Strategies Employed By Project Managers When Adopting Agile DevSecOps To Manage Software Development In The DoD

A Dissertation Presented in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Management

By

Noe Lorona

Department of Doctoral Studies, Colorado Technical University

October 2023

Committee Members

Deane Desper, DBA, Chair

Kelly Hughes, DCS, Committee Member

Jeffrey Butler, PhD, Committee Member

Signature Page

Strategies Employed By Project Managers When Adopting Agile DevSecOps To Manage Software Development In The DoD

Noe Lorona

Approved by Doctoral Committee:

DocuSigned by:
Dr. Deane Desper
C044BEB4FF864C5...

October 22, 2023

Dissertation Chair — Date

DocuSigned by:
Dr. Kelly Hughes
A1CE3AB4D72F49A...

October 22, 2023

Committee Member — Date

DocuSigned by:
Dr. Jeffrey Butler
3C8D88D6CCD146C...

October 22, 2023

Committee Member — Date

Abstract

The problem this qualitative study identified was the lack of established strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. The purpose of this study was to explore the strategies used by project managers to adopt Agile DevSecOps within the DoD software development lifecycle. The research question was, what are the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. The conceptual framework included Agile software development, DevOps culture and practices, DevSecOps security integration, and DoD software initiatives. Additional topics explored were Waterfall methodology, the Software Acquisitions and Practices (SWAP) study, and software DoD factories. The methodology was an exploratory qualitative approach using semi-structured interviews with a purposive snowball sample of 12 DoD contractor project managers. Data was analyzed using thematic analysis. The study revealed strategies and challenges across five key themes: Theme 1: Methodologies and Approaches, Sub Theme 1.1: Enabling Technologies, Theme 2: Quality and Security, Theme 3: Team Dynamics and Trust, Theme 4: Feedback Culture, Theme 5: Challenges and Tradeoffs. This study fills a gap in the literature by understanding real-world Agile DevSecOps implementations in the DoD. It provides insights into strategies used by project managers when adopting Agile DevSecOps within the DoD software development. The research contributes knowledge that can inform best practices for Agile DevSecOps adoption to enhance efficiency, security, and speed of delivery for critical DoD systems.

Keywords: Agile, DevOps, DevSecOps, Department of Defense, Project management, DoD Software Initiatives

Acknowledgments

First and foremost, I want to thank God for granting me the patience, serenity, and good health needed to complete this dissertation. There were many late nights and early mornings throughout this journey, and I am grateful for the strength to persevere. To my beloved wife, words cannot express how much your love and support have meant to me. Thank you for your incredible patience on the weekends when I disappeared for hours on end to work on this dissertation. Your willingness to listen to me vent my frustrations and also celebrate each accomplishment kept me going. I could not have done this without you by my side every step of the way. I am truly blessed to have such a supportive and understanding partner. You were always there to remind me of a light at the end of the tunnel. I look forward to spending those weekends with you now that this dissertation is complete!

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Rapid technological advancements and the growing complexity of software systems have led organizations to adopt new software development and delivery approaches. One such approach is the integration of Agile methodologies and DevSecOps practices to streamline the software development process, enhance collaboration, and ensure faster delivery of high-quality software (Khan et al., 2022; Usman et al., 2022). This dissertation explored the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the Department of Defense (DoD). The research was supported by an extensive literature review that covered various aspects of Agile and DevOps adoption, including tools, culture, and DoD initiatives. The Agile methodology has gained popularity for embracing change and uncertainty, allowing software development teams to deliver high-quality products in a shorter timeframe (Bolhuis et al., 2022). Meanwhile, DevOps practices focus on the seamless integration of development and operations teams, enabling continuous integration, delivery, and deployment of software (Luz et al., 2019). The combination of Agile and DevOps has improved technology organizations' efficiency, reliability, and security (Kim et al., 2016). However, adopting these practices in complex and highly regulated environments, such as the DoD, presented unique challenges (Miller et al., 2022).

The DoD recognized the potential of DevSecOps in delivering customer capabilities faster and less costly. The DoD Enterprise DevSecOps Strategy Guide emphasized the necessity for software development practices that meet industry standards for agility. The DoD Enterprise DevSecOps Reference Design document outlined the reference design to enable DevSecOps to scale across the DoD (Defense, 2021b). This growing interest in DevSecOps

stemmed from its ability to address the unique challenges of adopting Agile and DevOps practices in complex and highly regulated environments, such as military networks and combat systems (Miller et al., 2022). By moving security left in the Software Development Life Cycle (SDLC), organizations could detect and remediate vulnerabilities earlier, reduce costs, and enhance team collaboration (Kim et al., 2016; Rahy & Bass, 2020; Zaydi & Nassereddine, 2020). Success stories, like the Army Software Factory (ASWF) and the Air Force's Kessel Run Division (KR), have demonstrated the effectiveness of implementing Agile DevSecOps to speed up and secure software development (Command, 2023b; Run, 2022). This growing adoption highlighted the need for further research on strategies and factors contributing to the successful implementation of Agile DevSecOps in the DoD.

To better understand the challenges and benefits of adopting Agile and DevSecOps in the DoD, this dissertation examined the existing literature on the subject. For instance, McQuade et al. (2019) discussed the need for refactoring the acquisition code for competitive advantage in software development. Additionally, Miller et al. (2022) highlighted the challenges of adopting DevOps for the combat systems development environment, emphasizing the need for research on overcoming these obstacles. Several studies have examined the implementation of Agile and DevOps in various contexts, providing valuable insights into the challenges and benefits associated with their adoption (Bolhuis et al., 2022; Luz et al., 2019; Zampetti et al., 2022). These studies provided a foundation for understanding the specific challenges and opportunities that may arise when implementing Agile and DevSecOps in the DoD. In response to the need for further understanding, this dissertation employed an explorative qualitative research approach to explore the strategies used by project managers

when adopting Agile DevSecOps to manage software development in the DoD. The research was guided by a conceptual framework that synthesized the existing literature on Agile, DevOps, and DevSecOps and the challenges and benefits of their implementation in various environments.

Study Problem

The problem addressed in the proposed study was that the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD had not been established. The existing software development approach within the DoD presented significant risks. It was lengthy, costly, and exposed war fighters to unacceptable risks from delayed access to essential tools for mission success (McQuade et al., 2019). The DoD faced inefficiency and a lack of security in its software development and delivery processes (Gerstenberger, 2022; McQuade et al., 2019; Miller et al., 2022). The DoD's ability to adapt and respond to evolving threats was determined by its capacity to rapidly develop and deploy software to the field (McQuade et al., 2019). The Government Accountability Office (2021) reported that the DoD had experienced difficulties finding and hiring staff, transitioning from Waterfall to Agile software development, and managing technical environments. The DoD encountered challenges in scaling up Agile methodologies to accommodate its missions' complex and diverse needs (McQuade et al., 2019). The DoD spent $2.8 billion on 29 selected major business IT programs in FY 2019 and planned to invest over $9.7 billion between FY 2020 and FY 2022 (Office, 2021).

The DoD recognized the need to rethink its software development practices and culture by leveraging commercial sector approaches and best practices, such as Agile (Office, 2021) and

DevSecOps, to improve security and efficiency, reduce deployment friction, and risk management within the organization (Defense, 2021c). Many programs and missions within the DoD had inadequate software development practices and lacked industry-standard agility (Defense, 2021c). The Defense Innovation Board (DIB) study on Software Acquisition and Practices (SWAP) highlighted the central role of software development in National Defense and the need to remove hardware-centric bottlenecks and increase digital talent within the acquisition workforce (McQuade et al., 2019). In response to these challenges, the DoD explored the implementation of Agile and DevSecOps in software factories such as the Army Software Factory and the Air Force Kessel Run (Factory, 2022; Run, 2022) to improve security, efficiency, and speed of software delivery.

Study Purpose

The purpose of this qualitative exploratory study was to explore the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. A different approach to the current software delivery methods would help reduce risks to the warfighters by ensuring that they have the tools they need in a timely fashion to execute their missions in today's software-dominated environment (McQuade et al., 2019). Changing the current software delivery methods may have decreased risks to the warfighter. This research aimed to enrich the understanding of software development methodologies, mainly focusing on the integration of Agile and DevSecOps approaches into the DoD software development lifecycle. The study employed a purposive snowball sampling technique that allowed for the initial selection of project managers experienced in Agile DevSecOps within the DoD, who then recommended additional participants. This method

ensured an extensive understanding of the strategies employed when adopting Agile DevSecOps to manage software development in the DoD. The preferred number of participants was ten; the exact number was predetermined with an addition to follow the guiding principle of data saturation, where additional interviews did not yield any new insights.

Research Question

The study research question aimed to explore the strategies used by project managers when adopting Agile DevSecOps to manage software development in the DoD. The interview questions from the research instrument are shown in Appendix A. The research included project managers who use Agile DevSecOps in the DoD to manage software development. The responses were collected and analyzed to identify the strategies used. The answers then were against the literature to derive new findings that helped answer the research question.

Q1

What are the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD.

Conceptual Framework

The conceptual framework for this qualitative exploratory study (see figure 1) began by examining Waterfall and Agile methodologies to differentiate the software development approaches. The study then introduced DevOps, a software development approach that bridges the gap between development and operations teams. DevOps is a collaborative approach to software development and operations that emphasizes communication, integration, automation, and continuous improvement (Luz et al., 2019). The literature then diverged DevOps into maintaining a DevOps culture and using tooling and practices to enable DevOps.

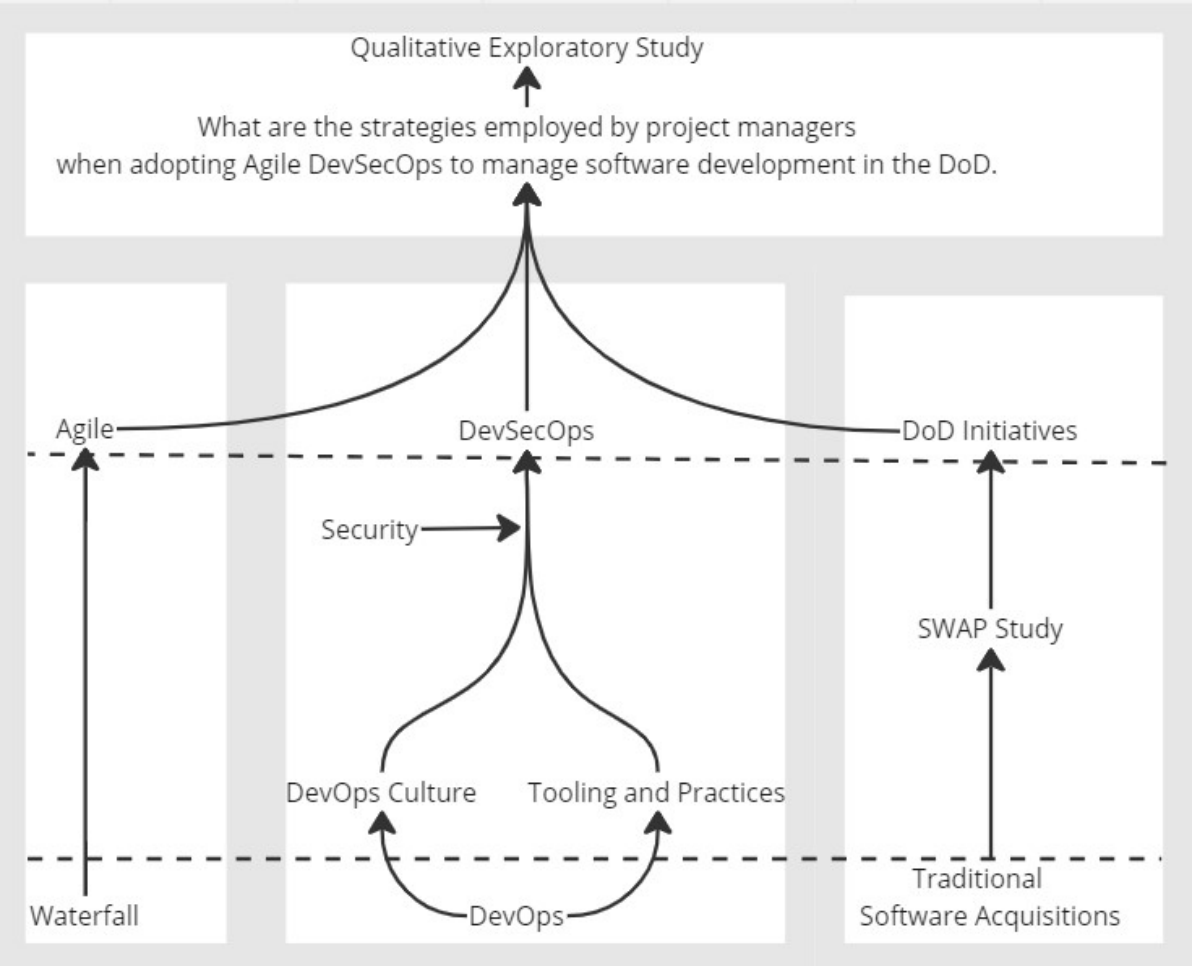
The literature notes the tools and practices used to support a DevOps approach and how they can streamline the software development process.

Security was introduced as a concern in the modern world with increasing cyber risks. The literature explored the importance of shifting security left earlier in the development process and not as the traditional after thought. The concept of shifting security left places security into DevOps, extending it into the new approach of DevSecOps. DevSecOps is an extension of DevOps that integrates security practices into the software development and operations process to reduce security risks (Zaydi & Nassereddine, 2020). The traditional Army software acquisition strategy was discussed in parallel, introducing the Software Acquisitions and Practices (SWAP) Study conducted by the defense innovation board in 2019. The SWAP Study was performed pursuant to Section 872 of the 2018 National Defense Authorization Act, shedding light on many of the DoD acquisition problems. The SWAP study was a significant influence of the DoD's problem: inefficiency and a lack of security in its software development and delivery processes.

Initiatives and software factories emerged and continue proliferating throughout the DoD to enhance delivery times. The Army Software Factory and the Air Force Kessel Run stood as two active examples of these software factory efforts within the DoD, as evidenced in the literature. The topics of Agile, DevSecOps, and DoD initiatives drove the research question, "What are the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD?" This study aimed to explore these strategies in-depth, shedding light on project managers' approaches when adopting Agile DevSecOps within the DoD's software development process.

Figure 1

Conceptual Framework



Significance of the Study

The exploration of the strategies employed by program managers in the adoption of Agile DevSecOps in the DoD stood as a potentially significant contribution to the existing body of knowledge in technology management, software development, security, and operations. This research had the potential to illuminate the intricate strategies of Agile DevSecOps implementation in the DoD, thereby offering valuable insights to scholars and practitioners alike. This study could potentially fill a gap in the DoD's and academia's understanding of these

modern software development practices. The DoD had shown a keen interest and has had several initiatives in adopting Agile and DevSecOps, but their practical implementation strategies have remained unexplored in the academic body of knowledge. This study hastened the accumulation of knowledge in this field by focusing on project managers' strategies when adopting Agile DevSecOps to manage software development. The digital age has heightened the necessity for efficient software development, robust security, and rapid deployment. Any shortcomings or vulnerabilities in the software development process could lead to substantial repercussions and, in this case, risks to the warfighter.

The prevailing software development approach was identified as a significant risk, mainly due to its prolonged timeline, which put warfighters at risk by delaying access to crucial tools (McQuade et al., 2019). Therefore, the potential impact of this study, which sought to address this concern, was vast. The study's unique contribution to existing scholarship lies in its exploration of the strategies used by program managers in the DoD when adopting Agile DevSecOps to manage software development. By offering insights into the practical implementation of Agile DevSecOps, the study paved the way for future research in technology management. Examining these strategies could enhance understanding in the broader software development and delivery field within the DoD and comparable organizations. This study contributed to the existing knowledge on employing Agile DevSecOps in software development and shed light on the challenges and successes associated with implementing this approach in the DoD.

Researcher Positionality and Reflexivity

Reflexivity was a crucial aspect of research analysis, as it allowed me, as the researcher, to be aware of my biases and assumptions and how they may have influenced the research process. This ongoing self-reflection included data collection, data analysis, and interpretation of results. Ravitch and Riggan (2016) suggest that an effective approach to promoting researcher reflexivity is to ask a thought-provoking question that probes how the researcher perceives the connections between themselves, their research, their target audiences, and their participants. I continuously asked how I see myself as the researcher among the research and the participants. As a Platform Engineer and Army Software Factory member, I was an active duty Soldier in the organization's platform operations team. I provided troubleshooting and platform development, using the platform software developers and security engineers utilize to develop, test, scan, and deploy applications to cloud infrastructure. Although I did not have a direct relationship with the interviewees, I may inherently be biased to think that Agile DevSecOps was a successful process due to my status at the ASWF.

To reduce bias in the research process, I continuously reflected on my personal biases and assumptions and how they influenced my work. Consistent reflection helped to gain a more comprehensive understanding of the research topic and reduced the potential for bias in the findings. I also ensured that my data collection process was as objective as possible by using reliable and validated methods and avoiding sources that may have a bias. Mitigating the potential for preconceptions and bias involved the use of journaling detailing the decision process throughout the research (Mills & Birks, 2017). One potential bias that could have been raised was my inclination to think that Agile DevSecOps was a better approach to software

development than traditional Waterfall methods. As I analyzed the data collected from the interviews, I closely noted my thought process, ensuring I looked at it through a neutral, unbiased lens.

Additionally, I engaged in peer review and collaboration with university staff to gain constructive feedback and new insights, which helped validate the research findings and identify any areas where my biases may have influenced the process. I aimed to produce rigorous, accurate, and trustworthy research by remaining vigilant and proactive in reducing bias. Recognizing that biases and assumptions can shape the research process, my responsibility as a researcher was to counteract these influences. I was confident producing high-quality research by focusing on reflexivity and continuously questioning my biases.

Delimitations and Limitations

Delimitations are limitations that a researcher sets consciously by themselves. Theofanidis and Fountouki (2019) present that delimitations require challenging the researcher's assumptions. They are shortcomings of the research and should be openly exposed to the reader. Delimitations are acknowledged and set by the researcher. A delimitation was maintaining the scope of theoretical frameworks to Agile, DevOps, and DevSecOps. The data collection did not include other frameworks such as Lean, Rapid Application Development (RAD), Spiral, or Dynamic Systems Development Method (DSDM).

Limitations of a study are weaknesses and usually outside of a researcher's control (Theofanidis & Fountouki, 2019). In this study, one major limitation was focusing solely on current and prior project managers within the DoD. While these individuals play a significant role in managing software development using Agile DevSecOps, their perspectives and

strategies may not fully represent or capture the holistic view of the process. There may have been nuances and important details that can be provided by other stakeholders involved in the process, such as developers, security professionals, and end-users. Another limitation of this study relates to the potential issue of self-report bias during interviews. Participants may have tended to provide responses that they believe are socially desirable or favorable to their professional reputation rather than accurately reflecting their true practices and experiences. The study's scope was another limitation. It only focused on Agile DevSecOps implementation within the DoD. Therefore, the results may not be generalizable to other organizations or sectors employing Agile DevSecOps for software development.

Definition of Terms

Defining the terms will help with the understanding of the research. The research contents were very infantile; some had not gained standard definitions. This section's operational definitions appear alphabetically and include citations for their sources. Terms are closely related to modern software development practice but are not inclusive.

Agile

Agile describes a mindset of values set by the Agile manifesto (Institute, 2021)—an iterative approach to managing non-Waterfall projects or products.

Containers

A container is a standardized software unit that combines code and its dependencies to ensure consistent performance across diverse computing environments. Similar to virtual machines, containers share resources with host operating systems, resulting in greater efficiency (Orechovesky, 2021).

Continuous Integration/Continuous Delivery (CI/CD) pipeline

A CI/CD pipeline is a set of automated processes that enable developers to build, test, and deploy software changes quickly and reliably (Zampetti et al., 2022).

DevOps

DevOps is a collaborative approach to software development and operations that emphasizes communication, integration, automation, and continuous improvement (Luz et al., 2019).

DevSecOps

DevSecOps is an extension of DevOps that integrates security practices into the software development and operations process to reduce security risks (Zaydi & Nassereddine, 2020). An approach to move security left earlier in the software development lifecycle.

GitOps

GitOps is a model for operating Kubernetes clusters and cloud-native applications using the version control system Git as a single source of truth (Beetz & Harrer, 2022).

