DISSERTATION

HYBRID MBSE-DEVOPS MODEL FOR IMPLEMENTATION IN VERY SMALL ENTERPRISES

Submitted by

Cailin R. Simpson

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2024

Doctoral Committee:

Advisor: Steven Simske

Erika Miller
Brad Reisfeld
Ronald Sega

ABSTRACT

HYBRID MBSE-DEVOPS MODEL FOR IMPLEMENTATION IN VERY SMALL ENTERPRISES

This work highlights the challenge of implementing digital engineering (DE) practices, specifically model-based systems engineering (MBSE) and DevOps, in very small entities (VSEs) that deliver software products. VSEs often face unique challenges due to their limited resources and project scale. Various organizations have authored strategies for DE advancement, such as the Department of Defense's Digital Engineering Strategy and INCOSE's System Engineering 2035 that highlight the need for improved DE practices across the engineering fields. This work proposes a hybrid methodology named FlexOps, combining MBSE and DevOps, to address these challenges.

The authors highlight the challenges faced by VSEs and emphasize that MBSE and DevOps adoption in VSEs requires careful consideration of factors like cost, skill availability, and customer needs. The motivation for the research stems from the difficulties faced by VSEs in implementing processes designed for larger companies. The authors aim to provide a stepping stone for VSEs to adopt DE practices through the hybrid FlexOps methodology, leveraging existing MBSE and DevOps practices while accommodating smaller project scales. This work emphasizes that VSEs supporting government contracts must also adopt DE practices to meet

industry directives. The implementation of FlexOps in two case studies highlights its benefits, such as offering a stepping stone to DE practices, combining Agile, MBSE, and DevOps strategies, and addressing VSE-specific challenges. The challenges faced by VSEs in adopting DE practices may be incrementally improved by adopting a hybrid method: FlexOps. FlexOps was designed to bridge the gap between traditional practices and DE for VSEs delivering software products.

ACKNOWLEDGEMENTS

I would like to thank the following folks who have supported and assisted me in pursuing this degree:

First and foremost, I cannot thank my advisor, Dr. Steve Simske, enough for patiently teaching, supporting, and encouraging me throughout this process. Thank you to Dr. Ron Sega for believing that I was a good fit for the CSU Systems Engineering PhD Program and introducing me to Dr. Simske. Thank you to my dissertation committee members, Dr. Erika Miller and Dr. Brad Reisfeld for their thoughtful feedback and support.

Thank you to Jimmy Allen and Ryan Hooper, who sparked my interest in pursuing a PhD; without your encouragement, I don't think I would have begun this journey. I'd like to recognize the assistance and support that I received from Chase Wortman.

I will be forever grateful to the CSU Systems Engineering Department as a whole for their kind faculty, staff, and students. I could not have picked a better community of scholars to learn from.

DEDICATION

This dissertation is dedicated to my family; without their love and support this wouldn't have been possible. To my parents, thank you for sacrificing more than I'll ever know to give me the opportunity to receive a good education. To my brothers, thank you for being a listening ear and making me laugh through the hard times. To my grandparents, thank you for supporting me at every turn throughout my life.

Lastly, to my loving husband: Carl, you gave me the courage and the strength to complete this work. Thank you for believing in me more than I could ever believe in myself. I am grateful everyday that you choose me to be your partner in this life.

TABLE OF CONTENTS

ORGANIZATION OF DISSERTATION

This work is organized into six chapters to communicate the connectedness of two separate case studies through the lens of the FlexOps process. Many VSEs don't have the luxury to do a "clean implementation" of DevOps or MBSE. The research was born out of an experienced difficulty of small entities to implement a full-scale DevOps or MBSE processes while starting a software organization. One of the objectives of this work and its subsequent organization is to give examples of how organizations may incrementally implement elements of each paradigm whether they are doing so from scratch or if they are already utilizing some elements of either or both processes. Each case study is a stand alone technical paper that is preceded by a background section and followed by a discussion section, further elaborating the FlexOps process context.

A detailed description below is outlined to help guide the reader:

- **Chapter 1. Introduction**: this chapter outlines the problem statement and motivation for the research.
    - *1.1 Problem Statement*
    - *1.2 Motivation*
- **Chapter 2. Literature Review**: this chapter details a literature review showcases relevant existing paradigms for implementing MBSE and DevOps in general and particularly in VSEs.
- **Chapter 3. Overview of FlexOps**: this chapter includes an INCOSE Symposium published conference paper introducing the FlexOps method and an accompanying background and discussion.
    - *3.1 Background*
    - *3.2 Scalable, Flexible Implementation of MBSE and DevOps in VSEs: Design Considerations and a Case Study (Simpson & Simske 2023)*
    - *3.3 Discussion*

- **Chapter 4. Case Study 1: Fiber Morphology Automated Metrics**: this chapter includes a Society of Imaging Science and Technology published conference paper with an accompanying background and discussion section.
    - *4.1 Background*
    - *4.2 Fiber Morphology Analysis for Directed-Energy Deposition Manufacturing Process (Simpson et al 2019)*
    - *4.3 Discussion*
- **Chapter 5. Case Study 2: RAW Video Analysis Framework**: this chapter includes an unpublished paper with an accompanying background and discussion section.
    - *5.1 Background*
    - *5.2 A RESTful Framework for Analyzing RAW Video in Common Open-Source Multimedia Document*
    - *5.3 Discussion*
- **Chapter 6. Discussion, Summary, and Conclusions**: this chapter provides a closing discussion, summary, and conclusion of both works through the lens of the FlexOps method.
    - *6.1 Discussion*
    - *6.2 Summary*
    - *6.3 Conclusion*

LIST OF ACRONYMS

ASOT	authoritative source of truth
DAU	Defense Acquisition University
DE	Digital Engineering
DevOps	development operations
DoD	Department of Defense
IEC	International Electrotechnical Commission
INCOSE	International Council on Systems Engineering
ISO	International Organization for Standardization
MBSE	model-based systems engineering
OOSEM	Object-Oriented Systems Engineering Method
R&D	research and development
ROI	return on investment
SAFe	Scaled Agile Framework
SERC	Systems Engineering Research Council
SBIR	Small Business Innovation Research
V&V	verification and validation
VSE	very small entity
WG	working group

Chapter 1. Introduction

1.1 Problem Statement

1.1.1 Background

As we push further into the so-called 4th Industrial Revolution, software systems have increased in complexity but the expectations for development time and cost for systems have remained the same. As engineering project complexity increases along with accompanying development time and cost, industry, academia, and government alike look to digital engineering (DE) practices as the solution. The Defense Acquisition University (DAU) defines DE as "an integrated digital approach that uses authoritative sources of systems' data and models as a continuum across disciplines to support lifecycle activities from concept through disposal" (DAU 2023). Industry and government organizations are putting out strategies like the Department of Defense's (DOD) *Digital Engineering Strategy* and the International Council on Systems Engineering's (INCOSE) *System Engineering 2035* that detail existing challenges and provide potential approaches on how to achieve ideal future DE practices (DoD 2018, INCOSE 2022).

Two popular software engineering practices are MBSE and DevOps which are used to implement DE strategies into a business's project management and engineering practices. MBSE provides a methodology that has documented perceived reduced cost and meet schedules through system modeling and cross- organizational and discipline collaboration (Henderson & Salado 2021). For software systems, DevOps provides a method of continuous delivery and improvement of software. These paradigms both have accompanying benefits and challenges with adoption and execution. This work will focus on the benefits and drawbacks of implementing MBSE and DevOps practices in VSEs that deliver software products.

1.1.2 Intended Scope

The scope of this work is focused on software engineering entities that are engaged in very little MBSE and DevOps activities or none at all. The model presented is intended as a stepping stone to implement MBSE and DevOps practices within their organization but is not intended as a "best practice" implementation of either approach.

1.1.3 VSE Challenges

VSEs tend to develop products that are integrated into larger products or systems made by larger enterprises. Most international and industry project management and product development standards like the International Organization for Standardization (ISO) standards are tailored to larger businesses. According to an article published by the American Society of Mechanical Engineers (Crawford 2023), some of the top issues faced by small businesses in 2023 are:

1. Inflation (Business costs)
2. Supply chain disruptions
3. Having enough skilled engineers
4. Sustainability
5. High customer expectations
6. Technology implementation

All of these issues affect project management strategies in VSEs. However, when it comes to picking and establishing a project execution strategy, cost, skilled workers, high customer expectations, and technology implementation are all factors to consider. Both DevOps and MBSE require implementation cost, skilled engineers, and technology implementation to meet customer expectations. These project execution strategies have their benefits and drawbacks, but have the advantages of being customer-communication and product-delivery centric.

A standard has been developed by Working Group 24 (WG) for project management in VSEs: ISO/IEC 29110. ISO/IEC 29110 established the Systems Engineering Basic Profile, shown in Figure 1.1, that provides a model for VSEs to implement systems engineering project management processes. Houde et al. provided various methods of implementation to accompany each element of the model called deployment packages (Houde et al 2016). There are many examples and case studies of VSEs incorporating MBSE processes like Houde et al's case study for how to incorporate MBSE practices by incorporating SysML requirements diagrams (Houde et al. 2016). However, there are not many examples of models of how to incrementally implement processes as a VSE matures and provides an example of what the "minimum steps required" are.

