 0.009) between high and low priority and at (t = 1.964, p < 0.001) between high and medium priority and reject the null hypothesis. Our results provide support for decreased delivery times for high priority features as compared to medium and low priority features. We failed to reject the null hypothesis at (t = 1.964, p < 0.749) between medium and low priority and concluded there is no significant difference for story workdays.

Table 27. Anova of Story Workdays by Priority



Means and Std Deviations

Level	Number	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%
Low	79	36.936367	39.579706	4.4530648	28.070996	45.801738
Medium	257	39.075412	54.254479	3.3843014	32.410796	45.740029
High	229	54.789895	53.158832	3.5128345	47.868125	61.711666

Means Comparisons

Comparisons for each pair using Student's t

Confidence Quantile

t	Alpha
1.96419	0.05

Ordered Differences Report

Level	- Level	Difference	Std Err Dif	Lower CL	Upper CL	p-Value
High	Low	17.85353	6.785828	4.5248	31.18221	0.0087*
High	Medium	15.71448	4.725988	6.4317	24.99724	0.0009*
Medium	Low	2.13905	6.690342	-11.0021	15.28017	0.7493

Value Stream

A t-Test assuming unequal variances was conducted for stories between GEOINT and Infrastructure value streams. We observed that there is a significant difference at (t = -4.736, p < 0.000). Our results provide support for our hypothesis that delivery times are different based on value streams and reject the null hypothesis. The SIGINT and MULTIINT value streams did not have enough data to analyze. The results of the t-Test are shown in Table 28.

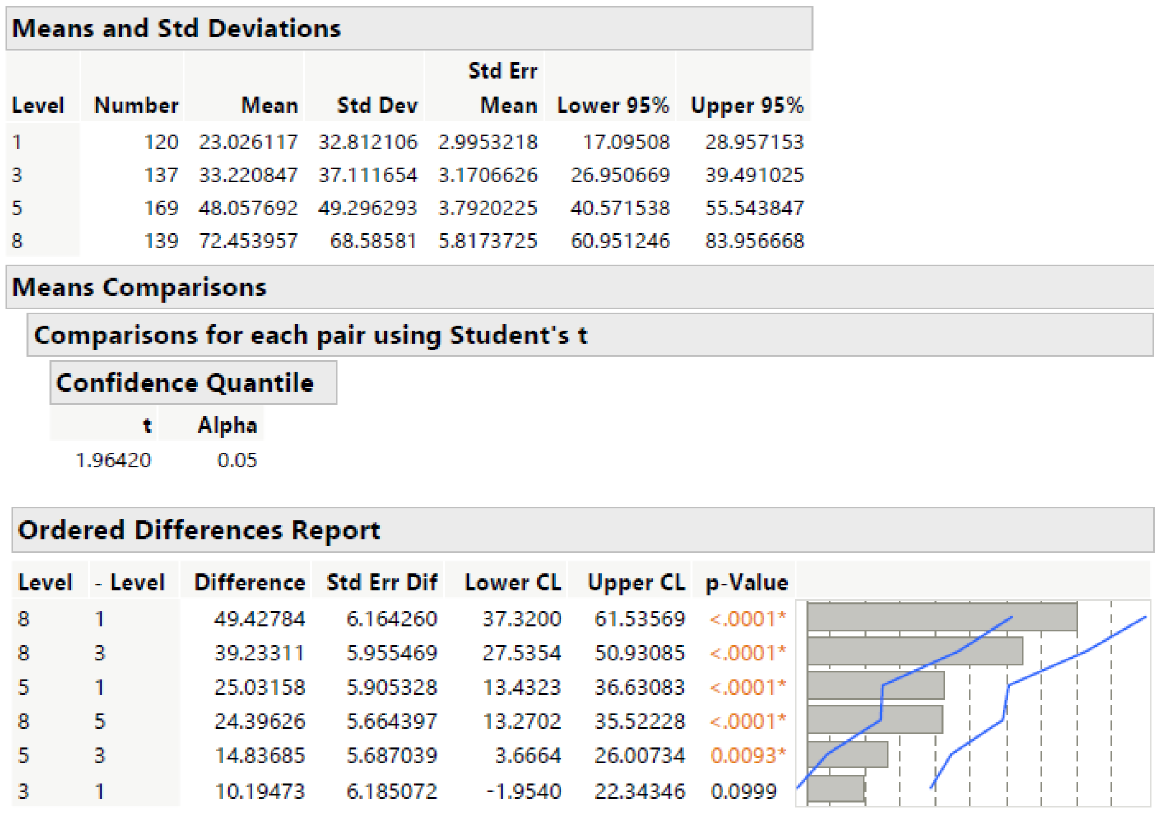
Table 28. Two-Sample t-Test Story Value Stream

t-Test: Two-Sample Assuming Unequal Variances		
Issue Type: Story	*Infrastructure*	*GEOINT*
Mean	35.809	57.824
Variance	2,258.216	2,443.082
Observations	393.000	154.000
df	270.000	
t Stat	-4.736	
P(T<=t) one-tail	0.000	

Story Points

We ran a one-way analysis of variance for the story workdays by four story points (1,3,5,8). Examining Table 29 we observed that there is a significant difference at (t = 1.964, p < 0.000) between story point (8:1, 8:3, 5:1, 8:5) and at (t = 1.964, p < 0.010) between story point (5:3) and reject the null hypothesis. We failed to reject the null hypothesis at (t = 1.964, p < 0.100) between story point (3:1) and conclude there is no significant difference for delivery times between story point 1 and 3.

Table 29. Anova of Story Workdays by Story point



Means and Std Deviations

Level	Number	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%
1	120	23.026117	32.812106	2.9953218	17.09508	28.957153
3	137	33.220847	37.111654	3.1706626	26.950669	39.491025
5	169	48.057692	49.296293	3.7920225	40.571538	55.543847
8	139	72.453957	68.58581	5.8173725	60.951246	83.956668

