3-22-2018

An Analysis of Multi-domain Command and Control and the Development of Software Solutions through DevOps Toolsets and Practices

Mason R. Bruza

AN ANALYSIS OF MULTI-DOMAIN COMMAND AND CONTROL AND THE DEVELOPMENT OF SOFTWARE SOLUTIONS THROUGH DEVOPS TOOLSETS AND PRACTICES

THESIS

Mason R. Bruza, Capt, USAF

AFIT-ENG-MS-18-M-016

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-18-M-016

AN ANALYSIS OF MULTI-DOMAIN COMMAND AND CONTROL AND THE DEVELOPMENT OF SOFTWARE SOLUTIONS THROUGH DEVOPS TOOLSETS AND PRACTICES

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

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 Cyber Operations

Mason R. Bruza, BS

Captain, USAF

March 22, 2018

AFIT-ENG-MS-18-M-016

AN ANALYSIS OF MULTI-DOMAIN COMMAND AND CONTROL AND THE DEVELOPMENT OF SOFTWARE SOLUTIONS THROUGH DEVOPS TOOLSETS AND PRACTICES

Mason R. Bruza, B.S.

Capt, USAF

Committee Membership:

Lt Col Mark G. Reith, Ph.D.
Chair

Lt Col Logan O. Mailloux, Ph.D.
Member

Maj Alan C. Lin, Ph.D.
Member

AFIT-ENG-MS-18-M-016

Abstract

Multi-Domain Command and Control (MDC2) is the exercise of command and control over forces in multiple operational domains (namely air, land, sea, space, and cyberspace) in order to produce synergistic effects in the battlespace, and enhancing this capability has become a major focus area for the United States Air Force (USAF). In order to meet demands for MDC2 software, solutions need to be acquired and/or developed in a timely manner, information technology infrastructure needs to be adaptable to new software requirements, and user feedback needs to drive iterative updates to fielded software. In commercial organizations, agile software development methodologies and concepts such as DevOps have been implemented to meet these demands. However, the USAF has been slow to adopt modern agile software development concepts such as DevOps in favor of traditional software development lifecycles and large contracts that can go nearly a decade without any value being released to the users. This work explores MDC2 software use cases and aims to show that MDC2 software can be successfully developed using modern agile software development practices in a timely manner. The contributions in this work have been published in two conference papers, and are pending publication in one journal article.

Acknowledgments

I would like to thank AFIT for granting me this rare opportunity. To my fiancé: thank you for your love and encouragement. I am also greatly indebted to Lt Col Enrique Oti for the opportunity to work with the outstanding DIUx team. My experiences there clarified my thinking on many subjects. Finally, I would like to express my sincere appreciation to my faculty advisor, Lt Col Mark Reith, for his guidance and support throughout the course of this thesis effort.

Mason R. Bruza

Table of Contents

Page

List of Figures

Page

List of Tables

Page

AN ANALYSIS OF MULTI-DOMAIN COMMAND AND CONTROL AND THE DEVELOPMENT OF SOFTWARE SOLUTIONS THROUGH DEVOPS TOOLSETS AND PRACTICES

I. Introduction

1.1 Framing the Problem

Command and Control (C2), in essence, is the passing of decision-quality information from one person or group of people to another in order to make decisions and/or take actions based on the decisions that have been made. This process of decision-making and information passing becomes more difficult as the number of people involved and the amount and/or complexity of the information increases. Multi-Domain Command and Control (MDC2) is in essence no different from single-domain command and control, but is nevertheless considerably more difficult because adding more operational domains to the command and control process greatly increases the number of people involved and the amount and complexity of information needed to make a decision. MDC2 is further complicated because information regarding one domain may be simple to someone familiar with that domain, but indecipherable to a person from another domain, so the same information can be simple or complex depending on who is looking at it.

Enhancing C2 involves reducing the number of people involved in making a decision, reducing the time it takes to pass information from one person/group to another person/group, making complex information easier to understand, and/or reducing the amount of information that is passed up and down the chain through changes in structure/processes as well as technical solutions. For example, using a decentralized C2

structure allows decisions to be made at lower levels which both reduces the number of people required to make a decision and it can ensure that the decisions are made by people with the greatest understanding of the information used to make the decision. An example of a technical solution would be an application that pulls needed information from several data sources and displays it in an intuitive format which reduces the time it takes for C2 personnel to receive the information they need and makes complex information more understandable with intuitive formatting. These same concepts apply when you add multiple domains to the C2 process, only the problem becomes more complex because of the additional personnel and information involved.

Enhancing the Air Force's MDC2 capabilities will require experimental campaigns to test new ideas whether they are technical solutions, changes to C2 structure or decision-making processes. As C2 operators test different solutions, feedback is collected and changes are made accordingly. Since enhancing MDC2 will be an iterative process, the software development lifecycle should be iterative as well in order to respond to changing user requirements. In other words, MDC2 software should be developed using agile software development concepts such as DevOps.

DevOps is an agile software development concept that seeks to use practices and toolsets that integrate Information Technology (IT) infrastructure operation teams with software development teams in order to release new code frequently into the production environment. It has become popular in the private sector, but has not been adopted by the military. In light of these observations, this thesis answers the following research questions:

1. **How can software enhance the Air Force's ability to conduct MDC2?**

2. **What software development practices have attributes that will aid MDC2 software development and experimentation?**
3. **What are the key attributes of the AOC Pathfinder Project that enabled it to develop software using DevOps?**

1.2 Hypothesis

DevOps toolsets and practices are effective at developing C2 software, and can be successfully implemented by the Air Force to enhance its MDC2 capabilities. This is shown by first examining the MDC2 problem and developing use cases that have possible software solutions and then demonstrating the effectiveness of DevOps for developing software for command control centers.

1.3 Methodology

The research for this work involves two major lines of effort. The first focuses on examining the MDC2 problem, and includes participation in two different MDC2 working groups that produced reports for USAF senior leaders and site visits to the 609th Combined Air Operations Center and the 624th (cyberspace) Operations Center. The second line of effort focused on DevOps and the ability of the USAF to successfully use DevOps toolsets and practices to develop C2 software, and involved more than 4 months of hands on experience with the Defense Innovation Unit Experimental (DIUx) working on the Air Operation Center (AOC) Pathfinder project security and platform team. These two lines of effort provided the background knowledge and observational data to answer the research questions

1.4 Document Structure

The bulk of the work in this thesis has been published or submitted for publication by the author in [6], [8], and [9]. Additionally, the author has presented material at two academic conferences and as lectures for three AFIT courses. Chapter II covers the content of [6], focusing on analyzing the MDC2 problem and answering research question one. Chapter III covers the content of [8], and examines how software should be developed for MDC2 use cases and shows how DevOps meets the requirements for MDC2 software development more effectively than traditional military software development practices in order to answer research question two. Chapter IV, covering the content of [9], is an experience report of the work being done by AOC Pathfinder, and shows that DevOps toolsets and practices can be used to effectively develop C2 software for Air Force operations centers. It examines key lessons learned for implementing DevOps for military software development in order to answer research question three. Chapter V presents a final analysis of the research done for this work and concludes.

II. Multi-Domain Command and Control: The Need for Capability Transparency

2.1 Introduction

Over the past century military technology has advanced at an unparalleled rate resulting in the creation of three distinct operational domains, so in addition to Land and Sea we now have Air, Space, and Cyberspace. Not only have new domains been created, but there is an unimaginable number of different assets and capabilities in each domain for commanders to choose from to accomplish their mission. Modern militaries need to operate in all of these domains, and be able to synergize assets and capabilities from different domains to create desired effects in order to accomplish the mission. A key issue faced by multi-domain planners is the lack of asset transparency; assets in domains like Space and Cyberspace are not likely owned by the planner's organization, so the planner may not even know they are available for use. Then there are additional considerations unique to each domain that need to be taken into account, such as the possible loss of an intelligence source when a cyber capability is used offensively. As we examine solutions for MDC2 we must also look at what approval authority level is appropriate for any given asset/capability. This chapter examines these problems facing MDC2 and proffers a technical solution that gives planners the full picture of all the assets and capabilities available to them, and allows them to see what effects they can create by combining assets together. The solution will be compatible with current command and control paradigms, but also provides a framework for future advancement in MDC2.

2.2 Motivation

Many United States Air Force (USAF) leaders have recognized the need for more agile command and control, and this can be seen in documents such as the Air Force Future Operating Concept (AFFOC) which makes operational agility the central idea of how the USAF will operate in the year 2035 [15]. In the AFFOC, operational agility is defined to be: “the ability to rapidly generate—and shift among—multiple solutions for a given challenge.” Because the solution(s) for a given challenge could come from any operational domain, in order to fully achieve operational agility in modern environments it is necessary to develop integrated MDC2 capabilities. Any organization that operates in multiple domains needs to be able to exercise C2 over these domains. This need is substantiated not only in doctrinal documents like the AFFOC, but by senior military leaders including the Air Force Chief of Staff who has made enhancing MDC2 one of his personal focus areas as Chief of Staff saying:

“Achieving a military advantage will depend on harnessing the vast amount of information our sensors can generate, fusing it quickly into decision-quality information, and creating effects simultaneously from any domain, region or command anywhere in the world. This is why my third focus area is enhancing multi-domain command and control.[25]”

It's clear that MDC2 is a vital area for C2 research with immediate applications for military organizations operating in the modern world. The goal of this chapter is to identify key requirements for MDC2 and offer a possible technical solution with attributes that can meet these requirements. The question will be examined primarily

from a USAF perspective with special focus on cyberspace integration, but without loss of generality for other organizations.

2.3 Definition of Multi-Domain Command and Control

According to the Air Force Chief of Staff's directive, MDC2 is not just commanding and controlling operations in multiple domains at the same time, or even just utilizing capabilities one domain to support operations in another domain. A vision

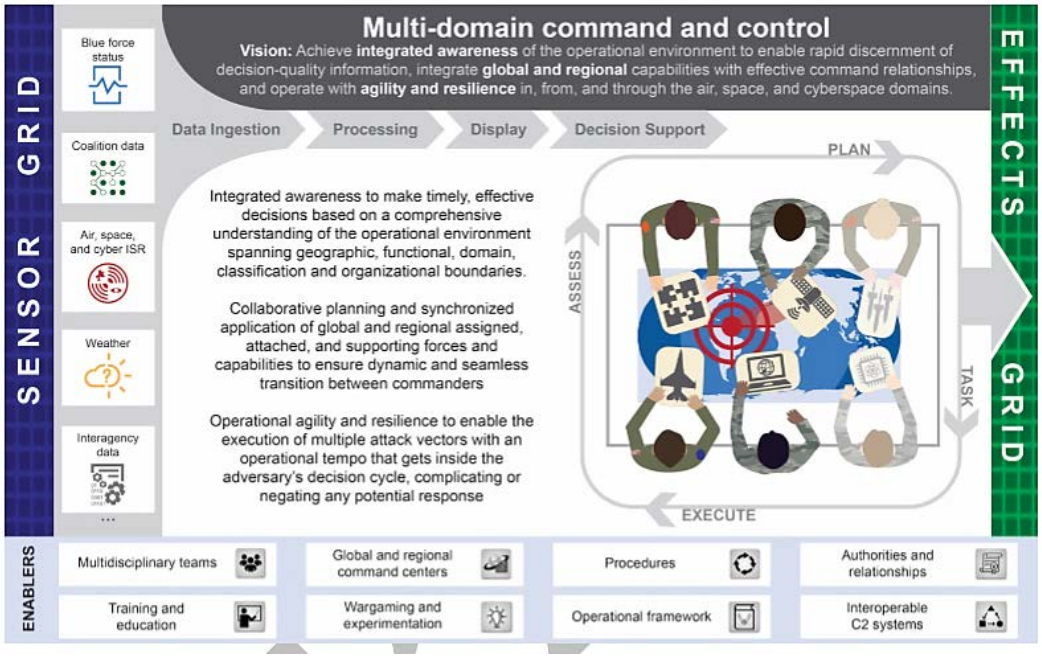

Figure 1: MDC2 Vision [58]

for MDC2 is neatly summarized in Figure 1. MDC2 starts with data collection from multiple sources; this includes Intelligence, Surveillance, and Reconnaissance as well as other data sources such as weather forecasts, maintenance schedules, or other information regarding the status of blue forces. This data must then be processed and displayed effectively in order to enable rapid decision making and produce effects. The key is that this process should be completely domain agnostic. Data from multiple domains should

be collected and processed by experts in those domains, and then presented to decision makers who will select a course of action to produce the effects needed to accomplish their mission. This course of action can include capabilities and assets from multiple domains all working together to produce the needed effects in the battlespace.