Information Security

Information security is the protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction to ensure confidentiality, integrity, and availability (Dempsey et al., 2011).

Waterfall

The Waterfall model is a software development process that follows a predetermined sequence of phases, which must be completed before moving on to the next phase. Each phase

requires thorough documentation before the next one can begin (O'Brian & Green-Mack, 2018).

Chapter Summary

Chapter 1 provided a contextual overview of the problem, purpose, and research question of this study. The problem to be addressed is the lack of exploration regarding the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. The purpose of this qualitative exploratory study was to uncover these strategies and their impact on software development outcomes. This research was significant because it contributed to the existing body of knowledge and academic discourse surrounding Agile DevSecOps within the DoD. By focusing on the strategies used by project managers in the DoD, this research aimed to fill gaps in understanding and offer valuable insights into implementing Agile DevSecOps in DoD settings.

Chapter 2 will present a comprehensive literature review encompassing various software development methodologies, including Waterfall and Agile. The literature review will explore DevOps, DevSecOps, and related topics like continuous integration and continuous delivery (CI/CD). Traditional software acquisition and management initiatives within the DoD will also be examined. This will set the stage for exploring the unique strategies project managers employed in the DoD when implementing Agile DevSecOps for software development.

Chapter 2: Review of the Literature

The problem addressed in this study was that the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD had not been established. The purpose of this qualitative exploratory study was to explore the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. This research aimed to enrich the understanding of software development methodologies, mainly focusing on the integration of Agile and DevSecOps approaches into the DoD software development lifecycle. This concise literature review provides a comprehensive overview of key, up-to-date empirical literature within the last five years, focusing on peer-reviewed sources to ensure validity. The review followed the structure of the conceptual framework in chapter one, guiding the reader through a clear path to understanding current conversations, concepts, and terminologies in the literature. Initially, the review contrasts the Waterfall and Agile methodologies to differentiate their approaches in software development. It then explores DevOps, a software development methodology that facilitates cohesion between development and operations teams. The literature further delineates DevOps as maintaining a DevOps culture and employing tools and practices to enable it. It highlights how these tools and practices can optimize the software development process.

Literature Search Strategies

A comprehensive literature review was conducted for this study, utilizing various sources, including ACM, ProQuest Computer Science Database, IEEE Xplore, Sage Journals, Google, Google Scholar, and the Colorado Technical University library search. Keywords

searched to identify relevant literature were "Agile," "DevOps," "DevSecOps," "DoD software acquisitions," "DoD DevSecOps initiatives," "DoD software factories," “AirForce Kessel Run,” “ASWF,” and "Army Software Factory." In addition to traditional academic sources, grey literature such as government white papers, memorandums, conferences, and directives were included to capture relevant information due to the rapid evolution of technology and approaches in the technology field. Citation management software EndNote 20 was utilized during the literature collection process to ensure maximum metadata capture and management. The collected information was coded and analyzed using Nvivo and thematic analysis techniques to identify common practices.

Waterfall

The traditional software development life cycle (SDLC) Waterfall methodology is a step-by-step process in which each phase of the software development process must be completed before moving on to the next. This approach involves following a set sequence of activities, starting with planning and ending with implementation. Each phase must be completed before proceeding to the next, resulting in a linear progression through the different stages of the SDLC (Drew, 2020). The Waterfall methodology has been widely used in federal government projects, including the DoD; its inflexibility can lead to challenges such as contracts, proprietary limitations, and vendor lock-in (Drew, 2020; Goljan et al., 2021). Stark (2022) contended that the Waterfall method may be effective for developing physical products like trucks, aircraft, and tanks but may not be as suitable for software development within the DoD.

The Project Management Institute published the 7th edition of the PMBOK (Project Management Body of Knowledge) in 2021. Industry project management standards are set

through the PMBOK by this publication Institute (2021). According to the 7th edition of the PMBOK, different approaches, such as predictive “Waterfall” and iterative “Agile,” need to be selected based on the requirements of the project and its cadence (Institute, 2021). Aerospace, automotive, and infrastructure industries follow predictive project management guidelines (Institute, 2021). Similarly, developing training programs and producing equipment like trucks, aircraft, and tanks may use a predictive approach (Stark, 2022). Conversely, a software application should use an Agile approach called an incremental one. This perspective aligns with the arguments made by Drew (2020), Goljan et al. (2021), and Stark (2022), who suggest that while the Waterfall approach may be suitable for specific projects with well-defined requirements, Agile methodologies are recognized as being more effective in handling the dynamic nature of software development projects, offering greater flexibility and adaptability in response to changing requirements and project scopes.

The Waterfall methodology, a linear approach to software development, has been criticized for its inflexibility and inability to adapt to the rapid pace of technological change. The Waterfall approach has several disadvantages in the context of government software acquisitions. The DoD faced significant producibility challenges as it worked to develop innovative, unprecedented software systems (National Research et al., 2010). The Waterfall approach is inflexible, as it is difficult to change the software once the development process has begun. This can lead to delays, cost overruns, and programmatic risks if requirements change or new issues are identified during development (National Research et al., 2010). The Waterfall approach is inefficient and unable to provide incremental solutions, as it does not allow for the simultaneous development and testing of different software components (Crook et al., 2020).

Research by Bolhuis et al. (2022) showed that professionals practicing Agile development in the Internet of Things (IoT) industry reported difficulties using the Waterfall methodology, particularly in aligning differences in organizational norms and cultures. The complexity and interdependence of technologies involved in IoT projects can also make using the Waterfall methodology difficult. Therefore, many professionals in the IoT industry have embraced Agile development methods as a more flexible and adaptable approach to software development (Bolhuis et al., 2022). The current Waterfall-based model with milestones and gates that the DoD uses to manage its software life cycle does not adequately support the rapid pace of technological change and the need for flexibility and agility in software development. This approach assumes that software can be developed in a linear, step-by-step manner but does not consider software's complexities and rapid evolution and the need for agile development processes (Iyer, 2022). The limitations identified in the Waterfall methodology prompted companies to adopt more agile methods in software development (Marini-Wear, 2019).

Agile

Agile is an approach that allows flexibility and the ability to adjust to changes; this literature review uses the term in software development. The term "Agile" emphasizes the significance of being able to react to changes in a rapidly evolving environment. Agile describes a mindset of values set by the Agile manifesto (Institute, 2021)—an iterative approach to managing non-Waterfall projects or products. The approach is focused on continuously analyzing the current situation, identifying uncertainties, and adapting accordingly (Alliance, 2018). On February 2001, seventeen people representing various software development

methodologies met in a ski resort in the Wasatch mountains of Utah. They created the Manifesto for Agile Software Development, also known as the “Agile Alliance” (Beck et al., 2001a). Beck et al. (2001b) claimed that the Agile Manifesto set a better way to develop software by following a set of values, prioritizing "individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan" (Beck et al., 2001b, p. 1).

The principles behind the Agile Manifesto prioritize customer satisfaction by providing early and continuous delivery. The Agile mindset encourages agility, allowing for adding elements to the process while in development and welcoming changes even late in development (Beck et al., 2001a; Eze et al., 2022). Multiple authors state that the concept of a minimum viable product (MVP) is central to Agile development; the MVP has just enough features to meet basic functionality and serves as the foundation for future improvement (Defense, 2021a; Factory, 2022; McQuade et al., 2019). Delivery of the MVP should occur within a couple of weeks to a couple of months, with shorter time scales preferred (Beck et al., 2001c).

Agile teams prefer an incremental development approach, in which each product version is usable and builds upon the previous one by adding new functionality visible to the user (Alliance, 2018). Providing users with an application early allows for early evaluation and feedback, enabling iterations of the development process and continuous software improvement. Program and product managers can then use this feedback to re-prioritize the backlog and portfolio (McQuade et al., 2019). Beck et al. (2001c) stated that continuous changes to the software harness competitive advantage for the customer. Automated small chunks of code changes in an Agile approach enhance the feedback loop for addressing code

quality issues and speeding up the delivery of a project or product with frequent customer feedback (Marini-Wear, 2019).

The Agile Manifesto encouraged developers and customers to work together to garner the most benefit, promoting face-to-face communication in a collaborative environment as the best way to exchange information quickly. Communication and trust are essential early in the process due to the fast project turnaround. Rahy and Bass (2020) mentioned that Agile software development could become overwhelming, necessitating early trust and communication to prevent communication gaps that may generate low-quality or misleading information, leading to incorrect or late work. They also mentioned that If negative first impressions occur, a decline of trust may begin, violating the element of individuals and interactions. Beck et al. (2001c) stated that working software is the primary measure of progress; software that is done is better than perfect and never delivered. Finster (2021) offered a recent definition of Agile as being "[a] set of principles and practices focused on small batch sizes, rapid delivery, and continuous improvement. Agile processes are a required subset of DevOps (Finster, 2021, p. 13).

Approaches and Tooling

Introducing Agile and Waterfall software development practices helps understand their differences. The literature so far demonstrated added benefits to using Agile approaches in software development as it states that working software is the primary measure of progress; software that is done is better than perfect and never delivered (Beck et al., 2001c). The following subtopics will cover DevOps, cultural aspects, tooling and practices, security considerations, and the introduction of DevSecOps.

DevOps

DevOps is the amalgamation of development and operations. Combining the two words fits the idea of improving collaboration between development and operations (Beetz & Harrer, 2022). DevOps is the combination of people, processes, and products continuously delivering value to end-users (Díaz et al., 2021; Finster, 2021). The term DevOps was accidentally coined by Patrick Debois, a Belgian consultant and IT professional, in a presentation at the Agile Conference in Toronto in 2009 (Kim et al., 2016). In the presentation, Debois discussed the need for organizations to improve collaboration and communication between development and operations teams to support software's rapid deployment. The term "DevOps" was and is still intended to reflect this focus on collaboration and integration between these traditionally separate teams (Eze et al., 2022). DevOps directs automation, testing, management, and deployment (Quillen, 2022).

The concept's inception occurred after seeing the seminal presentation by John Allspaw and Paul Hammond, “10 Deploys per Day,” at the 2009 Velocity Conference (Allspaw & Hammond, 2009). In the presentation, Allspaw and Hammond (2009) provided valuable insights into the relationship between development and operations teams in software development from their experiences at Flicker. They challenged the conventional views of these teams as adversaries and suggested that the role of the operations team goes beyond just keeping the site stable and fast. Instead, the operations team should be seen as enablers of the business, and their job should encompass a wide range of responsibilities. Allspaw and Hammond (2009) emphasized the crucial role of cross-functional teams, collaboration, and communication in enabling successful DevOps practices. Their talk highlighted the importance of breaking down

silos between development and operations teams and fostering a culture of collaboration and teamwork. The presentation provided valuable insights for organizations looking to adopt DevOps practices and highlighted the criticality of strong communication and collaboration between development and operations teams.

The 2000s enveloped the Agile Manifesto, establishing iterative software development practices to bring the time for developing new functionality to months or weeks. However, the deployment of applications was still very lengthy and required those from development and operations to work together. The outcomes sometimes resulted in catastrophic deployments (Kim et al., 2016). By 2010 DevOps began to take flight, and organizations started to adopt it as a method to integrate Development and Operations to reduce silos and improve communication increasing the speed of software delivery. According to various researchers, DevOps decreases development and engineering time while increasing deployment frequency. (Forsgren et al., 2018; Kim et al., 2016; Otwell, 2022).

Alnafessah et al. (2021) presented the stages of the DevOps lifecycle, which include: Plan, Develop, Verify, Test, Deploy, Operate, and Monitor. The focus of each stage is defined and described briefly, as the planning stage aims to define objectives and requirements, the development stage focuses on code development and review, and the verification stage evaluates the correctness of software artifacts. The testing phase includes continuous automation testing, and the deployment phase focuses on continuous re-deployment. The operating stage deals with the configuration and management of the software after deployment, while the monitor stage tracks the performance of deployed applications. The authors noted that these stages are similar to those in traditional software engineering

methodologies but with a focus on the continuous and highly-automated release of software artifacts (Alnafessah et al., 2021). There is no standardized application of DevOps in organizations as of the publishing of this research. However, this literature review will cover elements of standard practices and technologies to draw a well-rounded understanding of the topic. Paired Programming, Culture, Containers, Container Orchestrations, Kubernetes, Microservices, Monitoring, Pipelines, Git, Security, and the transition to DevSecOps.

Paired Programming

Paired programming, or Extreme Programming, is a form of collaboration used in Agile software development, where two developers work together on the same computer to complete a task. Jones (2019) emphasized the significance of collaborative programming in Agile Software Development, with team members working together in short, iterative cycles while integrating user feedback. This approach diverged from traditional practices by prioritizing cooperation over the conventional handover process. Team members work closely together, regularly joining forces to address challenges collectively. Gerstenberger (2022) elaborated on the benefits of collaborative programming, stating that when paired, extreme programming can foster innovation by aligning with design thinking principles. In the conclusion of Gerstenberger (2022) research, the researcher stated that Test-Driven Development (TDD) is influenced by various factors, including adopting paired programming as a standard practice among engineers. Bolhuis et al. (2022) further support the importance of paired programming in Agile Software Development, mentioning it as a critical element that enables Agile team growth and high-quality software production.

DevOps Culture

Culture is a collection of shared beliefs, values, and practices that shape group or organizational interactions (Díaz et al., 2021). Culture plays a fundamental role in successful DevOps environments, often characterized as a cultural shift towards enhanced cooperation between development and operations (Díaz et al., 2021; Luz et al., 2019). The DevOps culture facilitates automation and efficiency, teamwork, and collaboration, which collectively expedite market entry, improve software quality, and heighten customer satisfaction (Díaz et al., 2021). The development of a cooperative culture is a prerequisite for the successful implementation of DevOps, with a specific emphasis on breaking down silos between development and operations teams, as stated by (Luz et al., 2019). Quillen (2022) illustrated this by describing how security engineers under pressure for faster assessments must collaborate with operations engineers striving for 99.999% uptime under pressure to expedite secure development.

Crook et al. (2020) stated that an effective DevOps culture should foster a growth mindset and support continuous learning and innovation. It also necessitates addressing behavioral changes within development and operations teams (Eze et al., 2022). This cultural shift poses a substantial challenge when integrating security teams into the software development lifecycle, requiring significant collaboration and security considerations at every stage (Eze et al., 2022). The 'human factor' is fundamental to the system's success; thus, special attention should be devoted to individuals managing the systems (Crook et al., 2020). This correlates with one of the ideas of the Agile Manifesto to prioritize individuals and interactions over processes and tools (Beck et al., 2001b). Culture may also serve as a barrier to organizational change. Addressing this barrier necessitates a deliberate organizational culture

design, aligning it with the system's architecture to facilitate successful transformation (Crook et al., 2020). In a digital transformation context, the COVID-19 pandemic has highlighted the need for leaders to define clear transformation outcomes and implement bold DevOps strategies. These strategies include prioritizing tasks, increasing transformation investments, demonstrating the value of DevOps, and modernizing technical stacks (Walters et al., 2020). Quillen (2022) reiterates the importance of automation in complementing the human element, enabling engineers to focus on more complex tasks, thus potentially increasing profitability.

Play one of the DoD DevSecOps Playbook (Defense, 2021b) promotes collaboration and integration among development, deployment, security, and operations teams. Its success depends on buy-in from all stakeholders, including leadership, acquisition, contracting, middle-management, engineering, security, operations, development, and testing teams. Such a culture fosters collective responsibility and encourages the breaking down silos, leading to a more effective software and infrastructure engineering process (Defense, 2021b). Adopting a DevOps culture is not without its challenges. A systematic review of 66 studies by Khan et al. (2022) highlighted critical issues such as inadequate collaboration and communication, a deficiency in DevOps skills and knowledge, and a lack of shared understanding of DevOps. The study results showed that a lack of collaboration and communication was the most critical challenge, with a frequency of 68%. The lack of skill and knowledge was challenging, constituting 56% of the published articles.

Criticism practices had a frequency of 50% as a critical challenge that harms the DevOps culture. Criticism practices were characterized as a blaming culture that often focused on punishment, conflict, finger-pointing, and fostering negative perceptions. Such a culture can

lead to destructive behavior and disrupt the workflow (Khan et al., 2022). The most important issue in integrating DevOps into an organization is that people oppose attitudes that are typically an impediment to organizational change success. A lack of shared understanding of DevOps is a significant issue in adopting DevOps culture, with a frequency of 47%. Lack of management was also a significant challenge, with a frequency of 45%. Trust and confidence problems were other difficult barriers for DevOps culture, with a frequency of 45% (Khan et al., 2022). Díaz et al. (2021) further supported these findings, indicating that the primary motivation for adopting a DevOps culture is to reduce software delivery time. Their study, based on case studies of 30 multinational software companies, also pointed to common patterns and anti-patterns in adopting DevOps. Companies typically focused on breaking silos and improving collaboration, optimizing and automating processes, and maintaining high-quality products while aiming for a faster market entry (Diaz et al., 2019).

Containers

An application container is a lightweight, standalone, executable software package encompassing all the necessary components to run an application. Containerization is a technological strategy that facilitates the execution of software applications in a standalone environment, encompassing all required components apart from the operating system, such as code, runtime, system tools, libraries, and configurations (Defense, 2021b). Containers operate within isolated processes, distinguishing them from other containers and ensuring their independence. They offer a scalable, efficient, and secure method to package and deploy software applications, which is highly significant in DevSecOps.

A qualitative study by (Orechovesky, 2021) examined the use of container security within DevSecOps pipelines. Orechovesky (2021) discovered that containerization and automated security practices enhanced cybersecurity measured and alleviated burdens on cybersecurity engineers. The study suggested that containers can significantly contribute to security initiatives within DevSecOps. Otwell (2022) further discussed the role of containers in the automated deployment within a DevSecOps Continuous Integration/Continuous Deployment (CI/CD) pipeline. They additionally proposed that using a Platform as a Service (PaaS) is essential to complete automated deployments, reinforcing the prominence of container technologies in current software deployment strategies.

The DevSecOps Playbook (Defense, 2021b) called for an adoption of a modular open system approach (MOSA) that is predicated on microservices and software containers is beneficial. MOSA is an acquisition and design strategy with a technical architecture conforming to open standards and promoting a modular, loosely coupled, and highly cohesive system structure. Containers, as a part of this strategy, allow for an implementation that meets MOSA requirements. Conversely, the complexities associated with emerging architectures like containers pose significant challenges in edge performance observability, according to Usman et al. (2022). As containers become more integral to these architectures, a more nuanced understanding of their role in ensuring optimal performance is necessary.

The Open Container Initiative (OCI) was launched by industry leaders like Docker and CoreOS on June 22, 2015, to create open industry standards surrounding container formats and runtimes (Initiative, 2023). The OCI's responsibility encompasses three specifications: the Runtime Specification, the Image Specification, and the Distribution Specification. These

specifications delineate the process for running an OCI Image as a filesystem bundle, underlining the interoperability of containerization in contemporary software practices (Initiative, 2023). Containerization is a key element in DevSecOps, given its role in enhancing cybersecurity, facilitating automated deployments, and contributing to establishing open system architectures.

Container Orchestration

Container orchestration refers to the management and coordination of containers in a distributed system. It involves using specialized tools and technologies, such as Docker and Kubernetes, to automate the process of deploying and managing containerized applications at scale. Birkeland et al. (2020) elucidated that containerization leverages technologies like Docker and Kubernetes to simplify the development, testing, deployment, and orchestration of microservices, APIs, and service meshes. In the domain of DevSecOps, these containerization technologies streamline the construction and maintenance of complex applications and services. In particular, container orchestration is fundamental in managing and coordinating containers in distributed systems. It employs specialized tools and technologies to automate the deployment and management of containerized applications at scale, enhancing the efficiency and effectiveness of the application lifecycle management (Lopez Garcia et al., 2020).

Kubernetes, or K8s, is a popular open-source container orchestration platform. This platform, founded on Google's extensive experience managing production workloads, facilitates the organization of containers into logical units for efficient management and discovery (Kubernetes, 2023). Kubernetes offers deployment automation, scaling, self-healing, and service discovery, significantly aiding organizations in managing and optimizing their

containerized applications. Birkeland et al. (2020) asserted that Kubernetes, among other containerization technologies, greatly eased the development, testing, deployment, and management of microservices, APIs, and service meshes. Thus, these technologies appear critical to modern software development and operations. However, containers and container orchestration are not devoid of challenges.