Means Comparisons

Comparisons for each pair using Student's t

Confidence Quantile

t	Alpha
1.96420	0.05

Ordered Differences Report

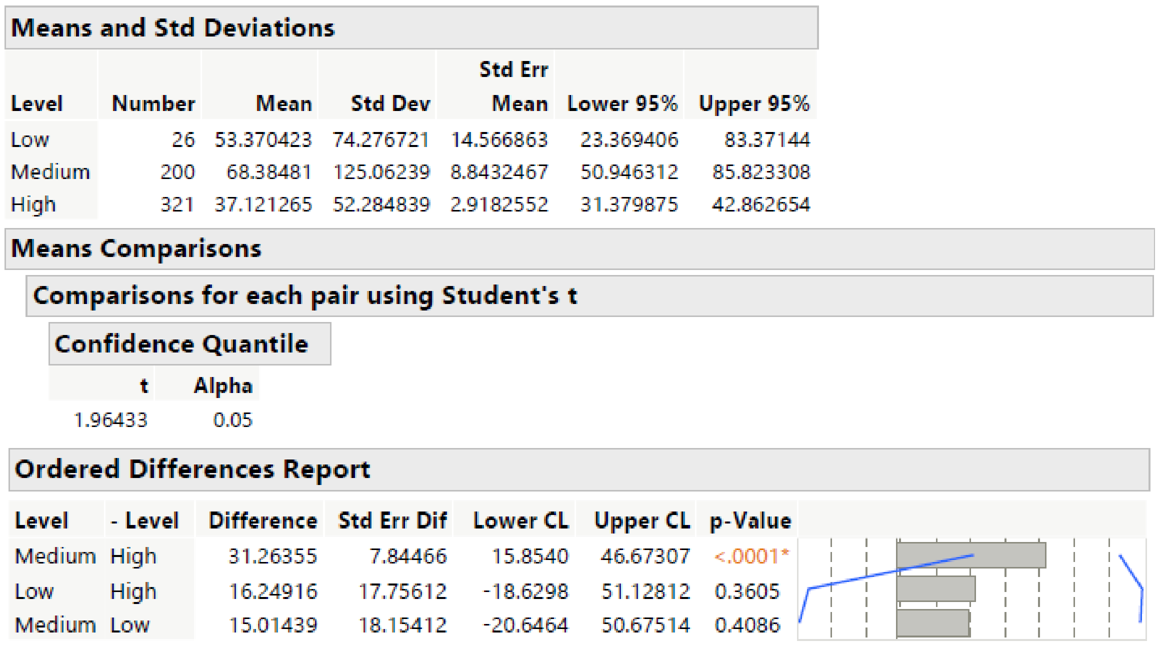
Level	- Level	Difference	Std Err Dif	Lower CL	Upper CL	p-Value
8	1	49.42784	6.164260	37.3200	61.53569	<.0001*
8	3	39.23311	5.955469	27.5354	50.93085	<.0001*
5	1	25.03158	5.905328	13.4323	36.63083	<.0001*
8	5	24.39626	5.664397	13.2702	35.52228	<.0001*
5	3	14.83685	5.687039	3.6664	26.00734	0.0093*
3	1	10.19473	6.185072	-1.9540	22.34346	0.0999

Task

Priority

We ran a one-way analysis of variance for the task workdays by the three priorities (Low, Medium and High). Examining Table 30 we observed that there is a significant difference at (t = 1.964, p < 0.000) between medium and high priority and rejected the null hypothesis. Our results provide support for decreased delivery times for high priority features as compared to medium priority features. We failed to reject the null hypothesis at (t = 1.964, p < 0.361, p < 0.409) between low and high priority and medium and low priority and conclude there is no significant difference for task workdays. There were a small number of low priority tasks at 26 data points compared to the medium and high at 200 and 321 respectively that could have been a factor for the high *p*-value.

Table 30. Anova of Task Workdays by Priority



Means and Std Deviations

Level	Number	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%
Low	26	53.370423	74.276721	14.566863	23.369406	83.37144
Medium	200	68.38481	125.06239	8.8432467	50.946312	85.823308
High	321	37.121265	52.284839	2.9182552	31.379875	42.862654

Means Comparisons

Comparisons for each pair using Student's t

Confidence Quantile

t	Alpha
1.96433	0.05

Ordered Differences Report

Level	- Level	Difference	Std Err Dif	Lower CL	Upper CL	p-Value
Medium	High	31.26355	7.84466	15.8540	46.67307	<.0001*
Low	High	16.24916	17.75612	-18.6298	51.12812	0.3605
Medium	Low	15.01439	18.15412	-20.6464	50.67514	0.4086

Value Stream

A t-Test assuming unequal variances was conducted for tasks between GEOINT and Infrastructure value streams. We observed that there is a significant difference at (t = 2.355, $p < 0.010$) for $\alpha = 0.05$. Our results provided support for our hypothesis that delivery times are different based on value streams and reject the null hypothesis. The SIGINT and MULTIINT value streams did not have enough data to analyze. The results of the t-Test are shown in Table 31.

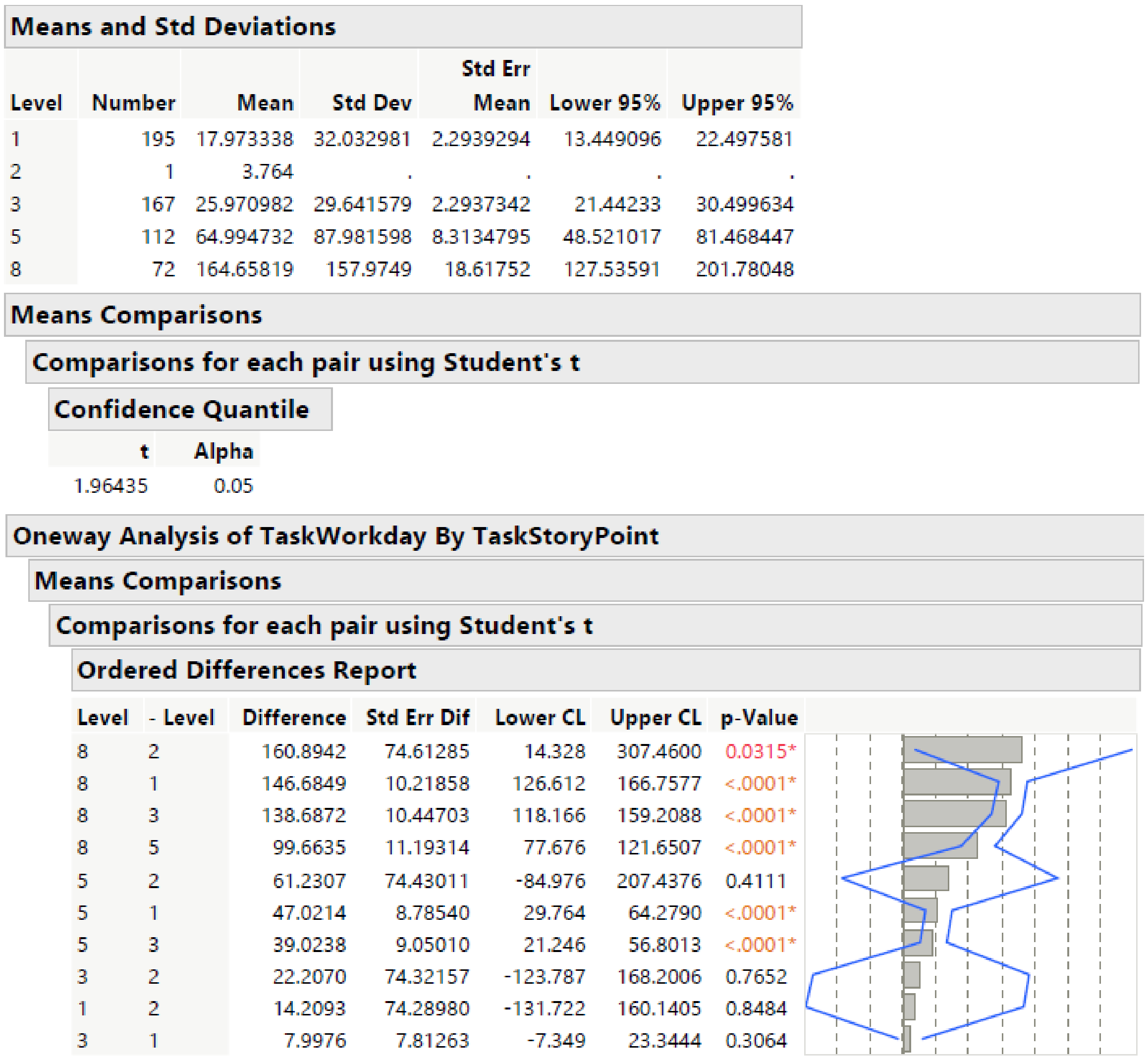
Table 31. Two-Sample t-Test Task Value Stream