Now we can summarize the Air Force's vision for MDC2 as "integrated awareness of the operational environment to enable rapid discernment of decision-quality information, integrate global and regional capabilities with effective command relationships and operate with agility and resilience in, from, and through the air, space, and cyberspace domains." If we look closely at that vision then we can pick out two key requirements:

- Integrated Awareness
- Effective Command Relationships

Without these two attributes, it is impossible to conduct MDC2. Decision makers must have an integrated awareness of all the domains in order to be able to make good decisions, and once the course of action is decided there must be an effective command relationship to execute that course of action. Now we will look at these requirements in detail.

First, we will look at integrated awareness. Integrated awareness refers to the ability for MDC2 planners to have a full picture of the battlespace that integrates all of the operational domains, and have full awareness of capabilities that are available to create effects. Currently, the Air Force and other military services are limited to seeing an operational picture for a single domain at a time without any way of correlating the information with the other domains (there may be some elements of other domains in

these pictures, but full awareness of the domain is not provided). In order to have effective MDC2, we must be able to present a full picture of the battlespace to MDC2 planners with all of the domains seamlessly integrated. Without this full awareness, MDC2 planners may end up over focusing on one domain to the detriment of the others and miss opportunities to create synergistic effects. Achieving this will require new ways of visualizing the operational domains and showcasing the relationships between them, and innovative technical means of presenting this information to the MDC2 planners. It is tempting to try and achieve this by trying to throw all of the data on a screen, but we should be careful to not overwhelm operators with too much information nor should we force one domain to present its data in an inefficient manner in order to compromise with another domain.

One possible solution would be to have different views as required to optimally represent the battlespace for each domain with links from one view to the next as the domains interact. For example, there could be a geospatial view with a traditional map showing all of the air/land assets and targets in theater according to their location, but when you select a target (such as a building) you can see that it has a cyber footprint and follow a link to the cyberspace view. Now you see a network diagram which shows how that target fits into the overall cyberspace picture, how the target can be affected in cyberspace, and has geographic data to link cyberspace targets to geospatial targets. The planners can go back and forth between the geospatial and network views, and effects represented on one view are automatically reflected on the other views. In our example that means a cyber effect against a router in the network view could be reflected in the geospatial view by showing that a target can’t communicate. A series of separate but

linked visualizations could provide the integrated awareness we need without resulting in a data deluge.

In addition to presenting a full battlespace picture, we also must present all of the effect producing capabilities available to MDC2 planners, and ensure that they are aware of the constraints on these capabilities as well as the effects that they can produce. This is especially important in non-physical domains such as Cyberspace because cyber capabilities are not likely to be under the direct control of the MDC2 planners. Because of classification constraints, the details of some capabilities will have to be abstracted away, and they will have to coordinate with the owners of these capabilities in order to produce the desired effects. Which ties into the second major concept.

Next, we examine the requirement for effective command relationships. In order to operate in multiple domains, there also needs to be effective command relationships between owners/executors of the capabilities and the MDC2 planners charged with creating effects in theater. Unfortunately, planners in an operation center will not always have direct operational control over the assets/capabilities they need, so the relationship between planners and those with operational control over capabilities/assets needs to be fostered and effective channels of communication maintained. Since the U.S. operates in multiple theaters at once, resources are scarce and many capabilities are high-demand and low-density, so planners in different theaters may desire the same high-demand capability and only one will be able to utilize it. Who will decide which planner gets the capability? How do we ensure that the decision is made quickly and effectively? In addition to conflicts there are concerns specific to each domain that may necessitate a higher approval authority for planners to use the resource. There is not a one-size-fits-all answer

to these questions so they will often have to be dealt with on a case by case basis for each asset/capability. In order to overcome these obstacles, we must have effective lines of communication between planners and the approval authority for the capabilities that they need to use.

Meeting these two requirements should be the goal of any proposed MDC2 solutions. It's also highly unlikely that any single solution, whether it be technical, doctrinal, or organizational, is going to perfectly meet these requirements. Which means that developing effective MDC2 solutions should be an iterative process. We need to be open to experimenting with imperfect solutions and ensure that C2 operators are highly involved in the development process.

2.4 Distinguishing MDC2 from Joint Operations

Now that we have defined MDC2, we can examine it in light of current Joint Operational doctrine because MDC2 and Joint Operations are closely related concepts. At first glance, it may seem like MDC2 is simply Joint C2. However, though they are related, they are indeed different concepts, so it is important that we distinguish the two. According to Joint Publication 3-30, Joint C2:

"encompasses the exercise of authority and direction by a commander over assigned and attached forces to accomplish the mission. Command includes both the authority and responsibility to use resources to accomplish assigned missions. Control is inherent in command. To control is to manage and direct forces and functions consistent with a commander's command authority. Control provides the means for commanders to

maintain freedom of action, delegate authority, direct operations from any location, and integrate and synchronize actions throughout the operational area (OA). [14]"

From the JP 3-30, we can see that Joint C2 is similar to MDC2 because the Joint Forces Commander (JFC) must exercise command and control over forces in multiple domains and ensure integration and synchronization between forces in multiple domains in order to accomplish the mission. However, in practice this has meant that the JFC relies on component commanders who are given distinct objectives for their domain that they must accomplish. While the objectives of each component are integrated so that they all contribute to accomplishing a single goal, the result is that the component commanders primarily operate in a single domain with limited support from other domains. More often than not, forces in different domains operate in parallel rather than being fully integrated. MDC2 seeks to avoid this stove piping of the components, and encourage multi-domain thinking at lower levels. The Joint Forces Air Component Commander (JFACC) should not be limited to the air domain to accomplish the mission, the Joint Force Maritime Component Commander (JFMCC) should not be limited to the sea domain, and the Joint Force Land Component Commander (JFLCC) should not be limited to the land domain. MDC2 breaks down the walls between the domains, and ensures that we are truly operating jointly.

We should not see MDC2 doctrine as something opposed to Joint Operational doctrine as if we can only do one or the other, but rather we should understand MDC2 as a more effective means of accomplishing the objectives set by the JFC. When our forces are stove-piped into their domains the component commander's options are severely limited. However, if each component commander is able utilize

multiple domains then they are given a full range of options, are able to use multiple attack vectors, and ultimately can more effectively accomplish their mission.

2.5 Air Force Specific MDC2 Challenges

For the Air Force, developing MDC2 capability means reevaluating the Air Operation Center (AOC) model, and making the necessary changes in Doctrine, Organization, Training, Materiel, Leadership and Education, Personnel, Facilities and Policy (DOTMLPF-P) in order to eventually transform them into what the AFFOC calls Multi-Domain Operating Centers (MDOC). According to the AFFOC "The MDOC provides the Air Force with the ability to plan, conduct, and assess integrated multidomain operations. [15]" While there is already some integration between domains within the AOC in the form of liaisons from land, sea, space, and cyberspace working alongside the air operators, these liaisons do not currently have the integrated awareness nor the effective command relationships to properly conduct MDC2. Cyberspace in particular is difficult to integrate with the physical domains, so we will look closely at some specific challenges for integrating cyberspace operations within the AOC.

One major obstacle is the 72-hour Air Tasking Order (ATO) cycle. In an AOC, a new ATO is planned, drafted, and approved every 72 hours, and operations within an AOC revolve around this cycle. This is a challenge for cyber planners because of the additional complexities inherent in cyber operations. If an air planner is given a set of targets that need to be destroyed, then they can look at the air assets available to them, select the munitions needed to destroy the target (maybe with the aid of SMEs), and plan the operation. However, in order to produce effects in cyberspace it is not that simple

because “cyber munitions” are inherently more complex than munitions in the physical domains. For a physical munition, we know exactly what the effect is going to be, and we can know with almost absolute certainty that it will always be effective against a target. Whereas, any given cyber munition relies on the target having specific technical vulnerabilities, and these differ widely from target to target. For example, a cyber capability may be effective against Unix systems, but not Windows systems; it may even require a specific version of Unix that is running a vulnerable service. This means it takes a significantly greater amount of preparation to create a cyber effect (known as operational preparation of the environment or OPE). Also, once the cyber munition is used there is a chance that the enemy is able to patch their systems and effectively render that munition obsolete (also known as technical loss). If we can imagine this in the air domain, it would be like having to design and manufacture a new missile for each target we identify, and once the missile is used it may not be effective again even against the same target. This makes it very difficult to plan cyberspace operations within the 72-hour ATO cycle. Especially when the planners have difficulty even knowing what is available to them.

Which brings us to another major challenge, which is that unlike physical domains, cyberspace assets and capabilities are scattered across many different organizations, including many outside of the DoD. This means that the cyber planners in an AOC often do not have direct control over any cyberspace resources at all. Any cyber effects that planners within the AOC need are going to come from outside organizations.

The current force structure can be seen in Figure 2 below. We can see that there is an

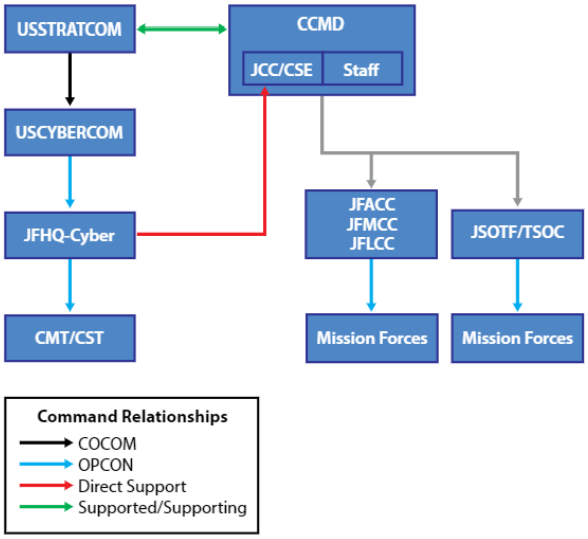

Figure 2: Force Structure [21]

avenue for the JFACC to request cyber effects from Joint Forces Headquarters Cyber (JFHQ-Cyber) which has operational control of the Cyber Mission Teams, but in practice this is incredibly difficult and doesn't usually happen. We can get a first-hand look at this problem from Capt. Michael Dunn who served as a cyber liaison within the AOC at al-Udeid during the summer of 2015, and explained in an interview that when they needed a cyber effect as part of an operation there was essentially one Air Force cyber unit that they went to for support, and they relied on email and personal contacts to request effects rather than a formal C2 system or process. There was also not an effective operational relationship with the cyber component of CENTCOM; there were regular teleconferences that gave AOC planners awareness of what the cyber component was doing at the strategic level, but there was not any awareness at the operational level or an efficient means of requesting cyber effects. How can cyber planners effectively integrate cyber

operations with Air operations if they have no way of knowing what capabilities are available to them?

2.6 Related Work

While MDC2 itself is a relatively new concept, it builds on areas of C2 that are well researched. Especially recent work highlighting the need for decentralized/distributed approaches to C2.

Vassiliou explains the trend towards decentralized C2 paradigms in which you have “edge” organizations making the decisions and sharing information [49]. The key tenets of this “net-centric” approach are:

1. A robustly networked force improves information sharing.
2. Information sharing and collaboration enhance the quality of information and shared situational awareness.
3. Shared situational awareness enables self-synchronization
4. The above dramatically increase mission effectiveness.

Decentralized organization have been proven to make more effective decisions than organizations with traditional hierarchies. When information is shared directly from those who initially ingest/analyze the data to those who need the data to make a decision it has a much lower chance of being misunderstood than if the information was passed through several hierarchical layers before getting where it needs to be. US military doctrine has been moving towards a decentralized C2 structure, but implementation has been slow. As Vassiliou explains:

"Moving from a centralized, hierarchical C2 paradigm to a more decentralized one, even in only a subset of situations, requires considerable effort in changing the command culture. Higher levels of command must become accustomed to delegating and not over-specifying or micromanaging missions. Lower levels must become accustomed to taking initiative and not receiving highly detailed orders."

The key to an effective decentralized C2 paradigm is that all of the edge organizations share the same situational awareness. Information must be freely flowing for this model to be effective. As we saw earlier, the situational awareness needed to make decentralized C2 effective is also required for effective MDC2.