Usman et al. (2022) identified that despite the complexities arising from Agile development methodologies, cloud-native deployments, and novel DevOps practices, Kubernetes was an invaluable tool in modern IT management. It simplifies the orchestration of edge infrastructures and reduces underlying complexity, enabling efficient and effective IT ecosystem monitoring, yet observability became a new challenge requiring additional tools. Adding to the narrative, Marini-Wear (2019) emphasized the importance of Infrastructure as Code (IaC) in container orchestration. IaC leverages automated tools like Ansible, Chef, or Puppet to manage configuration changes. The core element of IaC is the use of definition manifest files, like Dockerfiles and Kubernetes configurations, which detail the components required by the microservice running inside the cloud-based container to build the image. These components include library versions, operating system patch levels, and more. While container orchestration technologies provide significant benefits, such as simplifying development and deployment processes, they also present certain challenges that necessitate further exploration and research (Marini-Wear, 2019).

Microservices

Microservices is an architectural style for building and deploying software applications that involves decomposing a complex application into a collection of smaller, independent

services that can be developed, tested, and deployed independently. Siewruk and Mazurczyk (2021) stated that a change from monolith to microservice design could deploy a new version of an application more quickly and be accessible to the end user as soon as possible. This approach has gained popularity recently, particularly in DevOps, as it enabled organizations to improve their applications' agility, scalability, and maintainability. Alnafessah et al. (2021) wrote about some microservices challenges, including monitoring them, determining container placement strategy, and autoscaling them. These challenges are complex and require careful consideration to ensure that microservices function optimally. Play 3 of the DoD's Defense (2021b) DevSecOps Playbook recommended adopting a modular open system approach (MOSA) using containerized microservices. MOSA is a technical architecture that uses open standards and supports a modular, loosely coupled, and highly cohesive system structure. It was required by U.S. Code Title 10 Section 2446a and DoD Instruction 5000.02. (Defense, 2021b).

Containerized microservices, where discrete business services are bundled in lightweight, executable software packages, meet MOSA requirements. Containers run in isolated processes and can be quickly composed using lightweight protocols, providing benefits such as scalability, upgrade options, cyber hardening, and support for failure and recovery (Defense, 2021b). Key characteristics of a containerized microservice architecture include componentization, organization around business capabilities, infrastructure automation, and design for failure. The checklist for implementing this play includes researching the benefits of microservices, using CNCF Certified Kubernetes for container orchestration, leveraging Iron

Bank for hardened containers, injecting the Sidecar Container Security Stack (SCSS), and adopting a service mesh to secure east-west network traffic.

Monitoring

Monitoring enables tracking and analyzing the performance and health of systems in real-time. According to NIST SP 800-137, continuous information security monitoring (ISCM) involves constant awareness of information security, vulnerabilities, and threats to support organizational risk management decisions. This is achieved by continuously inventorying all system components, monitoring the performance and security of those components, and logging relevant events related to applications and systems (Dempsey et al., 2011). Monitoring consists of checking applications' availability and responsiveness, utilizing node status, memory, and network performance resources, and identifying potential errors and issues. Monitoring has been a core function of information technology; however, it has become limited by development approaches like DevOps and Agile (Usman et al., 2022). Applying traditional monitoring tools will not serve a complex architecture like Kubernetes. The out-of-the-box solutions are insufficient for a dynamic environment outside containerized applications (Usman et al., 2022). By monitoring systems, organizations can gain insights into their applications and infrastructure performance and proactively address issues before they impact the user experience. Usman et al. (2022) concluded that observability in cloud-native, containerized infrastructures and microservices still needs more research.

Continuous Integration / Continuous Delivery Pipelines (CI/CD)

Continuous Integration/Continuous Delivery (CI/CD) pipelines in DevSecOps gained significant attention recently. CI/CD is a software development practice involving continuously

integrating code changes, testing them, and deploying them to production. Automating the build, test, and deploy process enhances software development and deployment's speed, reliability, and efficiency. This, combined with quick deployments and automated pipelines, helps companies reduce the time to launch and deploy changes to the application (Marini-Wear, 2019). Walters et al. (2020) concluded that by utilizing open-source software and automating the pipeline, the team's attention is freed to navigate external changes and shifts in the industry. In order to effectively implement CI/CD pipelines, it is necessary to carefully design the process flows and automation activities across the various stages and to implement them through an iterative, automated approach (Defense, 2021b). Finster (2021) emphasized the importance of Continuous Delivery (CD) in this context, as delivering the latest changes on demand while minimizing change size can reduce risk and improve time to market.

GitOps

Git, a widely utilized version control system in DevOps, enhances the speed and reliability of software development and deployment through tracking and managing code changes, enabling collaboration and sharing among teams, and providing a codebase change history for debugging and problem-solving. GitOps is a methodology that leverages code to describe and observe systems declaratively (Defense, 2021b), emphasizing git and a git-driven workflow. Its benefits stem from Infrastructure as Code (IaC) and encompass the use of merge requests and a Continuous Integration/Continuous Deployment (CI/CD) pipeline (Defense, 2021b). GitOps extends the use of Infrastructure As Code (IaC), adding the capability of correcting misbehaving infrastructure through pull-based deployments (Beetz & Harrer, 2022).

Information Security

Security is critical in app development because it protects sensitive information and systems from unauthorized access and tampering. This includes all stages of app development, from design and implementation to deployment and ongoing maintenance. Ensuring comprehensive, reliable, and secure source code development is critical to any software development project (Lopez Garcia et al., 2020). Failing to address security concerns can result in data breaches and other vulnerabilities that can compromise the integrity of the app and harm users. Therefore, it is important to prioritize security at every step of the app development process and afterward through application and foundational infrastructure upgrades. Foundational erosion from not upgrading creates brittle and unstable systems. Upgrading improves performance, removes infrastructure code, fixes bugs, and ensures application security (Uchitelle, 2021).

There are many challenges and considerations regarding security in app development. App developers must consider the potential risks and vulnerabilities associated with the app and take steps to protect against these risks. Lopez Garcia et al. (2020) stated that source code security is crucial in software development. They suggested implementing security controls and testing throughout the development process and using tools such as static code analysis and penetration testing to identify and address potential vulnerabilities. By taking a proactive approach to security, software developers can help prevent potential attacks or data breaches and ensure the reliability and integrity of their systems. In a separate study, Orechovesky (2021) found a strong correlation between DevSecOps build cycles and Linux containers, and other research has emphasized the importance of Linux container security early in the software

development process. Security is paramount for the military's software posture, as it protects sensitive information and systems from unauthorized access and tampering.

The security of military systems is critical due to the sensitive and often confidential nature of the information they handle and the potential consequences of a security breach (Gerstenberger, 2022). Cybersecurity is a critical issue in the military and government sectors, where the Internet of Things (IoT) is increasingly being used to track personnel and equipment and handle sensitive information. To respond to cybersecurity threats, courses of action such as active cyber defense measures and seeking assistance from foreign countries through mutual legal assistance treaties or the Budapest Convention on cybercrime have been evaluated; however, their legality and feasibility may be uncertain and limited by various factors. To ensure the security of IoT devices, the U.S. military has established standards and frameworks such as the Internet of Things Cybersecurity Improvement Act, the Common Criteria, the Risk Management Framework, and the Cybersecurity Maturity Model Certification. However, implementing these standards and frameworks can be challenging due to the complexity and constantly evolving nature of cybersecurity threats (Barnes, 2018; Stanley, 2019)

The use of DevSecOps can increase cybersecurity posture by incorporating security considerations into every phase of the app development process. By "baking in" cyber resiliency and moving towards a Continuous Authorization to Operate (cATO) through a transparently defined and well-understood continuous monitoring program, organizations can effectively enhance their cybersecurity posture (Defense, 2021b). The Defense (2021b) DevSecOps Playbook emphasized the importance of incorporating cyber resiliency into the DevSecOps pipeline process. Organizations implementing DevOps practices often face challenges in

balancing the conflicting goals of the development team, which prioritizes shorter development cycles and faster feature delivery, with the stability goals of the operations team. Integrating security into the development process, known as DevSecOps, introduced the added challenge of prioritizing security concerns. These tensions can hinder the successful adoption of DevOps practices (Eze et al., 2022).

Play 9 of the DoD Defense (2021b) DevSecOps Playbook emphasized the importance of cyber resilience in DevSecOps adoption, aiming to "baking in" cyber resiliency into applications as part of the software factory's DevSecOps pipeline process. Cybersecurity is a vital component of all eight phases of the DevSecOps lifecycle, and control gates serve as decision points for ensuring that cybersecurity is both "baked in" and transparently identified. Moving to DevSecOps also involves moving towards a Continuous Authorization to Operate (cATO) for applications developed using DevSecOps processes, including a software factory with a CI/CD pipeline. A transparently defined and well-understood continuous monitoring program is essential for cATO (Defense, 2021b).

DevSecOps

DevSecOps is a software development methodology that integrates security into the DevOps process. DevOps is a software development approach that emphasizes collaboration and communication between development, security, and operations teams, with the goal of improving the agility, efficiency, and reliability of the software development process. DevSecOps builds on this approach by incorporating security earlier into the DevOps process, with the aim of ensuring that the software being developed and deployed is secure and

compliant with relevant security standards (Eze et al., 2022). In the past, security processes were isolated and left to a specific team at the end of development.

However, the DevOps model's fast and frequent development cycles require a more integrated approach to security, known as DevSecOps. In this approach, security is a shared responsibility throughout the entire development process, from start to finish, to ensure the security of the application and infrastructure (Zaydi & Nassereddine, 2020). Research by Zaydi and Nassereddine (2020) showed that aligning DevOps and security teams can lead to positive outcomes and enable the Business, DevOps, Operations, and Security teams to work together effectively. By collaborating and co-creating value, these teams can meet business needs in an agile and secure way, highlighting the importance of integrating security into the DevOps process, as outlined in the DevSecOps approach. Similarly to DevOps, automation is critical to building capabilities and improving software delivery (Eze et al., 2022).

Transitions to DevSecOps from traditional Waterfall methods involve the introduction of an overwhelming amount of new tools for planning, tracking, and automation (Eze et al., 2022). The transition from DevOps to DevSecOps reflects the growing recognition of the importance of security in the software development and deployment process. Eze et al. (2022) state that the DevSecOps market is expected to experience a compound annual growth rate of 31.2% and reach $5.9 billion by 2023. As DevOps practices have become more widely adopted, there has been an increasing awareness of the need to incorporate security into the DevOps process to protect systems and data and maintain the trust of customers and stakeholders.

In a DevSecOps environment, development, security, and operations teams work together to integrate security into the software development and deployment process. In the

academic literature, the distinctions between DevOpsSec, DevSecOps, and SecDevOps were unclear and often used interchangeably. However, some authors have found that DevSecOps is the more widely accepted definition (Eze et al., 2022). This involves using tools and technologies to support the automation and integration of security into the DevOps process and adopting practices and processes that promote collaboration and coordination among teams. The adoption of DevSecOps was very infantile, and much more research is needed to fully understand the benefits and drawbacks.

DoD Software Acquisitions and Initiatives

DoD software initiatives are crucial in addressing the inefficiencies and challenges the DoD faces in its software development and delivery processes. These initiatives aimed to improve software development speed, effectiveness, and security within the DoD, ensuring timely access to critical tools and capabilities for mission success. This section explored traditional DoD software acquisitions and initiatives to transform the software development landscape. Additionally, it highlighted the role of the Army Software Factory (ASWF) and the Air Force Kessel Run (KR) Software Factory as pioneering organizations within the Army and Air Force dedicated to building transformative tech talent and promoting digital transformations.

Traditional DoD Software Acquisitions

Due to inefficiencies, hardware and software acquisitions in the DoD have been scrutinized and questioned. A study by Miller et al. (2022) stated that the DoD struggled with software delivery. Software development and delivery were usually performed by acquiring a contract and matching it to a military need, causing extensive negotiations and delivery spanning a long timeline (Gerstenberger, 2022). The excessive time it takes to suffer

procurement is a problem that was also stated by the defense innovation board in the software is never done report. “DoD’s current procurement processes treat software programs like hardware programs but can no longer take years to develop software for our major systems” (McQuade et al., 2019, p. 1).

The literature review stated that the inefficiency of timely software delivery is well documented (Miller et al., 2022). Software delivery generally has taken the Waterfall method to delivery, applying gates and handoffs in the process from inception to delivery. The Waterfall method is tedious and takes a long time for software delivery. The reasons for the inefficiency in software delivery by the DOD are complex. Potential causes include using outdated or inflexible procurement processes and lacking clear communication and coordination between teams and stakeholders. In the current Waterfall approach, the Defense Acquisition System takes a decade or more to come to fruition; this obsolescence of hardware creates a tech refresh spiral that leads to nearly endless requirements creep and the eventual death of programs (Miller et al., 2022). Results from the SWAP study reflected that many software programs take too long and are too large and complex to provide value to the users (McQuade et al., 2019).

The problem was worsened by the safe approach from contractors delivering the software, ensuring the contract was protected and not jeopardized. The Waterfall approach has flaws, causing contractual obligations and vendor lock-in. Funding is strictly managed, creating strict measures for software delivery. Contract technology companies use the Waterfall method due to the military assurances to give what it needs. The Waterfall method requirements are

set before the code is written, crossing silos in the development process (Gerstenberger, 2022). Their rigidity and lack of flexibility create inefficiencies.

One of the main disadvantages of the Waterfall approach is that it can be inflexible and rigid. Since each phase must be completed before the next one can begin, it is difficult to make changes or adjustments to the project once it is underway. This makes incorporating new requirements or user feedback challenging, leading to delays or failures in the development process. Another disadvantage of the Waterfall approach is that it can be difficult to accurately predict and plan for all project requirements at the outset. As a result, it is common for projects to experience scope creep, where the scope of the project expands over time, leading to delays and cost overruns; due to the strict budgeted guidelines, project scope changes are not necessarily welcome. Return on investment (ROI) for software development projects using the Waterfall methodology can be challenging to predict and measure.

Software is Never Done

On March 2019, a report called software is never done; Refactoring the acquisition code for competitive advantage was published by the defense innovation board (McQuade et al., 2019). The study was performed pursuant to Section 872 of the 2018 National Defense Authorization Act. Section 872 mandated the Defense Innovation Board (DIB) to conduct a study on software acquisition and practices for national defense. The study aimed to identify best practices and provide recommendations for improving software development, procurement, assurance, deployment, and maintenance efficiency and effectiveness in the DoD (McQuade et al., 2019).

The study found that “The current approach to software development is a leading source of risk to DoD; it takes too long, is too expensive, and exposes warfighters to an unacceptable risk by delaying their access to the tools they need to ensure mission success” (McQuade et al., 2019, p. 1). The study focused on three key themes: speed and cycle time, which are the most important metrics for managing software; software is made by people and for people, so digital talent matters, and software is different from hardware. Miller et al. (2022) stated that the most significant difference between software and hardware is that software must be constantly updated through its lifecycle. All military branches are affected by software development and release management inefficiencies. Research by Goljan et al. (2021) found a projected annual growth rate of 15 to 25% in demand for developing and maintaining all defense software.

After experiencing issues with the delivery of the F-35, the department began using Agile software management techniques to prioritize timelines and ensure the timely delivery of critical mission capabilities. These techniques allow for the rapid sense, share, integrate, coordinate, and act process of software development (McQuade et al., 2019). According to the SWAP study, enhancing software approaches is critical for increasing military readiness and the ability to succeed in any conflict or domain. Goljan et al. (2021) also noted that improving software can increase the overall efficiency of operations. McQuade et al. (2019) suggested that the quickest implementation of software-defined military capabilities will emerge victorious in future conflicts. The study also presents the best software development practices for a successful software company, which will be discussed in a separate section of this research.

This inquiry is not the first time the DoD has asked for an expert assessment of the state of software capabilities and improvement plans. Studies have been traced by McQuade et al. (2019) back to 1982, with many proposed recommendations not being implemented, and in 1982, a study concluded that software is an essential opportunity for military missions. In 1987, another study by the Defense Science Board covered the STARS program, DARPA Strategic computing Initiative, and others, stating that five initiatives were uncoordinated. It was recommended that the Under of Secretary (Acquisitions) formalize a coordination mechanism to increase collaboration. Ten more reports were synthesized by McQuade et al. (2019) on software acquisition and practices in the DoD. The themes identified were the issue of current approaches to software development, including long development cycles and a lack of focus on speed and agility. These reports made recommendations for improving the DoD's approach to software development, including long development cycles and a lack of focus on speed and agility. The reports recommended improving the DoD's approach to software development, including refactoring statutes and regulations, establishing continuous and rapid software deployment, and increasing the understanding of modern software within the acquisition workforce. The reports also highlighted the importance of software in national defense and the need for the DoD to prioritize speed and agility in its software development approach to keep pace with changing threats.

DoD Initiatives

The DoD has faced challenges in IT program management. According to a report from the U.S. Government Accountability Office (GAO), 20 of 29 selected major business IT programs have experienced cost or schedule changes since January 2019, and several programs could be

underreporting risks (Office, 2021). The DoD established several initiatives in place to support the adoption of DevSecOps and the development of software factories. These initiatives aimed to improve the efficiency, effectiveness, and security of the DoD's software development and deployment processes to support the DoD's mission and objectives. Play 6 of the DoD DevSecOps Playbook (Defense, 2021b) recommended using a DevSecOps Software Factory for custom software development within the DoD. The use of a DevSecOps managed service provider, such as Platform One, is one option for establishing a Software Factory or establishing a Software Factory using a CSP with a DoD ATO or PA, leveraging managed services and infrastructure as code practices (Defense, 2021b).

The Army Software Factory was stood up as one of those initiatives; this occurred after the creation of the Air Force Kessel Run initiative. Iyer (2021) identified the three key objectives that had defined the Army's digital transformation strategy. Initially, the strategy had aimed to create a data-driven Army that was digitally enabled, catalyzing the digital transformation process. The second objective focused on maximizing the value delivered to the Army by optimizing digital investments in line with the mission. The third objective was to develop a technologically proficient and operationally effective workforce in the digital realm. This workforce was intended to establish solid alliances with a comprehensive network of allies, industry stakeholders, and academia. Through these ambitious undertakings, the Army sought to initiate a cultural change that encouraged agility, innovation, and technological savvy. The ultimate goal of this transformation was to ensure the Army retained a competitive edge in the digital domain against strategic rivals.

The DoD Software Modernization Strategy offered an in-depth look at the concept of software factories. It delineated a software factory as an assembly plant for software development and integration, which consists of various pipelines (Defense, 2022). These pipelines are fortified with various tools, process workflows, scripts, and environments, enabling the creation of deployable software artifacts with minimal human intervention (Defense, 2022). The concept of automation was emphasized in the context of software factories, particularly in the phases of development, building, testing, releasing, and delivering. The document further elaborated on the software factories' support for multi-tenancy, expanding the scope and functionality of these factories (Defense, 2022). The 2019 DoD modernization strategy indicated that an effective pairing of technology, such as tools and platforms, and process changes, such as security authorization and testing, was imperative to actualize the full advantages of software factories. Therefore, the optimal functioning of software factories hinges on the symbiotic relationship between technological capabilities and process alterations (Defense, 2022).

Army Software Factory

The Army Software Factory (ASWF), based in Austin, Texas, was identified as an innovative initiative overseen by the Army Futures Command. Established in response to several challenges faced in software development within the Department of Defense (DoD), the ASWF represented a significant departure from traditional military practices and set a new standard for talent development and future force design (Errico & Zuniga, 2022). The literature revealed that the ASWF successfully promoted a scalable organic capability, enabling autonomous Soldier-led teams to discern requirements and engineer software solutions

applicable across the Army (Command, 2023b). These efforts led to the development of applications and tools used by more than 25,000 Soldiers, illustrating the breadth of the ASWF's impact. A key achievement attributed to the ASWF was establishing the Army's first accredited development, security, and operations platform, CREATE (Coding Repository and Transformation Environment) (Iyer, 2021). CREATE, operating in support of the Army Software Factory, was acknowledged as a significant milestone in the Army's evolution of its software development capabilities.