t-Test: Two-Sample Assuming Unequal Variances		
Issue type: Task	*GEOINT*	*Infrastructure*
Mean	60.909	43.238
Variance	3548.009	9123.878
Observations	108.000	389.000
df	276.000	
t Stat	2.355	
P(T<=t) one-tail	0.010	
t Critical one-tail	1.650	

Story Points

We ran a one-way analysis of variance for the task workdays by four story points (1,3,5,8). Examining Table 32 we observed that there is a significant difference at (t = 1.964, $p < 0.000$) between story point (8:1, 8:3, 8:5, 5:1, 5:3) and rejected the null hypothesis. We failed to reject the null hypothesis at (t = 1.964, $p < 0.306$) between story point (3:1) and concluded there is no significant difference for delivery times between story point 1 and 3. There was only one data point for story point 2 and it is assumed erroneous data after looking at the project in Jira.

Table 32. Anova of Task Workdays by Story point



Means and Std Deviations

Level	Number	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%
1	195	17.973338	32.032981	2.2939294	13.449096	22.497581
2	1	3.764	.	.	.	.
3	167	25.970982	29.641579	2.2937342	21.44233	30.499634
5	112	64.994732	87.981598	8.3134795	48.521017	81.468447
8	72	164.65819	157.9749	18.61752	127.53591	201.78048

Means Comparisons

Comparisons for each pair using Student's t

Confidence Quantile

t	Alpha
1.96435	0.05

Oneway Analysis of TaskWorkday By TaskStoryPoint

Means Comparisons

Comparisons for each pair using Student's t

Ordered Differences Report

Level	- Level	Difference	Std Err Dif	Lower CL	Upper CL	p-Value
8	2	160.8942	74.61285	14.328	307.4600	0.0315*
8	1	146.6849	10.21858	126.612	166.7577	<.0001*
8	3	138.6872	10.44703	118.166	159.2088	<.0001*
8	5	99.6635	11.19314	77.676	121.6507	<.0001*
5	2	61.2307	74.43011	-84.976	207.4376	0.4111
5	1	47.0214	8.78540	29.764	64.2790	<.0001*
5	3	39.0238	9.05010	21.246	56.8013	<.0001*
3	2	22.2070	74.32157	-123.787	168.2006	0.7652
1	2	14.2093	74.28980	-131.722	160.1405	0.8484
3	1	7.9976	7.81263	-7.349	23.3444	0.3064

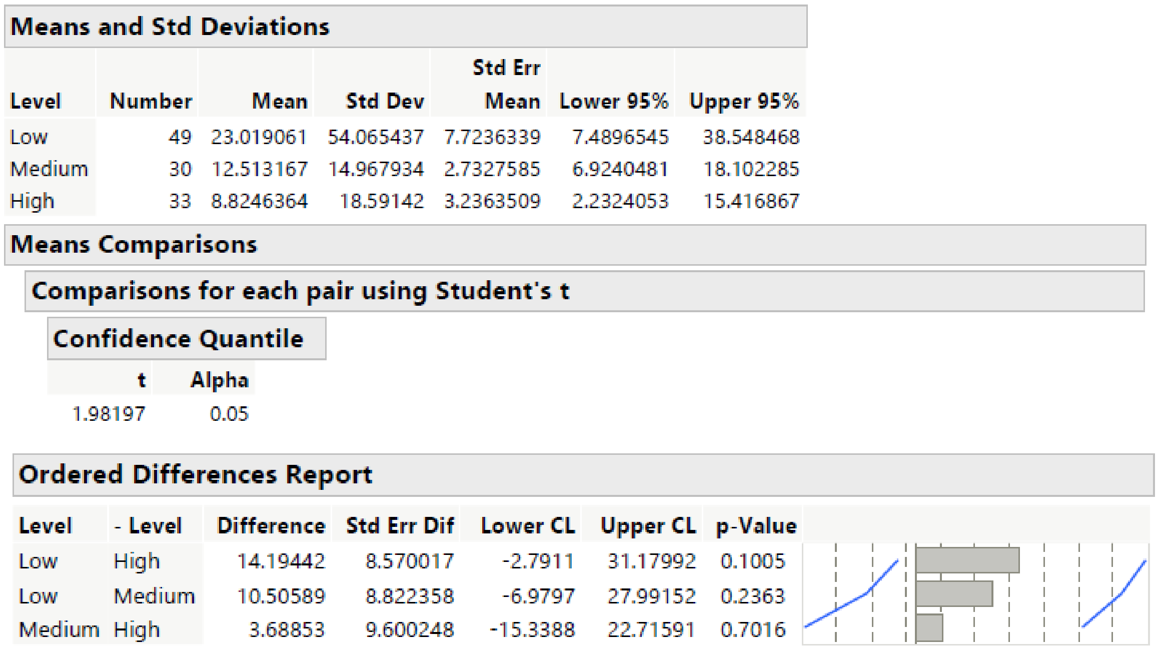
Bug

Priority

We ran a one-way analysis of variance for the bug workdays by the three priorities (Low, Medium and High). We failed to reject the null hypothesis at (t = 1.982,

$p < 0.101$, $p < 0.236$, $p < 0.702$) for all priorities and concluded there is no significant difference for bug workdays.

Table 33. Anova of Bug Workdays by Priority



Means and Std Deviations

Level	Number	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%
Low	49	23.019061	54.065437	7.7236339	7.4896545	38.548468
Medium	30	12.513167	14.967934	2.7327585	6.9240481	18.102285
High	33	8.8246364	18.59142	3.2363509	2.2324053	15.416867

Means Comparisons

Comparisons for each pair using Student's t

Confidence Quantile

t	Alpha
1.98197	0.05

Ordered Differences Report

Level	- Level	Difference	Std Err Dif	Lower CL	Upper CL	p-Value
Low	High	14.19442	8.570017	-2.7911	31.17992	0.1005
Low	Medium	10.50589	8.822358	-6.9797	27.99152	0.2363
Medium	High	3.68853	9.600248	-15.3388	22.71591	0.7016

Value Stream

The MULTIINT and GEOINT value streams did not have enough data points and there were no data points for SIGINT. We were not able to determine if the attribute value stream has any effect on delivery times for bugs.