Levchuk and Pattipati investigated whether C2 structures designed by algorithms could be as effective as structures designed by C2 experts [33]. Their results not only showed that optimization algorithms are capable of producing more effective C2 structures, but also that the reason these structures are more effective is because they were more distributed/decentralized architectures. The structure designed by the algorithm effectively distributed engagement loads among all commanders in the simulation, minimized coordination requirements between commanders by giving them more authority and resources, and ultimately it reduced the number of commanders needed per operation. The results of Levchuk and Pattipati's study shows that distributed C2 structures are better able to achieve decision superiority which confirms the trend analysis done by Vassiliou.

In a 2012 paper, Alberts and Vassiliou examine four distinct C2 trends, and analyze their scientific and technological implications [50]. The four trends identified are:

- The extremely broad availability of advanced information and communications technologies that place unprecedented powers of information creation, processing, and distribution in the hands of almost anyone who wants them—friend and foe alike
- The increasing complexity of endeavors as military establishments form coalitions with each other, and partnerships with various civilian agencies and non-governmental organizations
- The rising importance of decentralized, net-enabled approaches to C2
- The data deluge—the unprecedented volumes of raw and processed information with which human actors and C4ISR systems must contend.

MDC2 can be seen as a development resulting from first three trends identified here. The "data deluge" is something that will be compounded by attempts to engage in MDC2 because adding additional domains dramatically increases the amount of data that C2 operators need to shift through in order to make decisions. The key is to identify the information that C2 operators need to do their mission, present that information to them, and cut out everything that is not needed.

General Hostage has also highlighted distributed control as a necessary component for resilient C2 structures [27]. Reinforcing the conclusions reached by the C2 academic community, General Hostage proposes that the Air Force replace its long-held tenant of "centralized control; decentralized execution" with "centralized command, distributed control, and decentralized execution." By this he means that command authority still remains centralized, but direct control is distributed among subordinate units. As the literature shows, this results in better decisions and faster decision making.

However, what is not directly covered in the academic sources is the resiliency that this model also provides. If subordinate units are prepared to exercise control, then operations can continue to run smoothly even if communications with the command authority are disrupted. In a personal interview, General Hostage also highlighted how distributed control can allow us to keep critical command centers stateside rather than being in theater, and thereby reducing the attack surface presented to the adversary.

As the literature shows, decentralization and distribution should be key attributes of any proposed MDC2 structures or technical solutions. We've also identified the need for shared situational awareness and the ability to know what capabilities are available in other domains. This ability to have awareness of all available capabilities can be called "capability transparency", and it is a necessary attribute for effective MDC2. If planners do not know what is available to them from other domains, then they will neglect those capabilities even if they would more effectively accomplish the mission. Right now, there are not any systems for planners to see the non-air capabilities available to them. As Capt. Dunn explained, cyber liaisons in the AOC today rely on informal channels to gain this awareness, and even the awareness they do get is extremely limited [18]. This chapter will now propose a technical solution that can support distributed control and provide planners with capability transparency to enable effective MDC2.

2.7 Proposed Technical Solution

We've identified distributed control and capability transparency as requirements for effective MDC2. Figure 3 shows us key attributes of a technical system to address this

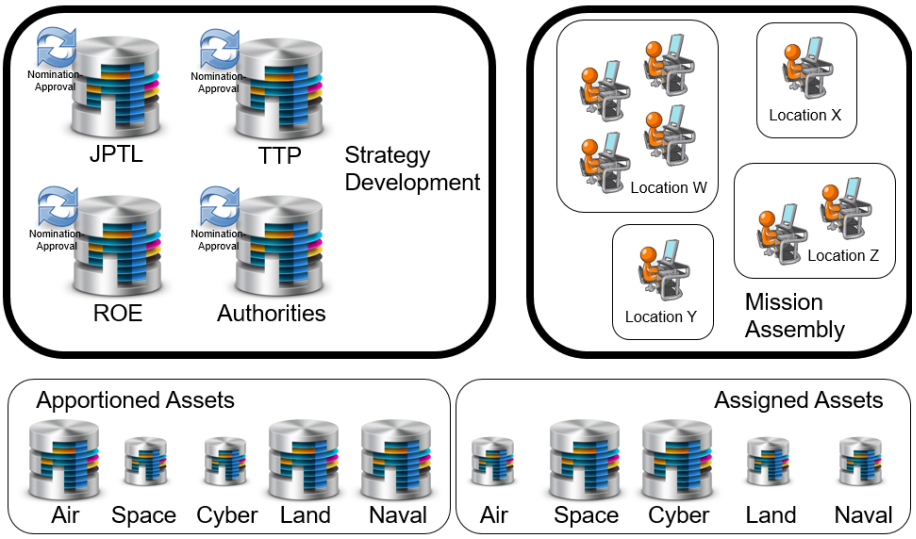

Figure 3: Proposed System Attributes

need. In this system, resources (objectives, intel, target lists, assets, and capabilities) are decomposed into logical units that can be assembled dynamically based on the effects that the planners need to create to accomplish the commander's intent. The Joint Prioritized Target List (JPTL) and the Rules of Engagement (ROE) would both be maintained to keep in line with the JFC's intent, and the data for assets and capabilities would be provided by the owners of those capabilities/assets themselves. Capability owners would input the capabilities they are able to offer, and the attributes of that capability such as the kind of effect it creates, what kinds of targets it's effective against, estimate of preparation time needed, contacts for coordination, or any other information that planners need for decision making. They can then assemble these units together using established formulas to produce synergistic effects; new formulas could be added and old formulas can be changed as Tactics Techniques and Procedures (TTPs) develop as the system is used. All assets are able to be seen by the "mission assemblers" including those directly under their control (Apportioned Assets) and those which they must request

from other organizations (Assigned Assets). This final process is known as “mission assembly” and is the heart and soul of the system, and it’s essentially the process of aligning available assets/capabilities against the targets, constraints, and requirements are given by the JFC.

A system with these attributes would be able to provide cyber planners in the AOC with the capability transparency they need to efficiently integrate cyber effects into the Air Tasking Order. There may be some cyber capabilities that are capable of producing effects almost immediately, but many capabilities will still require a lengthy OPE period. This system would let planners see an estimate of how much time is needed for OPE, so that they can begin the preparations well ahead of time. This likely means that cyber planning really has to start in the target identification step of the planning process, so that when effects are needed the they can be planned within the 72hr ATO timeframe. More importantly, it can be useful for planners in all branches, and brings us one step closer to truly domain-agnostic planning.

Figure 4 gives us a sample application architecture with the primary components necessary to enable effective MDC2. The application itself is simple, reliable, and

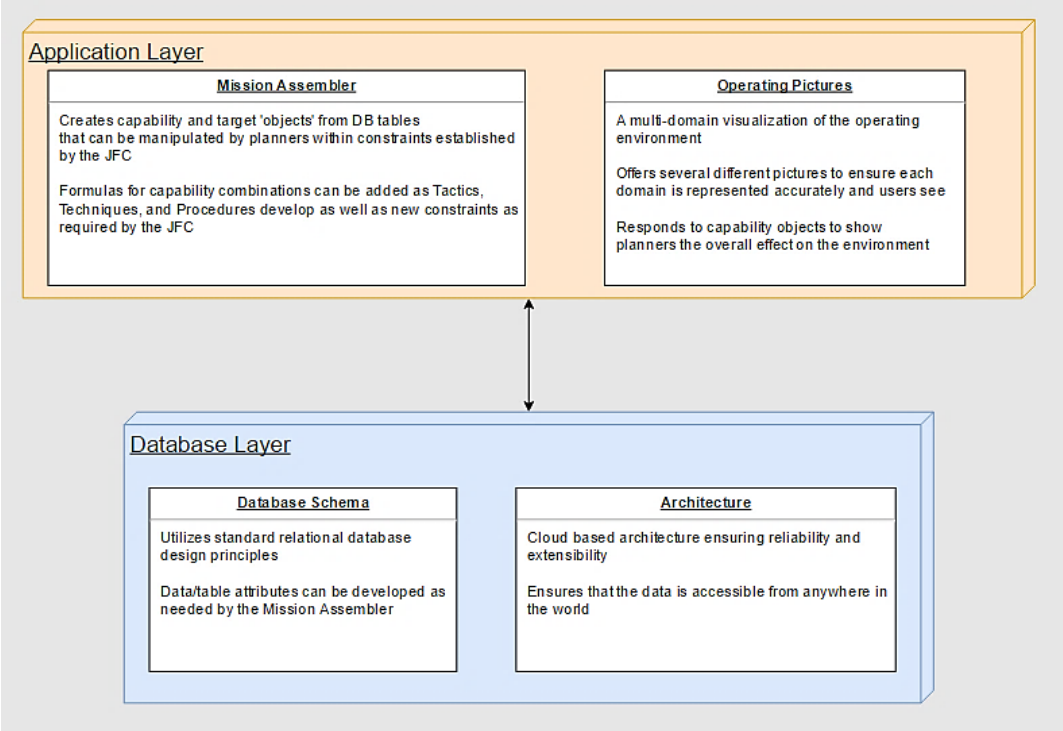

Figure 4: Application Architecture

endlessly extensible. These attributes are absolutely necessary for any technical MDC2 solution since MDC2 doctrine is still in its early development stages. The proposed solution is a two-layered system that will allow capability owners to publish capabilities, targets and constraints to be set by the JFC, and gives MDC2 planners a Mission Assembler that can match capabilities to targets and constraints in order to produce the effects necessary to accomplish their objectives.

The database layer takes input from capability owners and the Joint Staff to provide the application layer with the data necessary to do mission assembly (as explained previously). Targets are identified, constraints established, and capabilities are

published. Then capabilities can be selected based on whether they are effective against a target and meet the established constraints (e.g. lead time or rules of engagement), and if these capabilities aren't under the direct control of the planners then the planners will be able to contact the appropriate authority to request the capability. Then once the mission is assembled, those responsible for execution can then start carrying out the preparation needed to accomplish the mission as planned in this system. Note that the proposed system is primarily a planning solution, so the execution of the assembled mission is not a concern at this point.

The system will be built on a cloud platform, making it easily accessible and resilient. The distributed system architecture also places the system in line with the decentralized C2 paradigms suggested by the latest research because it can be utilized by edge organizations and provides the situational awareness necessary for net-centric decentralized C2. It also enables distributed control by providing a platform that gives planners full situational awareness and the ability to rapidly assemble missions based on commander's intent. Most importantly, since no system is perfect, it's extensible and can be quickly updated as new requirements are identified.

This proposal should be seen as a starting point not the final solution. As mentioned in previous sections, application developers should keep C2 operators involved throughout the entire development process because ultimately the measurement of success is how well the application helps the operators do their job. As features are implemented, the developers can collect feedback from operators and refactor as needed. This iterative development cycle continues as needed to meet the requirements of the operators.

2.7 Future Work and Conclusion

The next step for this research effort is to implement a prototype of the system and gather feedback from subject matter experts. There are also many other questions that need to be answered:

- General Robert Elder has suggested that MDC2 entails the full spectrum of the instruments of power ranging from traditional military assets/capabilities to diplomatic actions. Should non-military effects be part of MDC2? How can they be integrated?
- What does Battle Management look like in MDC2? Should planning systems also be used for Battle Management?
- Cloud computing has many benefits for C2 systems, but how can we ensure that the systems are secure?
- What can be automated? How can we create the most effective human-machine teams?
- How do we resolve conflicts over limited resources (namely cyber capabilities)?
- Who should make decisions regarding intel gain/loss and technical gain/loss for cyber capabilities?
- Is there a civilian equivalent to MDC2? What can military organizations learn from commercial organizations and vice versa?
- How can we foster the culture shift necessary to make distributed MDC2 effective?

All of these questions need to be explored, and there are no easy answers. However, the research effort into these questions could directly impact how military organizations operate in the future.

This chapter defined MDC2 from an Air Force perspective. It compared and contrasted MDC2 with current Joint Operational doctrine and showed how MDC2 is a way to avoid the stove-piping that happens in Joint Operations. It analyzed the current trend towards decentralized C2 structures with distributed control and identified capability transparency as a key requirement for MDC2. Finally, we proposed a technical solution that can provide capability transparency as well as enable effective distributed control. It allows commanders to provide their intent, and then can they can leave it to edge organizations to assemble missions based on this intent. Overall, this is a single step towards a general solution that promotes integrated operations across multiple domains and ultimately achieves decision superiority.