The ASWF was recognized for establishing unclassified and classified cloud environments with commercial cloud service providers under the name cArmy and for implementing a centralized contract for discounted procurement of cloud services. The ASWF's training programs were found to encompass a variety of fields, including Software Engineering, UX/UI Design, Product Management, and Platform Engineering, augmented by Agile and Lean 6 Sigma management principles, user-centered design, and the application of Agile DevSecOps in software development (Errico & Zuniga, 2022). The ASWF's alignment with digital transformation strategies characteristic of Fortune 500 companies was suggested to have enabled the operationalization of cloud technology, Agile methodologies, and DevSecOps within the Army (Command, 2023a). A novel approach highlighted by Gerstenberger (2022) was the ASWF's strategy of enabling Soldiers to control the entire process, from problem submission through discovery and framing to cloud hosting, development, and maintenance. This Soldier-centric methodology further distinguished the ASWF from conventional military models. The literature also highlighted the ASWF's rigorous training and selection process, culminating in the production of fully qualified software developers (Magnuson, 2022).

Air Force Kessel Run

While the ASWF has initiated a strategic shift within the Army, adopting Agile methodologies and DevSecOps in its software development process, the Air Force's Kessel Run emerged as a similarly revolutionary initiative. It was initiated in response to the DoD's requirement for Agile and DevSecOps software development practices. The Kessel Run initiative succeeded in delivering mission-critical software to warfighters around the globe, setting a new benchmark for military software development and operations (Forces, 2022). One example that appears was that Kessel Run created the 'Jigsaw' application, which streamlined air refueling operations by providing real-time data to pilots. Jigsaw enabled pilots to access data during flights, thereby significantly improving their decision-making process and operational efficiency (Forces, 2022).

Kessel Run's endeavors mirror the ASWF's shared goal of adopting Agile and DevSecOps practices to enhance operational efficiency and mission effectiveness. The lessons and successes from Kessel Run have provided useful insights to other teams within the DoD and have inspired similar software and product development teams within other military branches (Beachkofski & Patt, 20022). The Air Force's Kessel Run initiative and the Army Software Factory exemplify the DoD's efforts to address the growing demand for agility and the efficient use of DevSecOps. These initiatives demonstrate the critical importance of modern software development practices such as Agile and DevSecOps in military operations and offer insightful perspectives for further examination of this topic.

Gaps in the Literature

The literature on DevSecOps within the DoD context had several notable gaps that merit further exploration. While a significant amount of research has investigated the principles and benefits of integrating security into the development process (Kim et al., 2016; Rahy & Bass, 2020; Zaydi & Nassereddine, 2020), there was a lack of comprehensive studies exploring the practical implementation of Agile DevSecOps in the uniquely demanding environments of the DoD (Miller et al., 2022; Uchitelle, 2021). Most existing research has focused heavily on the technical aspects of DevSecOps, with limited emphasis on the critical cultural and organizational factors that contribute to its successful implementation. The cultural element of DevSecOps may have unique implications in the DoD and military setting, where organizational cultures are less agile and decision-making and often heavily influenced by rank and position (Theofanidis & Fountouki, 2019; Walters et al., 2020).

There were also noticeable gaps in the literature concerning the challenges and opportunities associated with developing a skilled workforce for secure software development within the DoD. It is an area that remains largely unexplored and would benefit from further examination; however, this will be outside this research's scope. Existing literature on Agile and DevSecOps within the DoD context had primarily relied on case studies and anecdotal evidence rather than rigorous empirical research. The lack of academic literature has resulted in a lack of comprehensive data on the factors contributing to the success or failure of Agile DevSecOps initiatives within the DoD. The need for more empirical research is needed to add to the literature.

Conclusions

Chapter 2 extensively reviewed the literature on Agile and Waterfall approaches to software development, DevOps culture, and tooling, the introduction of DevSecOps, the SWAP study, and DoD initiatives. The literature highlighted the growing importance of DevSecOps in the DoD (Miller et al., 2022). Furthermore, the literature emphasized the need to shift security left in the SDLC for early detection and remediation of vulnerabilities, reducing costs, and enhancing team collaboration (Kim et al., 2016; Rahy & Bass, 2020; Zaydi & Nassereddine, 2020).

Despite the wealth of knowledge provided by the reviewed literature, gaps persist in understanding the strategies used by project managers when implementing Agile DevSecOps while managing software development in the DoD. As Miller et al. (2022) identified, empirical research is needed to explore the challenges, best practices, and lessons learned from organizations implementing DevSecOps. This proposed study aimed to address these gaps by examining the strategies used by project managers when adopting Agile DevSecOps software development in the DoD, offering valuable insights and guidance for other organizations seeking to adopt similar practices.

Chapter Summary

In chapter 2, the literature review provided a comprehensive analysis of the existing knowledge surrounding Agile DevSecOps and its importance in the DoD. The chapter began by examining the historical development of DevOps and its evolution into DevSecOps, highlighting the need for integrating security practices earlier in the Software Development Life Cycle (SDLC). Subtopics covered paired Programming, CI/CD pipelines, monitoring, observability, and

security. A significant portion of the literature focused on the Agile and Waterfall approaches to software development and the advantages and disadvantages of each methodology in different contexts. The importance of moving security left in the SDLC was emphasized, resulting in early detection and remediation of vulnerabilities, reduced costs, and enhanced team collaboration (Kim et al., 2016; Rahy & Bass, 2020; Zaydi & Nassereddine, 2020).

The chapter further explored the DevOps culture, tooling, and the introduction of DevSecOps, as well as how these elements influenced the development and deployment of secure software. The literature review also covered the SWAP study, which provided valuable insights into the challenges and opportunities in building a skilled workforce for secure software development. DoD initiatives aimed at fostering the adoption of DevSecOps, including the Army Software Factory and the Air Force's Kessel Run, demonstrated its effectiveness in speeding up and securing software development (Command, 2023b). Chapter 3 defines the research methodology to address the research question and purpose. The chapter discusses the exploratory qualitative research design, outlining the data collection methods, sampling strategies, and data analysis techniques used to gain insights into the strategies used by project managers when adopting Agile DevSecOps to manage software development in the DoD.

Chapter 3: Methodology, Design and Methods

The problem addressed in this study was that the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD had not been established. The purpose of this qualitative exploratory study was to explore the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. The conceptual framework covered the transition from Waterfall to Agile, DevOps, tools, practices, and culture. DevSecOps was derived from shifting security left in the SDLC. DoD traditional acquisition methods and initiatives and examples of software factories in the Army and Air Force were covered. This research aimed to enrich the understanding of software development methodologies, mainly focusing on the integration of Agile and DevSecOps approaches into the DoD SDLC.

Research Methodology and Design

The selected research design for this study was qualitative exploratory. Determining an appropriate research method to answer the posed research questions is often challenging, especially when initiating a dissertation (Winerman, 2005). Qualitative designs are commonly used when there is a need to delve deeply into the nuances of a particular problem or issue (Creswell & Poth, 2016). The relatively unexplored nature of Agile DevSecOps in the DoD's software development landscape, with minimal academic documentation, justified this methodological choice. As Edmondson and McManus (2007) have indicated, this field's early stage of development necessitates open-ended learning and environmental input, which qualitative research methods facilitate through deep, immersive interaction (Duesbery & Twyman, 2020). Qualitative research methods are implemented when dealing with data types

like audiovisual materials, documents, observations, and interviews, as pointed out by Creswell and Creswell (2018). In this study, the primary population sample was project managers who worked on software development projects using Agile DevSecOps in the DoD. Purposive snowball sampling was used, and interviews were the primary data source.

The exploratory qualitative research design provided an enriched understanding of strategies used by product managers using Agile DevSecOps when managing software development in the DoD. The snowball sampling technique leveraged the networks of identified participants to reach others who could contribute to the study. By asking initial participants to identify others who meet the study criteria, there was an increase in the breadth and depth of the data collected. Snowball sampling is a potent tool to recruit participants that can help unravel the meaning, develop understanding, and discover insights relevant to the research question. Findings from the interviews provided a basis for uncovering patterns or themes within the collected data (Burkholder et al., 2020); this study aided in uncovering strategies used. This qualitative methodology diverges from quantitative research, which primarily explores causal or correlational links between variables. Instead, this research sought to comprehend phenomena within their real-world contexts using the method of interviews (University, 2022). Despite the inherent limitations of qualitative research, such as potential biases and the dependence on participants' networks in snowball sampling, this method offered an in-depth exploration of Agile DevSecOps strategies in the DoD used by project managers. The interview data and insights significantly contributed to the expanding body of literature in this area.

Population, Sample, and Participant Recruitment

This study's population encompassed all contractors who have been involved in managing software development projects within the DoD using Agile, DevOps, or DevSecOps methodologies, or a combination of them. These contractors provided a unique perspective, comparing and contrasting their experiences across different projects, departments, or even between government and private-sector clients. The population included those who have worked for different branches of the military or DoD agencies, were involved in diverse types of software development projects, and possess varying levels of experience and tenure. Therefore, it is essential to note that this population was broad and diverse, and its exact size was challenging to determine, given the classified nature of some projects and the expansive size of the DoD. According to the Bureau of labor (2023), as of 2022, there were 843,910 Project Management specialists in the United States. Project managers may manage different types of projects in which they evaluate and organize the schedule, timeline, procurement, personnel allocation, and budgeting for a product or service on an individual project basis.

The sampling method for this study followed purposive and snowball sampling strategies. The initial sample included contractors recommended based on their relevant experience with Agile DevSecOps in the DoD. A request was made on LinkedIn to initialize contact with potential participants. Identified participants were contacted through email if acquired or the LinkedIn Messager using the participation invitation email template from Appendix B. For this study, a minimum of (12) participants were surveyed. Snowball sampling was employed as the sampling progressed, where initial participants recommended other potential candidates. This approach is consistent with McCallister's(2019) strategy and

increases the likelihood of achieving data saturation. Prospective participants were contacted and provided an overview of the study, its purpose, and its implications. Those who expressed interest were given informed consent forms in Appendix C outlining their rights, the voluntary nature of their participation, the measures taken to ensure confidentiality and anonymity, and the possibility to withdraw from the study at any time without consequences. These precautions ensured that the study met ethical guidelines and contributed to the trustworthiness and integrity of the research.

Data Collection Instrumentation and Procedures

The data collection procedures for this research study were constructed on the solid foundation of established methodologies, combining elements from various researchers (Alvarado, 2022; Ford, 2020; McCallister, 2019; Thomas, 2022), all within the guiding principles delineated by Creswell and Creswell (2018) for qualitative research. The primary data collection instrument was semi-structured interviews, as per the format outlined in Appendix A, conducted through the Zoom online platform. Details on the research protocol and initial contact are also detailed in Appendix A.

This approach, which allowed for both in-depth exploration and emergent insights, is consistent with the methodology used by Alvarado (2022). As Alvarado (2022) suggested, this approach allows for a profound exploration of the research topic while leaving room for unexpected insights to emerge. The Zoom video conferencing platform was chosen as it enables extensive reach and accessibility, especially when participants are geographically dispersed, supporting recommendations from Creswell and Creswell (2018) for using technology in data collection. While the interviews were in progress, notes were taken using a

reMarkable e-ink tablet, capturing non-verbal cues and immediate impressions to supplement the audiovisual data. This methodology mirrors Thomas's (2022) approach, further enriching the data collected.

In keeping with the methodology from Ford (2020), participants received the research questions a day in advance, allowing them to prepare and reflect. This method has been shown to enhance the depth and quality of the data collected. By weaving together the strategies and methodologies of Alvarado (2022), Ford (2020), Thomas (2022), McCallister (2019), and Creswell and Creswell (2018), this research study's data collection process was thorough, robust, and ethically sound, paving the way for a comprehensive exploration of the research question. The data analysis phase, which follows data collection, will be discussed in the next steps of the research process.

Data security was taken with upmost care in this research. All digital data, including recorded interviews, transcriptions, electronic notes, and other digital documents, were stored on a password-protected computer with access limited to the researcher. The data files were encrypted and backed up on a password-encrypted external drive. All digital and physical files were anonymized, replacing participant names with unique identification numbers. Physical data, such as handwritten notes and printed transcripts, were kept in a secure, locked cabinet only accessible to the researcher. In compliance with the university's guidelines, all data was retained securely for seven years post-research conclusion, after which digital files will be permanently deleted, and physical documents will be securely shredded. Access to the collected data will be strictly limited to the researcher. All data was thoroughly anonymized to protect participant identities if shared for academic or presentation purposes.

Data Analysis Procedures

A comprehensive and rigorous procedure utilizing NVivo qualitative analysis software was followed for the data analysis in this study (see figure 2). First, data from individual interviews was imported into NVivo as individual files for coding and aggregated into one large file to allow for mass analysis with the built-in NVivo tools such as word cloud generation and auto coding. To familiarize myself with the data, I undertook an initial review of all collected transcripts, following the approach recommended by Fusch and Ness (2015). This repeated reading provided early insights into possible themes, patterns, or anomalies, setting a strong foundation for achieving data saturation, where no new insights emerged from the data, marking its sufficiency to allow study replication (Fusch & Ness, 2015).

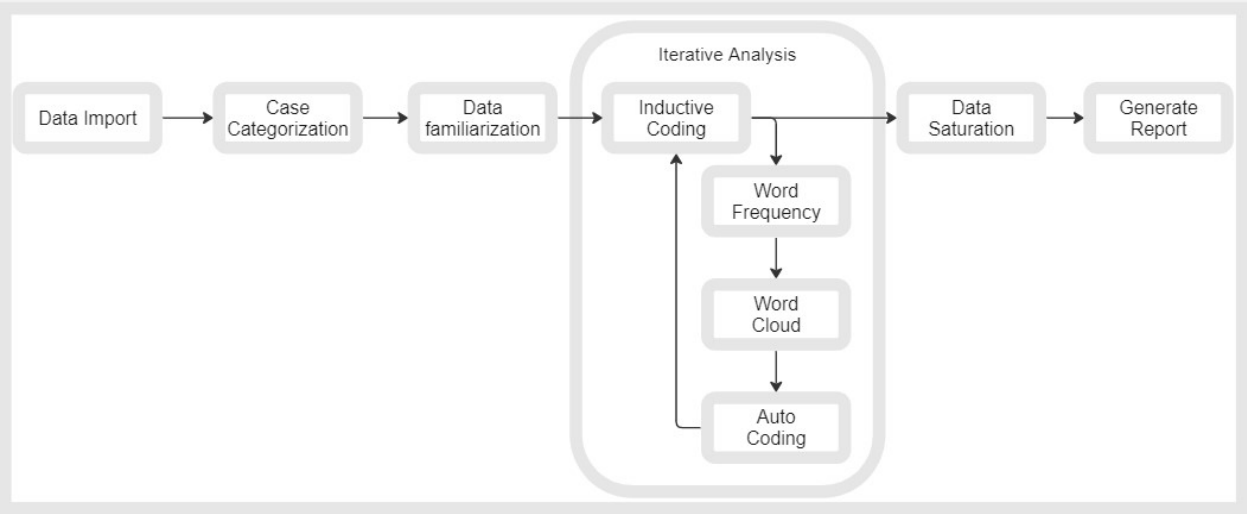
As the researcher, it was critical to maintain a mindful awareness of personal biases and their potential influence on the analysis (Creswell & Creswell, 2018; Sebastian, 2019). Therefore, the inductive coding process began with manually identifying significant codes and emerging themes using NVivo (Sebastian, 2019). Once initial codes were established, I leveraged NVivo's word frequency feature to spot commonly reoccurring words. Following the creation of the word frequency table, a word cloud was generated to visualize common reoccurring words that stood out in the data (see figure 3). Building upon the manual coding, NVivo's auto-coding function validated existing codes and suggested new ones, ensuring no potential theme was overlooked (Kuckartz & Rädiker, 2019).

This iterative approach to inductive coding was repeated until data saturation was reached, focusing on the depth and quality of information rather than sheer quantity. As part of the process, I remained particularly aware of the distinction between DevOps and DevSecOps,

recognizing that the latter integrates security into the software development life cycle. If applicable, these themes were explored in greater depth in the subsequent chapters of the research. In the final stage, all codes were validated to ensure consistency, and similar codes with variations in spelling or capitalization, such as "Agile" and "agile", were noted for accurate representation. This thorough, iterative approach to data analysis contributed to the study's reliability and replicability, allowing other researchers to duplicate the process in the future.

Figure 2

Data Analysis



Trustworthiness

To achieve trustworthiness in this study, I meticulously upheld the four pillars of credibility, transferability, dependability, and confirmability (Morgan & Ravitch, 2018). Each of these criteria contributed to the overall trustworthiness of the research. Each of those four pillars must remain supported and embraced for the entire study to be trustworthy (Gerstenberger, 2022). Trust must be placed in the academic landscape to have credibility over time in follow-on publications.

Credibility

Credibility was established in this study by accurately interpreting and representing participants' perspectives and experiences in using Agile DevSecOps in managing software development within the DoD. To create a comprehensive understanding, this was achieved through detailed descriptions, participant quotations, and multiple data sources, including interview transcripts and field notes.

Transferability

Transferability in this study was supported by providing a thorough and detailed description of the research context, the characteristics of the participants, and the Agile DevSecOps practices within the DoD. This level of detail allows readers and other researchers to discern the applicability of the findings to similar settings or populations. The widespread relevance of Agile DevSecOps practices within the DoD and other organizations further enhanced the study's transferability. A comprehensive account of the research, including in-depth descriptions of the participants, their roles, data collection processes, and the contexts from which data was collected, allowed for a better understanding of the situations that led to the findings. This transparency enables other researchers to assess the potential applicability of the findings to their respective scenarios. This study also emphasized transferability by meticulously documenting all research procedures, including participant selection, data collection methods, and data analysis techniques. Such detailed record-keeping forms a clear roadmap for any researcher who wishes to replicate the study and verify the findings. As Morgan and Ravitch (2018) recommended, an extensive audit trail was maintained, promoting transparency and enabling a thorough examination of the research process.

Dependability

The dependability of the findings in this study was bolstered by maintaining a detailed and transparent record of all research processes, decisions, and changes that occurred throughout the investigation. This ensured that the findings were consistent and could be reliably repeated by other researchers. To facilitate a clear and repeatable process, the NVivo software was employed for coding the data and managing the themes. Building on the recommendations of Thomas (2022) and Morgan and Ravitch (2018), the entire research process was meticulously documented. This includes comprehensive details on participant selection, data collection, and data analysis. The interview process was also detailed, including the research question in supporting appendices. Such thorough documentation allowed other researchers to replicate the study and verify the findings, thereby enhancing dependability.

A rigorous analysis and interview cross-verification was undertaken to validate the study's dependability further. Transcripts were cross-referenced for consistency, and recurring themes or strategies related to Agile DevSecOps were identified. This thorough analysis further guarantees the reliability of the findings. A dissertation committee review was conducted to ensure that the conclusions drawn from the research were robust and free from personal biases. The study's findings, interpretations, and conclusions were shared with experts in the field. Their feedback helped to confirm that the conclusions drawn were consistent with the data and could withstand scholarly scrutiny. This approach to ensuring dependability aligns with the principles Thomas (2022) and Morgan and Ravitch (2018) laid out.

Confirmability

To enhance the confirmability of the study, I undertook several steps, drawing from the guidance of Gerstenberger (2022), Thomas (2022)**,** and Morgan and Ravitch (2018). Firstly, I documented a detailed account of my research process, which explicitly acknowledges any potential biases that may have influenced the study. This approach aligned with the tenets of confirmability in qualitative research, as it demonstrates how the data was analyzed to derive codes and themes. I ensure the elimination of researcher bias at the data interpretation stage, providing a comprehensive record of my data handling procedures. I also made an audit trail available, empowering other researchers to follow my investigation steps. I maintained a reflective journal throughout my research to further bolster confirmability. This journal served as a comprehensive record of all activities during the research process, including personal reflections. To ensure that the interpretation of data remained true to participants' experiences, direct quotes from participants were included in the findings and used to support the thematic analysis. This strategy ensured that the research findings were rooted in the data collected and not influenced by personal preconceptions, solidifying the confirmability of the study.