Spike

Priority

We ran a one-way analysis of variance for spike workdays by the three priorities (Low, Medium and High). Examining Table 34 we failed to reject the null hypothesis at

(t = 1.982, p < 0.403, p < 0.822, p < 0.843) and concluded there is no significant difference in delivery times based on priority for spike workdays. We observed that the low priority spikes are delivered faster than the medium priority spikes. There was not enough data on the spikes to determine why low priority spikes are delivered faster.

Table 34. Anova of Spike Workdays by Priority

Means and Std Deviations

Level	Number	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%
Low	7	42.997143	76.494052	28.912034	-27.74806	113.74234
Medium	69	49.764928	90.924969	10.94608	27.922365	71.60749
High	37	36.80027	30.873958	5.0756474	26.50638	47.09416

Means Comparisons

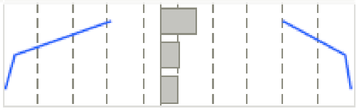
Comparisons for each pair using Student's t

Confidence Quantile

t	Alpha
1.98177	0.05

Ordered Differences Report

Level	- Level	Difference	Std Err Dif	Lower CL	Upper CL	p-Value
Medium	High	12.96466	15.44021	-17.6342	43.56354	0.4029
Medium	Low	6.76778	30.05793	-52.8000	66.33554	0.8223
Low	High	6.19687	31.23218	-55.6980	68.09171	0.8431



Value Stream

A t-Test assuming unequal variances was conducted for spikes between GEOINT and Infrastructure value streams. Examining Table 35 we failed to reject the null hypothesis at (t = -0.235, p < 0.407) and concluded there is no significant difference in delivery times based on value stream for spikes. The MULTIINT and SIGINT value streams did not have enough data points to perform an analysis. The results of the t-Test are shown in Table 35.

Table 35. Two-Sample t-Test Spike Value Stream

t-Test: Two-Sample Assuming Unequal Variances		
Issue type: Spike	*GEOINT*	*Infrastructure*
Mean	30.693	32.793
Variance	1042.063	3961.383
Observations	30.000	88.000
df	98.000	
t Stat	-0.235	
P(T<=t) one-tail	0.407	
t Critical one-tail	1.661	

Results of Within-Subjects Study

Our analysis did not provide very much insight into differences in the CI/CD pipeline delivery speed based on the attributes that were selected. We did observe some differences but the majority of the differences were not statistically significant or there wasn't enough data. Overall, feature, story, and task had the most significant differences in delivery times based on the attribute. Except for one case, bug and spike were either statistically not significant or not enough data to analyze. The results will be discussed in detail in Chapter V. The results of the hypothesis tests are summarized in Table 36.

Table 36. Results of Within-Subjects Study

Within-Subjects Study: Value Stream, Priority, Story Points				
Issue Type	Attribute	*Null Hypothesis*	*Alternative Hypothesis*	*Result*
Feature	Value Stream	H_0 = There is no difference in the delivery time for features between value streams	H_a = The features have different delivery times based on value streams	🟩 *Rejected the null* *The CIE delivery times are different for features based on value streams.*
	Priority	H_0 = There is no difference in the delivery time for features based on priority	H_a = The features have decreasing delivery times as the priority increases	🟩 *Rejected the null* *The CIE delivers high priority features faster than medium priority*
Story	Value Stream	H_0 = There is no difference in the delivery time for stories between value streams	H_a = The stories have different delivery times based on value streams	🟩 *Rejected the null* *The CIE delivery times are different for stories based on value streams.*
	Priority	H_0 = There is no difference in the delivery time for stories based on priority	H_a = The stories have decreasing delivery times as the priority increases	🟨 *Rejected the null for high/low priority* *Rejected the null for high/medium priority* *The CIE delivers high priority stories faster than medium and low priority stories* *Failed to reject the null for medium/low priority*
	Story Points	H_0 = There is no difference in the delivery time for stories based on story points	H_a = The stories have increasing delivery times as the story points increase	🟨 *Rejected the null for storypoint (8:1, 8:3, 8:5, 5:1, 5:3)* *The CIE delivery times for stories are different based on story points*

Legend:

🟩 CI/CD improved delivery timeline	🟥 CI/CD did not improve delivery timeline
🟨 Mixed results	⬜ Not enough data

Within-Subjects Study: Value Stream, Priority, Story Points				
Issue Type	Attribute	*Null Hypothesis*	*Alternative Hypothesis*	*Result*
				🟨 *Failed to reject the null for (3:1)*
Task	Value Stream	H_0 = There is no difference in the delivery time for tasks between value streams	H_a = The tasks have different delivery times based on value streams	🟩 *Rejected the null* *The CIE delivery times are different for stories based on value streams.*
	Priority	H_0 = There is no difference in the delivery time for tasks based on priority	H_a = The tasks have decreasing delivery times as the priority increases	🟨 *Rejected the null high/medium priority* *The CIE delivers high priority stories faster than medium priority tasks* *Failed to reject the null for low/high priority*** *Failed to reject the null for low/medium priority*
	Story Points	H_0 = There is no difference in the delivery time for tasks based on story points	H_a = The tasks have increasing delivery times as the story points increase	🟨 *Rejected the null for storypoint (8:1, 8:3, 8:5, 5:1, 5:3)* *The CIE delivery times for tasks are different based on story points* *Failed to reject the null for (3:1)*

Legend:

🟩 CI/CD improved delivery timeline	🟥 CI/CD did not improve delivery timeline
🟨 Mixed results	⬜ Not enough data

Within-Subjects Study: Value Stream, Priority, Story Points				
Issue Type	Attribute	*Null Hypothesis*	*Alternative Hypothesis*	*Result*
Bug	Value Stream	H_0 = There is no difference in the delivery time for bugs between value streams	H_a = The bugs have different delivery times based on value streams	*Inconclusive - Not enough data*
	Priority	H_0 = There is no difference in the delivery time for bugs based on priority	H_a = The bugs have decreasing delivery times as the priority increases	*Failed to reject the null for all prioirities*
Spike	Value Stream	H_0 = There is no difference in the delivery time for spikes between value streams	H_a = The spikes have different delivery times based on value streams	*Failed to reject the null for all value streams*
	Priority	H_0 = There is no difference in the delivery time for spikes based on priority	H_a = The spikes have decreasing delivery times as the priority increases	*Failed to reject the null for all prioirities*

Legend:

🟩 CI/CD improved delivery timeline	🟥 CI/CD did not improve delivery timeline
🟨 Mixed results	⬜ Not enough data

Summary

The AF DCGS CI/CD pipeline was analyzed for speed of delivery. The data set consisted of observable projects from July 2018 to September 2020 with no controlled experiment. The data set was created by extracting project data from Confluence and Jira on the DI2E DevTools platform for both CI/CD and non CI/CD projects. The results of the Between-Subjects study identified improvements in delivery times for the five issue types analyzed. The CI/CD pipeline was identified as a faster delivery method for AF DCGS request for changes. The within-subjects study was inconclusive due to insufficient data for some of the attributes and the inability to clearly identify positive improvements in the CI/CD delivery times due to a particular attribute. Improvements to the integration and test phase, the RFC process, and the metrics collection and reporting were recommended.