III. Teaming with Silicon Valley to enable Multi-Domain Command and Control

3.1 Introduction

In recent years cyberspace has become a major focus as a new military operational domain alongside land, sea, air, and space (in this context, “domain” refers to a unique environment in which military forces operate). However, being excellent at cyberwarfare is not enough if we cannot effectively coordinate warfighting efforts across all operational domains. General Goldfien, the Air Force Chief of Staff, highlighted this need: “Achieving a military advantage will depend on harnessing the vast amount of information our sensors can generate, fusing it quickly into decision-quality information, and creating effects simultaneously from any domain, region or command anywhere in the world. This is why my third focus area is enhancing multi-domain command and control.” Additionally, developing MDC2 capabilities was identified by Reith et al (2016) as a key research area for operationalizing cyberspace [41]. Though the research for this paper is being done in a U.S. Air Force context (thus the focus is on air, space, and cyberspace), the concepts are applicable to any organization. This paper will define MDC2, pull out some of the key problems that may have technical solutions, and show how integrating DevOps, cloud-native platforms, and microservices-based applications into the AOC can help solve these problems.

3.2 Definition of Multi-Domain Command and Control

MDC2 is difficult to define concisely. General Goldfien offers the following: “Multi-domain battle is more than the ability to work in multiple domains. We already do this quite effectively in today's Air Operations Centers. It is also more than operations in

one domain supporting or complementing operations in another domain. An advanced multi-domain operating concept (CONOPS) will exploit current and new capabilities as well as integrate joint and coalition capabilities across all military operations. It will allow us to both see more opportunities and generate more options for our nation's leaders. Nominally, as either the Joint Forces Air Component Commander or Joint Forces Commander facilitating a campaign, we will be responsible for the delivery and articulation of joint fires. This responsibility mandates that we master MDC2. [26]"

MDC2 begins with data collection from multiple sources; this includes intelligence, surveillance, and reconnaissance as well as other data sources such as weather forecasts, maintenance schedules, or any other information pertinent to decision makers. This data must then be processed and displayed effectively in order to enable rapid decision making and produce effects. The key is that this process should be completely domain agnostic. Data from multiple domains should be collected and processed by intelligence experts in those domains, and then presented to decision makers who will select a course of action to produce the effects needed to accomplish their mission. This course of action includes capabilities and assets from multiple domains all working together to produce the desired effects in the battlespace. In short, the goal of MDC2 is to be able to efficiently utilize the unique capabilities/attributes of land, sea, air, space, and cyberspace forces in an integrated manner to provide the most options to friendly leadership, ensure military forces are being used most effectively, and as a result presenting the enemy with multiple dilemmas. This is not an entirely new goal, but should be seen as further evolution of joint operations.

3.3 Identifying Problems and Requirements

Now that we've defined MDC2, we can start looking at some major problems that need to be solved. For the Air Force, command and control (C2) functions are primarily based in regional AOCs, so the first step to conducting MDC2 is to ensure that the AOCs are equipped to plan and execute multi-domain operations. Today there are personnel from air, space, and cyberspace career fields working together, and there has been real progress made in multi-domain operations. However, there are still "silos of excellence" where collaboration is limited and overall effective MDC2 is limited to individual missions rather than being the normal flow of operations. The main obstacles to conducting MDC2 in the AOC today include:

- Target development is done with kinetic operations in mind which limits non-kinetic options during the planning phases
- There is no way of providing planners with a multi-domain operational picture (lack of integrated awareness)
- There is no way for planners to be able to view all of the capabilities available to them from multiple-domains
- Though MDC2 happens because of SMEs that are able to coordinate efforts, there is not a clearly defined workflow for MDC2 to make it part of the ordinary workflow in the AOC
- Current software cannot adapt to organizational/process changes in a timely fashion

These problems have the potential to be solved with new software tools. In the past, new software has been released as part of major updates of the entire system (both hardware and software). Which, for many reasons, means that it often takes nearly a

decade to implement new software in the AOC, and if new requirements are identified it can be months or even years before the changes are made. This in turn results in software that can't quickly adapt if the users find it to be counterintuitive, it lacks data sources that they need, or their workflow has changed since the product was released. The result of all of this is that when we look at the AOC today, we often find operators using spreadsheets for tasks rather than tools designed for their work. Additionally, AOC users often express frustration that the tools they use don't talk to each other which results in duplication of effort. Now we see that in order to develop effective MDC2 applications/tools we need a practice with the following attributes:

- Software needs to be developed in a timely manner (~6 months or less)
- Applications need to be designed so that data can be easily shared between themselves and new data feeds should be able to be quickly added as needed
- User feedback needs to be collected for the entire application lifecycle and inform further development
- New features and/or changes to the application need to be implemented rapidly without breaking the application and/or system
- Software needs to be secure and reliable

The technical solution for MDC2 won't be an application or even a collection of applications. The solution will be a system architecture and software development process that quickly releases applications and tools to the war fighter that can be adapted as C2 operators learn how to more effectively conduct MDC2. Instead of reinventing the wheel, we can look at the private sector and see what technologies and practices are out there that the Air Force can adopt.

3.4 DevOps and Microservices

In the private sector, one of the more successful software development practices is known as DevOps [13]. The goal of DevOps is to closely integrate software development with operation of the production environment. This is done through a collection of automated tools for continuous testing, integration, and deployment as well as infrastructure that can be easily configured through scripts (known as “infrastructure as code”). DevOps practices and tools were specifically developed to build on the success of agile development practices to enable more frequent deployment of software into the operational environment which ultimately leads to faster release times. Though DevOps is about uniting development and IT in order to promote more frequent deployment cycles, the result is that users are able to give their feedback and see changes implemented in a matter of weeks or months. The increased deployment cycle can also result in a more secure product than traditional software acquisition methods.

The most common objection raised to DevOps and other agile software development methodologies is that security is compromised in favor of faster development time. However, the question itself seems to have an underlying assumption that traditional software acquisition/development has produced secure software which is not a safe assumption. DevOps toolsets include automated software scanning tools to ensure developers are using best practices secure software design (e.g. ThreadFix, SonarQube, and/or OWASP). Frequent deployment of new code can also be a security feature itself. The frequent deployment cycles produced by DevOps allow for quick fixes to security vulnerabilities when they are uncovered. Additionally, there has been momentum recently to include cyber security professionals in the release cycle and

DevOps tools (e.g. SDElements) designed specifically to promote collaboration and communication between cyber security, development, and IT operation teams. DevOps can be just as security-focused as any other software development cycle, and the more frequent deployment of code means that security vulnerabilities can be identified and fixed in a timely manner.

DevOps is often tied to another trend in software development, and that's building applications as a distributed system of independent microservices such that new microservices can be added or changed without affecting the rest of the codebase. The following figure shows how a simple application looks when implemented using both microservices and monolithic architectures:

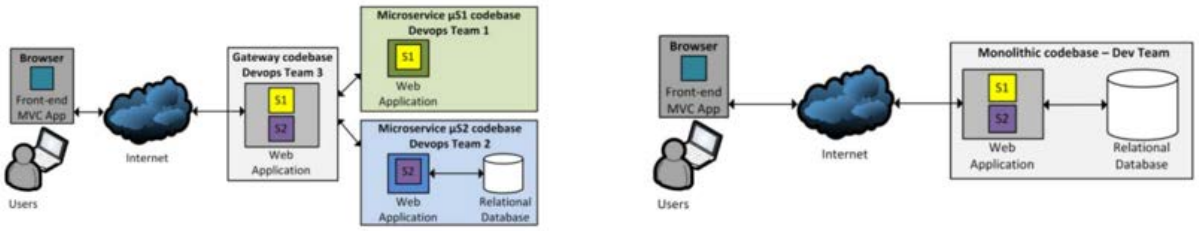

Figure 5: Microservices Architecture vs. Monolithic Architecture [51]

The application on the left is implemented using microservices, and we can see that it contains three distinct, relatively small codebases, compared to the single large codebase of the monolithic application. Each of these "microservices" do their specific tasks and they communicate with other services using an API. This modular approach provides several advantages. First, it allows the developers to make changes to their codebase without having to worry about the changes affecting the other codebases which

leads to more frequent deployments. Second, it can make high-availability more effective by allowing each service to be scale as needed as opposed to having to scale the entire monolithic application [42]. Finally, if the users end up requiring additional services, then these can be added without changing the entire codebase. These advantages, however, come at the cost of increased complexity up front. Villamizar et al explain: "microservices introduce many problems of distributed systems (failures, timeouts, distributed transactions, data federation, responsibility assignments, etc.) which make the development process more complex in some areas. [51]" For small/simple applications it may not make sense to use microservices. However, for larger, complex C2 applications that need to be adaptable, this additional complexity up front is worth the increased agility.

Bringing this all together, the table below shows how DevOps meets the identified requirements for MDC2 application development:

Table 1: MDC2 Requirements and DevOps

DevOps Answers to MDC2 Requirements	
MDC2 Application Development Requirement	DevOps Answer
Rapid Software Release	Designed for reduced development timelines
Data Sharing	Additional data pipelines can be added as microservices and/or application programming interfaces
User Feedback Informs Development	Development is iterative based on user feedback for the entire lifecycle of the application
Frequent Changes and Updates	Designed to enable frequent deployment of new code
Security and Reliability	DevOps toolsets ensure secure software design principles are followed

We see that DevOps along with microservices based application development provides the attributes identified as necessary to develop effective MDC2 applications, but can

these practices be implemented in military organizations? Does this approach produce better results than traditional military software acquisition?

AFLCMC and DIUx are attempting just this. They utilize an open source cloud-native platform-as-a-service known as Cloud Foundry (which can be run in a commercial or an on-premise cloud) along with several other continuous integration and automated security scanning tools as a DevOps tool-chain to build and operate applications for the AOC. Their work provides a case study to see if DevOps can be effectively implemented in military organizations that have been attached to traditional software development practices.

3.5 Analysis and Preliminary Results

The DevOps platform being used by AFLCMC/DIUx has built-in high availability to ensure reliability of the applications being run on it as well as support for a nearly infinite number of security tools and/or services. It should be noted that a "cloud-native platform" does not necessarily mean that the platform exists in a commercial cloud managed by an outside entity because they can be just as easily run in an on-premises cloud managed by the AOC itself with reach back to the development teams in the United States. An on-premises cloud/data center would allow the AOC to use their C2 software even in a contested environment, but also provide the agility inherent in cloud-native platforms/infrastructure. The AOC is able to get the best of both worlds in this sense: the flexibility of cloud-native architecture, and the control/security of traditional data centers.

The application developers also took advantage of microservices concepts in their implementation. There are some dependencies between some of the modules/services

because in early development stages these were implemented as a single module/service but separated later on. However, during the development history new microservices have been added to the application as distinct codebases without impacting the other modules. The ability to add microservices later on was necessary to release the product quickly, and it allowed the users to start using the application as a "minimum viable product" while more advanced/luxury features were implemented later. One major issue with traditional military acquisitions is that the scope becomes unmanageable because all requirements are expected to be met at delivery, but with microservices the scope can start small and be increased as needed during the iterative development/deployment cycle.

At the time of writing, DIUx and AFLCMC had released one application for production and daily use in the CAOC at Al-Udeid Air Base. This application is a tanker planning tool used to plan all of the air refueling sorties in the AOC. The investment for the project was $1.5M, and development began in November 2016, and the minimal viable product was released in March 2017. The user feedback was overwhelmingly positive, and the task of planning went from taking three people 6-8 hours to two people taking 4 hours with the application. Work on the application continues, and AFLCMC/DIUx is able to release a new update each week to production using their DevOps toolchain. In contrast, AOC 10.2 (the project to update AOC systems using traditional acquisitions methods) was nearing a decade of development, and the cost by the end was $745 million [29]. This is one data point showing the success of DevOps over traditional military software acquisitions.

3.6 Future Work and Conclusion

More work is needed to continue to evaluate DevOps and microservices as the paradigm for military software development, and specifically how it will impact the Air Force's C2 systems. Future questions include:

- Do microservices enhance security by incentivizing regular updates or does the additional complexity provide a larger attack surface?
- What are the gaps in current DevOps toolsets?
- Will a collection of small, independent, cloud-based applications, developed iteratively with DevOps function better than the large system packages acquired with traditional methods or is DevOps only successful on a small scale?
- DevOps often relies on cloud-based platforms, how can we ensure the security of these platforms?
- Military software must meet rigid security/accreditation requirements; can DevOps become "SecDevOps"?