Ethical Assurances

In conducting this research study, I remained committed to upholding rigorous ethical standards and protecting the rights of participants. I adhered to the ethical principles for human subjects research outlined in The Belmont Report, including respect for persons, beneficence, and justice (*The Belmont report: Ethical principles and guidelines for the protection of human subjects of research*, 1979). The principle of respect for persons involves recognizing

the autonomy of individuals to make their own decisions and protecting those with diminished autonomy. To uphold this, I obtained informed consent from each participant by clearly explaining the study's purpose, risks, and benefits see Appendix C. I ensured that participation was completely voluntary see Appendix C. The principle of beneficence means maximizing possible benefits while minimizing possible harms. I took measures to protect the confidentiality of participants by removing identifiers, using P1, P2, and P3 identifiers, and securely storing data. The risks involved were minimal. The principle of justice relates to the equitable distribution of benefits and burdens—my selection of participants involved fair subject selection procedures without systematic exclusion of certain groups. I will equitably distribute any benefits and minimize burdens on participants.

I collaborated with the Institutional Review Board (IRB) to ensure rigorous review and approval of the study's ethical procedures as outlined in the federal regulations for human subjects research (*The Belmont report: Ethical principles and guidelines for the protection of human subjects of research*, 1979). Obtaining IRB approval was essential to validate that my study design incorporated safeguards for participants' rights and welfare. Through training, knowledge of ethical principles, IRB review, and diligent implementation of protections for confidentiality, informed consent, and participant well-being, I strived to conduct this research study according to the highest ethical standards. The board played a crucial role in scrutinizing and validating the ethical considerations of studies (Thomas, 2022).

Chapter Summary

Chapter three thoroughly explained the research methodology, data collection, data analysis, trustworthiness, and ethical considerations applicable to this study. The research

methodology delineated for this investigation employed a qualitative approach, focusing specifically on interviews as a primary means of data collection. This data was analyzed via inductive procedures, a cycle that continues iteratively until data saturation is attained (Fusch & Ness, 2015). For data validation, the auto-coding feature of NVivo was employed, as proposed by (Kuckartz & Rädiker, 2019). This process included consideration of similar codes with varying verb forms to guarantee comprehensive data representation (Romero, 2022).

The study's trustworthiness was built upon the four cornerstones of credibility, transferability, dependability, and confirmability (Gerstenberger, 2022; Morgan & Ravitch, 2018; Thomas, 2022). Credibility was established via triangulation of the interviews, notes, and transcription. Transferability was achieved through exhaustive documentation of the research processes and methodology. Dependability in the study was ensured through detailed record-keeping, a process aligned with the strategies proposed by Thomas (2022) and Morgan and Ravitch (2018). For confirmability, a reflective journal was consistently maintained throughout the research, and an audit trail was provided to facilitate transparency and further research. Ethical considerations adhered to the highest standards secured through the successful completion of CITI certification and collaboration with the Institutional Review Board (IRB). The collected interview data used for this study were kept confidential and secure, both during and after the study, ensuring the principles of confidentiality, anonymity, and data security were upheld. Chapter four will delve into the results of this research approach and data analysis.

Chapter 4: Findings

The problem addressed in the proposed study was that the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD had not been established. The purpose of this qualitative exploratory study was to explore the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. This research aimed to enrich the understanding of software development methodologies, mainly focusing on the integration of Agile and DevSecOps approaches into the DoD software development lifecycle.

Description of the Study Sample

This study's population encompassed all project management contractors managing software development within the DoD using Agile DevSecOps methodologies. These contractors provided a unique perspective, comparing and contrasting their experiences across different projects, departments, and between government and private-sector clients. The population included those who have worked for different branches of the military or DoD agencies, are involved in diverse types of software development projects, and possess varying levels of experience and tenure. Therefore, it is essential to note that this population was broad and diverse, and its exact size was challenging to determine, given the classified nature of some projects and the expansive size of the DoD. According to the Bureau of labor (2023), as of 2022, there were 843,910 Project Management specialists in the United States. Project managers were defined as those who managed different types of projects in which they evaluated, organized the schedule, timeline, procurement, personnel allocation, and budgeting for a product or service on an individual project basis.

To address this problem and fulfill the study's purpose, semi-structured interviews were conducted with 12 contractor project managers with direct experience and expertise in managing software development projects using Agile DevSecOps approaches and tools within the DoD. Although initially 10 participants were planned for recruitment, two additional project managers were included in the final sample. Expanding to 12 participants provided more perspectives and allowed for potentially deeper insights to emerge during data collection and analysis, creating richer results (see table 1). The additional inclusion aligned with the flexible, exploratory nature of the qualitative approach. Alvarado (2022) indicated that leaving room for unexpected findings to surface can profoundly enrich a study's outcomes.

The key findings that emerged from rigorous inductive coding and thematic analysis of the interview transcripts are presented in this chapter. These findings are structured based on the overarching research question regarding the strategies employed by project managers employing Agile DevSecOps to manage software development in the DoD. Supporting data in the form of relevant quotes from the interviews were included to substantiate the identified themes. The findings provided rich, diverse insights into the real-world strategies and insights of project managers employing Agile DevSecOps in software development in the DoD. The results help address critical knowledge gaps regarding Agile DevSecOps implementation in the DoD, anchored in empirical data instead of anecdotal evidence.

Table 1

Participant Demographics

ID	Title	Experience (years)	College Level	Relevant Certifications	Gender	Age Range
P1	Sr. Product Manager	6	Bachelors	None	Female	25-34
P2	Platform Manager	14	Masters	Security+, Certified Scrum Masters, Certified Scrum Product Owner	Male	35-44
P3	Product Manager	5	Bachelors	CKAD, Product Strategy, Professional Scrum Master	Male	25-34
P4	Sr. Project Manager	20	Masters	None	Female	45-54
P5	Product Manager	25	Masters	None	Female	55-64
P6	Product Manager	30	Masters	CKAD, CKA	Male	55-64
P7	Sr. Product Manager	8	Masters	None	Female	35-44
P8	Product Manager	8	Bachelors	Scrum Product Manager, Pragmatic Marketing	Female	35-44
P9	Agency Director	16	Masters	None	Male	35-44
P10	Technical Manager	40	Masters	MCSE, MCSD, MCDBA, CISM, PMP, PMI-ACP	Male	60-74
P11	Product Manager	8	Associates	Certified Scrum Master, Sec +	Female	35-44
P12	Product Manager	11	Masters	Certified International Configuration Management	Female	35-44

Results

As the researcher, I conducted multiple rounds of coding on the interview data to identify emerging themes and strategies. I removed all identifiable information from the interviews and mixed up the participant identifiers before beginning the analysis to reduce potential bias. Removing identifiable information allowed me to code the data without knowing which participant made each statement. Which proved to be very true after many iterations of data coding and familiarization. I read through all the interview transcripts thoroughly before starting to code to approach the data neutrally without drawing early conclusions. During coding, I remained detached and focused on accurately representing the data rather than confirming any pre-existing expectations or biases. By coding the entire dataset before analyzing, conducting multiple coding passes, and reflecting critically on my objectivity, I aimed

to conduct a rigorous qualitative analysis that minimized bias and accurately captured the strategies employed by participants when adopting Agile DevSecOps in the DoD.

The data collected for this qualitative exploratory study was derived from ten interview questions and optional probing questions that were given 24 hours prior to the interview to the participant to give them time to prepare their answers and be able to reflect on their experiences prior to the interview. As outlined in the interview protocol, the questions were asked during semi-structured interviews as part of the data collection process (Appendix A). The ten interview questions aimed to elicit participants' insights regarding:

1. The overall approach to managing software development within their organization (project vs. product-based).

2. Whether an Agile DevOps or DevSecOps approach was adopted and what prompted it.

3. Project management methodologies used, specifically any transition from Waterfall to Agile.

4. Cultural elements influencing Agile DevSecOps adoption and execution.

5. Technologies used in the software development process.

6. Use of open-source tools and their influence on development.

7. Specific tools like CI/CD pipelines, containers, and microservices used.

8. The way security was managed in the development process.

9. Ways the DoD could improve software development with Agile DevSecOps.

10. Strategies used as a project manager to employing Agile DevSecOps.

Through inductive coding and thematic analysis of the 12 interview transcripts using NVivo 12, 47 unique codes were identified and applied 884 times throughout participants'

responses (Appendix D). The codes were consolidated into five overarching themes related to project managers' strategies for employing Agile DevSecOps in DoD software development. The five key themes were methodologies and approaches, quality and security, team dynamics and trust, feedback culture, and challenges and tradeoffs. Descriptions of the themes are listed in Table 1. The themes, associated code groupings, and frequencies are outlined in Appendix D. There was also a subtheme of enabling technologies supporting the theme of methodologies and approaches that could stand independently but may bring more value when directly tied to how it enables some of the methodologies and approaches. The themes and subthemes are represented in Table 2. The key findings within these five themes are expanded upon in the following section to provide insights into the strategies employed by project managers for implementing Agile DevSecOps in DoD software development.

Table 2

Theme and Subtheme Descriptions

Theme	**Description**
Methodologies and Approaches	Explores software development methodologies and product management techniques
Subtheme: Enabling Technologies	Focuses on supporting technologies and their enablement of methodologies and approaches
Quality and Security	Centers on code quality and security measures
Team Dynamics and Trust	Delves into team dynamics and trust across teams
Feedback Culture	Examines practices and tools for continuous feedback loops
Challenges and Tradeoffs	Discusses challenges and tradeoffs in the DoD

Theme 1 Methodologies and Approaches

The most prevalent theme across the interviews was the discussion of various software development methodologies and approaches. 12 of the 12 participants mentioned the theme during the interviews. Agile development was emphasized as foundational by all 12 participants, with DevSecOps, Lean Principles, Scrum Practices, Extreme Programming, and Kanban mentioned repeatedly. While specific methodologies varied across teams, the overarching cultural shift towards iterative, collaborative processes was clear. An important supporting subtheme was the technologies enabling the implementation of these approaches in practice. Version control, Microservices, Containers, and Continuous Integration/Continuous Deployment (CI/CD) Pipelines were consistently noted as critical tools allowing for distributed development, automated testing, delivery, and rapid iteration. Methodologies and technologies were tightly aligned and loosely coupled, with flexible frameworks leveraging modern tools to enable developer productivity, security, and team agility. This subtheme will be discussed in the following section.

The theme of methodologies and approaches was the most mentioned across the interviews by all 12 participants and coded 239 times. Key methodologies included Agile, DevSecOps, Lean, Scrum, Extreme Programming (XP), and Kanban. The predominant approach described across the interviews was product-focused, to deliver value to users over extended timeframes continuously, contrasting with a project-based approach, which is more focused on fixed-duration efforts. P2 explained that their platform team aimed to "help specific customers achieve their missions" by providing capabilities and features needed by end users. P3 also highlighted this perspective, noting: "A product is more like it is going to last forever."

To enable this product focus, the Agile development methodology was widely adopted across all 12 participants. P11 stated that "Agile is about iteration and continuous improvement." Multiple participants highlighted integrating security earlier through DevSecOps to shift security left. P2 shared that this allowed application teams to "focus on end-user value" while platform teams handled underlying security controls. P1 discussed how their organization followed lean principles and Agile product management by stating that "It follows lean methods which encourages limiting waste and reducing risk for the organization thereby saving people time, saving business money and a lot of times it is lean thinking and implemented through agile product management."

P2 explained how their team uses Agile practices like extreme programming, weekly iterations, retrospectives, and demos: "We follow the lean XP or extreme programming model with paired programming...our iterations for my team specifically are one week, so we have an iteration planning meeting...then at the end, we will have a retrospective." P3 shared how their organization adopted Agile more formally from leadership: "There is that decision coming from higher up new organization, you know specific policies and playbooks signed by our previous authorizing official to set out that model." P11 emphasized that Agile focuses on continuous improvement: "Agile is about iteration, it is about looking back at what you did and saying how do I do this better." Agile development methodologies were highlighted as foundational by all 12 participants, with practices like iterations, retrospectives, and collaboration mentioned.

Eight participants mentioned extreme programming. P2 discussed using paired programming, a practice from extreme programming: "We follow the lean XP or extreme programming model with paired programming. There is a set of two engineers, sometimes

three, to work a specific problem, and the idea is you have the real-time code reviews instead of having one person develop and then wait for somebody else to review." P11 also highlighted the value of pairing from extreme programming: "I think pairing, having this social aspect, I think that was the biggest thing I saw that really made a difference." P5 mentioned using test-driven development, another extreme programming practice by stating that "we do test-driven development and pair programming and mob programming...we write the test first before we write the code."

P6 talked about using continuous integration: "We practice continuous integration, so any code that gets checked in goes through a whole slew of automated tests." P10 discussed adopting engineering practices like pair programming and test-driven development: "We have adopted a lot of standard engineering best practices, like pair programming and test-driven development." The idea of building, measuring, and learning through rapid iteration was emphasized across the interviews. Participants discussed adopting Agile development, Lean thinking, and DevOps to enable faster feedback cycles. P1 highlighted how their organization follows lean principles to limit waste.

P2 also discussed getting feedback quickly through Agile practices like weekly iterations and retrospectives. P2 explained: "Let us get things out in front of users as quickly as possible...so we can get feedback of hey is this useful? Should we keep going with this, or do we pivot and work on something else?" P2 advocated for "fast feedback" and accepted that "failure is part of the job." P11 also stated: "Agile is about iteration; it is about looking back at what you did and saying, how do I do this better?" Specific Agile practices mentioned included Scrum, Extreme Programming, and Kanban. P2 described using one-week iterations, daily

standups, retrospectives, and paired programming. However, strict adherence to a specific methodology was not seen as important.

P3 explained that their operations-focused team found time-boxed Scrum sprints did not work as well and tailored their process using Kanban boards instead. Enabling practices like CI/CD and automation were utilized to accelerate delivery. As P2 explained, committing code changes triggered automated deployment, greatly speeding up release cycles. However, challenges in fully implementing DevSecOps were noted when interfacing with external teams not following Agile methods, which caused delays. The flexible Agile frameworks require complementary technologies to enable rapid iteration, continuous integration and deployment, and distributed development. Key enabling tools are highlighted in the subtheme of Enabling Technologies.

Subtheme 1 Enabling Technologies

Transitioning from the development methodologies and approaches, the interviews highlighted how technologies acted as key enablers for implementing Agile DevSecOps. The interview data included version control systems such as Git for distributed code collaboration, container platforms like Kubernetes for packaging and deploying microservices, and CI/CD pipelines to automate testing and releases. These tightly aligned yet loosely coupled toolsets allow Agile teams to deliver value quickly while maintaining high quality. Although the tools themselves could form an independent theme, they are tightly coupled with the methods and approaches discussed previously.

Version control systems were highlighted by 7 out of 12 participants. Git was commonly cited as a tool for enabling distributed collaboration, implementing CI/CD pipelines, and

adopting GitOps workflows. P2 explained: “We follow a GitOps model where developers will commit their code, and then we use Kapp [Controller] as a GitOps tool that will watch [the] repo, we will watch certain branches, and anytime that we see a change, [Kapp Controller] will automatically deploy that change out, committing code changes to GitLab, trigger automated testing and deployment of code changes.” P11 stated: "Git is a really great tool for DevOps because it kind of encourages this distributed model of development. It really lets us do things like having feature branches and the master branches."

Containers and Kubernetes emerged as key enabling technologies for implementing Agile DevSecOps approaches. Multiple participants highlighted how these technologies empower developer productivity, accelerate delivery, and support resilient distributed systems. P2 and P4 specifically discussed using Kubernetes as a container orchestration platform to deploy and run microservices-based applications. P2 explained: "We use Kubernetes on a cloud provider, so it is virtual machines running; you know there is an operating system with Kubernetes on top of it." Containers package up application code with dependencies, while Kubernetes handles deploying and networking containers across clusters of servers. Eight participants talked about using containers to standardize and simplify deployments across environments.

P8 said: "The use of Docker containers really helps us maintain consistency across development, testing, and production environments." Containers provide portability across infrastructure. P3 added to this discussion by stating: “Another kind of example of that is really prioritization being heavily into containers and Kubernetes, the idea of packaging applications as containers to ship software more reliably instead of having to deal with oh I have got a

server in production but acts a little different than the server in QA.” Maintaining consistency across deployment environments was noted to help enable development Agility and reduce deployment friction. Containers and orchestrators like Kubernetes were consistently cited as foundational technologies allowing teams to accelerate delivery through standardized packaging and run distributed fault-tolerant applications aligned with Agile and DevSecOps approaches.

Kubernetes was also coded under the theme of challenges and tradeoffs. Seven participants discussed adopting a microservices architecture composed of small, modular, independently deployable services. This helps enable continuous delivery and scaling of applications. Transitioning from monolithic architectures to microservices was a commonly mentioned approach. The microservices approach aligns well with Agile DevSecOps and cloud-native development. P2 discussed their platform using a microservices architecture:

> [In] Agile, you break a big problem into smaller concealable chunks to add value; you are breaking down this big monolithic application or platform into these smaller more manageable microservices so that if one service causes you a problem you can focus on that specific piece and not have to worry about breaking other parts of the platform.

P2 also explained how microservices enable swapping components without impacting other parts of the system: "You should be able to swap out a visualization software for another tool if something else better meets your need, and it should not have any impact on the other components of the architecture." P4 discussed leveraging microservices to build their applications: "We leverage microservices to build our applications so they are made up of small, modular components." P5 provided more detail on how their team leveraged microservices:

"We were using Kubernetes, but we needed to support a lot of the software development teams to use microservices. Some had already initiated their project in that manner, and some did not, so we were supporting both." The Agile methodology and its associated practices, such as Scrum, extreme programming, and Kanban, were consistently cited as foundational across teams, even if specific implementations varied. Enabling technologies like version control, containers, Kubernetes, and CI/CD pipelines were critical tools allowing developers to collaborate, iterate rapidly, and continuously deliver value. While challenges integrating legacy systems and processes exist, the overarching cultural shift towards flexible, collaborative development processes was clear across the interviews.

Theme 2 Quality and Security

Quality and security were a key theme that emerged from the participant interviews. All 12 participants mentioned the importance of being in sync with security to operate in the DoD space. Five participants emphasized the importance of automated security scanning and analysis tools like SonarQube and Fortify to catch vulnerabilities early in development. P10 explained:

> The first thing it does is it goes through the source code analyzers SonarQube and Fortify, and then the Fortify reports are then harvested and sent back to the team's Slack channel…it is possible to have a merge request get approved, and then a Fortify flag the code for vulnerability and then we feed that back into Slack so the developer becomes aware.

P3 and P11 also discussed how these tools enable faster feedback loops and can be integrated into developer workflows. Five participants emphasized extensive testing

throughout the development lifecycle, using tools like Selenium, Postman, Burp Suite, and ZAP. P10 stated: "We talked about our static application security tools, and then we also have two dynamic application security tools which we run Burp Suite, which is a common standard, and the other we run is ZAP." P10 also highlighted test automation using frameworks like Selenium to reduce manual testing efforts. P5 and P7 highlighted the value of standardized frameworks and starter apps to accelerate development while baking in baseline security controls. P1 said: “Some organizations and industries value engineers knowing how to create secure code…you have to think about competencies and how you want your people to grow if you want them to grow as security-minded people”. P1 further expanded on reducing security friction:

> We decided we would actually just abstract [security] away and put that into the starter application put a lot of the basic boilerplate security fundamentals for every app we just put that into the starter application, and so teams now are doing a little bit less security engineering than they might have a year ago.

10 of the 12 participants discussed the importance of integrating security specialists early in the process, though challenges were noted in breaking down silos between developers and security teams. P6 suggested techniques like getting security staff to write user stories from the persona of end users to encourage collaboration. P3 proposed including security engineers directly on feature teams to build on their “existing success and become even better.” Six participants emphasized security documentation and compliance with standards, including STIGs and NIST. However, P8 and P12 noted inconsistencies between development teams as areas for improvement. P12 questioned security documentation management:

> Should it just go in line with the code? Should it be just all in one big document repository? How does that get structured? Is it a combination? Should [the security documentation] only be owned by the app team, and then they just kind of let the security team know where it is?

Inconsistencies in common security practices proved to be a challenge. P1 and P3 also highlighted the need to balance security with other priorities like speed of delivery. P1 stated that: “It can feel sometimes like security is at odds with fast product development.” P3 said, “There is a lot of debate among the different teams eventually… it was brought over to the cyber team, and they definitely scrutinized it.” Participants highlighted extensive, automated testing, early security integration, standardized frameworks, and a focus on compliance as key elements in merging quality and security practices into a streamlined Agile DevSecOps workflow in the DoD. However, silos between developers and security teams persist as a challenge to be addressed, along with finding the right balance between security and speed.