V. Conclusions and Recommendations

Chapter Overview

This chapter begins with conclusions of the research based on the results found in the analysis. The limitations of this study and recommendations for action are discussed in the second section. This chapter concludes with recommendations for further research and a summary of this research.

Conclusions of Research

This research identified several questions to investigate and answer on the performance and effectiveness of the AF DCGS CI/CD pipeline. This research attempted to draw conclusions using data from many teams performing work on the actual weapon system and using the integration and test labs. The projects over the past two years have varied widely from simple user interface changes to upgrades for new mission planning and sensor control. Studying the effect attributes have on each issue type was constrained to the available attributes that had enough data. In spite of these limitations, some insight was gained on the questions that were proposed in Chapter I.

The CI/CD Pipeline is Faster

The AF DCGS CI/CD pipeline has a faster delivery speed than the RFC process based on the research and analysis. Hypothesis tests were conducted on five different types of agile work products (referred to as issues) that went through both processes. The study was not a controlled experiment; rather, it was conducted on historical program data that was stored over the past 2 years in the DI2E environment. We were not comparing the exact same changes going through both processes but the types of changes that went through each process were similar enough to make our findings valid. We saw improvement on speed of delivery for all issue types from the smallest improvement of 22% for tasks to the largest improvement of 119% for bugs. The results are detailed in Table 37.

Table 37. Detailed delivery speed results

Issue Type	Delivery workdays		Days reduced	% Improvement
	CI/CD	RFC		
Feature	135.82	176.84	41.02	**26%**
Story	45.15	83.74	38.59	**60%**
Bug	16.01	63.14	47.13	**119%**
Spike	40.68	58.35	17.67	**36%**
Task	49.32	61.56	12.24	**22%**

We observed that each issue type had different averages for delivery workdays and we observed no similarities between issue types. We conclude that the issue type has an effect on the speed of delivery.

CI/CD pipeline is improving over time

There was evidence that the CI/CD process has improved speed of delivery over the past 2 years for all of the issue types except for task as shown in Table 38. The detailed study highlighted a couple of atypical events that would not occur once the pipeline was fully operational. As the pipeline was matured and the project teams rolled into the CIE it typically created a large spike in workdays as the projects overcame the

Table 38. CI/CD Pipeline Time-phased Results

Quarter	Feature		Story		Task		Bug		Spike	
	Workdays	#Points	Workdays	#Points	Workdays	#Points	Workdays	#Points	Workdays	#Points
3Q18			31.059	19	267.641	19	31.810	2		
4Q18			24.000	24	32.262	65	55.000	1		
1Q19	99.151	7	106.000	106	44.035	100	23.747	3	79.541	17
2Q19	250.658	6	64.000	64	40.891	63	31.245	3	54.365	15
3Q19	180.462	25	69.000	69	32.400	67	120.208	3	96.667	6
4Q19	193.460	17	73.000	73	36.918	39	1.664	16	17.719	15
1Q20	163.336	29	59.000	59	54.432	31	3.962	19	31.789	15
2Q20	129.307	32	53.000	53	48.620	56	13.082	37	43.798	17
3Q20	65.402	34	82.000	82	48.485	96	20.899	27	30.769	28

hurdles and issues associated with first-time integration. The projects were complex from a technical perspective and it took a lot of work to integrate them into the environment. They also were overcoming the learning curve of the new process, the automated tools, and a completely new way of doing integration and test. These extremely high spikes in workdays were more than 10 times longer than the majority of the other software changes as could be seen on the Histograms.

Selected attributes had inconclusive impact on speed of delivery

The attributes that were studied (priority, value stream, and story point) had minimal impact that could be quantified on the speed of delivery. We were able to conclude very little about improvement or differences in delivery speeds based on the selected attributes. We failed to reject the null hypothesis for many of the cases and for several of the cases there was not enough data for some of the attributes to perform an analysis. Table 39 shows that priority had limited significant differences with only four cases being statistically significant. Also, the high priority changes were not always

Table 39. CI/CD Delivery times for Priority

Priority

Issue Type	Delivery workdays								
	High	Medium	Reject Null?	High	Low	Reject Null?	Medium	Low	Reject Null?
Feature	125.843	165.141	Y	125.843	194.525	N	165.141	194.525	N
Story	54.790	39.075	Y	54.790	36.936	Y	39.075	36.936	N
Task	37.121	68.385	Y	37.121	53.370	N	68.385	53.370	N
Bug	8.825	12.513	N	8.825	23.019	N	12.513	23.019	N
Spike	36.003	49.765	N	36.003	42.997	N	49.765	42.997	N

delivered faster than the medium and low. The method for assigning the priority and the overall managing of the projects needs to be examined based on these results. Work that is marked as high priority does not appear to be given the emphasis necessary to deliver quickly. Table 40 shows that value streams gave us slightly more insight with six cases being statistically significant. Overall the Infrastructure value stream delivers faster than the other value streams. The faster delivery could be attributed to easier changes, but it

Table 40. CI/CD Delivery times for Value Stream

Value Stream

Issue Type	Delivery workdays								
	GEOINT	Infrastructure	Reject Null?	GEOINT	SIGINT	Reject Null?	SIGINT	Infrastructure	Reject Null?
Feature	135.227	86.054	Y	135.227	191.625	Y	191.625	86.054	Y
Story	57.824	35.809	Y	57.824	140.520	N/A	140.520	35.809	N/A
Task	60.909	43.238	Y	60.909	100.352	N/A	100.352	43.238	N/A
Bug	29.188	14.380	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Spike	30.693	32.793	Y	30.693	168.247	N/A	168.247	32.793	N/A

could also be because they are doing things better than the other value streams. Finally, we determined story point was not a good candidate for an attribute that will affect the delivery speed as shown in Table 41. There was significant differences between all of the

Table 41. CI/CD Delivery times for Story Points

Story Points

Issue Type	Delivery workdays				
	1	3	5	8	
Feature	N/A	N/A	N/A	N/A	
Story	23.026	33.221	48.058	72.454	not significant diff between sp 1 and 3
Task	17.973	25.971	64.995	164.658	not significant diff between sp 1 and 3
Bug	N/A	N/A	N/A	N/A	
Spike	N/A	N/A	N/A	N/A	

story points for stories and tasks except for story point 1 and story point 3. However, there was no story point data for features, bugs or spikes. Story points are not widely used by all the projects and most projects did not assign story points. Two of the attributes selected (story points and priority) are subjective rather than a factual attribute that is calculated. This human factor could account for some of the inconsistencies with the results. For instance, the lack of any standards for determining story points could result in a wide variation between the individuals assigning the story points. The limited data for many of the attributes may also have been a factor for the lack of any real significant findings or impacts on speed of delivery. Between the lack of a full data set and attributes that are not based upon any evident standards, the results from this portion of the study are inconclusive.