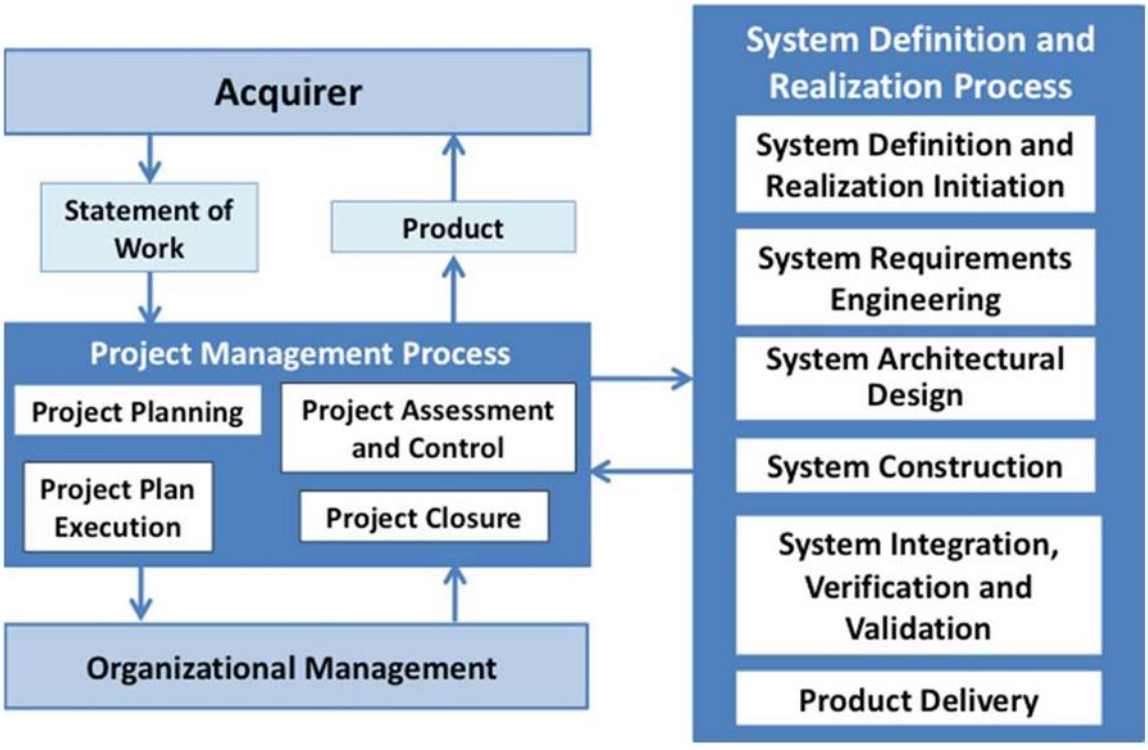

Figure 1.1. The Systems Engineering Basic Profile (Laporte 2014)

There are also examples of DevOps implementations in small businesses. Bucena and Kirkova propose a DevOps implementation method and maturity metrics (Bucena & Kirkova 2017).

However, DevOps methods alone lack a mapping back to the requirements that drove the continuous improvement and continuous deployment that DevOps is popular for.

With one of the main goals of DE being creating an authoritative source of truth (based on requirements) to create models, a VSE can get lost in the weeds trying to implement either MBSE or DevOps in an attempt to meet current and future industry standards. These methods are difficult to implement in larger businesses, even with proper coaching and resources.

1.1.4 MBSE and DevOps Adoption Challenges

Hamunen outlines 4 general-level challenges with DevOps adoption as (Hamunen 2016):

- Lack of awareness in DevOps
- Lack of support for DevOps
- Implementing DevOps technology
- Adapting organizational processes to DevOps

Chami et al. list many adoption challenges for MBSE (Chami et al., 2018):

- Upfront investment
- Adoption strategy
- Purpose and scope definition
- Awareness and change resistance
- Executive level sponsorship
- Method definition and extension
- Modularity and reusability
- Complexity management

- Tool dependency and integration
- Large models visualization

1.1.5 The Need for a "Stepping Stone" to DE Practices

Although both Hamunen and Chami et al. provide a path forward for addressing these challenges, neither provide a solution specific to VSEs. VSEs certainly can learn from the solutions provided, but may not have a project with the scope or the company resources to implement either methodology in its entirety.

1.2 Motivation

This work was born out of the difficulty experienced in implementing processes that are designed for larger companies in VSEs that deliver software products; in particular, VSEs that support government contracts. Most VSEs that support government contracts are not the prime contractor: they are subcontractors that deliver sub-system or other support software that contributes to larger systems or a larger project scope. The scope of the VSEs projects are usually small, however the quality of a VSE's software is a part of the quality of the overall system it supports. One of the goals of the DoD's DE Strategy is to "transform the culture and workforce to adopt and support DE across the lifecycle" and that cannot happen unless contractors of all sizes adopt DE practices.

The goal of this work is to provide a potential stepping stone for VSEs to DE practices for software products through a hybrid MBSE and DevOps methodology. This work will leverage existing MBSE and DevOps practices and the ISO/IES 21190 standard to account for a smaller project scale that a VSE may have.

1.3 Limitations

This work is presented as a potential solution for VSEs to iteratively implement DE practices. Although the authors attempt to leverage the benefits of MBSE and DevOps, the work presented is not a full scale of either methodology. This results in a small scope of businesses that this strategy may be conducive to.

This work will highlight two technical projects that the hybrid model was deployed on which is a limited amount of case studies. The highlights of this work will be lessons learned from those case studies.

Chapter 2. Literature Review

2.1 MBSE Implementation Benefits

MBSE adoption is getting increasingly more popular and its benefits are being studied as more organizations adopt the method. The Systems Engineering Research Center (SERC) published results from a survey of those in the DE/MBSE community and benchmarked the benefits metrics of MBSE across participants in government, industry, and academia (McDermott et al 2020). Some reported benefits of implementing MBSE from the survey participants were categorized as follows (McDermott et al 2020):

- Better requirements generation
- Reduce errors
- Increased traceability
- Better requirements management
- Improved system design
- Reduced cost
- Reduced time
- Increased capacity for reuse
- Improved system quality
- Increased effectiveness
- Higher level support for automation
- Higher level support for integration

When participants were asked their reasons for integrating MBSE they noted reduced cost, reduced time, better accessibility to information, increased efficiency, improved consistency, increased traceability, and improved system understanding as their leading motivators. When questioned

about the value leveraged from consistent model management, the participants listed increased capacity for reuse, improved consistency, improved system understanding, reduced time, better communication and information sharing, better accessibility of information, and reduced cost as the top valued results (McDermott et al 2020). Less than 25% of the survey participants were from organizations with then than 500 employees and only 17 had less than 1 year of experience with MBSE/DE practices. Therefore, most of the survey participants are participating in the survey from the perspective of a midsize to large business with good working knowledge of MBSE.

SERC's survey's literature review of MBSE benefits that encompassed reviewing 847 papers and using 360 papers that mentioned benefits of MBSE emerged 5 meta categories and top mentioned benefits under each category (McDermott et al):

- Quality
    - Reduce errors/defects
    - Improve system quality
    - Improve traceability
    - Reduce cost
- Knowledge Transfer
    - Better access to information
    - Better communication/info sharing
    - Collaboration
- Velocity/Agility
    - More reuse
    - Improve consistency

    - Increase efficiency
    - Support integration
    - Reduce time
- User Experience
    - Manage complexity
    - Improved system understanding
    - Automation
- Adoption
    - Methods/processes
    - Roles/skills
    - Training/tools
    - Leadership support
    - Change management process
    - Resources

Realizing the above benefits are key to an organization embracing DE/MBSE. Enabling the adoption of MBSE can be a challenge. Bonnet et al outlined the MBSE implementation enablers that helped them coach organizations in MBSE adoption/implementation (Bonnet et al 2015):

- Expected benefits
- Commitment to MBSE
- Deployment strategy
- Coaching

The expected benefits of MBSE usually are a big motivator for organizations to commit to MBSE until these benefits are realized.

2.2 DevOps Implementation Benefits

Faustino et al listed the following DevOps benefits that showed up in over 30% of DevOps literature in a 2022 literature review (Faustino et al 2022):

- Faster time to market
- Cross team collaboration and communication
- Decrease of manual work
- Increase of team performance
- Increase of code quality
- Better deployment management
- Improvement of system reliability
- Faster and better feedback

Lazuadri et al 2021 compiled perceived benefits of DevOps implementation in organizations showed the following benefit categories:

- Frequent software deployment improvement
- Software quality improvement
- Collaboration and sharing improvement between the development team, operation team, and stakeholders
- Reduce software downtime
- Team productivity improvement
- Market competition improvement
- Software artifacts management improvement

- Feedback cycle improvement
- Reduce development cost

DevOps is known for championing teamwork and eliminating silos in order to automatically and rapidly deliver software products to customers while still adopting the Agile method of allowing teams to be self organized.

2.3 Relevant Maturity Metrics

INCOSE published the Model-Based Capability Matrix (which throughout this document we will call the Capability Matrix), that assesses an organization's current and desired model-based capabilities (Hale & Hoheb 2020). The authors of the Capability Matrix define a capability as skills that: produce an outcome, are activated by resources, have both input and outputs, and changes over the lifecycle; the matrix has 42 capabilities that are provided as an excel spreadsheet and are tailorable to suit an organization's needs. The Capability Matrix has stages 0-4 defining the maturity of the 42 capabilities that also align to the DoD DE Strategy goals:

1. Use of Models
2. Authoritative Source of Truth (ASOT)
3. Innovation
4. Establish Environments
5. Workforce Transformation

VSEs can use the Capability Matrix to self-assess their MBSE maturity while they implement DE/MBSE practices with the above goals in mind.

In Zarour et al's review of DevOps maturity models, it was found that most DevOps maturity models comprise of 4 to 5 levels that can be summarized by the adopting organization doing the following (Zarour et al 2019):

- Level 1 (Initial):the starting point of using DevOps.
- Level 2 (Managed): the process is documented sufficiently.
- Level 3 (Defined): the process is defined as a standard organizational process.
- Level 4 (Measured): the process is managed according to metrics.
- Level 5 - optional (Optimized): the process is managed efficiently.

These summarized 5 levels are simple to understand and can be adapted by organizations of various sizes.

Laporte's ISO/IEC working group WG24 defined a software engineering lifecycle standard for VSEs that outlines "who qualifies for a VSE" and their challenges like that center around "a lack of standards used in their business". ISO/IEC 29110 Systems Engineering Basic Profile defines processes for VSEs to follow for successful project management. Reasons that the standard was developed are:

- Lack of resources
- Standards are not required
- The perception that standards are bureaucratic and do not provide adequate guidance for use in the VSE environment.