Theme 3 Team Dynamics and Trust

Enabling open communication, building trust, cross-functional teams, informal collaboration, and psychological safety were highlighted across interviews as important for improving team dynamics, as 11 of the 12 participants mentioned. P2 discussed the importance of open and candid communication: "It is a kind of a culture we are trying to foster empathy putting yourselves in user's shoes...we need to be able to talk about it openly and try to create a blameless culture." They emphasized creating a blameless environment where problems can be discussed without blame, ensuring psychological safety. P3 also highlighted their team's shift to Kanban, which enabled more fluid collaboration compared to rigid sprints as opposed to

Scrum, by stating that “[Scrum] did not work well for us.” Ten participants emphasized the importance of team interaction, collaboration, and open communication.

P2 and P5 highlighted pairing practices, while P10 and P11 discussed using tools like Slack and informal communication channels to enable "osmotic communication" and awareness between team members. P11 strongly stated open communication as they explained the concept of having a main video call room:

> We started doing a thing where internally, as a team, for a couple of hours during the day, we would all dial into our kind of main room for our team video conferencing room, and this all came out of retro, actually, by the way, like just little things about little complaints or things that we are working with are not working well, and we came up with this thing where we would all work together, we would go out into breakout rooms based on what stories we are working on, and it was just a very psychologically safe place where I was kind of allowed to go into the breakout rooms with the teams and just see what they were working on and those sorts of things so I felt like that really helped accelerate the team's performance with those factors.

P5 added to the importance of easy access to collaboration environments and their importance to being Agile: “This whole idea behind Agile and having these constant daily sync ups, we were geographically dispersed having a Zoom link or any kind of meeting link Google link whatever where people would come to collaborate and work and so even as a product manager between meetings we could go in there and to really get a pulse as to what is going on with the group.” Practices like informal communication, social interactions, and psychological safety were noted as helping team dynamics and performance.

When they were all asked the question from the protocol: “How would you describe the overall approach to managing software development within your organization – is it more project-based or product-based?” They all answered identically by saying that they were product-based. The interviews highlighted that taking a product-based approach focused on outcomes over process enables more effective software delivery. 10 of the 12 participants discussed adopting a product-focused approach centered on outcomes rather than finite projects. As P3 explained, their organization designates teams as "product teams" to maintain and incrementally improve platforms over time rather than completing defined projects. Validating product direction through early user feedback was also emphasized.

P2 advocated quickly getting capabilities in front of users to validate a product's usefulness and make any necessary pivots. They explained that organizations using Waterfall approaches could benefit from earlier user feedback to avoid wasted effort on unused products. P1 gave an example of how adopting Agile practices helped their team increase goal achievement from 50% to 80-90% per quarter by enabling faster pivoting based on outcomes. The data from the interview showed that fostering open communication, psychological safety, and informal collaboration helps build trust and improve team dynamics, though challenges bridging silos between teams persist. Adopting product-focused approaches centered on outcomes over process enables more effective software delivery through rapid validation and pivoting. While specific practices vary, the overarching cultural shift towards flexible, collaborative team structures and environments was emphasized across participants.

Themes 4 Feedback Culture

The importance of continual feedback emerged as a key theme across many interviews. Participants emphasized practices like user-driven development, CI/CD pipelines, retrospectives, and automation as critical for enabling rapid iteration and improvement. 10 of the 12 Participants consistently emphasized practices like soliciting user feedback, automating through CI/CD pipelines, and conducting retrospectives as instrumental for embedding rapid feedback loops, enabling continuous incremental improvement versus delayed feedback and long development cycles. A culture focused on feedback ties the Agile processes together through transparency, user-centricity, automation, and reflection. Continual feedback loops emerged as an essential priority across the interviews, with participants highlighting various practices as critical for enabling rapid iteration and improvement.

Nine participants emphasized user-driven development as a key way to gather feedback and validate assumptions early and often. As P2 explained: "From the product side, we want to make sure that we are giving our customers the features that they need to be successful, and that has really been done through you listening to user feedback, taking feature requests, and actually setting defined goals." P6 also noted the importance of "getting user feedback and being able to pivot your product as you are developing it in a way that meets your user needs better." Incorporating user feedback throughout the development process allows for course correction based on real user needs.

CI/CD pipelines were highlighted by 10 of the participants as enabling automated, repeatable processes for rapidly and reliably building, testing, and deploying applications. As P12 explained: "The value that you really get out of that is the automation, the repeatable

processes." Three participants discussed how CI/CD pipelines facilitate continuous delivery across their products. P9 stated: "For the most part, a CI/CD pipeline is used across every single one of our products." Integrating security tools and testing into the pipelines was noted by 2 participants. P11 mentioned: "having test-driven development," and P5 discussed pipelines that "had to go through a toolchain to pass all the security stuff." P9 provided significant detail on leveraging CI/CD and security tools: "with established thresholds or some type of scans or tests" to enable automated security checks and real-time risk management. P1 and P5 noted some implementation gaps, but 4 participants highlighted the shift towards CI/CD, especially in government contexts. P7 explained: "We are always working to move our clients towards CI/CD pipelines and particularly with government institutions."

Overall, CI/CD pipelines were seen as a critical Agile DevSecOps practice by 10 participants, enabling a transition from slower, manual releases to rapid, automated, and secure continuous delivery. Participants consistently emphasized practices like user-centric development, CI/CD, retrospectives, and test automation as critical for embedding rapid feedback loops within their Agile DevSecOps implementations. This enabled continuous incremental improvement versus delayed feedback from long development cycles. Soliciting continual user feedback, implementing CI/CD pipelines, and conducting retrospectives were consistently cited as instrumental practices for embedding rapid feedback loops. Automating processes through CI/CD and test frameworks accelerates delivery while maintaining quality and security. Transitioning to a culture focused on transparency, user-centricity, automation, and reflection facilitates continuous incremental improvements versus delayed feedback.

Theme 5 Challenges and Tradeoffs

Managing open-source technologies and dependencies was cited by 7 participants as creating obstacles in their DevSecOps implementations. The need to properly vet open-source licenses and versions was an area of concern noted by multiple participants when adopting these tools. As P10 explained: "I cannot run containers on in what we call our sandbox environment, and I can run them on the DREN if that is legal for us to do. I cannot run them on SIPR or JWICS." The use of open source presents tradeoffs between the flexibility and innovation these communities enable and the diligence required to ensure compliance, compatibility, and security in government networks. Seven participants acknowledged the growing reliance on open source in modern software delivery but highlighted lingering challenges in leveraging these technologies within highly regulated environments such as the DoD. Kubernetes, in particular, was mentioned by P12, who mentioned: "So CI/CD pipelines yes, Microservices yes, Containers are a very useful technology, but you have to make sure whatever container orchestration system using is supported by your environment.” The participant showed hesitation in the support for Kubernetes in regulated environments.

P2 mentioned frustration with "upstream cloud providers" and "critical services" that do not support containers and Kubernetes. According to four participants, the immaturity of Kubernetes for some regulated contexts makes integration and implementation complex. The cutting-edge nature of Kubernetes can conflict with the risk-averse culture of many government agencies, as highlighted by these participants. The cutting-edge nature of Kubernetes can conflict with the risk-averse culture of many DoD agencies. Finding the right balance between

innovative technologies like Kubernetes and enterprise security/stability requirements remains an ongoing challenge based on the experiences highlighted in these interviews.

The DoD's highly regulated environment and low tolerance for risk pose unique tradeoffs and obstacles for adopting innovative Agile DevSecOps practices, according to the challenges highlighted by 4 participants. As P1 explained: "In the private sector, high-risk tolerance is ok, we can break go fast and break things and in the DoD low-risk tolerance we cannot... we don't want to lose lives, we don't want to risk our national defense." P1 stated:

> I feel like at the end of the day, there needs to be a coming to Jesus moment of, like ok, DoD software, what's our risk tolerance? Can we accept a little bit more risk than we used to so that we can move more quickly with critical software?

The difference between P1's statements showed that low-risk tolerance could be in good intent for the DoD but also slows down the quick delivery of software, which is necessary to implement methodologies such as Agile DevSecOps. P1 noted the dichotomy between the flexibility and speed enabled by DevSecOps in the private sector and the controlled, risk-averse culture of the DoD. P2 explained: "Where we have much frustration is when we [come] up against external organizations [such as] upstream cloud providers that we rely on for critical services... [they] do not follow an Agile methodology and the process tends to slow down."

The need to balance agility through Agile DevSecOps with the security and stability imperatives for mission-critical systems was emphasized. Given the life-and-death nature of many DoD systems, change inherently carries risk, as noted by the participants. However, they acknowledged the efforts underway in various DoD organizations to find the right equilibrium between speed and security. Managing open-source tools like Kubernetes creates challenges.

These tools are new and change rapidly. Teams must make sure they follow the rules and are secure. The DoD has many strict rules that limit risks. These rules can slow down new tools and ways of working, like Agile DevSecOps. Nevertheless, DoD teams are trying to find the right balance between speed and security. Using very new tools always involves tradeoffs. There will be obstacles to work through. However, participants see the DoD making progress in adopting modern software methods, even for sensitive defense systems.

Discussion of Study Findings

An examination of the study findings in relation to the current scholarship is presented below. The data collected within this research is compared to what was previously discovered in existing literature. The discussion of study findings focuses on the major themes that emerged from the highest frequency codes in participant responses to the semi-structured interview questions. The 12 participants offered insights gained from their experiences as project managers adopting Agile DevSecOps in the DoD. The key themes included methodologies and approaches, quality and security, team dynamics and trust, feedback culture, challenges and tradeoffs, and a subtheme of enabling technologies. A word cloud visualization is introduced in Figure 3 to complement the discussion of the study findings and provide an alternative perspective on the key terms and concepts that emerged from the interviews.

Figure 3

NVivo Interview Word Cloud

The word cloud was created to represent the frequency of words in the collected data visually. By examining the size of the words in the word cloud, I could quickly identify prominent words that occurred more frequently in the data, which provided valuable insights into the emerging themes. Larger words in the word cloud indicated higher frequency, while smaller words represented less common occurrences. This visualization proved to be an effective tool for uncovering patterns and potential themes within the data, as it allowed me to spot reoccurring words that stood out easily.

In exploring the themes discovered and the word cloud, I found that the words with higher frequency, such as people, trying, security, kind, team, organization, product, and

development, played a significant role in visualizing the overarching themes in the study. The words “team” and “organization” emphasized the importance of collaboration and communication in Agile DevSecOps environments, a key aspect of the Team Dynamics and Trust theme. “Product” is connected to the theme of Methodologies and Approaches, which explores various software development methodologies and product management techniques. “Security” highlighted the focus on code quality and security measures, which is central to the theme of Quality and Security. The words “trying” and “people” indicated the challenges and tradeoffs project and product managers face when implementing Agile DevSecOps in the DoD, relating to the theme of Challenges and Tradeoffs. By analyzing the frequency of these words in the word cloud, I was able to supplement the identification of the key themes and patterns that emerged from the data, helping me answer the research question.

Theme 1 Methodologies and Approaches

The interview findings related to the prevalence of Agile methodologies align closely with what is covered extensively in the literature review. The literature highlighted the growing adoption of Agile approaches like Scrum and Extreme Programming in software development, especially in contrast to traditional Waterfall methods (Beck et al., 2001b; McQuade et al., 2019). The benefits of Agile highlighted in the interviews, such as rapid iteration, continuous integration and deployment, and gathering user feedback, mirror the advantages noted in sources like Beck et al. (2001c) and the Agile Manifesto. The emphasis on shifting security left through DevSecOps also connects strongly with the literature review's discussion of integrating security earlier in the SDLC (Kim et al., 2016; Rahy & Bass, 2020; Zaydi & Nassereddine, 2020).

The finding that DevSecOps enables application teams to focus on delivering value while platform teams handle security aligns with sources like Crook et al. (2020) that note DevSecOps facilitates development teams concentrating on features while security is embedded by design. The references to Extreme Programming practices like paired programming and test-driven development match what is covered in sources like Gerstenberger (2022) and Bolhuis et al. (2022) on the role of extreme programming in Agile development. The tailored use of methods like Scrum and Kanban based on team needs also reflects the flexibility encouraged in the Agile principles that the literature covers (Beck et al., 2001c). The importance of CI/CD and enabling tools is also extensively discussed concerning implementing Agile approaches, connecting closely to what Marini-Wear (2019) and Finster (2021) cover regarding essential DevOps tooling. The prevalent theme of Agile methodologies matches what existing literature establishes as the growing emphasis on iterative approaches over traditional Waterfall methods in contemporary software development. The findings confirm evidence and real-world examples of the Agile transition described in sources like Institute (2021) and Beck et al. (2001c).

Subtheme 1 Enabling Technologies

The existing literature strongly supports the findings related to enabling technologies like version control, containers, Kubernetes, and CI/CD pipelines. The emphasis on distributed version control systems like Git matches what sources like Finster (2021) and the DoD DevSecOps Playbook (Defense, 2021b) discuss regarding the role of version control in DevOps. Git enabling collaboration, implementing pipelines, and adopting GitOps workflows align closely with what the literature review covers on leveraging Git for code management. The literature

review also extensively discusses the vital role of containers and Kubernetes—sources like Lopez Garcia et al. (2020), Birkeland et al. (2020), and the DevSecOps Playbook highlight the benefits of containers and orchestrators like Kubernetes in packaging, deploying, and managing microservices. The standardization, portability, and resilience findings match the advantages noted in sources like Marini-Wear (2019). Transitioning to microservices architectures is also covered in the literature review, with sources like Siewruk and Mazurczyk (2021) and the DevSecOps Playbook (Defense, 2021b) describing the alignment with Agile approaches. The modular nature enabling continuous delivery and scaling connects to points made in sources like Alnafessah et al. (2021). The literature's key theme is the use of enabling tools and technologies to support cultural shifts like Agile and DevSecOps methodologies. For example, Finster (2021) emphasized the automation technologies complementing iterative processes. The findings confirm evidence of the tight coupling between methods like Agile and tools like version control, containers, and CI/CD pipelines covered in the literature review. The findings related to enabling technologies strongly reinforce what existing sources have established regarding these tools' vital role in effectively implementing Agile, DevOps, and DevSecOps in practice. The findings provide real-world examples and evidence supporting the linkage between approaches and technologies highlighted in the literature.

Theme 2 Quality and Security

The theme of quality and security aligns closely with the literature review's emphasis on shifting security practices earlier in the development lifecycle through DevSecOps. The discussion of automated security scanning tools like SonarQube and Fortify matches sources like Lopez Garcia et al. (2020), which covered static and dynamic testing tools to catch

vulnerabilities during development. The goal of faster feedback loops is consistent with the literature's theme of "shifting security left" through DevSecOps (Kim et al., 2016; Rahy & Bass, 2020). The focus on extensive, automated testing throughout the lifecycle also connects to points made in sources like Finster (2021) and the DoD DevSecOps Playbook (Defense, 2021b).

Regarding comprehensive test automation in CI/CD pipelines. Using standardized frameworks and starter apps to bake in security aligns with Orechovesky's (2021) findings on the value of container platforms in improving cybersecurity practices. The challenges around integrating security teams and balancing security with speed of delivery reflect issues noted in sources like Eze et al. (2022) and Crook et al. (2020) regarding organizational obstacles in adopting DevSecOps. The inconsistencies in security documentation also relate to the DoD Enterprise DevSecOps Strategy Guide (Defense, 2021b) point about a need for software development practices that meet industry standards for agility. The quality and security theme confirms evidence for the literature's emphasis on taking a proactive approach to security through practices like extensive automation, standardized frameworks, and early integration of security specialists. The findings reinforce what studies like Orechovesky (2021) and Lopez Garcia et al. (2020) have established regarding the role of DevSecOps in shifting security left. The challenges noted also reflect known obstacles discussed in sources like Eze et al. (2022).

Theme 3 Team Dynamics and Trust

The findings related to enabling open communication, building trust, cross-functional teams, and psychological safety connect strongly to points made in the literature review regarding the cultural shifts in adopting Agile and DevSecOps approaches. Sources like Díaz et al. (2021), Khan et al. (2022), and the DoD DevSecOps Playbook (Defense, 2021b) emphasized

the importance of promoting collaboration and integration among development, security, and operations teams. The discussion of blameless cultures and psychological safety reflects Díaz et al.'s (2021) point about culture playing a fundamental role in successful DevOps environments and Luz et al.'s (2019) point about cooperation over silos. Informal collaboration practices like pairing, “mob programming,” “osmotic communication,” and shared Slack channels match Jones' (2019) and Gerstenberger's (2022) coverage of collaborative programming, stating that when paired, extreme programming can foster innovation.

The challenges bridging team silos also relate to Eze et al.'s (2022) findings on cultural obstacles in DevSecOps adoption. The product-based approach ties back to Beck et al.'s (2001b) points in the Agile Manifesto about customer collaboration and responding to change. The emphasis on outcomes over process also connects to the Institute's (2021) coverage of Agile's adaptability over predictive methods like Waterfall. The examples of P2’s increased goal achievement support Walters et al.'s (2020) findings on Agile increasing development speed. The cultural findings reinforce existing literature regarding the teamwork, collaboration, and customer-centric view needed to enable Agile DevSecOps approaches. While specifics like Kanban differ from the literature, the overarching themes of flexibility, openness, and focusing on outcomes rather than process match the cultural transformations described in sources like Díaz et al. (2021) and Luz et al. (2019). The findings confirm evidence for the literature's emphasis on these cultural elements.

Themes 4 Feedback Culture

The theme of continual feedback loops connects strongly to core Agile and DevOps principles covered extensively in the literature review. The emphasis on user-driven

development reflects the literature's discussion of gathering user feedback and responding to change as key Agile values (Beck et al., 2001c). Sources like McQuade et al. (2019) also highlighted user-centricity. CI/CD pipelines were a central focus of the literature review, with multiple sources (Defense, 2021b; Finster, 2021; Marini-Wear, 2019) all describing their role in enabling automated testing, rapid delivery, and repeatable fast feedback. Retrospectives, although not directly mentioned in the literature review, align with the Agile Manifesto's principle: "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly" (Beck et al., 2001c).

Examples of increased speed and quality tie back to points made in sources like Walters et al. (2020). The overall theme of feedback loops reinforces core Agile and DevOps principles that are extensively discussed in sources like the Agile Manifesto (Beck et al., 2001c) and (Finster, 2021). The specific practices noted in the findings confirmed evidence and demonstrated real-world implementation of the continual improvement literature emphasizes as critical. While specifics like CI/CD gaps differ, the broader emphasis on soliciting user feedback, automating via CI/CD, and regularly reflecting matches the literature's coverage of instilling a culture focused on transparency, user-centricity, automation, and reflection. The findings reinforce what sources like Beck et al. (2001c) establish as essential to Agile approaches.

Theme 5 Challenges and Tradeoffs

The findings related to managing open-source technologies and dependencies, especially Kubernetes, connect to some of the gaps and challenges discussed in the literature review. The emphasis on properly vetting open-source licenses and versions matches what

Usman et al. (2022) covered regarding the complexities of emerging architectures like Kubernetes. The cutting-edge nature of these technologies conflicts with the risk-averse cultures of the DoD. Kubernetes maturity issues noted by multiple participants relate to the literature review's discussion of Kubernetes still requiring additional tools for full observability, as highlighted by Usman et al. (2022). The challenges of balancing innovation with security and stability in the DoD environment connect to the gaps around implementing Agile DevSecOps in government environments noted by Miller et al. (2022), who stated that adopting these practices in complex and highly regulated environments, such as the DoD, presented unique challenges. Uchitelle (2021) further proves the need for more research, stating that there is a lack of comprehensive studies exploring the practical implementation of Agile DevSecOps in the uniquely demanding environments of the DoD.

The findings provide real-world examples of the tradeoffs discussed regarding the cultural element of DevSecOps, which have unique implications in the DoD setting, where organizational cultures are less agile and decision-making and often heavily influenced by rank and position (Theofanidis & Fountouki, 2019; Walters et al., 2020). While specifics like Kubernetes licensing differ, the broader theme of managing open-source innovation versus stability and compliance aligns with gaps like Usman et al.'s (2022) coverage of Kubernetes observability challenges and Miller et al.'s (2022) point about lacking DevSecOps implementation research in government contexts. Overall, these findings confirm that despite the growth of Agile DevSecOps, challenges remain around emerging and open-source technologies, especially in regulated environments like the DoD. The findings serve to reinforce these gaps in the literature.