In summary, MDC2 is absolutely necessary to make the most of cyberwarfare capabilities and ensure that cyberspace functions as an operational domain equivalent to air, land, and sea. As a result, effective software is needed to manage the complexities involved in multi-domain operations. The success of AFLCMC/DIUx shows that DevOps can be implemented in military organizations, and that it appears to produce high quality software much faster than traditional methods and is able to adapt to changing needs of the war-fighter.

IV. Implementing DevOps in the U.S. Air Force: A Case Study of the AOC Pathfinder Project

4.1 Introduction

The U.S. Air Force (USAF) has been putting a considerable amount of time and money into improving its command and control (C2) systems, concepts, and processes. Many USAF leaders have recognized the need for more agile command and control, and this can be seen in documents such as the Air Force Future Operating Concept (AFFOC) which makes operational agility the central idea of how the USAF will operate in the year 2035 [15]. In the AFFOC, operational agility is defined to be: “the ability to rapidly generate—and shift among—multiple solutions for a given challenge.” Because the solution(s) for a given challenge could come from any operational do-main, in order to fully achieve operational agility in modern environments it is necessary to develop integrated MDC2 capabilities.

New software is a major part of meeting these requirements. However, new software is ordinarily re-leased as part of a major update of the entire system baseline (both hardware and software); which, for many reasons, means that it often takes nearly a decade to implement new software in the Air Operating Center (AOC), and if new requirements are identified it can be months or even years before the changes are made. The result is software that can’t quickly be changed if the users find it counterintuitive, it lacks data sources that they need, or their workflow has changed since the product was released. We see that in order to develop effective MDC2 applications/tools the Air Force needs:

- Software to be developed in a timely manner (~6 months or less)
- Applications to be designed so that data can be easily shared between themselves and new data feeds should be able to be quickly added as needed
- User feedback to be collected during the entire application lifecycle and inform further development
- New features and/or changes to the application to be implemented rapidly without breaking the application and/or system
- Assurance that the software is secure and reliable

However, the effort to update the information systems in the AOC using traditional military software development/acquisition practices was over budget and approaching nearly a decade of work without any value being released to the users [29]. As a result, the U.S. Air Force decided to cancel the contract and stand up an experimental unit designated "AOC Pathfinder" to develop new software for the AOC using established agile software development and DevOps best practices.

DevOps (short for Development Operations) is a philosophy or strategy that leverages a set of automated tools and practices in order to integrate software development and information technology operations teams with the goal of increasing release frequency and reliability of software. DevOps takes advantage of tools that are designed to automate as much of development process as possible as well as concepts such as Infrastructure as Code (IaC) which allows the operations teams manage infrastructure configurations as developers would manage source code and microservices architecture which implements an application as a package of small, independent services [2,3,56]. These practices are well established in the private sector, but the pathfinder

project had to implement DevOps in a way that also satisfies the rigid security and reliability requirements demanded by the Air Force.

4.2 Risks

4.2.1 Personnel

Military software developers exist in limited numbers, but they are generally not familiar with culture, tools, and processes necessary for effective agile software development and DevOps, so every government-employed developer (and platform operator) needed to learn agile/DevOps methodology as well as the technical back-ground of the specific application they would be working on. Additionally, all the recruited developers were pulled from other units on a temporary basis, so they would have to leave the project after 3-6 months. The need for education plus the high turnover posed a major risk to the success of the project. People needed to be on-boarded and educated quickly, and they needed to be able to leave seamlessly without affecting the lifecycle of their projects.

4.2.2 Security and Reliability

All new technology in the Air Force needs to be accredited before it can be operated, and this process can take months if not years. The Air Force has been operating with the unspoken assumption that a lengthy security review process after the software has been developed is needed to ensure that the product is secure. This process required to approve new software has been a major obstacle for Air Force software development teams when trying to implement agile practices. AOC Pathfinder was especially challenging because the applications would have to run in a classified production

environment. Successfully implementing DevOps in this context would require a radical culture shift.

4.2.3 Multiple Environments

The development is done in an unclassified environment on a regular commercial network, but the production environment is classified, so a process needed to be established to move code from the unclassified to classified environment. Initially, AOC Pathfinder had to rely on another government organization's commercial and .mil environments to get its code approved and into production which created more complexity for the platform teams. AOC Pathfinder has since established its own DevOps environment, so only two environments need to be maintained at the time of writing.

4.3 Pathfinder Approach

After AOC Pathfinder was stood up, its leadership decided which software/tools would be targeted first based on what was feasible as a proof of concept and met a genuine need then they recruited a team consisting product managers, designers, and engineers to build the first product using the process outlined in Figure 6. In order to mitigate the risks outlined in section two, AOC Pathfinder utilized Pair Programming, a DevOps process focused on continuous security, and a toolset to help manage the

development testing and deployment in multiple environments.

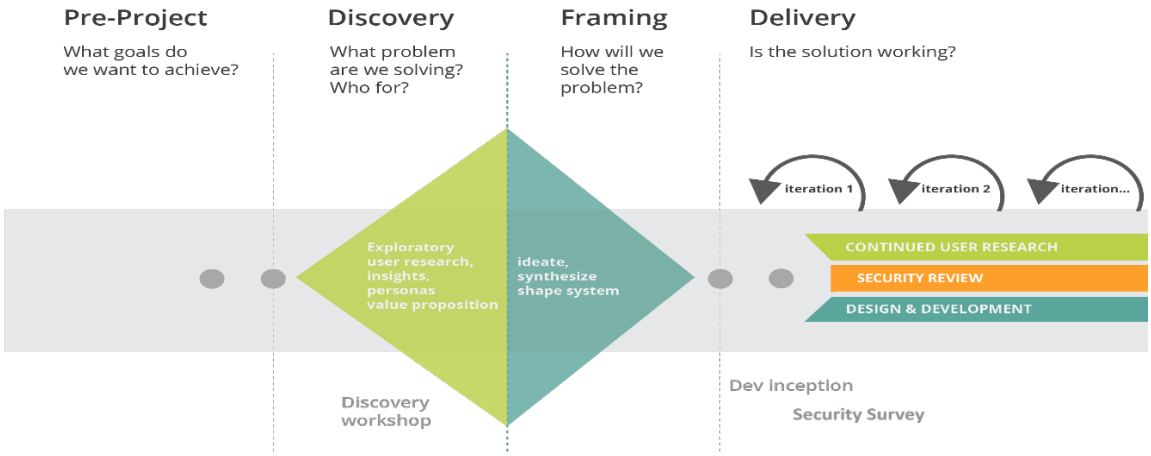

Figure 6: AOC Pathfinder Process [12]

4.3.1 Pair Programming

The Air Force itself did not have development teams with knowledge and culture required for successful agile software development, so we teamed with Pivotal Labs and utilized pairing to equip our personnel with the knowledge and culture of agile software development. Every government employee was teamed with an equivalent Pivotal employee (government developer with Pivotal developer, government designer with Pivotal designer, government product manager with Pivotal product manager, etc.) and given on the job training. Special “pairing stations” are set up so that a pair can share a computer while each person has their own monitor, mouse, and keyboard. One person in the pair is “driving” while the other is observing, learning, and correcting mistakes, and at any time they can switch roles. This process allows personnel to be onboarded to their project and start being productive almost immediately (2-4 weeks depending on individual ability), and ensures that no one person becomes a single point of failure. As more government employees become experienced with agile software development and

DevOps, the project is able to rely less on Pivotal employees, and more on government personnel.

4.3.2 SecDevOps

If DevOps is the term used to describe tools and practices that closely integrate the development and IT operations teams, then we can call the tools and practices intended to integrate security, development, and operations teams "SecDevOps" [35]. The AOC Pathfinder used two primary lines of effort in order to be able to deploy code frequently and at the same time meet the security and accreditation requirements.

First, it followed what is known as the "ATO-in-a-day" concept, an "ATO" is also known as "approval to operate". This entails accrediting the platform that the application code will run in, and accrediting the pipline that deploys new code to the production environment. Once trust is established in the platform and the pipeline, then the new code that's developed inherits this trust and re-quires little time to be approved before being deployed.

The second line of effort was to ensure that security became a continuous practice rather than being tacked on at the end. Initially, security scanning tools helped ensure that secure software design principles were being followed, but documentation was largely pushed until the time came to get approval for the first production release, so the initial release of an application would be delayed several weeks. To remedy this, AOC Pathfinder decided to add SDElements to the toolset to help the security team continuously create security task for the development and platform teams, track the projects, and document them in order to ensure that when the application is ready to be

released for the first time all security tasks and documentation needed by the approving official is also completed. This continuous security process with SDElements is shown in Figure 7.

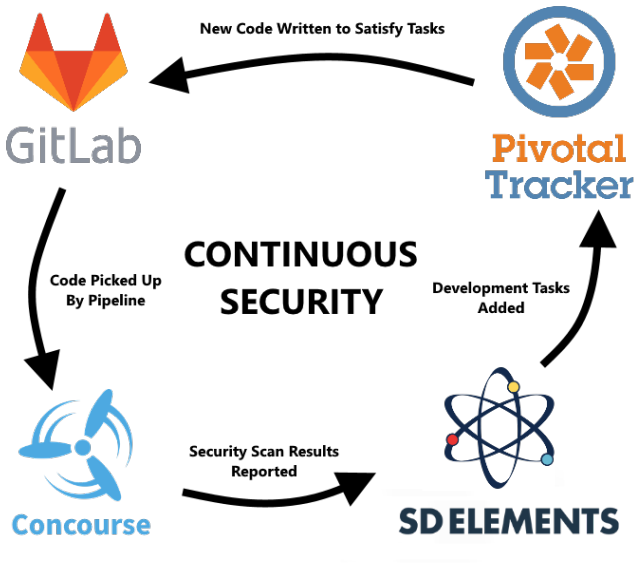

Figure 7: Continuous Security [47]

4.3.3 SecDevOps Toolset

All of the applications developed by AOC Pathfinder run inside Pivotal Cloud Foundry (PCF). PCF is a cloud-native Platform-as-a-Service that we ran on a commercial cloud for the development/testing environment and an on-premise cloud for the production environment. It provides the applications with any services they require as well as scaling, load balancing, and high-availability to ensure smooth operation.

Pivotal Tracker is the tool used for agile lifecycle management (ALM). It is used to track all tasks that need to be done as “stories” as well as metrics such as velocity

(number of stories completed over time) to help the product owner/manager gauge the health of a project.

GitLab is the code repository in all of the environments.

Concourse is the primary Continuous Integration (CI) tool which manages all of the pipelines and ensures that code goes through all the automated scans before being accepted. There are several automated scanning tools to ensure code quality and security notably:

- Lint-js-java: A tool used for the unit and integration tests for a given application. It runs at the very start of the pipeline to determine if any code regressions have occurred.
- Journeys: An acceptance test tool that walks through the application as if it were the end user. Ensures that the application functions as expected with the new code.
- OWASP: A dependency scanner that checks all of the code dependencies for known vulnerabilities
- Fortify: Static code analysis tool that checks to ensure the code follows best practices for se-cure software design
- SonarQube: Scanning tool that checks for code quality as well as security vulnerabilities
- ThreadFix: Consolidates the results from the vulnerability scanning tools and builds re-ports
- SDElements: Introduced later in the project timeline to manage the security/accreditation for each application. Takes security scans as input, creates

specific tasks based on the security policy baseline, integrates with the agile lifecycle management tool (Pivotal Tracker in our case), and automatically creates reports that meet the documentation requirements needed for approval

This toolset is designed to enable frequent deployments of new code into multiple environments, and at the same time ensure that all the documentation required for accreditation is ready by the time an application is ready for release into production.

4.4 Results

At the time of writing, the pathfinder project has success-fully released three applications into production using the concepts, processes and toolset presented in this article. Each of these applications were released as a Minimally Viable Product (MVP) three to four months after development started. The first application was released at the end of March 2017, and by June 2017 the users had completely abandoned their old tools/process in favor of the new application. The other two were released in November 2017, but have not yet reached the point that they have completely replaced the old tools/processes. None of the products were satisfactory at the time of initial release because the goal was to get something in the hands of users in order to gather feedback and improve the product. Users are able to use the applications while at the same time use their standard tools/processes when needed until the product is able to completely replace their old tools/process.

That being said, the development teams continuously gather feedback from 100% of the users who touch the applications, and all the users are satisfied with the project because they know that their concerns will eventually be addressed. The SecDevOps

process enables up-dates to be released into the production environment on a weekly basis, so pressing issues can be addressed quickly.