The working group has published several deployment packages to help define certain processes in the Systems Engineering Basic Profile and also provide Intermediate and Advanced Profiles as a part of the standard (Larrucea et al 2016). However, this standard currently lacks a continuous

improvement nature built into the model. The profiles provided by ISO/IEC 29110 includes a framework for project management and good elements of project execution strategies, but lacks the DE spirit of model-based engineering or continuous product improvement (like MBSE and DevOps). It also lacks the development of an ASOT that will enable other projects to take advantage of momentum/technical knowledge from past projects.

2.4 MBSE Adoption Challenges

Bonnet et al outlined the enablers and obstacles based on lessons learned from helping organizations implement MBSE. Their reported MBSE implementation obstacles included (Bonnet et al 2015):

- Cultural change
- Management commitment to MBSE
- Tooling
- IT policies and support
- Learning curve
- Return on Investment (ROI) measurement

Even with an organization that has management that is committed to MBSE and a culture that is ready for change, challenges like MBSE tooling, IT support, the learning curve, and ROI measurement can be issues in even large organizations adopting MBSE. VSEs may have personnel and financial resources that are additional barriers to implementation.

SERC's survey in 2020 reported the following MBSE adoption obstacles that were perceived by the survey participants can be summarized as (McDermott et al 2020):

- MBSE methods and processes
- Organizational culture
- Communicating success stories and practices
- Customer and stakeholder buy-in and engagement
- MBSE terminology, ontology, and libraries
- Legacy and current processes
- MBSE tooling use, cost, and training

These adoption challenges can be amplified in VSEs that have projects with limited scope, limited personnel, and likely limited MBSE implementation financial resources.

2.5 DevOps Adoption Challenges

In Faustino et al's DevOps literature review several adoption challenges were highlighted from the literature (Faustino et al 2022):

- Industry constraints
- Deep-seated company culture
- Insufficient communication
- DevOps is unclear

Faustino et al proposed possible enablers to overcome these challenges: executive leadership to encourage the change to DevOps and to provide proper training and time for adoption.

Leite et al's survey of DevOps literature listed challenges that are not handled by state-of-the-art implementations of DevOps (Leite et al 2019):

- How to re-design systems toward continuous deliver
- How to deploy DevOps in an organization
- How to assess the quality of DevOps practices in organizations
- How to qualify engineers for DevOps practice

Echoing what was addressed in the MBSE section: the above DevOps adoption challenges could be amplified in VSEs that have projects with limited scope, limited personnel, and likely limited DevOps implementation financial resources.

2.6 Relevant Case Studies: MBSE and DevOps Adoption in Small Businesses

Revell et al detail the adoption of MBSE in Phase I of an engineering Small Business Innovative Research (SBIR) project (Revell et al 2023). The project team struggled to land on a MBSE methodology in the beginning, but landed on some particular tooling and the Object-Oriented Systems Engineering Method (OOSEM) in the end. Despite some challenges, the team was able to demonstrate the adaptive and extensible nature of MBSE within the small timeline and scope of their SBIR project. The team was able to implement MBSE in a short time frame and claimed their MBSE practices enable the listed follow-on project activities:

1. Making detailed design decisions
2. Modeling test cases to enable traceability to requirements and stakeholder needs
3. Identifying key interfaces and physical parameters for production planning

Bucena and Kirkova developed a DevOps adoption method that was intended to be applied in small enterprises (Bucena & Kirkova 2017). The method can be summarized into the following 8 steps:

1. Detect impediments to software development flow and prioritize list of desired DevOps practices.
2. Establish the organization's existing maturity level and select the desired maturity level.
3. Get the list of DevOps practices prioritized in step one and identify gaps between the existing and desired maturity level.
4. Identify existing DevOps tools in organization and identify any additional tools needed to support selected DevOps practices.
5. Choose a DevOps adoption project.
6. Identify metrics for measuring success for the chosen project.
7. Draft the phases to DevOps adoption, collect data, and share results.
8. If applicable, choose another project and repeat the steps.

The method was applied to Company X (separate from Company X mentioned in Chapter 3) and was found to at least partially address the challenges listed by Hamumen (Hamumen 2016):

- Lack of awareness
- Lack of support
- Problems linked to the DevOps technological implementation
- Problems with adapting organizational processes to DevOps

Chapter 3. Overview of FlexOps

3.1 Background

Some small government contracting businesses are struggling to implement some of the digital transformation directives that are implemented by the DoD and bigger private industry partners.

Chapter 3.2 details a project execution strategy model that was implemented to aid a small business in developing a software system. The software system needed to be rapidly developed and incorporate requirements on a rolling basis as they evolved for research and development (R&D) systems. The VSE aimed to incorporate DE practices into their project execution strategy, but knew they did not have the personnel or financial resources to adopt MBSE or DevOps in their entirety. However, the VSE was interested in the benefits that MBSE and DevOps could provide. In reference to the Capability Matrix and the DevOps Maturity Levels mentioned in Chapter 2, the solution below is meant to aid an organization to address certain MBSE capabilities in stages 0 through 2. perhaps advancing to stage 3 in some capabilities, and Level 1-3 of DevOps maturity (INCOSE 2020), (Zarour 2019).

3.2 Scalable, Flexible Implementation of MBSE and DevOps in VSEs: Design Considerations and a Case Study

3.2.0 Overview

Model-based systems engineering (MBSE) and development operations (DevOps) both provide organizations with increased system development capabilities by defining development process controls and defining sequential development, testing, and other processes. As the use and

complexity of MBSE and DevOps processes for software systems development continues to rise, very small entities (<25 employees) may struggle to implement full scale implementations of these engineering project management processes. This paper uses principles from MBSE and DevOps models to explore a reduced, simplified scope of both system development paradigms that VSEs can implement to obtain benefits of both processes.

3.2.1 Introduction

Model-based systems engineering (MBSE) and development operations (DevOps) are increasingly being adopted by large and midsize businesses, and in particular by government contractors, to both increase system and DE efficiency and to implement new Department of Defense (DoD) directives. Recently, the DoD published their *Digital Engineering Workforce Plan*. This plan outlines their need to increase their DE workforce, which must have knowledge of MBSE and "modern software development practices" such as DevOps (DoD 2022). Larger government contracting enterprises usually not only have the resources to implement MBSE and DevOps on a company wide scale, but they also can preemptively implement these processes before they become a standard (which may even influence the standard to be issued). This dynamic makes it hard for a very small entity (VSE), defined as less than 25 employees (Laporte et al 2006) and thus having less personnel and resources, to implement the new standard if they have no foundation of knowledge and/or resources to allocate to keep up. This paper will focus on software engineering applications of DevOps and MBSE, but the authors want to acknowledge that there are uses of these models outside of software engineering; for example, other engineering and system development disciplines.

A VSE's success hinges on the ability to be adaptable and compete with other businesses. Due to their size, teams usually are cross-functional by design and necessity. A VSE may have a team of one or two. It may have a CEO who is project managing while also developing stakeholder code, a software engineer, and a technician both user acceptance testing and using the code. With scenarios like those, it may seem futile to implement DevOps and MBSE practices; at that size, it could even seem ridiculous to implement any processes at all, especially to those "in the trenches" developing the system. However, this is precisely the time to begin implementing those paradigms, while tailoring them to the existing team at hand.

Many have attempted to simplify the implementation of popular software engineering methodologies for small organizations. For example, Rising and Janoff's method of implementing the Scrum software process in small teams within their company (Rising & Janoff 2000). The findings from their case study showed the three teams studied had vastly different instantiations of the "scrum" method; that flexibility allowed their teams to successfully deliver software systems in their respective projects. Laporte et al. developed a standard, ISO/IEC 29110, for VSE-specific systems and software lifecycle profiles and guidelines that was implemented across many countries and organizations (Laporte et al., 2008). Both MBSE and DevOps adoption are hot topics for method improvements. Ploeg et al. highlights the best practices for implementing MBSE vary by application, and recommend that businesses put together an implementation team for assessment and subsequent adoption (Ploeg et al., 2022). Bucena et al proposed a step-by-step method of adopting DevOps practices within an organization and a maturity model for assessing the progression of implementation (Bucena & Kirikova, 2017).

To stay competitive with larger organizations, VSEs may want to pursue implementation of MBSE and DevOps. Currently, this process is documented separately and requires both financial and

personnel resources that the VSE may not have for a full-scale implementation. This paper attempts to allow a parallel implementation by presenting a simplified joint model, including elements of both MBSE and DevOps which allow an organization to get their "foot in the door" by iteratively implementing these practices while the organization grows.

3.2.2 Challenges of MBSE and DevOps Adoption

The adoption of MBSE and DevOps both present a list of challenges. For MBSE, Chami et al lists the adoption challenges as (Chami et al 2018):

- Upfront investment
- Adoption strategy
- Purpose and scope definition
- Awareness and change resistance
- Executive level sponsorship
- Method definition and extension
- Modularity and reusability
- Complexity management
- Tool dependency and integration
- Large models visualization

Chami proposed the D3 Toolbox to ease the challenges and simplify the phases of adoption: definition, development, and deployment. For DevOps, Hamunen describes the adoption challenges in the following four categories (Hamunen 2016):

- Lack of awareness in DevOps
- Lack of support for DevOps

- Implementing DevOps technology
- Adapting organizational processes to DevOps

These challenges are themes throughout many organizations, and extend to other operations and even governance processes. Our proposed method attempts to address elements of the upfront investment, adoption strategy, and adapting organizational processes to DevOps challenges. As a VSE incorporates more elements of each process, the above challenges may arise. Addressing these challenges are outside of the current scope of this work.

3.2.3 Overall Vision

The overall vision for the MBSE and DevOps method proposed in this paper is providing a more flexible model for VSEs to implement, and where necessary incrementally implement, pieces of each existing process while applying personnel and financial resources proportional to their availability. As the organization grows, it can implement more of the conventional steps as it seems appropriate to meet customer and internal operational efficiency needs.

Implementing the MBSE V-model shown in Figure 1 can provide benefits such as (McDermott et al. 2020), (Ploeg et al 2022):

- Finding inconsistencies in design earlier in system development, reducing lifecycle system cost since changes become more expensive later in the development cycle.
- Allowing changes in design specifications, lowering risk of inconsistencies because the model provides improved requirement traceability and impact analysis.
- Providing better understanding of the project due to improved design representation.