Automated metrics

The CI/CD pipeline needs to add automated metrics for tracking delivery speed, transition times for each Jira step and metrics to track quality. The data collection was a very arduous, time-consuming process to locate, interpret, and extract from Confluence, Jira, and eazyBI. There was a lot of data but most of it not relevant to speed of delivery. Metrics exist for the RFC process and the more waterfall-like steps and also for the agile process, regardless of whether the project goes through the pipeline. This study had originally intended to study improvement to the quality of deliveries, but the data was not available for quality measures. Navigating both processes and attempting to relate the disparate metrics was extremely difficult. The program office deliberately selecting meaningful metrics and putting the mechanism in place to capture and then build reports will be of great value as the CI/CD pipeline continues to mature.

Process constraints

The RFC and the CI/CD processes were awkwardly overlaid and the RFC process is obviously a hybrid mix of agile and waterfall. The result is a very confusing process flow with what appears to be a lot of time spent waiting in queues for signatures, reviews, testing, and boards. This research was not able to study those attributes due to the inability to distinguish between CI/CD projects and RFC projects in the eazyBI portion of DI2E.

Study Limitations

This study was limited by the lack of data, the difficulty to locate and collect the data, and suspected erroneous data. The data was from current projects generated from many teams across various types of projects. As a result, while the data represented a large variety of work it had uneven data sample sizes. In many cases there was less than 30 data points so no analysis was able to be performed on those data types. There were also data integrity issues. The data had to be cleaned and many data points were removed due to missing data. The Jira statuses were not standardized across projects which required interpretation. The issue types were also not standardized so a mapping had to be determined to ensure like-kind data was being compared. This study required access to a subject matter expert on the AF DCGS agile software development process and someone familiar with the ARTs and the weapon system to draw conclusions from the results.

Recommendations for Action

1. Modify the RFC process

The current AF DCGS process is a mix of agile, waterfall, and Air Force instructions - specifically formal readiness reviews, formal acceptance testing, formal boards. These two processes are in conflict with each other. The teams are using agile practices and documenting everything using Confluence and Jira. More importantly, they are using agile practices and language to perform their daily tasks. At the same time they are using a RFC process to track progress on software changes using a form on Confluence. Not only does this create two process flows and increase the work to document the same status in two locations, it generates two sets of metrics for the same changes using a different set of statuses. Also is was noted that the RFC process uses agile methods such as scrums, backlog, program increments, Kanban boards and other processes intended to empower the team and push decision making at the lowest level but then inserts gates and approvals. There was not an easy method to extract data to determine how much wasted time is spent waiting on these approvals so that was not studied. It is recommended that AF DCGS remove the RFC process and use nothing but Jira to track projects and fully adopt agile principles and remove the formal CCBs/CABs.

2. Establish quality metrics

Another recommendation is to capture quality metrics to determine if the CI/CD pipeline is improving the quality of software deliveries over time. Measures could be gathered for rework using the existing transition statuses, for failed tests, and a method to track the number of deficiencies generated during the integration and test activities in the CI/CD pipeline. It is not recommended to compare the quality of the CI/CD pipeline with the RFC process due to lack of easily accessible data on the RFC process. Quality data would have to be gleaned from reading through CCB results, test readiness reviews, and documentation for the RFC process. The labor required to gather this data would be better spent establishing methods to measure the quality of the CI/CD pipeline.

3. Remove the Integrated Test Cycle

Another recommended improvement based on the findings of this research is to remove the Integrated Test Cycle (ITC). AF DCGS has implemented an ITC that is a cadence driven test on a rigid six-week cycle that clashes with a CI/CD pipeline. The CIE Tiger team recognized the 6 week ITC needs to be phased out to stand up the CI/CD pipeline but no action was taken to remove the ITC or evidence of a transition plan has been observed.

4. Consolidate the labs

The last recommendation is to consolidate the multiple lab environments. Each lab has their own internal processes with numerous integrations and tests and workflow steps occurring that do not align with the CI/CD philosophy. Preliminary integration and testing occurs at Rome in their G7 lab for changes they are making, the managed lab environment at Robins AFB for integration and testing for all changes, then through the ITC for Integrated Acceptance Test and Development Test. There appears to be a lot of duplicative tests that do not align with agile or the CI/CD philosophy that are adding significant time to the workflow.

Recommendations for Future Research

A study of the transition times within the 28 step Jira process would be very beneficial to the AF DCGS program office. The RFC Leadership board has indications that some areas may be resource constrained. Researching the workflow steps and the time spent in each transition would give more insight into how each activity and team affects the overall speed of delivery and may uncover additional process improvements. Another study that would be very helpful to the program is identifying attributes that affect the speed of delivery. The projects are not currently capturing data that will provide information at a detailed enough level to understand how to improve the speed of the pipeline.

Summary

This chapter summarized the results from the between-subjects and within-subjects studies that were performed using actual data for the past two years from the AF

DCGS program. The speed of delivery for the CD/CD pipeline and associated CIE automated tools was first compared to the RFC process that does not use the CI/CD pipeline. The purpose was to demonstrate that the CI/CD pipeline delivers capability faster than the RFC process. The second study focused on different attributes that were available from historical project data and attempted to see if they had any effect on speed of delivery. This section also discussed recommendations for action and recommendation for future research and covered limitations of this study. We conclude that the CI/CD pipeline has increased speed of delivery and the program needs to converge on one process for sustainment and modernization of the weapon system. Based on these findings, recommend a full transition to the CI/CD pipeline.

Appendix A. Definitions

Bug - A bug is a problem with code identified outside of formal integration test and does not have to be linked to a feature. A bug is part of a release as a fix and the assigned priority is important (Jemilo, et al., 2019).

CI/CD Pipeline – The implementation of continuous integration and continuous delivery that is an embodiment of a culture, a set of operating principles, and practices that enable software development teams to deliver code changes more frequently and reliably. A key component of the pipeline is continuous testing and feedback loops (Jemilo, et al., 2019).

Confluence – Software product developed by Atlassian that is a team workspace to help structure, organize and share work so team members have visibility of and access to the information needed to perform their work (©2020 Atlassian, 2020).