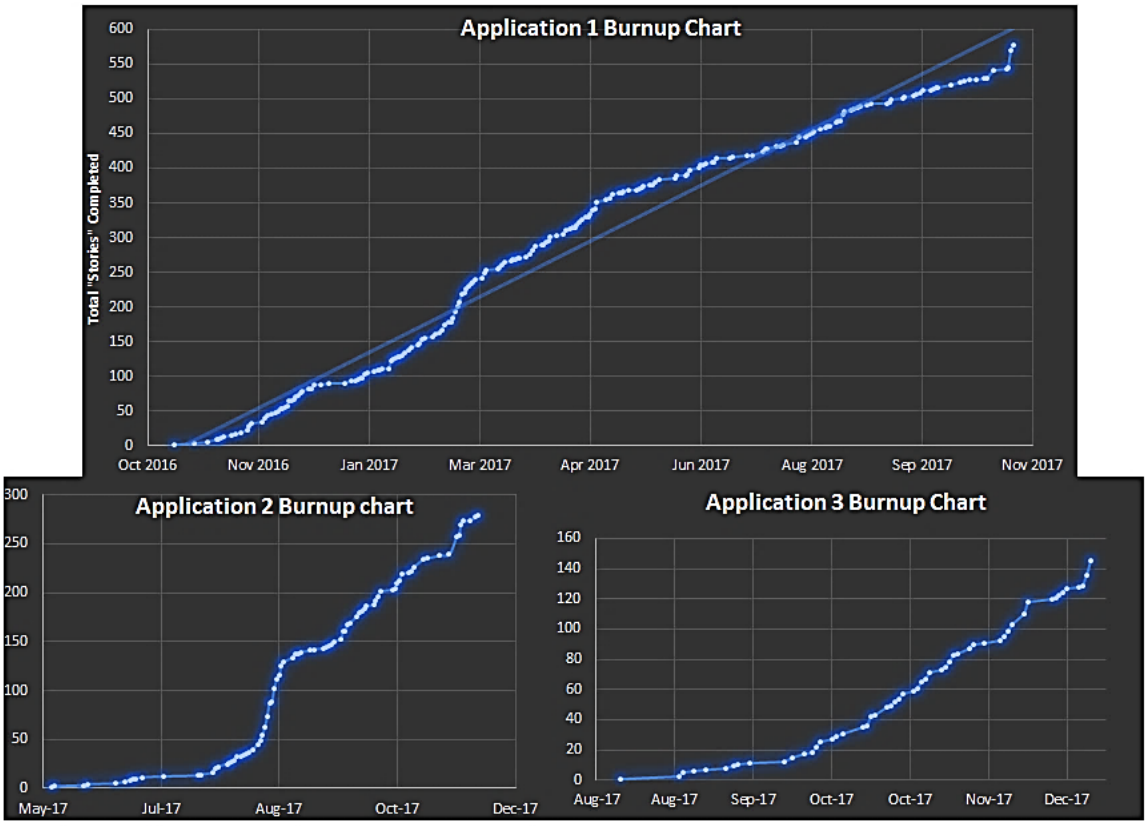

Figure 8: Burnup Charts

The practice of paired programming allowed the team to develop the products at a consistent rate even with the heavy training/education requirements and frequent turnover of personnel. Figure 8 contains burnup charts showing the number of tasks or “stories” completed over time for the first three applications released. The significance of Figure 3 is that the development teams were able to maintain a consistent development rate for even with the three to six-month turnover rate. Application 1 shows this most clearly because work on it has been going on for more than a year. The other two have only been in development for six months or less, so they haven’t yet had to replace a significant

number of the developers. It should be noted that the spike in August 2017 for Application 2 is due to the development team taking a two-week trip to the AOC where they worked much longer hours than normal (you can also see a brief dip where the team was recovering from this trip), but after this sprint the development progresses at a normal rate. The bottom line from this data is that AOC Pathfinder was able to teach new developers the tools, languages, culture, and processes they needed to know in order to be successful without significantly impacting ongoing development.

Though AOC Pathfinder is still in its early stages, the success so far shows that organizations with large beurocratic obstacles and stringent software security and accreditation requirements are able to use (Sec)DevOps processes and toolsets to produce software that meets security and accreditation requirements and ultimately satisfies their customers

4.5 Lessons Learned

Three key lessons have been learned over the course of the AOC Pathfinder Project.

First, due to different uses of the term “operations” in the military, many people think that DevOps refers to developers and users working together because the users in this case are often known as “operators”, so there sometimes can be an overemphasis on the development team over the platform operations team, so it’s crucial for leadership to understand the importance of IT operations in DevOps. The pipelines, staging and production environments all need to be maintained by an effective platform team to ensure smooth operations.

Second, a dedicated security team that is integrated with the development and operations teams is needed to ensure that the project is meeting all security requirements. Through the continuous security process, the security team enforces established security policies and tracks the tasks that need to be completed so that the application can be approved in a timely manner. The security team for AOC Pathfinder also took steps to ensure sensitive project information was being properly handled by the development/platform teams. Integrating continuous security into the DevOps process is necessary for organizations like the U.S. Air Force that have to meet stringent security requirements. If there's an effective security team monitoring the project, then the rapid release cycle of SecDevOps could in fact be a net gain for software security since security flaws can be fixed soon after they are identified.

Finally, working in pairs helps ensure that new people are quickly trained and become productive team members. It also helps ensure that critical project information is distributed across the team, and no one person becomes a single point of failure. In addition to the training benefits, pairing has built in quality assurance because there is always someone reviewing the work being and it also can have a positive impact to morale. However, some personalities may not be conducive to a paired working environment, so it's important to recruit people who are able to work well in this environment.

4.6 Remaining Questions

Now that AOC Pathfinder has shown the success of its SecDevOps process as an experimental project we will have to see if SecDevOps can be institutionalized across an

organization like the Air Force. Some could argue that the success of the project is due to a unique circumstance where the right people were brought together at the right time. Can this process scale and sustain itself long term?

Along similar lines, the applications developed so far are tightly scoped for specific tasks, and are or will be very good at doing those tasks. The long-term vision is that eventually all of the software in the AOC will be replaced by a set of applications built and maintained through the SecDevOps process either by refactoring the legacy application or building a new one from scratch. The open question then is whether a set of cloud-native applications replace the entire AOC software suite. Can the whole function as well as the individual components?

This process so far has worked very well for command and control software which is the primary scope for this project, but there are many other software development efforts in the U. S. Air Force. It's doubtful that the toolset used to build and sustain command and control software will be able to be used for scenarios where hardware and software are tightly coupled such as embedded systems because the agility of the toolset relies on the cloud-platform that operates the applications. AOC Pathfinder was able to release code frequently be-cause Cloud Foundry was trusted platform that was hardware agnostic. Can non-C2 software development teams benefit from some of these practices? How can software that is tightly coupled with hardware be re-leased frequently?

4.7 Conclusion

The example of AOC Pathfinder shows that DevOps does not have to be antithetical to software security and reliability, so companies that have to meet tight security regulations can look at the example of AOC Pathfinder and experiment with similar tools and processes for their software. Though this case-study is most relevant to military organizations and command and control software, the tools and practices are applicable for nearly any set of software applications that can be operated in an on-premises data center or commercial cloud. The toolset listed here accomplished AOC Pathfinder's goals, and their team found these tools to be most useful to them, but there are many different tools available for organizations to experiment with and find what works best for their situation. AOC Pathfinder was ultimately successful because they were able to adopt the proper DevOps toolsets and practices that met their unique requirements.

V. Final Analysis and Conclusion

This chapter presents additional analysis of DevOps, vignettes of the first three applications developed by AOC Pathfinder, answers the research questions in light of the material presented thus far, proposes future research areas, and concludes.

5.1 Significance of Security in DevOps

Cybersecurity is a major concern for C2 systems, especially when classified data is involved. As the previous chapters highlight, security teams should be integrated with DevOps to ensure that the teams are meeting the standards required for accreditation. The team is responsible for operational security of the project itself, this includes password management, offboarding team members who leave the project, controlling access to sensitive project information, etc. However, the approval authority for the development pipeline and platform should be independent to maintain proper checks and balances.

While the frequency of new code being released into production with DevOps is a net gain for security, this is only true for the applications themselves and the platform they operate inside. DevOps by itself will not contribute to the security of the enterprise network, or the security of accounts outside the environment such as email. It may eventually be helpful to consolidate all C2 systems in an operations center into one environment that can be managed, but that also may introduce a reliability problem because the DevOps platform becomes a single point of failure.

5.2 Limitations of DevOps

DevOps requires an environment with flexible infrastructure and software that is hardware agnostic. The infrastructure needs to be able to adapt quickly to changes in the application, so the practices presented in this work will not be as effective for software development projects where software is tied to unique hardware that is not flexible such as embedded systems. Software development may still be able to use agile software development lifecycles, but it will be difficult for those projects to use DevOps toolsets. . DevOps is best suited for developing applications designed to run in an on-premise data center or commercial cloud environment.

Also, DevOps toolsets must be used by people who are willing to embrace agile software development culture and work as a team. When recruiting personnel for AOC Pathfinder, a deciding factor was the ability to work in a team. Especially since training was done primarily through the practice of pairing new team members with experienced team members in the culture, processes, and tools for DevOps.

5.3 Pathfinder Application Vignettes

The first application developed by AOC Pathfinder is known as Tanker Planner Tool (TPT). Tanker planners in the AOC have to ensure that every plane on the ATO had enough fuel to complete its mission using the aerial refueling planes available to them. The task is more complex than simply assigning a new refueler to each plane that needs fuel both because of the limited number of refuelers, and because it is more fuel efficient to reroute a nearby refueler already in the air. TPT manages all the calculations needed for this process (e.g. how much fuel each plane needs, how many refueling missions a

specific refueler can meet, etc.), and can even automatically build a refueling plan on its own with a single click.

The second application is known as Dynamic Targeting Tool (DTT) and it manages the entire workflow for dynamic targeting. Dynamic targeting is when a target of opportunity is identified and action must be taken within hours or minutes rather than the days needed to go through the normal ATO cycle. Each target must be identified, be approved using Law of Armed Conflict (LOAC) criteria, and assets with appropriate munitions must be identified and routed to the target. DTT must interface with all the legacy software that hosts the data needed by the dynamic targeting team, and ensure that targets are tracked through the entire process.

The third application, known as Target Production Manager (TPM), manages the longer targeting cycle known as "deliberate targeting." It was originally envisioned to manage the entire lifecycle of a target from the earliest identification stage to mission analysis. However, the scope of the project was limited to the last stages of target development that is actually done by users in the AOC. Like DTT, it also must interface with legacy software to acquire needed data, and manage the entire workflow of the deliberate targeting team.

5.4 Research Questions

1. How can software enhance the Air Force's ability to conduct MDC2?

Chapter II examined the MDC2 problem and concluded that two key requirements for MDC2 are integrated awareness and effective command relationships and presented some examples of how software can address these requirements. One way to help provide

integrated awareness would be to build an application that can present a multi-domain common operating picture (COP); such an application would enable operators to understand the battlespace for each domain and how the domains are connected. The capability transparency application proposed in that chapter would both help provide integrated awareness by showing planners what capabilities are available to them in a domain-agnostic manner, and it also would provide more effective command relationships by providing planners with the channel they need to use to gain approval to use any given capability.

Also, though AOC Pathfinder was not intended to develop MDC2 software, the target development applications could have features implemented that enhance the Air Force's MDC2 capabilities. For example, the TPM development team could increase the scope of their application in the future and ensure that the proper intelligence is collected during the target development cycle to provide multi-domain options at the end of the cycle rather than developing a target with kinetic effects in mind.

The ultimate goal of MDC2 software is to help operators make better decisions and to make them faster. The above examples are good starting points, and more requirements will be developed as MDC2 experimentation continues.

2. What software development practices have attributes that will aid MDC2 software development and experimentation?

Chapter II highlighted that the Air Force is still experimenting with MDC2, so software needs to be able to adapt to changing user requirements. Chapter III then examined DevOps since the goal of DevOps is to develop new software in months or weeks rather than the years that it often takes using traditional military software

development/acquisition, and DevOps is designed to enable frequent release of new code so software can adapt to the user's changing needs. Chapter III also examined microservices architecture as another means of making MDC2 software more flexible to changing needs. AOC Pathfinder has utilized both of these concepts as it is developing C2 software for the AOC. In the near term, the AOC is where the Air Force conducts MDC2, so the methodology used by Pathfinder should be adopted by early MDC2 developers as a baseline. Then the methodology can be adapted as needed.