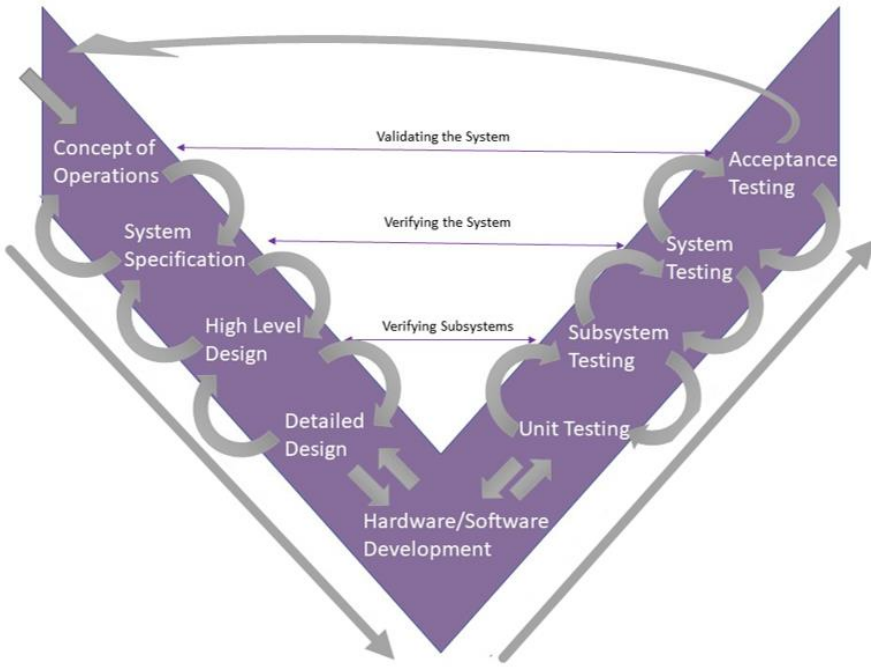

Figure 3.1. MBSE System V Diagram (Wymore 2018)

The above benefits of MBSE can help a VSE reduce system development cost and risk, but a full-scale implementation of the systems engineering process may not be feasible at their size or scope of work. The main themes from the benefits listed above focus on the requirements definition, requirements management, and design model. According to the Air Force, DevOps can provide benefits such as: promoting teamwork and eliminating silos, allowing rapid experimentation in the face of uncertainty, while still allowing a focus on mission end goals (meeting requirements) and improving performance and quality of applications (Air Force).

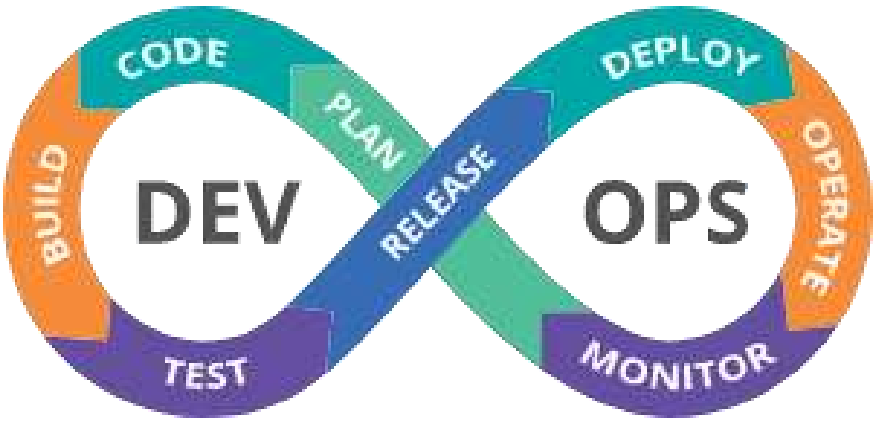

Figure 3.2. DevOps Diagram (Air Force)

Our model will focus on these main themes in an effort to give a VSE plan an optimized set of MBSE benefits without requiring a complete implementation. Hugues and Yankel proposed a joint-model, ModDevOps, that includes both DevOps and MBSE in a functional way. The ModDevOps model includes requirements management and modeling at each stage of the DevOps. Hugues and Yankel extended the Air Force's definition of DevOps (Air Force) to define ModDevOps as (Hugues & Yankel 2021):

> *A systems/software co-engineering culture and practice that aims at unifying systems engineering (Mod), software development (Dev), and software operation (Ops). The main characteristic of ModDevOps is to strongly advocate abstraction, automation, and monitoring at all steps of system construction, from integration, testing, releasing to deployment and infrastructure management.*

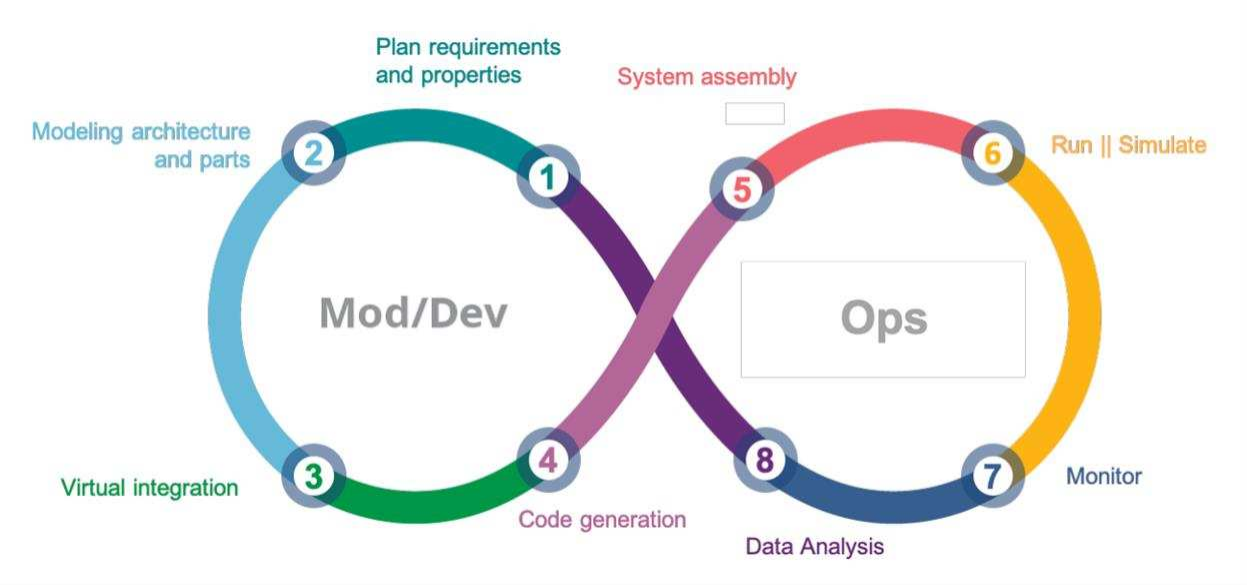

Figure 3.3. ModDevOps Model (Hugues & Yankel 2021)

Our model seeks to sustain the advocacy of systems engineering in DevOps demonstrated in this model while including a more iterative implementation approach with a simplified initial process focusing on requirements and model design that can be elevated to the full ModDevOps process as appropriate to the VSE. This approach will allow an organization to implement elements of MBSE and DevOps whether they have an established team or they are beginning from business inception, while simultaneously lowering the barrier to entry for challenges such as upfront investment, adoption strategy, and adapting organizational processes to DevOps.

3.2.4 The FlexOps Method

Whether a VSE is starting with one or two employees or already has an established team, a full scale DevOps and MBSE implementation is likely infeasible due to personnel limitations and/or financial limitations. Both processes are inherently collaborative and if a company only has one to two employees who also have other responsibilities, the VSE isn't equipped with the personnel to make some of the tools required for implementation of MBSE and/or DevOps financially

worthwhile. Based on the ModDevOps model, FlexOps extends the base DevOps model and injects MBSE elements to promote systems engineering practices within this popular software engineering process. Traditionally, DevOps has eight well-defined and measurable steps: plan, code, build, test, release, deploy, operate, and monitor, informed by its evolution of the agile movement (Christensen 2016). ModDevOps extends the eight DevOps steps by replacing some with relevant MBSE practices: plan requirements and properties, modeling architecture, virtual integration, code generation, system assembly, run/simulate, monitor (test and measure), and data analysis (Hugues & Yankel 2021). FlexOps aims to prioritize the core of DevOps with continuous improvement and delivery (Sharma & Coyne 2015) with MBSE's core to instantiate and reference requirements and model throughout the engineering process (Aceituna 2014) while also considering the VSE's main goal: delivering a system to remain profitable. FlexOps defines this with six core steps:

1. **Plan**
2. **Model**
3. **Code**
4. **Test**
5. **Deploy**
6. **Monitor**

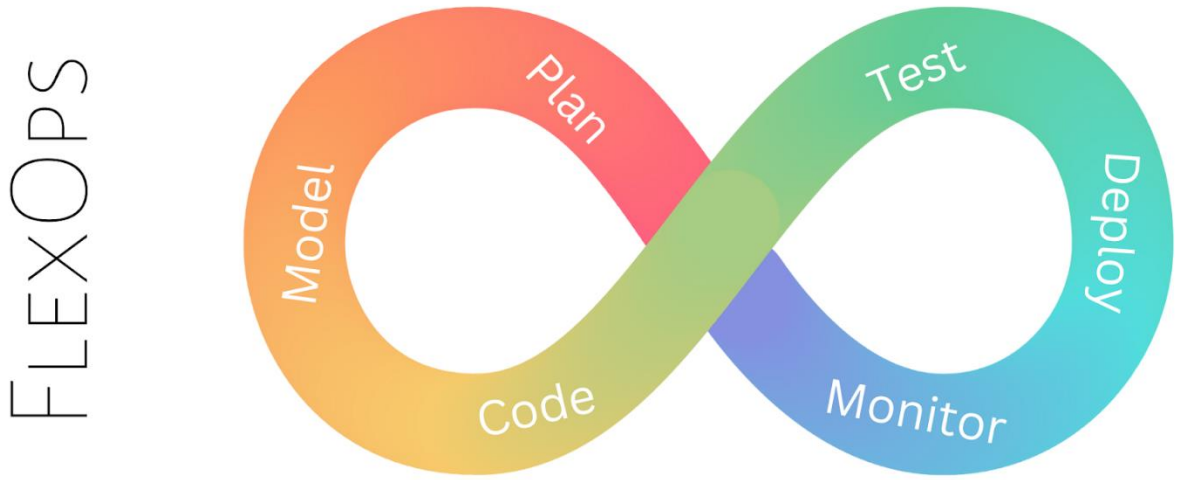

Figure 3.4. FlexOps Model

3.2.5 FlexOps Steps

Plan. Planning requirements and properties from ModDevOps as well as Agile planning techniques included in DevOps is critical to a customer focused system. VSEs cannot afford to create irrelevant or out of scope features. Requirements management reduces project and system risk. Also included in the planning stage are any project management paradigms that the VSE may want to implement: Scrum, Kanban, waterfall, etc.