Continuous Delivery (CD) – An extension of continuous integration that automatically deploys all code changes to a testing/production environment after the build stage. This is different from continuous deployment that releases the changes to the customer (Jemilo, et al., 2019).

Continuous Integration (CI) – A fundamental DevOps practice where a team of developers integrate their code early and often to a centralized code repository. The goal is to reduce risk during integration by not waiting until the end of a sprint to merge all code together (Jemilo, et al., 2019).
Jira – Software product developed by Atlassian that was originally designed as a bug and issue tracker but is now a work management tool for agile teams, project management teams, software development teams and product management teams (©2020 Atlassian, 2020).

DevOps – A mindset, a culture and a set of technical practices that provides communication, integration, automation, and close cooperation among all the people needed to plan, develop, test, deploy, release, and maintain a solution (Jemilo, et al., 2019).

Feature – A service that fulfills a stakeholder need. Each feature includes a benefit hypothesis and acceptance criteria, and is sized or split as necessary to be delivered by a single Agile Release Train (ART) in a Program Increment (PI) (Jemilo, et al., 2019)

Kanban Board – An agile project management tool designed as a visual method for managing workflow, limiting work in progress, and maximizing efficiency (©2020 Atlassian, 2020).

Spike – Activity to gain the knowledge required to reduce the risk of a technical approach, better understand a requirement, or increase the reliability of a story estimate (Jemilo, et al., 2019)

Sprint – A short, time-boxed period that a scrum team works to complete a set amount of work. Story points and the amount of team capacity for a given sprint determines the amount of work (velocity) assigned to a sprint (©2020 Atlassian, 2020).
Story – A short requirement or request written from the perspective of the end user that are intended to be something the team can commit to finish within a one to two-week sprint. Also called a user story (©2020 Atlassian, 2020).

Story point – An agile estimation of the relative effort of work in a Fibonacci-like format: 0, 0.5, 1, 2, 3, 5, 8, 13, 20, 40, 100 that determines the velocity of teams. The estimation is a team effort that focuses on the difficulty of the task rather than setting dates. The work is broken into the smallest unit and story points should be under 20 points to increase accuracy of the estimates (©2020 Atlassian, 2020).

Value Stream - Value streams represent the series of steps that an organization uses to implement solutions that provide a continuous flow of value to a customer (Jemilo, et al., 2019). AF DCGS organized the value streams around the categories of intelligence to align with requirements flow from the user.

Bibliography

©2020 Atlassian. (2020). *Confluence Basics*. Retrieved October 13, 2020, from Atlassian: https://www.atlassian.com/software/confluence/guides/get-started/confluence-overview

Alberts, D. S., Garstka, J. J., & Stein, F. P. (1999). *Network Centric Warfare: Developing and Leveraging Information Superiority, 2nd edition.* C4ISR Cooperative Research Program (CCRP).

Babbitt, D. (2019, June 20). Introduction to CI/CD. Retrieved August 26, 2020, from https://confluence.di2e.net/pages/viewpage.action?pageId=375359786

Billings, T. (2019, October 23). *Tailored SAFe for DCGS.* Retrieved from AF DCGS Confluence Page: https://confluence.di2e.net/display/AFDCGS/Tailored+SAFe+for+DCGS

Bridgwater, A. (2019). Changes in the CI/CD pipeline. *ComputerWeekly*.

Bush, W. (2019, June 14). *AF DCGS Open Architecture Schedule.* Retrieved from AF DCGS Schedules: https://intelshare.intelink.gov/sites/isrdcgsaf/DCGS/scheduling/Scheds/OA%20IMS/OA%20IMS%206_14_19%20PDF.pdf#search=OA%20OUE

Castellon, G., & Hurst, J. (2019, June 2019). CI/CD with Jenkins. Retrieved August 26, 2020, from https://confluence.di2e.net/pages/viewpage.action?pageId=375359816

Cazares, S. (2019, December 2). *Request for Change.* Retrieved from RFC Management Tool Transition Plan: https://confluence.di2e.net/display/AFDCGS/Request+for+Change+%28RFC%29+Management+Tool+%28RMT%29+Transition+Plan

Cazares, S., & Hamilton, S. (2020, January 21). *RFC Transition Plan*. Retrieved from AF DCGS Configuration Management: https://confluence.di2e.net/display/AFDCGS/Request+for+Change+%28RFC%29+Management+Tool+%28RMT%29+Transition+Plan

Clark, L. (2019). Deliver quality software at speed with CI/CD. *Computer Weekly*.

Dasovich, M. G. (2017, May 8). AF DCGS High-Level Operational Concept Graphic Operational View (OV)-1 Version 1.1. ACC/A5/2D.

D'hara, S. (2020, 05 28). *AF DCGS Architectural Runway*. Retrieved from AF DCGS Architecture Standards: https://confluence.di2e.net/display/AFDCGS/AF+DCGS+Architecture

DI2E. (2020). Retrieved April 10, 2020, from DI2E: https://www.di2e.net/

dos Santos, P. S., Beltrao, A. C., de Souza, B. P., & Travassos, G. H. (2018). On the benefits and challenges of using kanban in software engineering: a structured synthesis study. *Journal of Software Engineering Research and Development*.

Durante, R., & Haga, W. (2015, November 13). OA DCGS Transition Plan.

Gilmore, J. M. (2016). *DOT&E FY2016 Annual Report.* The Office of the Director, Operational Test and Evaluation. Retrieved August 24, 2020, from https://www.dote.osd.mil/Portals/97/pub/reports/FY2016/af/2016afdcgs.pdf?ver=2019-08-22-105429-373

Gutierrez, M. (2020, August 4). DCGS Immersion Brief v6.

Haga, M. W. (2017, October 16). How the US Air Force Made Its ISR Network Cheaper to Run and Easier to Upgrade. *Defense One*. Retrieved July 10, 2020, from https://www.defenseone.com/ideas/2017/10/how-us-air-force-made-its-isr-network-cheaper-run-and-easier-upgrade/141806/

Hamilton, S. (2020, July 30). *Request for Change Form Instructions*. Retrieved from AF DCGS Confluence: https://confluence.di2e.net/display/AFDCGS/Request+for+Change+%28RFC%29+Form+Instructions

Hamilton, S. (2020, November 6). *RFC Leadership Dashboard*. Retrieved from https://confluence.di2e.net/display/AFDCGS/RFC+Leadership+Dashboard

Hilton, M., Tunnell, T., Huang, K., Marinov, D., & Dig, D. (2016). Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects. *2016 31st*

IEEE/ACM International Conference on Automated Software Engineering (ASE), (pp. 426-437). Singapore.

Howard, D. (2014, September 30). AF DCGS Draft Reference Architecture. Robins AFB, GA: MITRE.