Additionally, it should be noted that MDC2 software is not qualitatively different from single-domain C2 software (it is quantitatively different because there is more information and people to manage), so software development practices that work for C2 software can work for MDC2 software. Which is why the Pathfinder project is used as a case study for MDC2 software development. The software development practices that are successful for Pathfinder should also be successful for MDC2 developers.

3. What are the attributes of the AOC Pathfinder Project that enabled it to develop software using DevOps?

Chapter IV examined the AOC Pathfinder project to show that DevOps can be implemented in organizations with a highly regulatory environment, and highlight the attributes that made that project successful. Specifically, the chapter highlighted pairing as a means of training new team members, integrated security teams, the ATO-in-a-day concept, and the DevOps toolset used by Pathfinder to accomplish its goals as the main attributes contributing to the project's success thus far.

In addition to these attributes, the Pathfinder project embraced Silicon Valley's entrepreneurial culture. DevOps is designed to encourage developers to test new

solutions, and be able to quickly recover when something doesn't work, and Pathfinder leadership was risk-tolerant and understood that failure is often part of the process, so developers were able to take risks without fear of reprisal. Grade/Rank was also secondary to knowledge and skill, so a non-commissioned officer could be a project manager leading a team of developers that included commissioned officers if he or she was the best person for the position. The project also actively worked with personnel from private industry in order to develop this culture in the government team members. Actively encouraging this culture is crucial to promote creative problem solving and avoid falling back into old patterns.

5.5 Future Work and Conclusion

5.5.1 Platform Evaluation

As mentioned in the previous chapters, Pivotal Cloud Foundry has been the platform used to manage the infrastructure, deploy, and operate applications by the DevOps teams examined in this work (see Appendix A for an overview of PCF). Though it's been successfully used by AOC Pathfinder, future work should evaluate PCF against competitors such as Amazon's Elastic Beanstalk. How reliable is PCF in the long term compared to the competition? Are there significant performance differences on similar hardware? The success of any DevOps project is intricately tied to the platform used to manage the hardware resources, so research is needed to investigate the advantages and disadvantages of available platform services and determine which one(s) best fit the needs for developing MDC2 software.

5.5.2 Tool Evaluation

This work does not evaluate or advocate for specific tools since the scope of this work is limited to evaluating DevOps as a philosophy and practice in the military for developing C2 software. More research needs to be done to test the specific tools used to make up a DevOps toolset, and evaluate tools against each other since there is a wide variety of commercial and open source tools available designed for DevOps. This includes code scanning tools, pipeline management tools, agile project management tools, etc. While the tools used by AOC Pathfinder have been effective, a rigorous analysis and evaluation of the major available tools is necessary so that DevOps teams can make informed decisions when building their toolsets.

Future research should also examine whether the toolsets should be standardized from project to project or even application to application. For example, a code scanning tool that works well on a Java-based application may not be as effective for an application that is written mostly with Python. However, using different tools may make accreditation more difficult and introduce too many complexities for security teams to manage. How can the toolset flexibility that developers may want be balanced with the standardization that security and accreditation teams would like to have?

5.5.3 DevOps Pipeline Security

Confidence in the security of an application developed and deployed in a DevOps environment is tied to the security of the pipeline that deploys the code. What is the best way to ensure that the code released into the environment is hasn’t been altered from what the trusted developer wrote? How could an adversary get around current protections

and deploy malicious code? If malicious code is released how can it be detected? Research is needed to evaluate current techniques and detect/address any gaps in current pipeline security.

5.5.4 Conclusion

MDC2 is a concept that is still being actively studied by C2 operators and subject matter experts, so the software requirements are going to change as MDC2 concepts are developed and tested. DevOps can be implemented in highly regulated organizations like the U.S. military (or any other highly regulated organization) and is effective at promoting, interaction between users, software developers, and infrastructure operators resulting in rapid response to user feedback which will allow developers to meet the flexibility that MDC2 demands. This work supports the hypothesis that DevOps toolsets and practices are effective at developing C2 software, and can be successfully implemented by the Air Force to enhance its MDC2 capabilities. Therefore, DevOps should be embraced by the Air Force in order to continue enhancing its MDC2 capabilities.

Appendix A. Pivotal Cloud Foundry Overview

PCF is a platform for rapidly developing, deploying, and executing applications. It is a Cloud-Native Platform-as-a-Service (PaaS) that can run on any kind of hardware infrastructure whether it is in a commercial cloud (e.g. Amazon Web Services or Microsoft Azure) or on an on-premises data center using vSphere, Open Stack, etc. It provides the application with any and all services it requires and provides scaling and load balancing to ensure smooth operation. The following diagram gives us a high-level overview of PCF's components that make up the platform:

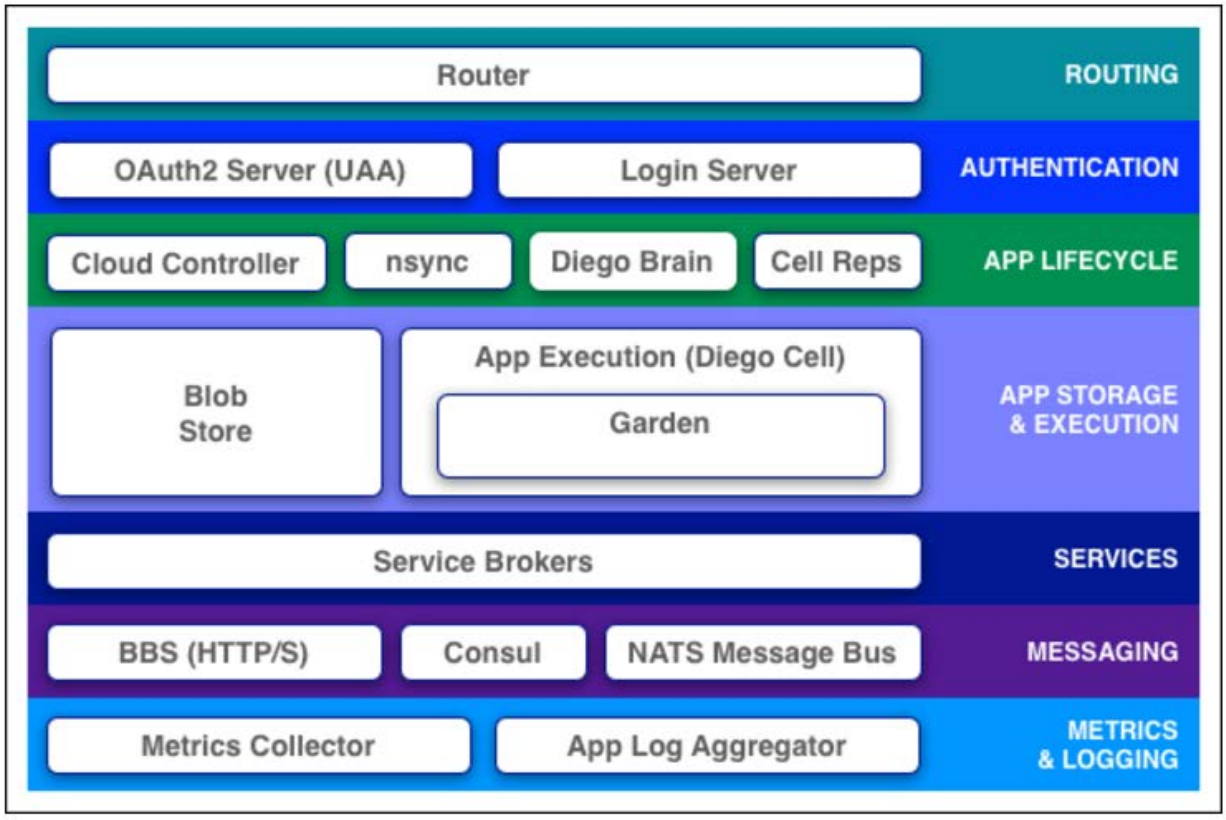

Figure 9: Pivotal Cloud Foundry Architecture [54]

Starting from the top of the above architecture we'll go through the layers to show how PCF functions at a high level:

- In the routing layer, the router handles all incoming traffic for both the hosted applications and the other PCF component VMs
- The authentication layer provides identity management and role-based access control for users
- The application lifecycle layer manages the deployment of applications through the cloud controller, allocates all application tasks/processes to cells/containers through an auction algorithm, constantly monitors the state of all application instances, and starts and stops processes as required
- The application storage and execution layers consist of Diego Cells that host application instances and the Blob Store that hosts large binary files such as application code packages
- The services layer consists of services internal or external to the PCF environment which implement PCF's Service Broker API in order to provide the applications with needed services such as databases or other third-party services
- The messaging layer allows components within PCF to send messages using HTTP/S with Consul being used to store data long term and the Bulletin Board System (BBS) being used to store frequently updated data
- The metrics and logging layer provides a Log Aggregator that streams application logs to the developers
- Finally, all of these components (including hosted applications) can be scaled horizontally by creating more instances and distributed into Availability Zones in order to provide high availability

PCF manages deployed applications through Diego (see Figure 10). Diego is the runtime powering PCF, and it does so by scheduling *tasks* and *long-running processes* (LRPs). A task is an action to be run at most once for a finite amount of time whereas LRPs can run

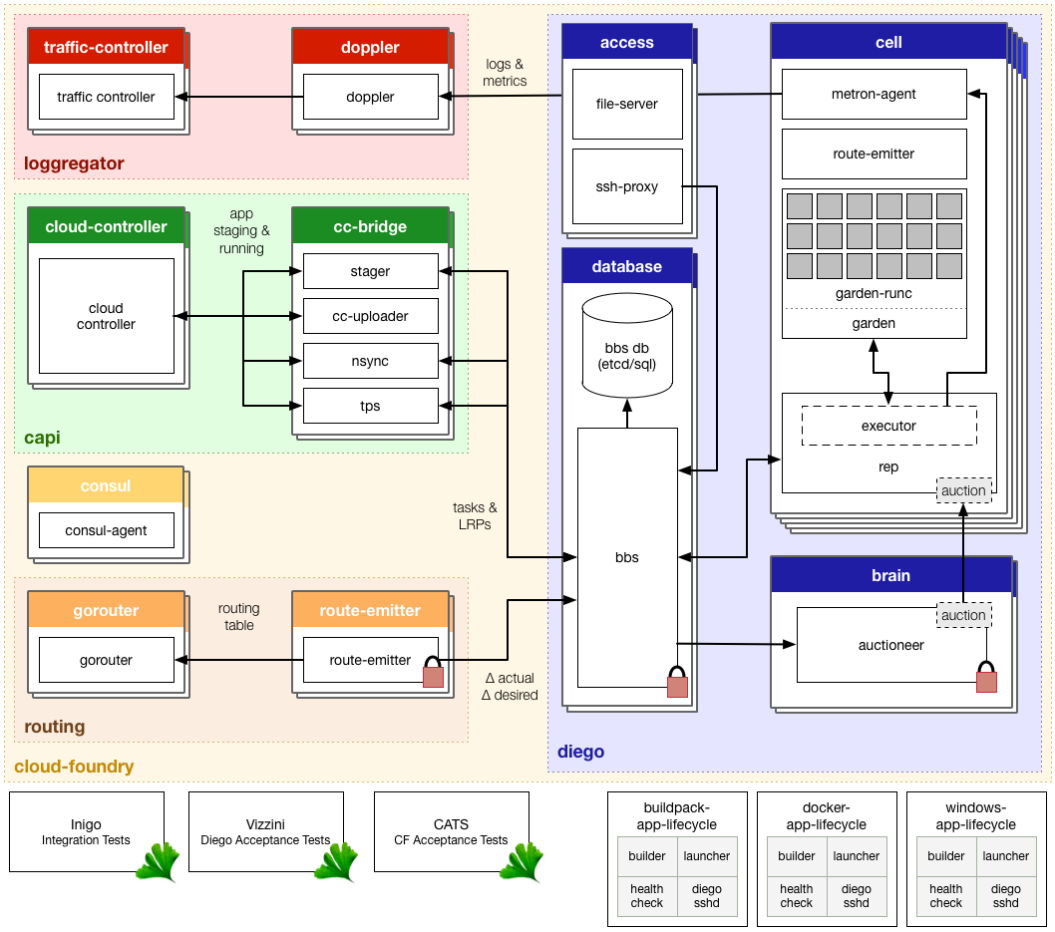

Figure 10 Diego Architecture [54]

continually. It receives requests from the Cloud Controller (which at the time of writing must be translated into tasks/LRPs by the CC-Bridge) and the router. These requests for tasks and LRPs are posted to the BBS. As tasks/LRPs are being posted to the BBS, the Auctioneer is allocating batches of these jobs to Cells for execution. The Auctioneer prioritizes the jobs, and then the cells “bid” with their suitability to complete the jobs, and

finally the Auctioneer assigns jobs based on the following criteria (in order of priority) [6]:

1. Allocate all jobs only to Cells that have the correct software stack to host them, and sufficient resources given their allocation so far during this auction.
2. Allocate LRP instances into Availability Zones that are not already hosting other instances of the same LRP.
3. Within each Availability Zone, allocate LRP instances to run on Cells that are not already hosting other instances of the same LRP.
4. Allocate any job to the Cell that has lightest load, from both the current auction and jobs it has been running already. In other words, distribute the total load evenly across all Cells.