Model. Modeling the architecture of the system and continuing to revisit if the model aligns with the system requirements to ensure that the software system stays within scope. The requirements defined in the planning step informs the initial model and continues to inform any updates to the model. Use of a standardized modeling language is not required, but is encouraged if the VSE has the knowledge. Otherwise, a document-based model can be utilized, but versioning will need to be closely managed. A simple yet not inelegant versioning can be simply increment the version number in the file name; e.g. *filename-vNN* where NN is the version number.

Code. Writing the code for the system is informed by the requirements, model, and plan for execution. The coding step is also where the execution of any Agile processes like Scrum or Kanban will be key. Additionally, version control software use is expected.

Test. Testing code functionality and verification against requirements continues the process of ensuring a VSE's software system success. This step will have elements of the system testing part of the MBSE process. It is up to the VSE to choose a testing plan that works for their development practices.

Deploy. Deploying code is an obvious step in the success of the system. The deployment process is up to the VSE. Automated deployment is encouraged, as one of the goals of DevOps is automation. However, the consideration that the scope of work a VSE may have may not require or allow this to occur. Other advantageous elements of an agile process, such as pair coding, may also be resource-limited if there is little talent overlap on the small team.

Monitor. Monitoring the system after deployment is key to understand if the system is meeting the defined requirements. This step boils down to testing and measuring the system and then logging that data to evaluate requirement fulfillment with every deployment.

3.2.6 From FlexOps to ModDevOps

The below model shows how a VSE may employ additional steps from the DevOps process to get closer to a full ModDevOps process. The in-between stage of FlexOps and ModDevOps is called FlexDevOps, shown in Figure 5. This transitional model can be scaled to incorporate steps as the VSE has personnel and financial resources to implement.

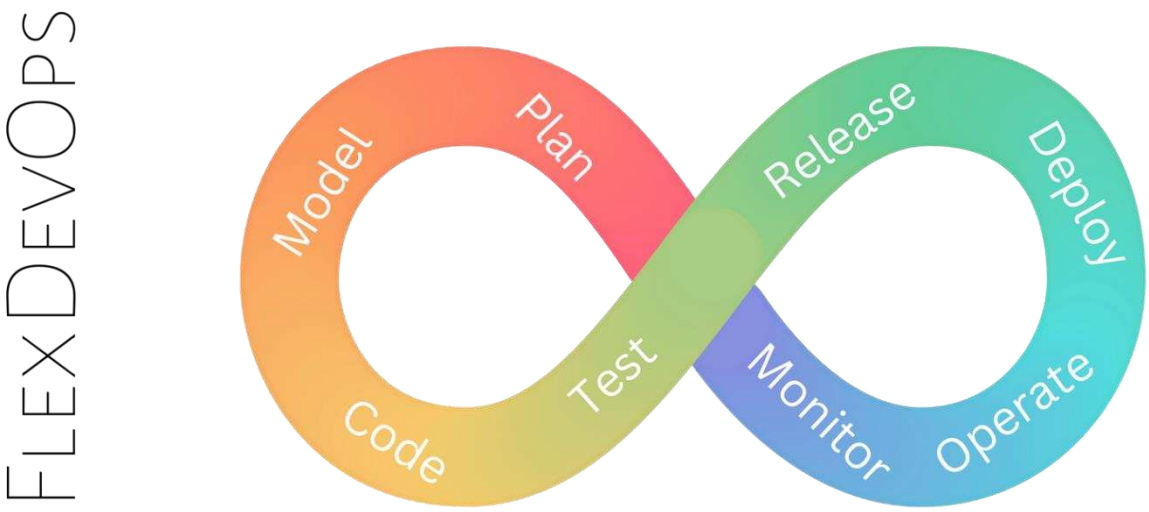

Figure 3.5. FlexDevOps Transition Model

From FlexDevOps, a VSE should work to incorporate models throughout the DevOps process increasing their competency to incorporate a ModDevOps culture.

3.2.7 Experimental Application of the Method

The FlexOps method was both applied and informed by developing a standard operating procedure at Company X. Company X was incorporated in 2017. The founder began the company with a consulting agreement developing software as both a service and system for a large company, Company Z.

3.2.7.1 Background

The company began as one employee, the founder and CEO. The CEO developed a script to be used internally to deliver data analysis reports from a monitoring sensor to Company Z. The script and data analysis report requirements were not defined and only had one user. The CEO was also acting as project manager for the contract.

Company X expanded the consulting agreement with Company Z to an ongoing three year contract, enabling the hire of a software engineer. The contract required the software be deployed on Company Z's systems six times per year. The hiring of another developer and new deployment requirements spurred the process of implementing additional operations.

3.2.7.2 Application of the FlexOps Method

Phase I. The first goal was to implement tools to increase collaboration and requirements tracking. FlexOps was implemented to capture and track software system requirements, foster collaboration, manage software versioning, and deploy the software on schedule. Table 1 details Company X's implementation including tools used to support each step.

Table 3.1: Company X FlexOps Implementation

FlexOps Step	**Supporting Tool(s)**	**Implementation Description**
Plan	PowerPoint, Word, YouTrack	Planning efforts within Company X included requirements definition (PowerPoint & Word), requirements management (PowerPoint & Word), and Scrum planning processes: sprint plannings, backlog grooming, etc (YouTrack).
Model	PowerPoint, YouTrack	Modeling the architecture (PowerPoint) of the system based on the requirements; and reassessing the model with the

		results of the monitoring and planning steps (YouTrack).
Code	PyCharm, Git, YouTrack	Writing code in the PyCharm Integrated Development Environment (IDE), tracking tasks/user stories within YouTrack, and utilizing version control (Git). Ensuring there is logging functionality within code to provide data for the monitoring step.
Test	PyCharm	Unit testing written within Company X's IDE of choice: PyCharm.
Deploy	Git, Operating System	Pulling the most recent stable commit from Git and installing any dependencies needed for the host operating system.
Monitor	Data logs, Jupyter Notebook, Excel	Analyzing data logs to test and measure requirements compliance.

Company X's implementation increased development efficiency between the CEO and engineer and paved the way for scaling to include new employees and processes. The increased efficiency and delegated work allowed the CEO to focus on increased project management duties.

Phase II. During year 1 on the contract, Company X honed their FlexOps process, positioning the company for increased software scope and additional hires. At the end of year 1, Company X hired two additional engineers. This allowed the CEO to increase oversight roles to having no involvement in day-to-day development. The original engineer was flexed to other projects won by the CEO, only supporting the Company Z project half-time. The two new engineers increased efficiency by implementing FlexDevOps. This allowed the half-time engineer and CEO to understand progress and "jump in" when needed without being lost on current code deployments (Release), development efforts, requirements management, and performance metrics. Table 2 details how the additional two steps were implemented to augment the FlexOps process.

Table 3.2: Company X FlexDevOps Implementation

FlexDevOps Step	**Supporting Tool(s)**	**Implementation Description**
Plan	PowerPoint, Word, YouTrack	Planning efforts within Company X included requirements definition (PowerPoint & Word), requirements management (PowerPoint & Word), and Scrum planning processes: sprint plannings, backlog grooming, etc (YouTrack).
Model	PowerPoint, YouTrack	Modeling the architecture (PowerPoint) of the system based on the requirements; and reassessing the model with the results of the monitoring and planning steps (YouTrack).

Code	PyCharm, Git/GitHub, YouTrack	Writing code in the PyCharm Integrated Development Environment (IDE), tracking tasks/user stories within YouTrack, and utilizing version control (Git). Ensuring there is logging functionality within code to provide data for the monitoring step.
Test	PyCharm	Unit testing written within Company X's IDE of choice: PyCharm.
Release	**Git/GitHub**	**Release tags implemented in GitHub, marking commits that were included in a release.**
Deploy	Git/GitHub, Operating System	Pulling the most recent release from Git and installing any dependencies needed for the host operating system.
Operate	**PowerPoint, Word, YouTrack**	**Accepting feedback from the customer and users on the software and data outputs after deployment. Feedback is logged as user stories in YouTrack and any requirements added/changed are captured (PowerPoint & Word).**

Monitor	Data logs, Jupyter Notebook, Excel	Analyzing data logs to test and measure requirements compliance.

Although the latter five steps are not automated at this phase of implementation, Company X has a workflow to execute each step and tools to accomplish each.

Phase III. During year 2 of the contract, Company X honed their FlexDevOps skills and used the software tools they had available to effectively release their system and Company X is currently in Phase III. These operations are working well for them as they continue to hire and expand their business capabilities. Company X hired a technician to support software operations and data collection. The technician assists with user acceptance testing (Test, Operate), documentation (Plan, Model), and data/feedback gathering (Operate, Monitor).

3.2.7.3 Observations & Discussion

Company X's project for Company Z began with just one employee (the founder) and expanded to 5 employees over a three year period. Over the three year period, funding for the project increased. This increase in funding was given by Company Z as a result of successful software deployments and data product quality. This increase in funding allowed additional employee hires. From year 1 to year 2, funding increased by 478%. From year 2 to year 3 funding increased by an additional 114%.

Company X was able to implement FlexOps within year one, spending approximately 40 hours of training and 80 hours of tool implementation time between two employees. With a mean team cost of $100/hour, implementation personnel cost was approximately $12,000. To increase from

FlexOps to FlexDevOps, implementation hours were 80 with a mean employee cost of $100. FlexDevOps implementation personnel cost was $8,000.