Additional Findings

The data collected from interviews with project managers adopting Agile DevSecOps in the DoD revealed several notable findings not fully explored in the literature review. While the literature provides a broad understanding of Agile, DevOps, and DevSecOps principles and practices, there remain gaps in knowledge regarding the specific strategies employed for implementing these approaches within the unique context of DoD software development. The findings from this study contribute new insights that help fill these gaps and further the understanding of Agile DevSecOps adoption strategies in the DoD.

One key finding from the data is the importance of taking an incremental, iterative approach to implementing Agile DevSecOps rather than simultaneously attempting a major transformational shift. The literature does not delve deeply into this implementation strategy for regulated environments like the DoD. However, participants emphasized starting small with pilot projects, demonstrating value and buy-in, and scaling practices more widely. The data revealed the significance of extensive upfront planning and alignment on vision, goals, and metrics when adopting Agile DevSecOps in the DoD. While the literature covered the importance of planning, the specific alignment challenges in a complex bureaucratic environment like the DoD warrant further examination. Participants noted that without diligent, collaborative planning across teams and leadership, Agile DevSecOps adoption could falter.

Cultural challenges also featured strongly in the findings; an area less developed in literature. Adapting to the Agile mindset and ways of working, overcoming resistance to change, and fostering greater collaboration and trust were cited frequently by participants as

obstacles to Agile DevSecOps adoption in the DoD. Further exploration of cultural transition strategies could provide value. Governance challenges balancing Agile approaches with DoD oversight and regulations emerged as unexplored concepts. Participants noted difficulties reconciling Agile values, like responding to change with DoD requirements such as documentation and reviews. The literature does not substantially address this governance paradox, representing an area for additional research. By exploring these initial findings in greater depth through future studies, researchers can further explore the nuances and complexities of adopting Agile DevSecOps within regulated government environments. The findings of this study lay the groundwork for future research focused on optimizing Agile DevSecOps strategies for the unique needs of DoD software development.

Chapter Summary

The findings from interviews with DoD contractor project managers provided insightful data on the strategies for adopting Agile DevSecOps in software development in the DoD. Through rigorous analysis, five key themes and one subtheme emerged. Several overarching conclusions can be drawn. First, Agile development methodologies enabled faster feedback, continuous improvement, and iterative value delivery through practices like extreme programming and retrospectives. Second, enabling technologies are an essential component to enable Agile software development. Version control systems, containers, and CI/CD pipelines are key enablers for implementing Agile DevSecOps approaches by empowering developer productivity, accelerating delivery, and supporting resilient distributed systems. Third, practices like automated security scanning, extensive testing, early security integration, and compliance with standards can reduce the burden on software development teams.

Challenges remain with persisting silos between developers and security teams, as well as balancing security with speed of delivery. Fourth, open communication, trust, collaboration, and psychological safety were highlighted as improving team dynamics. Product-focused approaches centered on outcomes over process enable effective software delivery through rapid validation and pivoting. Fifth, Continuous feedback loops through user-driven development, CI/CD pipelines, retrospectives, and test automation enable rapid iteration and improvement. A culture focused on transparency, user-centricity, automation, and reflection facilitates continuous incremental improvement. Sixth, Adopting innovative Agile DevSecOps practices involve balancing speed with strict security requirements in regulated DoD environments. The data findings create an empirically grounded foundation for practice, policy, and theory. Chapter 5 will further connect the findings to prior literature and examine implications.

Chapter 5: Discussion and Conclusions

The problem addressed in the proposed study was that the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD had not been established. The purpose of this qualitative exploratory study was to explore the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. This research aimed to enrich the understanding of software development methodologies, mainly focusing on the integration of Agile DevSecOps approaches into the DoD software development lifecycle.

Limitations of Study Findings

This study had several limitations that may have impacted the findings. The sample size of 12 participants, while sufficient for an exploratory qualitative study, showed data saturation but limited the generalizability of the findings. A larger sample could reveal additional insights. One major limitation was the focus solely on project managers within the DoD. While these individuals played a significant role in managing software development using Agile DevSecOps, their perspectives and strategies may not fully represent or capture the holistic view of the process. There may have been nuances and important details that other stakeholders involved could have provided, such as developers, security professionals, and end-users. Another limitation is the potential self-report bias during the interviews. Participants may have tended to provide responses they believe are socially desirable or favorable to their professional reputation rather than accurately reflecting their true practices and experiences.

The study's scope focusing only on Agile DevSecOps implementation within the DoD was done intentionally to bring focus on the gap in the literature as referenced in the gaps in the

literature section, while a significant amount of research had investigated the principles and benefits of integrating security into the development process (Kim et al., 2016; Rahy & Bass, 2020; Zaydi & Nassereddine, 2020). There was a lack of comprehensive studies exploring the practical implementation of Agile DevSecOps in the unique environments of the DoD (Miller et al., 2022; Uchitelle, 2021). This intentional focus enabled an in-depth exploration of this specific environment. Although other frameworks like Lean and Kanban were omitted, they came up in the interviews. Exploring them would have yielded additional findings. As an exploratory qualitative study, it did not assess causality or test hypotheses. Follow-up confirmatory research is needed to build on these initial findings.

The study relied on self-reported data from interviews, which can be subject to recall bias. Triangulation with other data sources could have improved validity. The rapidly evolving nature of Agile DevSecOps practices means the findings represent a snapshot in time and may not capture more recent developments. Researcher bias and subjectivity in data analysis and interpretation are inherent in qualitative research. Steps were taken to mitigate bias, but it remained a limitation. The study relied solely on interview data. Including other data sources like documents or observations could enhance triangulation.

Interpretation of Study Findings

This exploratory qualitative study addressed gaps in understanding the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. As highlighted in the literature review, prior research on Agile and DevSecOps implementation has largely focused on the private sector, with limited investigation into strategies for adoption within the DoD. The findings from the 12 interviews with DoD

project managers provided valuable insights that helped fill this gap and expand knowledge of Agile DevSecOps strategies tailored to the unique aspects of DoD software development. Five key themes emerged from the data analysis that align with and build upon the existing literature. The research question for this study was: “What are the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD?.” By exploring these strategies and challenges, this study offers valuable information that can inform future research and practice in software development, security, and delivery within the DoD.

Theme 1 Methodologies and Approaches

The interview findings revealed that Agile development methodologies were widely adopted across participants' teams as a foundational approach. Practices like iterations, retrospectives, and collaboration were consistently mentioned. There was flexibility in the specific methodology used, with Scrum, Extreme Programming, and Kanban all referenced. The tailored use of these methodologies based on team needs reflects the adaptability encouraged in Agile principles (Beck et al., 2001b). Enabling technologies like version control, containers, and CI/CD pipelines are critical tools that allow developers to collaborate, iterate rapidly, and continuously deliver value. While challenges integrating legacy systems and processes exist, the overarching cultural shift towards flexible, collaborative development processes was clear. The prevalent callout of Agile methodologies matches what existing literature establishes as the growing emphasis on iterative approaches over traditional Waterfall methods in contemporary software development (McQuade et al., 2019).

Subtheme 1 Enabling Technologies

The interview findings revealed that technologies like version control systems, containers, and CI/CD pipelines were consistently cited by participants as critical enablers for implementing Agile DevSecOps approaches. Distributed version control systems like Git align with the literature on their role in DevOps (Finster, 2021). Containers and orchestrators like Kubernetes were highlighted as providing key benefits like standardization, portability, and resilience, connecting to points made in sources like Marini-Wear (2019). The findings provide real-world examples supporting the literature on the tight coupling between Agile methodologies and enabling tools and technologies.

Theme 2 Quality and Security

The findings revealed the importance of integrating security practices throughout the software lifecycle rather than as an afterthought. Using automated security tools for early feedback aligns with the literature's emphasis on "shifting security left" in DevSecOps (Kim et al., 2016). However, challenges like persisting silos between developers and security teams were noted, reflecting issues identified in sources like Eze et al. (2022). The findings confirm evidence in the literature on proactively addressing security through practices like extensive automation and early security integration.

Theme 3 Team Dynamics and Trust

The findings highlighted the importance of open communication, trust-building, and informal collaboration in improving team dynamics, connecting to points in the literature regarding needed cultural shifts in adopting Agile and DevSecOps (Díaz et al., 2021). Adopting practices like Extreme Programming and focusing on outcomes over process facilitates effective

software delivery through rapid validation, aligning with sources like Beck et al. (2001b). The findings reinforce the literature's emphasis on flexibility, openness, and teamwork as essential cultural elements.

Theme 4 Feedback Culture

Continual feedback loops were emphasized across findings, reflecting core Agile and DevOps principles covered extensively in the literature. Specific practices like user-driven development, CI/CD, and retrospectives provide real-world examples of instilling a culture focused on transparency, user-centricity, automation, and reflection. The findings confirm evidence from sources like Play 7 of the DoD DevSecOps Playbook (Defense, 2021b). (2001b) on the value of continuous feedback loops.

Theme 5 Challenges and Tradeoffs

Managing innovative yet rapidly evolving open-source tools posed a key challenge, connecting to points in the literature about maturity and regulatory issues with technologies like Kubernetes (Usman et al., 2022). The findings provided nuance around balancing agility with stability and compliance in regulated DoD environments. The findings align with gaps identified by sources like Miller et al. (2022) regarding implementing DevOps government, reinforcing the literature gaps around emerging tools and regulated environments.

Practice Implications of Study Findings

The purpose of this qualitative exploratory study was to explore the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. This research enriched the understanding of software development methodologies, mainly focusing on the integration of Agile and DevSecOps approaches into the

DoD software development lifecycle. Based on the findings of this study, project and product managers in the DoD should consider the following implications and ways forward to improve Agile DevSecOps in software development.

Implication 1:

It is essential to understand the unique challenges and opportunities that may arise when implementing Agile DevSecOps in the DoD, including addressing the complexity and regulation. Agile DevSecOps practices must be tailored to fit the specific needs and requirements of the DoD, which may differ from those in the commercial sector.

Implication 2:

Project and product managers should focus on changing organizational culture and mindsets to be more collaborative, flexible, and open to experimentation. Cultural shifts include providing extensive training and coaching in Agile and DevSecOps values and principles, empowering teams to drive changes, and incentivizing based on delivering outcomes rather than outputs.

Implication 3:

Taking an iterative approach to software development, starting small by piloting Agile DevSecOps on low-risk projects to demonstrate benefits before scaling, is crucial. This involves planning frequent deliverables in short sprints, prioritizing a minimum viable product, and leveraging user feedback to improve software continuously.

Implication 4:

Integrating security from the start, involving security engineers early in development to shift security left and make security a shared responsibility, is essential. Integration includes

automating security scanning, testing, and compliance checks in CI/CD pipelines and educating all team members on secure coding practices.

Implication 5:

Aligning Agile DevSecOps practices with existing DoD policies and evolving acquisition models to support Agile processes while maintaining compliance and oversight is necessary. Alignment involves demonstrating the benefits of new approaches to leadership and helping change policies that create bottlenecks. By understanding and addressing these challenges, project and product managers can leverage the findings to improve Agile DevSecOps in software development in the DoD.

Researcher Reflections

During my dissertation research, I gained significant insights into my development as a scholar-practitioner and the importance of staying focused on the core research problem. I encountered various distractions throughout the research process, such as industry whitepapers and social media discussions. However, I remembered the importance of concentrating on my primary research goal, exploring the strategies employed by project managers who managed software development in the DoD using Agile DevSecOps. Some key takeaways from my experience include maintaining strong connections between my research activities and the core issues under examination and ensuring high-quality scholarship.

The importance of thorough preparation before data collection, including keeping my references updated and conducting an up-to-date literature review. I mitigated my biases through consistent self-reflection and acknowledgment, focusing on active listening during interviews, and setting up a dedicated, distraction-free environment for interviews. Finally, I did

not attach myself too firmly to the draft of the research prospectus since feedback is always welcome, and others looking at my paper may not have the same thoughts as me. Being flexible and Agile proved key to my success. By adhering to these principles, I discovered research findings firmly rooted in and supported by the data, ensuring a comprehensive understanding of the topic. This experience has taught me valuable lessons that I will carry with me throughout my academic and professional career.

Recommendations for Further Research

Based on the findings from this study, the following recommendations are made for further research to expand understanding of Agile DevSecOps implementation strategies tailored to the DoD.

Recommendation 1:

Conduct an in-depth case study of a DoD software development team successfully adopting Agile DevSecOps. Utilize interviews, direct observations, and document analysis to provide rich insights into how the team overcame challenges and optimized practices for their environment. Perform a comparative analysis of multiple DoD software teams at different stages of Agile DevSecOps adoption to identify key patterns and variances in strategies across DoD organizations. A multiple case study approach comparing teams could illuminate effective and ineffective strategies.

Recommendation 2:

Explore the role of leadership in enabling Agile DevSecOps success in DoD software teams through a phenomenological study of leaders' experiences and perspectives. Interviews

with leaders across levels could provide insights into how their support and commitment influence adoption.

Recommendation 3:

Investigate the cultural transition process in DoD software teams adopting Agile DevSecOps using an ethnographic approach. Observing team dynamics and culture could reveal how values and mindsets evolve during adoption.

Recommendation 4:

Conduct action research by partnering with a DoD software team to implement Agile DevSecOps and study the process collaboratively. Active participation in adoption activities could produce practical insights.

Recommendation 5:

Examine strategies for integrating external teams using different development approaches into DoD Agile DevSecOps environments through a Delphi study engaging expert perspectives.

Recommendation 6:

Develop a change management framework tailored to Agile DevSecOps adoption in DoD using input from project managers through focus groups. These recommended studies would build upon this research by providing greater breadth and depth of understanding regarding Agile DevSecOps implementation strategies for DoD software development. The findings could help produce models, frameworks, and practical guidance to enable more effective adoption approaches in this complex environment.

Conclusion

This dissertation explored the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. The research was conducted through semi-structured interviews with 12 project managers in the DoD who have experience employing Agile DevSecOps in software development projects in the DoD. The problem addressed in the proposed study was that the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD had not been established. The purpose of the proposed qualitative exploratory study was to explore the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD. The central research question that guided this study was: What are the strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD?

To answer this question, I conducted semi-structured interviews with a purposive snowball sampling of 12 project managers in the DoD. The data collected from these interviews was analyzed using thematic analysis techniques to identify strategies for adopting Agile DevSecOps in software development in the DoD. The study produced several notable findings regarding the strategies employed by project managers when implementing Agile DevSecOps in the DoD. Additionally, the study discovered gaps in the current literature about practical findings that can be used for further exploration.

The findings from interviews with DoD contractor project managers provided insightful data on the strategies for adopting Agile DevSecOps in software development in the DoD. First, Agile development methodologies enabled faster feedback, continuous improvement, and

iterative value delivery through practices like extreme programming and retrospectives. Second, enabling technologies are an essential component to enable Agile software development. Version control systems, containers, and CI/CD pipelines are key enablers for implementing Agile DevSecOps approaches by empowering developer productivity, accelerating delivery, and supporting resilient distributed systems. Third, practices like automated security scanning, extensive testing, early security integration, and compliance with standards can reduce the burden on software development teams.

Challenges remain with persisting silos between developers and security teams, as well as balancing security with speed of delivery. Fourth, open communication, trust, collaboration, and psychological safety were highlighted as improving team dynamics. Product-focused approaches centered on outcomes over process enable effective software delivery through rapid validation and pivoting. Fifth, Continuous feedback loops through user-driven development, CI/CD pipelines, retrospectives, and test automation enable rapid iteration and improvement. A culture focused on transparency, user-centricity, automation, and reflection facilitates continuous incremental improvement. Sixth, adopting innovative Agile DevSecOps practices involves balancing speed with strict security requirements in regulated DoD environments. The data findings create an empirically grounded foundation for practice, policy, and theory.

This research made several valuable contributions. It adds to the limited scholarly knowledge regarding Agile DevSecOps implementation strategies in the environment of the DoD. The findings provide actionable insights to inform practice, guiding project managers in the DoD and other regulated sectors to adopt Agile DevSecOps successfully. The study has

some limitations that provide opportunities for further research. The sample was limited to project managers and did not include developers, security experts, and end users. Including a wider range of stakeholders could offer additional insights. The findings also may not generalize fully beyond the DoD to other settings.

Further research could examine strategies in other complex regulatory environments. A quantitative study validating the success of identified strategies could be beneficial. This exploratory study provides an in-depth understanding of the strategies employed by project managers to adopt Agile DevSecOps for software development in the DoD. The findings offer practical guidance to practitioners seeking to implement Agile DevSecOps in complex, regulated environments while contributing valuable knowledge to the academic literature in this developing field. This research can serve as a foundation for ongoing scholarship and practice in Agile DevSecOps.

References

Alliance, A. (2018). *Agile 101*.

Allspaw, J., & Hammond, P. (2009). *Velocity 09: John Allspaw and Paul Hammond, "10+ deploys per day"*. Youtube.

Alnafessah, A., Gias, A. U., Wang, R., Zhu, L., Casale, G., & Filieri, A. (2021). Quality-aware DevOps research: Where do we stand? *IEEE Access*, *9*, 44476-44489. https://doi.org/10.1109/access.2021.3064867

Alvarado, S. F. (2022). *Department of Defense cybersecurity maturity model certification compliance: The impact on small business defense contractors* (Publication Number 29324025) [D.C.S., Colorado Technical University]. ProQuest One Academic. Colorado.

Barnes, I. (2018). Implementation of active cyber defense measures. *Homeland security affairs*.

Beachkofski, B., & Patt, D. (20022). Kessel Run shows how to bridge the gap between development and operations. *War on the rocks*.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001a). History the Agile manifesto.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001b). *Manifesto for Agile development*.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S.,

Schwaber, K., Sutherland, J., & Thomas, D. (2001c). Principles behind the Agile manifesto.

Beetz, F., & Harrer, S. (2022). GitOps: The evolution of DevOps? *IEEE Software*, *39*(4), 70-75. https://doi.org/10.1109/MS.2021.3119106

The Belmont report: Ethical principles and guidelines for the protection of human subjects of research. (1979). E. U.S. Department of Health, and Welfare.

Birkeland, P., Cockroft, A., Edwards, D., Kernaghan, C., Kissler, C., Nasello, S., Olshausen, R., & Sherwin, J. (2020). It's time for ERP disruption: Applying Wardley mapping and DevOps techniques to the ERP ecosystem. *The DevOps Enterprise Journal*, *2*(1), 160-190.

Bolhuis, W. M. v., Bernsteiner, R., Hall, M., & Fruhling, A. (2022). Enhancing IoT project success through Agile best practices. *ACM Transactions on Internet of Things*, *4*(1), 1-31. https://doi.org/10.1145/3568170

Burkholder, G., Cox, K., Crawford, L., & Hitchcock, J. (2020). *Research design and methods*. SAGE

Command, A. F. (2023a). *Army Software Factory catalog*.

Command, A. F. (2023b). Year in review 2023. 34.

Creswell, J. W., & Creswell, J. D. (2018). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed.). VitalSource.

Creswell, J. W., & Poth, C. N. (2016). *Qualitative inquiry and research design: Choosing among five approaches*. SAGE Publications.

Crook, J., Johnson, S., Koehnemann, H., Shupaclk, J., Spear, S., Yasar, H., & Yeman, R. (2020). Applied industrial DevOps 2.0. *The DevOps Enterprise Journal*, *2*(1), 7-20.

Defense, D. o. (2021a). *DevSecOps fundementals guidebook*.

Defense, D. o. (2021b). *DevSecOps playbook*.

Defense, D. o. (2021c). *DoD enterprise DevSecOps strategy guide*.

Defense, D. o. (2022). *Department of Defense software modernization* (Vol. 1). Department of defense.

Dempsey, K., Chawla, N., Johnson, L., Johnston, R., Jones, A., Orebaugh, A., Scholl, M., & Stine, K. (2011). *Information security continous monitoring (ISCM) for federal information systems and organizations*.

Díaz, J., López-Fernández, D., Pérez, J., & González-Prieto, Á. (2021). Why are many businesses instilling a DevOps culture into their organization? *Empirical Software Engineering*, *26*(2). https://doi.org/10.1007/s10664-020-09919-3

Diaz, J., Perez, J. E., Lopez-Pena, M. A., Mena, G. A., & Yague, A. (2019). Self-service cybersecurity monitoring as enabler for devsecops. *IEEE Access*, *7*, 100283-100295. https://doi.org/10.1109/access.2019.2930000

Drew, K. (2020). *Government software development project failures: A qualitative approach* (Publication Number 27958098) [DM/IST, University of Phoenix]. ProQuest One Academic.