In this way, the Diego Brain allocates all job requests to the Diego Cells hosting the appropriate applications and maximizes efficiency and availability by distributing the load evenly across the Cells. Then each Cell then has a Rep that represents the Cell in the BBS to receive and respond to requests as described above, one or many containers managed by Garden, and a Metron agent that forwards application logs to the developers From start to finish the platform manages all of the overhead so applications run smoothly and developers can focus on building high quality software.

In addition to being a platform for application execution, PCF makes it easy for developers to continuously integrate and deploy their code in accordance with the DevOps philosophy. Once code is “pushed” to PCF through the command line it is running within 30 minutes. Also, PCF supports tools such as Concourse CI or Jenkins

which allow developers to include automated static and dynamic analysis tools for code quality and security assurance (e.g. Sonarqube, Fortify) as well as dependency and vulnerability scans before deploying to the operational PCF environment. Using these automated tools enforce accepted secure software design principles and ensure code is being tested for quality throughout the entire development process. These tools also encourage frequent deployments which allows users to quickly test new features and provide feedback to developers creating the feedback loop that empowers effective DevOps.

Bibliography

[1] S. W. Ambler, “Agile Requirements Best Practices,” pp. 1–8, 2012.

[2] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, “DevOps: Introducing Infrastructure-as-Code,” in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 2017, pp. 497–498.

[3] A. Balalaie, A. Heydarnoori, and P. Jamshidi, *Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture*, vol. 33, no. 3. 2016, pp. 42–52.

[4] L. Bass, R. Holz, P. Rimba, A. B. Tran, and L. Zhu, "Securing a Deployment Pipeline," in *2015 IEEE/ACM 3rd International Workshop on Release Engineering*, 2015, pp. 4–7.

[5] M. Bruza and M. Reith, "AOC Pathfinder: Producing C2 Software Through DevOps," Presented at the *Air Force Institute of Technology*, 2017.

[6] M. Bruza and M. Reith, "Multi-Domain Command and Control: The Need for Capability Transparency," in *22nd International Command and Control Research and Technology Symposium Proceedings*, 2017.

[7] M. Bruza and M. Reith, "Multi-Domain Command and Control: The Need for Capability Transparency," Presented at *22nd International Command and Control Research and Technology Symposium*, 2017.

[8] M. Bruza and M. Reith, "Teaming with Silicon Valley to enable Multi-Domain Command and Control," in *International Conference on Cyber Warfare and Security*, 2018.

[9] M. Bruza, A. Lin, L. Mailloux, and M. Reith, "Implementing DevOps in the U.S. Air Force: A Case Study of the AOC Pathfinder Project," *IEEE Softw.*, 2018. (publication pending)

[10] M. Callanan and A. Spillane, "DevOps: Making It Easy to Do the Right Thing," *IEEE Softw.*, vol. 33, no. 3, 2016.

[11] C. A. Cois, J. Yankel, and A. Connell, "Modern DevOps: Optimizing software development through effective system interactions," in *IEEE International Professional Communication Conference*, 2015, vol. 2015–January.

[12] T. Cruz, "Personal Communication." 2017.

[13] P. Debois, "Agile infrastructure and operations: How infra-gile are you?," in *Proceedings - Agile 2008 Conference*, 2008.

[14] Department of Defense, "Joint Publication 3-30 Command and Control for Joint Air Operations" 2010.

[15] Department of Defense, "Air Force Future Operating Concept," pp. 1–48, 2015.

[16] E. Diel, S. Marczak, and D. S. Cruzes, “Communication challenges and strategies in distributed DevOps,” in *Proceedings - 11th IEEE International Conference on Global Software Engineering, ICGSE 2016*, 2016.

[17] L. Dodd and G. Markham, “7th ICCRTS: “Operationalizing C2 Agility” Paper 014: C2 agility, different models of change and reasoning with time.”

[18] Dunn, M. Personal interview. 2017.

[19] R. Elder, “Personal Communication. 2017.

[20] C. Esposito, A. Castiglione, and K.-K. R. Choo, “Challenges in Delivering Software in the Cloud as Microservices,” *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 10–14, Sep. 2016.

[21] A. Furfaro, T. Gallo, A. Garro, D. Sacca, and A. Tundis, “ResDevOps: A Software Engineering Framework for Achieving Long-Lasting Complex Systems,” in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, 2016, pp. 246–255.

[22] J. Gargan, “The Joint Force Air Component Commander and the Integration of Offensive Cyberspace Effects: Power Projection through Cyberspace,” 2016.

[23] P. Garretson and J. Sawtelle, “A Code of Conduct for Rapidly Adaptive Warfare.”

[24] B. Gilmary, M. Hostage, and L. R. Broadwell, “Resilient Command and Control The Need for Distributed Control,” pp. 38–43, 2014.

[25] W. Gloria, J. Shepard, and J. Ellsworth, “THE CYBER CHRONICLE Cyber Squadron Initiative : More Than Just A Name,” no. 1, 2017.

[26] D. Goldfein, “Enhancing Multi-Domain Command and Control.” 2016.

[27] M. Hostage, “Personal Communication.” 2016.

[28] J. Hukill, “Air Force Command and Control (C2): The Need for Increased Adaptability,” Air Force Res. Inst. Pap., 2012.

[29] V. Insinna, “Air Force cancels Air Operations Center 10.2 contract, starts new pathfinder effort,” *DevenseNews*. [Online]. Available: https://www.defensenews.com/air/2017/07/13/air-force-cancels-air-operations-center-10-2-contract-starts-new-pathfinder-effort/.

[30] H. Kang, M. Le, and S. Tao, “Container and Microservice Driven Design for Cloud Infrastructure DevOps,” in *2016 IEEE International Conference on Cloud Engineering (IC2E)*, 2016, pp. 202–211.

[31] J. Kim *et al.*, “Service provider DevOps for large scale modern network services,” in *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, 2015.

[32] G. M. Levchuk and K. R. Pattipati, “Design of Distributed Command and Control for Collaborative Situation Assessment,” *18th ICCRTS*. 2013.

[33] G. M. Levchuk and K. R. Pattipati, “Design of command and control organizational structures : from years of modeling to empirical validation,” *Proc. 15h Int. Command Control Res. Technol. Symp.*, 2010.

[34] L. E. Lwakatare, P. Kuvaja, and M. Oivo, “Dimensions of DevOps,” in *Lecture Notes in Business Information Processing*, 2015, vol. 212.

[35] V. Mohan and L. Ben Othmane, “SecDevOps: Is it a marketing buzzword? Mapping research on security in DevOps,” in *Proceedings - 2016 11th International Conference on Availability, Reliability and Security, ARES 2016*, 2016.

[36] R. Myers, Personal Communication. 2017.

[37] Pivotal Software, “Pivotal Cloud Foundry Documentation,” 2017.

[38] M. Rajkumar, A. K. Pole, V. S. Adige, and P. Mahanta, “DevOps Culture and its Impact on Cloud Delivery and Software Development,” in *Proceedings - 2016 International Conference on Advances in Computing, Communication and Automation, ICACCA 2016*, 2016.

[39] M. Reith, “Forging tomorrow’s air, space, and cyber war fighters: Recommendations for integration and development,” *Air Sp. Power Journal*, vol. 30, no. 4, pp. 96–107, 2016.

[40] D. Lyle, “Developing Strategic Advantage through C2 of the Global Sensor to Effects Grid,” *Blue Horizons, Air University*. 2016.

[41] M. Reith, S. Pentecost, D. Celebucki, and R. Kaufman, “Operationalizing Cyber: Recommendations for Future Research,” vol. 22, no. 8, pp. 610–620, 2016.

[42] D. Richter, M. Konrad, K. Utecht, and A. Polze, “Highly-Available Applications on Unreliable Infrastructure: Microservice Architectures in Practice,” in *2017*

IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2017, pp. 130–137.

[43] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, and T. Männistö, "DevOps Adoption Benefits and Challenges in Practice: A Case Study," 2016, pp. 590–597.

[44] B. Rueter. "CYBERSPACE INTEGRATION WITHIN THE AIR OPERATIONS CENTER," Air Force Institute of Technology. 2013.

[45] J. Shepherd, "The Cyber Chronicle Butter Bar Brief." 2017.

[46] D. Spinellis, "Being a DevOps Developer," *IEEE Software*, vol. 33, no. 3. 2016.

[47] W. Starr, "Personal Communication." 2017.

[48] A. Uruguay, "Collective C2 in Multinational Civil-Military Operations:A Topological Model for C2 Organizations," in *16th International Command and Control Research and Technology Symposium*, 2016.

[49] M. Vassiliou, "The Evolution Towards Decentralized C2," Inst. Def. Anal., 2010.

[50] M. Vassiliou and D. Alberts, "Operationalizing C2 Agility: Megatrends Reshaping C2 and their Implications for Science and Technology Priorities," *ICCRTS*, 2012.

[51] M. Villamizar *et al.*, "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," in *2015 10th Computing Colombian Conference (10CCC)*, 2015, pp. 583–590.

[52] M. Virmani, "Understanding DevOps & bridging the gap from continuous integration to continuous delivery," in *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, 2015, pp. 78–82.

[53] J. Wettinger, U. Breitenbücher, and F. Leymann, "DevOpSlang - Bridging the gap between development and operations," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8745 LNCS.

[54] D. Winn, *Cloud Foundry*. O'Reilly Media, 2016.

[55] Y. Yue, A. Kalloniatis, and E. Kohn, "A concept for 5th generation operational level military headquarters."

[56] L. Zhu, L. Bass, and G. Champlin-Scharff, "DevOps and Its Practices," *IEEE Softw.*, vol. 33, no. 3, 2016.

[57] “Cross-Domain Synergy in Joint Operations,” 2016.

[58] “Multi-Domain Command and Control Operating Concept,” 2016.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
22-03-2018	Master's Thesis	May 2016 – March 2018

4. TITLE AND SUBTITLE

An Analysis of Multi-Domain Command and Control and the Development of Software Solutions through DevOps Toolsets and Practices

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Bruza, Mason R, Capt, USAF

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, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-18-M-016

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

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

14. ABSTRACT

Multi-Domain Command and Control (MDC2) is the exercise of command and control over forces in multiple operational domains (namely air, land, sea, space, and cyberspace) in order to produce synergistic effects in the battlespace, and enhancing this capability has become a major focus area for the United States Air Force (USAF). In order to meet demands for MDC2 software, solutions need to be acquired and/or developed in a timely manner, information technology infrastructure needs to be adaptable to new software requirements, and user feedback needs to drive iterative updates to fielded software. In commercial organizations, agile software development methodologies and concepts such as DevOps have been implemented to meet these demands. However, the USAF has been slow to adopt modern agile software development concepts such as DevOps in favor of traditional software development lifecycles and large contracts that can go nearly a decade without any value being released to the users. This work explores MDC2 software use cases and aims to show that MDC2 software can be successfully developed using modern agile software development practices in a timely manner. The contributions in this work have been published in two conference papers, and are pending publication in one journal article.

15. SUBJECT TERMS

Multi-Domain Command and Control, MDC2, DevOps, AOC Pathfinder, C2

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Lt Col Mark G. Reith, AFIT/ENG
U	U	U	UU	78	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-3636 x4603 mark.reith@afit.edu

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