Company X used FlexOps and FlexDevOps to grow their development operations while maximizing their personnel resources and keeping software tool cost low. The cost breakdown of the tools utilized per year is detailed in Table 3-5.

Table 3.3: Company X Tool Costs Year 1

Tool Name	**Tool Cost**	**Employee #**	**Total Cost**
Microsoft 365 Business Premium	$276 per user/year	2	$552
YouTrack	Free	2	$0
PyCharm Professional	$265 per user/year	2	$530
GitHub	$252 per user/year	2	$504
Jupyter Notebook	Free	2	$0
		Total	**$1586**

Table 3.4: Company X Tool Costs Year 2

Tool Name	**Tool Cost**	**Employee #**	**Total Cost**
Microsoft 365 Business Premium	$276 per user/year	4	$1104
YouTrack	Free	4	$0
PyCharm Professional	$265 per user/year	4	$1060
GitHub	$252 per user/year	4	$1008
Jupyter Notebook	Free	4	$0
		Total	**$3172**

Table 3.5: Company X Tool Costs Year 3

Tool Name	**Tool Cost**	**Employee #**	**Total Cost**
Microsoft 365 Business Premium	$276 per user/year	5	$1380
YouTrack	Free	5	$0
Lucid Chart	Free	5	$0

PyCharm Professional	$265 per user/year	5	$1325
GitHub	$252 per user/year	5	$1260
Jupyter Notebook	Free	5	$0
		Total	**$3965**

Total FlexOps and FlexDevOps tool cost over the three years was $8,723. Total FlexOps and FlexDevOps implementation cost was $28,723 over three years. This is significant because implementation of large scale MBSE efforts requires an expert multidisciplinary team and significant personnel and financial resources. An example of this is the Submarine Warfare Federated Tactical Systems (SWFTS) program in which an investment of $3.28M and 26,117 personnel hours over two years was required for baseline MBSE and configuration management (Rogers & Mitchell 2021). SWFTS measures the program's ROI at 0.79 or 79%. DevOps implementations are very dependent on tools utilized and cost factors that are included. This is highlighted by the conclusion of Ravichandran et al. that “current measurement of DevOps ROI remains somewhat nascent, and difficult” (Ravichandran et al 2016).

Company X’s ability to implement MBSE and DevOps elements into their company with limited resources directly impacted their success over the three year contract. FlexOps and FlexDevOps allowed Company X to scale these practices with their funding and personnel availability. Total tool implementation cost and personnel hours to implement were kept low and practical for what the VSE could financially support. While the total ROI is difficult to measure, by keeping

implementation cost low while reaping the benefits, Company X benefitted from the implementation of FlexOps.

3.2.8 Limitations

FlexOps inherently misses the benefits of deploying full scale DevOps and MBSE processes, both independently and combined. A prime example: mission critical software like those developed for the automotive and aerospace industry would not benefit from this scaled back approach due to process assessment standards such as ISO/IEC 15504 also known as Software Process Improvement and Capability Determination (SPICE). This framework is suggested as a means to give organizations a simplified methodology that is intended to scale to include the full breadth and depth of what the organization needs as it has the resources to do so; compared to a larger scale implementation, the cost of implementation was kept relatively low, although an accurate ROI was not achieved within the scope of this work.

This method was created out of the necessity to implement elements of MBSE and DevOps at Company X that did not have the personnel and financial resources to completely implement either methodology, let alone both. However, Company X wanted to take advantage of the risk mitigation and increased efficiency that this model offers. FlexOps is presented with the assumption that other VSEs may have this particular struggle as a potential roadmap to implement similar processes in-house. The authors plan to implement in other VSEs to collect and aggregate more data on potential applications and benefits of this model.

3.2.9 Conclusions & Looking Forward

FlexOps and FlexDevOps was developed to provide VSEs a model to implement MBSE and DevOps practices within their organizations. The methodology was experimentally implemented in Company X, a startup that began as one employee and increased their ability to deliver a software system by applying FlexOps and then FlexDevOps. This model helped a VSE increase their footprint within the contract and increase credibility with their customer. Credibility was increased by delivering deployments on time and was shown through continued contract funding increases over the three year contract. The flexibility of a FlexOps implementation can provide VSEs the tools needed to make implementing MBSE and DevOps less challenging through simplification of requirements of what it takes to get started. FlexOps encourages an organization to assess what works (both time, personnel, and resources) at any present time to guide their process through the flow of the model. The authors intend to expand this work to other VSEs including small development groups in larger companies that may not have resources within their teams and accurately measure the ROI of FlexOps implementations. The intention of this work is to give VSEs a mechanism to implement standard DevOps and MBSE processes as financial and personnel resources grow and over time.

3.3 Discussion

The development of FlexOps occurred alongside the development of the Fiber Morphology tool described in Chapter 4, Section 4.2, and the RAW Video Analysis tool in Chapter 5, Section 5.2. Although the previous section is focused on the RAW Video Analysis tool, FlexOps concepts were implemented in the small R&D development team described in Chapter 5 Section 5.2 as the team's project execution strategy.

The above case study addressed many free tools to implement MBSE and DevOps concepts, but failed to suggest free systems modeling tools. Below are some suggestions for free tools a VSE may leverage to get started:

Table 3.6: Suggested Free Systems Modeling Tools for VSEs

Tool Name	FlexOps Step	Description
Eclipse Papyrus	Model	Open-source platform for Unified Modeling Language (UML) and Systems Modeling Language (SysML)
Modelio	Model	Open-source UML modeling tool.

Chapter 4. Case Study 1: Fiber Morphology Automated Metrics

4.1 Background

This work was developed in a small R&D department in a large aerospace and defense company. The R&D department had around fifteen employees; the majority of the team were engineers and scientists working on a specialized fiber manufacturing process. The development team was made up of two developers and a software manager. The development team created a data pipeline to support the engineering and science team's experimental test beds. The data pipeline automatically collected data from the sensors on the test bed and analyzed it to provide inputs into test experiment matrices, ultimately helping the lead scientists make decisions based on near-real time results.

Using the FlexOps process, the development team utilized an incremental approach to planning, modeling, developing, testing, deploying, and monitoring elements of the data pipeline until it was in sustainment. During the incremental approach of continuously integrating and deploying the pipeline, an unknown manufacturing issue presented unexpected fiber morphologies. The development team added capability to the pipeline to batch analyze the morphology of fibers using a machine vision methodology.

The following publication highlights the methodology used to analyze the fibers. The methodology was developed to continue the goal of informing the lead scientists' decision quickly to determine the causation of the morphology changes.

4.2 Fiber Morphology Analysis for Directed-Energy Deposition Manufacturing Process

4.2.0 Overview

The structural and mechanical properties of fibers manufactured from a batch-wise directed-energy deposition process are related to the fiber morphology. Mechanical property data of a batch is determined by testing a randomly-selected sample of fibers. Metrics such as edge roughness are utilized to quantify the variations in the outer edges of a fiber whose morphology is expected to be uniform. The amount of edge roughness present in a batch of fibers affects the mechanical properties of the batch. A correlation between reduced mechanical properties and edge roughness was possible through calculating edge roughness metrics programmatically for each batch of fibers. In order to automate the analysis of batches of fibers, a software tool was leveraged to gain understanding of the morphologies.

4.2.1 Introduction

The fiber manufacturing process highlighted in this chapter enables the manufacturing of low density, temperature resistant material with superior mechanical properties to other commercially available alternatives. Material properties are determined by down-selecting a sample of fibers from a batch, which currently contain thousands of fibers but will expand to include orders of magnitude more. Optical images are captured for the down-selected batch. Mechanical and morphological properties are then calculated from optical images. Additional characterization and analysis is performed on an as-needed basis using scanning-electron microscopy.

Variations in material properties were observed in several batches and were correlated to fiber morphology. Within these batches distinct fiber morphologies were identified. Straight fibers yield

acceptable mechanical properties. In this context, straight is defined as uniform in diameter and linear in shape. An example of a fiber classified as straight is shown in Figure 1.

Figure 4.1. Straight morphology. Diameter: 12.37 +/- 0.51 µm UTS: 3.36 Gpa

Figure 4.2. Non-uniform morphology. Diameter: 12.93 +/- 0.76 µm UTS: 1.57 GPa

Non-uniform morphologies yield lesser mechanical properties than straight fibers. An example of a fiber with decreased mechanical properties and a change in morphology is shown in Figure 4.2. After qualitatively classifying different fiber morphologies, a tool was created to quantify the variations in morphology in order to better aid in root-cause assessments of process variations.

4.2.2 Qualitative Assessment of Fiber Morphologies

In order to reverse-engineer the factor or factors in the manufacturing process that caused the change from straight fibers to non-uniform fibers, there was a need to classify the kinds of defects in the fibers that were being observed. Classifying the different defects that deviate from a straight

fiber was a qualitative process. This was done by comparing images from different batches with varying mechanical properties.

4.2.2.1 Jogs

Changes in axis direction along a length of a fiber can be defined as a jog. Figure 4.3 illustrates a representative example of a jogged fiber with arrows to highlight the changes in axes.

Figure 4.3. Jogged fiber with white arrows indicating changes in axes.

4.2.2.2 Fluctuations

Repeated and irregular changes in diameter where the top and bottom edges of the fibers are anti-correlated can be defined as fluctuations. Figure 4.4 shows an example of fluctuating fibers with arrows to highlight the changes in diameter and correlating edges.

Figure 4.4. Fluctuating fiber with red arrows indicating anti-correlated edges.

4.2.3 Quantifying Changes in Morphologies

The ability to automate the analysis of images and test data of fibers is key to quantifying the mechanical properties and morphologies of a batch of fibers. By using software to process batch-wise data, historical information about batch morphology variations can be utilized to better understand potential process control pathways for the production of consistent, high-quality material.

4.2.3.1 Automated Analysis

The fiberMeasurement module is a program designed to calculate fiber diameter statics and ultimate tensile strength for individual fibers and for batches of fibers. The module was written using Python3 and is designed to analyze optical images given a pixel to micrometer conversion ratio. With the addition of edge roughness metrics the functionality expands to include morphology characterization.