Jarnagin, A. (2020, January). *OADCGS Infrastructure Board.* Retrieved from https://jira.di2e.net/secure/RapidBoard.jspa?rapidView=18241&projectKey=OADCGS&view=planning&issueLimit=100

Jemilo, D., Leffingwell, D., Oren, I., Hohmann, L., Knaster, R., Koehnemann, H., . . . Willeke, E. (2019, January 20). *DevOps.* Retrieved September 30, 2020, from Scaled Agile Framework: https://www.scaledagileframework.com

Joint Chiefs of Staff. (2013, October 22). *JP 2-0, Joint Intelligence.*

Kim, G., Behr, K., & Spafford, G. (2014). *The Phoenix Project: A novel about IT, DevOps, and Helping Your Business Win.* Portland: IT Revolution Press.

Lambert, D., Arnold, L., Sylvester, C., Koyle, J., & Dent, C. (2020, Oct). *CIE Dashboard*. Retrieved from https://confluence.di2e.net/display/AFDCGS/CIE+Dashboard

Myrbakken, H., & Colomo-Palacios, R. (2017). DevSecOps: A Multivocal Literature Review. (p. 14). Halden, Norway: Ostfold University College,.

ODNI. (2020). *What is Intelligence.* Retrieved July 17, 2020, from Office of the Director of National Intelligence: https://www.dni.gov/index.php/what-we-do/what-is-intelligence

Oligmueller, F., & Smith, B. (2020, Aug 14). *AF DCGS Agile Development Model.* Retrieved from https://confluence.di2e.net/display/AFDCGS/AF-DCGS+Agile+Development+Model

Ott, L., & Longnecker, M. (2018). *An Introduction to Statistical Methods and Data Analysis.* Boston: Cengage Learning.

Phillips, A. (2014, July 29). The Continuous Delivery Pipeline — What it is and Why it's so Important in Developing Software. Retrieved July 22, 2020, from https://devops.com/continuous-delivery-pipeline/

Priddy, K. (2018, December 13). AF DCGS Agile Execution Guide.

Priddy, K. (2019, 06 11). *AF DCGS Standard Change Approval Memo for Record.* Retrieved from OA DCGS SPO RFC Tracker: https://confluence.di2e.net/display/AFDCGS/OA+DCGS+SPO+RFC+Tracker

Priddy, K. (2019). *AF DCGS Systems Engineering Plan.*

Russell, K., & Hamilton, S. (2020). *AF DCGS Portfolio.* Retrieved from AF DCGS Home Page: https://confluence.di2e.net/display/AFDCGS/AF+DCGS+Home

Sawyer, W., & Smith, B. (2019). *AF DCGS Agile Development JIRA Workflow Standardization CoP*. Retrieved from AF DCGS Communities of Practice: https://confluence.di2e.net/display/AFDCGS/AFDCGS+JIRA+CoP+-+Risk+Management+and+Communications+Plan

Scott, K. D., Weaver, G., Brown, M., & Browder, G. (2017, July 05). Retrieved September 15, 2020, from Joint Chiefs of Staff: https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp2_01_20170705v2.pdf

Spinelli, A., & Newton, C. (2016). *AFRL_INF Team Kanban Board*. Retrieved from OA DCGS: https://confluence.di2e.net/display/AFDCGS/Infrastructure+-+Dashboard

Steven, J. (2018, March 19). What's the difference between agile, CI/CD, and DevOps. Synopsys. Retrieved Aug 5, 2020, from https://www.synopsys.com/blogs/software-security/agile-cicd-devops-difference/

Stocum, D. (2019, June 20). Integration with Enterprise Services. Retrieved September 10, 2020, from https://confluence.di2e.net/display/AFDCGS/5.+Integration+with+Enterprise+Services

Sylvester, C. (2018, April 8). AF DCGS CIE Tiger Team Report.

Sylvester, C., Tschuor, T., & Dent, C. (2018, March 11). *CIE Guidebook.* Retrieved June 22, 2020, from CIE: https://confluence.di2e.net/display/AFDCGS/CIE+Guidebook

U.S. Air Force. (2015, October). Retrieved August 15, 2020, from About-Us: https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104525/air-force-distributed-common-ground-system/

Wellspring, M. (2020, August). AF DCGS CI/CD (Briefing).

Williams, M., & Anderson, A. (2020, May 4). *Software Delivery Guide.* Retrieved from https://confluence.di2e.net/display/AFDCGS/Software+Binary+Package+Delivery+Guide

Williams, M., & Clark, L. (2019, May 3). *DCGS RFC Form Rev 8 Instruction Guide.* Retrieved May 22, 2020, from AF DCGS CM Homepage: https://intelshare.intelink.gov/sites/isrdcgsaf/DCGS/dm/InstructionGuides/Request%20for%20Change_RFC/DCGS-FRM-CDM-0008_REV-8_RFC_Form.pdf?Web=1

Williams, M., Hamilton, S., Cazares, S., & Noreen, C. (2020). *Request for Change*. Retrieved from Configuration and Data Management: https://confluence.di2e.net/pages/viewpage.action?pageId=436666908

Zaydi, M., & Nassereddine, B. (2019). DevSecOps Practices for an Agile and Secure IT Service Management. *Journal of Management Information and Decision Sciences*, 14.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 19-12-2020	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From - To)* 29-06-2020 – 19-12-2020
4. TITLE AND SUBTITLE Continuous Integration/Continuous Delivery Pipeline for Air Force Distributed Common Ground System		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Fuller, Carolyn W		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way, Wright-Patterson AFB OH 45433-7765		**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENV-MS-20-D-042
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFLCMC/HBG, AF DCGS Chief Engineer, Mr. James Bechtel, 235 Byron Street, Suite 19A, Robins AFB, GA 31098-1670, james.bechtel@us.af.mil		**10. SPONSOR/MONITOR'S ACRONYM(S)** C2ISR
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

AF DCGS has a recognized need to improve speed of delivery for modification and sustainment of the weapon system. Given that the program office implemented a Continuous Integration /Continuous Delivery (CI/CD) process for the sole purpose of delivering capability to the field faster, there is a need to measure and report the pipeline throughput. This research conducts an independent evaluation of the newly implemented pipeline within AF DCGS's existing integration and test laboratories. Actual project data from the agile development work environments is studied and hypothesis tests are conducted to substantiate that the CI/CD pipeline improved the speed of delivery. The research definitively shows that the CI/CD pipeline improves speed of delivery for AF DCGS from a range of 22% to 119% depending on the type of work product. Lastly, from observation and detailed study of the processes and data, recommendations are made for standardization and automated metrics collection, with suggestions for additional research to further characterize the pipeline with the intent to create a predictive model for more accurate estimation of delivery timelines.

15. SUBJECT TERMS

Continuous delivery, continuous integration, continuous integration environment, AF DCGS, agile software development

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Brent T. Langhals, AFIT/ENV
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	135	(937)255-3636x7402 brent.langhals.1@us.af.mil

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18