Duesbery, L., & Twyman, T. (2020). 100 Questions and Answers about action research. In. SAGE Publications, Inc. https://doi.org/10.4135/9781544305455

Edmondson, A., & McManus, S. E. (2007). Methodological fit in management field research. *Academy of Management Review*, 1155-1179. https://doi.org/10.5465/amr.2007.26586086

Errico, V., & Zuniga, J. (2022). For Soldiers by Soldiers. *Army AL & T*, 130-134.

Eze, T., Khanand, N., & Onwubiko, C. (2022). Procedings of the 21st european conference on cyber warfare and security. Chester England.

Factory, A. S. (2022). *Army Software Factory annual report 2021*.

Finster, B. (2021). An integrated approach to enterprise improvement [Journal]. *The DevOps Enterprise Journal of Enterprise Information Management*, *3*(1), 1-21.

Forces, A. a. S. (2022). Making the Kessel Run. *Air and Space Forces Magazine*.

Ford, S. W., Jr. (2020). *Reducing voluntary turnover rates via motivational strategies for top-tier project team members in Denver, Colorado* (Publication Number 28255647) [D.M., Colorado Technical University]. ProQuest One Academic. United States -- Colorado.

Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate : The science behind DevOps : building and scaling high performing technology organizations. In (First edition. ed., pp. 1 online resource (220 pages)). IT Revolution,.

Fusch, P. I., & Ness, L. R. (2015). Are We There Yet? Data Saturation in Qualitative Research. *The Qualitative Report*, *20*(9), 1408-1416.

Gerstenberger, J. (2022). *Action research study on Soldier led software development to reduce security vulnerabilities* (Publication Number 29211747) [D.C.S., Colorado Technical University]. ProQuest One Academic.

Goljan, J., Ritschel, J. D., Drylie, S., & White, E. (2021). Cost-of-delay: A framework for Air Force software factories. *Air & Space Power Journal*, *35*(4), 47-57.

Initiative, O. C. (2023). *About the open container initiative*.

Institute, P. M. (2021). *A guide to the project management body of knowledge (PMBOK guide)* (7 ed.). Project Management Institute.

Iyer, R. (2021). The digital army of the future. *Army AL & T*, 30-35.

Iyer, R. P. (2022). The imperative. *Army AL & T*, 8-13.

Jones, A. J. (2019). *Factors for successful Agile collaboration between UX designers and software developers in a complex organisation* (Publication Number 27745552) [Ph.D., University of Chester (United Kingdom)]. ProQuest One Academic.

Khan, M. S., Khan, A. W., Khan, F., Khan, M. A., & Whangbo, T. K. (2022). Critical challenges to adopt DevOps culture in software organizations: A systematic review. *IEEE Access*, *10*, 14339-14349. https://doi.org/10.1109/access.2022.3145970

Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. (2016). *The DevOps handbook : How to create world-class agility, reliability, and security in technology organizations* (First edition. ed.). IT Revolution Press, LLC,.

Kubernetes. (2023). *Getting started*.

Kuckartz, U., & Rädiker, S. (2019). *Analyzing qualitative data with MaxQDA* (1, Ed.). Springer Publications.

labor, B. o. (2023). *Occupational employment and wage statistics*.

Lopez Garcia, A., De Lucas, J. M., Antonacci, M., Zu Castell, W., David, M., Hardt, M., Lloret Iglesias, L., Molto, G., Plociennik, M., Tran, V., Alic, A. S., Caballer, M., Plasencia, I. C., Costantini, A., Dlugolinsky, S., Duma, D. C., Donvito, G., Gomes, J., Heredia Cacha, I., Ito, K., Kozlov, V. Y., Nguyen, G., Orviz Fernandez, P., Sustr, Z., & Wolniewicz, P. (2020). A cloud-based framework for machine learning workloads and applications. *IEEE Access*, *8*, 18681-18692. https://doi.org/10.1109/ACCESS.2020.2964386

Luz, W. P., Pinto, G., & Bonifácio, R. (2019). Adopting DevOps in the real world: A theory, a model, and a case study. *Journal of Systems and Software*, *157*, 110384. https://doi.org/10.1016/j.jss.2019.07.083

Magnuson, S. (2022). *Army Software Factory touts early success*. National Defense Magazine.

Marini-Wear, N. (2019). *Qualitative case study software security in DevOps* (Publication Number 27997079) [D.Sc., Capitol Technology University]. ProQuest One Academic.

McCallister, A. (2019). *The technical challenges of implementing gamification: A qualitative exploratory case study* (Publication Number 13808537) [D.I.T., Capella University]. ProQuest Central. United States -- Minnesota.

McQuade, M., Murray, R., Louie, G., Medin, M., Pahlka, J., & Stephens, T. (2019). *Software is never done: Refactoring the aquisition code for competitive advantage*.

Miller, A., Giachetti, R., & Van Bossuyt, D. (2022). Challenges of adopting DevOps for combat systems development environment. *Defense Acquisition Research Journal*, *29*(99), 22-48. https://doi.org/10.22594/dau.21-870.29.01

Mills, J., & Birks, M. (2017). *Introducing qualitative research*. SAGE Publications.

Morgan, D. L., & Ravitch, S. M. (2018). Trustworthiness. *Sage encyclopedia of educational research, measurement, and evaluation*, 1729-1731. https://doi.org/10.4135/9781506326139 (Sage Publications Inc)

National Research, C., Division on, E., Physical, S., Computer, S., Telecommunications, B., & Committee for Advancing Software-Intensive Systems, P. (2010). *Critical code : Software producibility for defense*. National Academies Press.

O'Brian, S. P., & Green-Mack, A. D. (2018). Taking Agile all the way. *Army AL & T*, 109-114.

Office, U. S. G. A. (2021). *DoD faces risks and challenges in implementing modern approaches and addressing cybesecurity practices* [Report](GAO 21-351). G. A. Office.

Orechovesky, A. D. (2021). *Linux capabilities: Autonomously securing LXC containers within the DevSecOps pipeline* (Publication Number 28772160) [D.Sc., Capitol Technology University]. ProQuest One Academic.

Otwell, C. E. (2022). *DevSecOps: Design science research of the devsecops technology stack, definitions, concepts, and improvements identified thereof* (Publication Number 29257252) [D.C.S., Colorado Technical University]. ProQuest One Academic.

Quillen, N. C. (2022). *Tools engineers need to minimize risk around CI/CD pipelines in the cloud* (Publication Number 29059797) [D.I.T., Capella University]. ProQuest One Academic.

Rahy, S., & Bass, J. (2020). Overcoming team boundaries in Agile software development. *Journal of International Technology and Information Management*, *29*(4), 20-49.

Ravitch, S. M., & Riggan, M. (2016). *Reason & rigor: How conceptual frameworks guide research* (3rd ed.). SAGE Publications.

Romero, M. A. (2022). *Improving disaster planning in North Texas through emergency manager development: A qualitative study* (Publication Number 28968749) [D.M., Colorado Technical University]. ProQuest One Academic. United States -- Colorado.

Run, K. (2022). *Kessel Run's software pathway strategy approved*.

Sebastian, K. (2019). Distinguishing between the types of grounded theory: Classical, interpretive and constructivist. *Journal for Social Thought*, *3*(1).

Siewruk, G., & Mazurczyk, W. (2021). Context-aware software vulnerability classification using machine learning. *IEEE Access*, *9*, 88852-88867. https://doi.org/10.1109/access.2021.3075385

Stanley, J. (2019). *Setting the standard: An in-depth analysis of the growing need towards regulated security standards for embedded software devices utilized in military networks* (Publication Number Dissertation/Thesis) ProQuest Dissertations Publishing].

Stark, S. (2022). Before software, there were computers. *Army AL & T*, 114-119.

Theofanidis, D., & Fountouki, A. (2019). Limitations and delimitations in the research process. *Perioperative nursing*, 155-163. http://doi.org/10.5281/zenodo.2552022

Thomas, R. (2022). *Strategies technology managers need to improve organizational growth in the technology industry: An exploratory study* (Publication Number 29325497) [D.M., Colorado Technical University]. ProQuest One Academic. United States -- Colorado.

Uchitelle, E. (2021). The past, present, and future of Rails at GitHub [Journal]. *The DevOps Enterprise Journal*, *3*(1), 45-54.

University, S. (2022). *Qualitative vs quantitative research*.

Usman, M., Ferlin, S., Brunstrom, A., & Taheri, J. (2022). A survey on observability of distributed edge & container-based microservices. *IEEE Access*, *10*, 86904-86919. https://doi.org/10.1109/access.2022.3193102

Walters, A., DeGrandis, D., Moore, J., Shupack, J., Nygard, M., Clanton, R., Grinnel, B., & Thrasher, P. (2020). Bold moves you can make actions for positive transformation. *The DevOps Enterprise Journal*, *2*(1), 23-30.

Winerman, L. (2005). *Choose your research methods wisely*.

Zampetti, F., Tamburri, D. A., Panichella, S., Panichella, A., Canfora, G., & Penta, M. D. (2022). Continuous integration and delivery practices for cyber-physical systems: An interview-based study. *ACM Transactions on Software Engineering and Methodology*. https://doi.org/10.1145/3571854

Zaydi, M., & Nassereddine, B. (2020). DevSecOps practices for an Agile and secure IT service management. *Journal of Management Information and Decision Sciences*, *23*(2), 1-16.

Appendix A: Interview Protocol

The researcher will log into the video conferencing software 15 minutes prior to ensure connection and functionality.

Introduction

Thank you for your willingness to contribute to this study. My name is Noe Lorona, and I am a doctoral candidate at Colorado Technical University. I am conducting research on the strategies project managers utilize when integrating Agile DevSecOps in software development within the DoD. Your insights and experiences will greatly enhance this study. This conversation should take approximately 30 to 60 minutes. Before we begin, we will discuss the informed consent form.

Review of Informed Consent

Could you please confirm that you have read, understood, and signed the informed consent form, and that you are still okay with participating in this interview?

If yes: Thank you.

If no: Unfortunately, we must halt the interview. Thank you for your time.

Do you grant me permission to record this interview to ensure the accuracy of the collected data?

If yes: Thank you.

If no: Regrettably, we cannot continue with the interview. Recording is a crucial aspect of maintaining data accuracy.

Prior to this interview I had emailed you the research questions that we will be going over today. While you respond I will be taking notes and recording the Zoom call for later

reference which will be secured and kept confidential. Do you have any additional questions before we begin the interview?

If yes: Answer the questions as approximately and record the answers on notes.

In no: Thank you, let us begin.

Questions and Probing Questions

1.Interview Question: Are you serving as a project manager or a product manager in your role within the DoD, and does your organization approach software development from a project-based or product-based perspective?

Probing Question: Could you elaborate on how this orientation (project or product) influences your Agile DevSecOps implementation strategy and practices?

2.Interview Question: Could you share your responsibilities and experiences in implementing Agile DevSecOps within the DoD?

Probing Question: Can you recall a specific project where you utilized unique strategies to successfully incorporate Agile DevSecOps?

3. Interview Question: What were the challenges you have faced during the incorporation of Agile DevSecOps in your DoD software development projects?

Probing Question: How did you address these challenges?

4. Interview Question: Could you discuss the advantages you have observed from Integrating Agile DevSecOps into your projects?

Probing Question: Can you provide specific instances or projects where these benefits were notably visible?

5. Interview Question: How have Agile DevSecOps practices influenced project management within the DoD's software development?

Probing Question: Could you share a project where the integration of Agile DevSecOps had a significant impact on the project outcome?

6. Interview Question: What steps should the DoD take to improve Agile DevSecOps practices, in your opinion?

Probing Question: What advice would you give to other project managers in the DoD who are trying to implement Agile DevSecOps?

7. Interview Question: Could you describe the process of transitioning from traditional project management methodologies to Agile DevSecOps within the DoD?

Probing Question: How did you navigate the challenges associated with this transition?

8. Interview Question: How has Agile DevSecOps impacted collaboration within your project team?

Probing Question: Can you provide an example where Agile DevSecOps played a critical role in enhancing team collaboration?

9. Interview Question: Could you talk about specific technologies you have used while implementing Agile DevSecOps?

Probing Question: Have you made use of any open-source technologies in your Agile DevSecOps implementation?

10. Interview Question: Was there any special training provided to you or your team prior to or during the implementation of Agile DevSecOps?

Probing Question: How beneficial was the training in navigating the implementation process and overcoming any potential difficulties?

11. Interview Question: Was this the first time your organization attempted to implement Agile DevSecOps?

Probing Question: How did the first-time experience shape the process and outcomes of the project?

12. Interview Question: From your experience, how has the implementation of Agile DevSecOps improved or otherwise impacted the software development process in your organization?

Probing Question: Could you share specific aspects or areas of your projects that have seen significant improvement due to the implementation of Agile DevSecOps?

Closing Remarks:

Thank you very much for your invaluable contributions to this research study. Within a week's time, I will provide you with a transcript of our conversation for your review. Please feel free to make any corrections or adjustments as you see fit to ensure that your insights and experiences have been captured accurately. Do not hesitate to reach out if any follow-up questions or reflections arise. Once the research is complete, I will be more than happy to provide you with a digital copy of the final report, should you wish to read it. I appreciate the time you have taken and the insight you've provided for this study.

Interview protocol of this study:

1. Project managers in the DoD or that have worked for the DoD, potential participants for this research, will receive an invitation for participation through electronic mail, a LinkedIn message, or telephonic communication.

2. Interested and available candidates will be scheduled for an online interview, anticipated to last approximately 30-60 minutes, at a time suitable to their respective schedules. These individuals will receive a consent form for their review, signature, and return to confirm their agreement for participation.

3. Prospective participants will receive reminders prior to the scheduled interview date to manage any changes in availability and to replace any individuals who may become unavailable. Rescheduling will be done if needed.

4. To maintain anonymity, participants will receive unique identification codes. With consent from each participant, interviews will be recorded using a digital audio device, with important comments also noted manually as a secondary measure.

5. Upon commencement of the interview, the researcher will reintroduce themselves, reconfirm the participant's title and name, express gratitude for the participant's contribution, and reiterate the study's focus: the examination of strategies utilized by project managers when incorporating Agile DevSecOps within the DoD. The researcher will also emphasize the estimated 30-60 minute duration of the interview.

6. Before activating the recording devices, the researcher will review the consent form with the participant, ensuring comprehension and comfort with the proceedings. Should a participant choose to retract their participation at this point, their decision will be respected, and appreciation for their time will be expressed.

7. Upon agreement from the participant to proceed with the audio recording, the digital audio device and Zoom's call recording feature will be initiated to capture the conversation. This recording will be transcribed and analyzed, with an assurance given to participants that they will have the opportunity to verify the accuracy of their contributions via electronic mail.

Appendix B: Participation Invitation Email

Dear Research Participant,

My name is XXXXXX, I am pursuing my doctorate in management with a concentration in technology management at Colorado Technical University. I am conducting a doctoral research study titled: "Strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD." This research aims to understand the strategies project managers adopt when implementing Agile DevSecOps within the Department of Defense. The study involves an interview on the research topic that will last approximately 30-60 minutes.

Your participation in this research is entirely voluntary, and you have the freedom to withdraw at any point. This study maintains strict anonymity; you do not need to provide any personal or organizational identifying information. Your insights and experiences are incredibly valuable and will contribute to our understanding of Agile DevSecOps implementation in the DoD.

If you are interested in participating, please respond to this email at your earliest convenience. Upon your agreement, I will forward an electronic consent form for your review and signature. Once the form is signed and returned, we can schedule an interview at your convenience. The study period is from July 2023, through Aug 2023.

I appreciate your time and consideration.

Best Regards,

XXXXXX

Appendix C: Informed Consent Form

INFORMED CONSENT FORM – v2.2

Title of Study: Strategies employed by project managers when adopting Agile DevSecOps to manage software development in the DoD

Principal Investigator Name: Noe Lorona

CTU Email Address: XXX.XXX@student.ctuonline.edu

Contact Phone Number: XXX.XXX.XXXX

Study Purpose

You are invited to participate in a research study that I am doing for a doctoral dissertation. The goal is to understand the strategies project managers use or have used when working with Agile DevSecOps while working with software development in the Department of Defense.

Participant Criteria

This form can help you decide if you want to be part of this study. If you match these points, you are invited to join.

- You are or have been a contractor project or product manager managing software development.
- You have used Agile, DevSecOps, or DevOps to manage software development in the Department of Defense.
- You are currently working or have worked in the past managing software development in or for the Department of Defense.
- You are not a current Department of Defense civilian employee or and Active Duty service memeber.

Study Activities and Time

You will be asked to do the following in this study:

- Spend 30 to 60 minutes on a video call over zoom. Call will be recorded for later analysis.
- Share prior or current knowledge of what strategies you used as a project manager when managing software development in the Department of Defense.
- Answer questions about the effect of Agile DevSecOps in the software development project.

Benefits

Although you may not directly benefit from taking part in this study, your participation could help us understand how to make software development in the Department of Defense better.

Risks or Discomfort

Participating in this study may make you feel a bit uncomfortable when discussing your work, but always remember, you don't need to answer any question that you are not comfortable with. This study is low risk and does not involve any physical activities.

Incentives to Participate

There is no cost to participate in this study. I cannot offer any incentives or payments for your time.

Voluntary Participation

You are being asked to volunteer. You may choose not to participate in this study. You may stop the study at any time. There is no penalty. You may ask questions at any time.

Privacy

All of our conversation, recordings, notes, will be stored on a password-protected computer that only I can use. I will also save a copy on a separate drive that is password-protected. I will replace your name with a unique identifier such as P1, P2 and so forth to protect who you are. Any written notes or printed papers will be locked away in a safe place that only I have access to. You will receive a copy of your transcript for verify that it is a valid trascript of what you said. Once I finish the study, I will need to keep all the information for seven years, as Colorado Technical University requires. When that time is up, I will delete all the digital files and destroy all printed papers forever. Only I will see the information we gather. If I need to share anything for a school presentation or paper, I will make sure the info does not show who it is from. Your privacy is very important to me, and I will do everything I can to protect it.

You may ask questions. You may discuss concerns. You may contact the Principal Investigator. Contact information is noted above. Dr. Deane Desper, XXX.XXX@XXX (XXX) XXX.XXXX. You may contact the CTU IRB. You can email the CTU IRB. The email is CTUIRB@coloradotech.edu. You can visit IRB resources. You will find resources at https://careered.libguides.com/ctu/doctoral_students/irb

Participant Consent

I have read the above information. I agree to participate in this study. I am at least 18 years old. My signature confirms that I agree to participate in this study. A copy of this form has been given to me.

Signature:	**Date:**
☒ Consent for typed signature as legal signature	

Appendix D: Themes and codes

Theme	Codes	Files	Code Frequency
Methodologies and Approaches	Agile	12	36
	Build-Measure-Learn	10	56
	DevSecOps	10	28
	Product	11	31
	Extreme Programming	8	18
	Project	8	17
	Speed	5	16
	Lean	5	11
	Roadmap and Backlog	7	9
	Scrum	5	9
	Kanban	3	4
	Experiment	1	3
	Ceremonies	1	1
Subtheme Enabling Technologies	Tools	11	25
	Operations, Platform	7	29
	Containers	8	13
	Microservices	7	12
	Version Control	7	19
	Git	7	15
	Infrastructure	2	10
	Git Branching	3	4
Quality and Security	Security	12	62
	Testing	7	14
	SonarQube	5	10
	Quality	1	2
	Starter App	1	1
Team Dynamics and Trust	Collaboration, Communication	11	45
	Culture	8	26
	DevOps	4	12
	Human Element	5	7
	Outcomes	3	7
	Balanced Team	2	5
	Blockers	3	4
	DevEx	1	1
	Interviews	1	1
Feedback Culture	User-Driven Development	9	36
	CICD pipelines	10	24
	Automation	6	14
	Retrospective	1	1

Challenges and Tradeoffs	Open Source	11	21
	Kubernetes	8	20
	DoD	8	16
	Risk Tolerance, Reduce Risk	4	14
	Waterfall	8	10
	Cloud	3	5
	Friction	1	5
	Practices	4	6
	Deliver	2	3
	External Organizations	2	2
	Pain Points	1	2