4.2.3.2 Morphology Metrics

In order to better quantify variation in fiber morphology, the following metrics were added to fiberMeasurement module (Bunday et al 2004):

Table 4.1. Edge Roughness Metrics

Edge Roughness Metrics	
Line edge roughness (LER)	The sum of the absolute value of the residual edge positions divided by the number of positions.
Line width roughness (LWR)	The sum of the absolute values of the residual width positions divided by the number of positions.

Total Edge Roughness (TER)	The sum of the mean square roughness for both edges of a fiber.
Correlation coefficient (C)	LWR mean square roughness subtracted from the TER divided by the product of 2 and the square root of mean square roughness for both the top and bottom edge of the fiber. This metric allows for understanding if the fibers are not not correlated (straight) or either directly correlated or anti-correlated (oscillating). A value between -1 and 1.

The basis of the above metrics is the calculation of edge and width residuals from the top and bottom edge positions of a fiber, given N edge positions. Edge residuals must be calculated for both the top and bottom edges of a fiber. To calculate the edge residuals, x_i, subtract the local edge positions, X_i, from the line of best fit:

$$x_i = X_i - (ai - b) \tag{4.1}$$

To calculate width residuals, w_i, the local width, W_i, is subtracted, from the mean width, $\underline{W}$:

$$w_i = W_i - \underline{W} \tag{4.2}$$

4.2.3.2.1 Mean Roughness

LER and LWR are measures of mean roughness, based on their respective residual values:

$$LER = \frac{1}{N}\sum_{i-0}^{N-1}|x_i| \tag{4.3}$$

$$LWR = \frac{1}{N}\sum_{i-0}^{N-1}|w_i|$$

(4.4)

4.2.3.2.2 Mean Square Roughness

Mean square roughness metrics provide understanding for the quadratic effects on roughness. TER is based on the sum of the mean square roughness of the top and bottom edge, R_E^2:

$$R_E^2 = \frac{1}{N-2}\sum_{i-0}^{N-1} x_i^2$$

(4.5)

$$\text{TER} = R_{E_{top}}^2 + R_{E_{bottom}}^2$$

(4.6)

Width square roughness is calculated similarly to edge square roughness:

$$R_W^2 = \frac{1}{N-2}\sum_{i-0}^{N-1} w_i^2$$

(4.7)

The correlation coefficient is a metric that describes how each edge is related to the other.

$$C = \frac{TER - R_W^2}{2 \cdot R_{E_{top}}^2 \cdot R_{E_{bottom}}^2}$$

(4.8)

4.2.4 Morphology Data Analysis

After applying the updated fiberMeasurement module on all past collected data, system operators could use the mean C, LWR and TER values of a batch to quantify the batch morphologies at a glance without scrutiny of individual fibers. Displaying the mean values of the metrics, as well as

histograms of the values over a batch, allowed for a deeper understanding of the morphologies over the sample and the batch as a whole.

4.2.4.1 Straight Fibers

The C value is used to indicate if the edges are correlated in movement. For straight fibers, a mean correlation coefficient of 0 indicates no dominating features or shape defects. In the present instantiation of the tool the image resolution is 0.91 µm per pixel. Therefore, any value less than 0.91 for LWR, or TER is outside of the resolution that can be quantified. Any roughness below 0.91 is less than one pixel indicates the fibers are acceptably straight based on the current body of data. Example images from a batch of straight fibers along with property data is presented in Figure 4.5 and morphology metrics are summarized in the table below.

Figure 4.5. Batch of fibers with straight morphology. Mean Batch Diameter: 12.39 +/- 0.48 µm Mean Batch UTS: 3.05 +/- 0.48GPa

Table 4.2: Batch of Fibers with Straight Morphology Edge Roughness Metrics

Mean C	0.00

Mean LWR	0.54 µm
Mean TER	0.76 µm

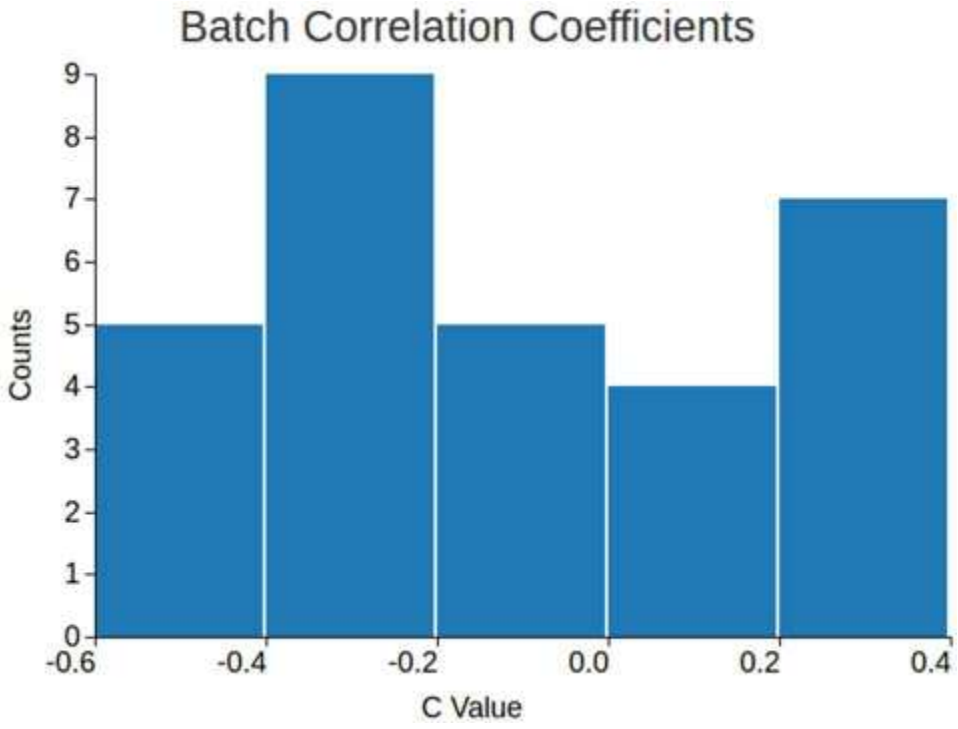

Figure 4.6. Histogram of batch with straight morphology correlation coefficients.

Straight fibers have the most desirable mechanical properties. The above histogram in Figure 6 shows a range of C values that center around 0, but the range is not normally distributed. This indicates a range in slightly different morphologies throughout a batch that results in a generally straight batch with desired mechanical properties.

4.2.4.2 Jogged Fibers

For jogged fibers, a mean positive C value indicates the top and bottom edges of the fiber move in the same direction and are correlated. The mean LWR, and TER values will be one pixel or more (higher than the pixel to micron ratio), significantly larger than the straight fibers. Example images from a batch of jogged fibers are shown in Figure X and the morphology metrics are summarized in the subsequent table.

Figure 4.7. Batch of fibers with jogged morphology. Mean Batch Diameter: 13.52 +/- 0.95 µm

Note: Resulting poor structural properties preventing mechanical testing.

Table 4.3: Batch of Fibers with Jogged Morphology Edge Roughness Metrics

Mean C	0.55
Mean LWR	0.98 µm
Mean TER	4.57 µm

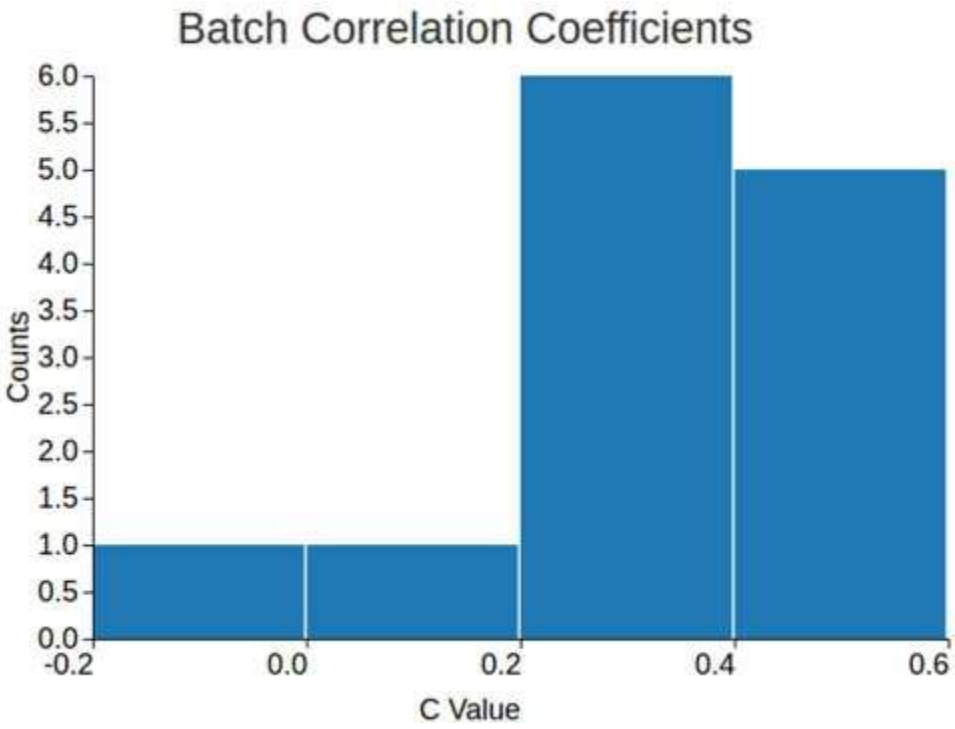

Figure 4.8. Histogram of batch with jogged morphology correlation coefficients.

Jogged fibers were determined to have the least desirable mechanical properties. In this example, the fibers were not able to be handled to be tested due to breaking. The histogram for these fibers (Figure 4.8) shows a heavy skew towards positive correlation between the two edges, which is expected in a jogged fiber.

4.2.4.3 Fluctuating Fibers

For fluctuating fibers, a negative mean correlation coefficient indicates the top and bottom edges of the fibers are anti-correlated. The mean LWR and TER are larger than one pixel, which is significantly larger than the straight fibers. Representative images from a batch of fluctuating fibers are presented in Figure 4.9. The results from the morphology analysis tool are summarized in the subsequent table.

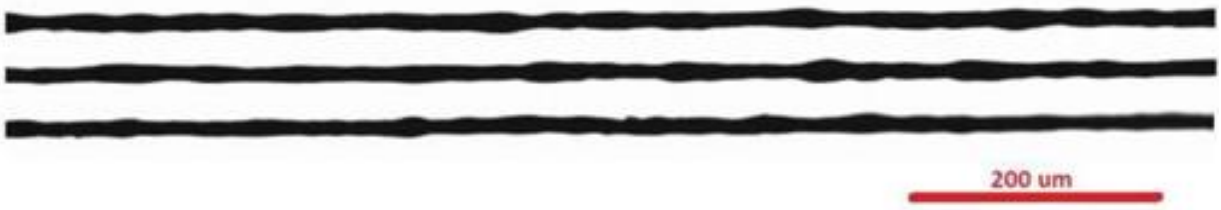

Figure 4.9. Batch of fibers with fluctuating morphology. Mean Batch Diameter: 11.93 +/- 0.58 µm Mean Batch UTS: 0.64 +/- 0.44 GPa

Table 4.4: Batch of Fibers with Jogged Morphology Edge Roughness Metrics

Mean C	-0.56
Mean LWR	1.22 µm
Mean TER	1.98µm

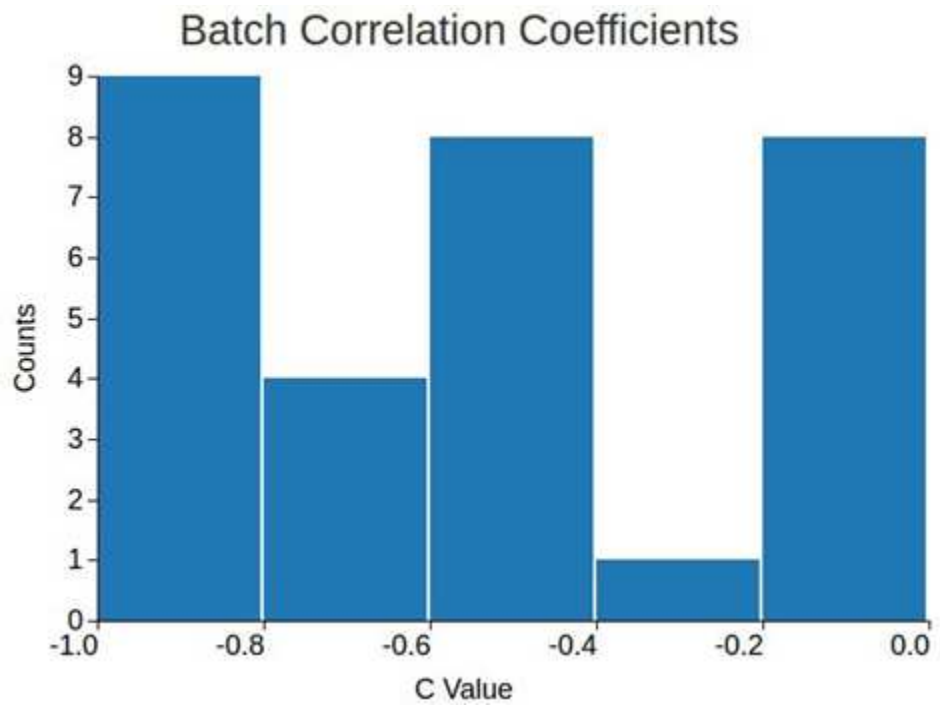

Figure 4.10. Histogram of batch with fluctuating morphology correlation coefficients.

Fluctuating fibers have properties that are not desirable, but are strong enough to be mechanically tested. The correlation coefficients in this batch of fibers skew towards a negative correlation between the two fiber edges (Figure 10). This is expected of fluctuating fibers because the edges are anti-correlated.

4.2.5 Conclusions

The addition of the edge roughness metrics allows for the comparison of different fiber morphologies. With the updates to the fiberMeasurement module, metrics for the uniformity of the fibers are made available at a glance. The fiberMeasurement module is currently being used during post-process batch testing for processing optical images of the fibers. Going forward, this functionality will be explored for expansion to monitor the morphology in-situ – providing real-

time feedback to the process control software. This evolutionary step could lead to reduced process development cycle times and a more stable manufacturing process overall.

4.2.6 Future Work

Refinement of the fiberMeasurement tool will continue in order to facilitate the continued development of the fiber manufacturing process. One of the challenges of taking edge roughness metrics is random error introduced by noise in images (Villarrubia 2005). To reduce random error and increase the accuracy, the software tool will read images with higher resolution at a higher magnification.

The addition of the metrics to the fiberMeasurement post- process analysis of fibers could potentially be expanded to monitor fiber morphology during the manufacturing process in situ. With this feedback, factors that affect the fiber shape could be adjusted in real-time as a part of the process control software in order to reduce product variability.

4.3 Discussion

As a result of the small department leveraging the infrastructure of larger company resources, it cost the department nothing to implement FlexDevOps. Enterprise tools like Microsoft Office, Git, and Jira were available to any teams across the company. Other free tools like PyCharm, LucidCharts, and open-source programming packages and tools were used to develop the pipeline. The development team used the following tools to support their FlexDevOps process in the development of this additional capability:

Table 4.5: R&D Department FlexDevOps Implementation

FlexDevOps Step	Supporting Tool(s)	Implementation Description
Plan	PowerPoint, Word, Jira, LucidCharts	Planning efforts included requirements definition (PowerPoint & Word), requirements management (PowerPoint & Word), and Scrum planning processes: sprint plannings, backlog grooming, etc (Jira).
Model	PowerPoint, Jira	Modeling the architecture (PowerPoint) of the system based on the requirements; and reassessing the model with the results of the monitoring and planning steps (Jira).
Code	PyCharm, Git/GitHub, Jira	Writing code in the PyCharm Integrated Development Environment (IDE), tracking tasks/user stories within Jira, and utilizing version control (Git). Ensuring there is logging functionality within code to provide data for the monitoring step.
Test	PyCharm	Unit testing written within the dev team's IDE of choice: PyCharm.

Release	Git/GitHub	Release tags implemented in GitHub, marking commits that were included in a release.
Deploy	Git/GitHub, Web Server	Merging from the dev branches to the main branch auto-deployed any changes to the web server.
Operate	PowerPoint, Word, YouTrack	Accepting feedback from the customer and users on the software and data outputs after deployment. Feedback is logged as user stories in Jira and any requirements added/changed are captured in Jira as Epics.
Monitor	Data logs, Jupyter Notebook	Analyzing data logs to test and measure requirements compliance.

Chapter 5. Case Study 2: RAW Video Analysis Framework

5.1 Background

The FlexOps model, as documented and detailed in Chapter 3, was applied to the RAW Video Analysis Framework presented in Chapter 5.2. Company X was in its infancy and was contracted to analyze RAW video captured from an existing system with in-situ restraints. The RAW video analysis tool was developed to automate creation of analysis documents based on the RAW video and metadata. Alongside the development of the tool, the FlexOps and FlexDevOps process model was documented and implemented. This tool and accompanying framework was not only used to deliver quality documentation and analysis based on RAW EO/IR videos, but used to develop company processes that paved the way for additional business.

5.2 Case Study: Rapid Development of a RAW Video Analysis Tool with In-situ Restraints

5.2.0 Overview

Utilizing open-source frameworks allow developers to keep development cost down even though it usually requires more code than a fully COTS (commercial off-the-shelf) option. This section presents a RESTful (representational state transfer) software stack utilizing open-source applications and frameworks to visualize and analyze large raw video files in Python using the Dash Framework. The methodology to transform the raw video type into a data type readily accepted by open-source Python web frameworks is abstracted for the reader to apply metadata to image objects in multimedia documents in other open-source web frameworks for rapid development of a post-process RAW video analysis tool.

5.2.1 Introduction

Development of open-source web frameworks has enabled software developers and analysts alike to build data analysis pipelines to support both research and industry endeavors. However, web frameworks are not designed to accept large, uncompressed data types such as RAW video. Although a customized desktop application may be more conducive to processing RAW Video and other large file types, developers can take advantage of the structure of RESTful web applications and open-source frameworks to lower the required development time. The authors had a need for rapid development of a tool to identify frames of interest in a large RAW video format from a system that did not allow in-situ frame processing (only post-process capability): 16-bit depth with 1 channel. Converting the video into a web document-acceptable format like MP4 or MPEG adds an extra step to generating deliverable analysis and encumbers the process for the analyst. In addition, this conversion reduces pixel selection accuracy which was essential for this application. To circumvent this step, the authors developed a method of directly reading the uncompressed data into the browser, thereby allowing the analyst to select frames and points of interest and attach distance data to those frames and points of interest.

Plotly Dash (Dash) is a python-based web framework designed to be a low-code, largely GUI and API driven solution for interactive visualizations. Dash accepts and visualizes input in native python data types like dictionaries, lists, and tuples as well as static images in the form of NumPy arrays and videos in types like MP4. Therefore, handling large RAW video and image formats is not natively supported by Dash. To accept native camera binary files, RAW video must be transformed into formats accepted by the web framework. The framework addresses the challenge of reading large RAW video (RAWs) formats into Dash using a task queue and packages like

NumPy to transform the video frames while attaching distance metadata from location files co-sourced with the video. This particular implementation can be used to rapidly develop a framework to produce analysis documentation from the RAW video.

5.2.2 Related Works

Luxem et al. detailed some best practices for setup and methods for using open source tools for behavioral analysis of animals in lab settings (Luxem et al 2023). Though this work focuses on static frame analysis, principles from Luxem's paper were used to calculate the pose of objects of interest and quantify distance of the objects of interest to create documents. McLeod used Flask, the Python-based micro-service web framework that Dash depends on: Celery, a task queue; RabbitMQ, a message broker; MongoDB, a NoSQL database; and popular Python-based scientific and machine learning packages to support the upload and processing of large data sets to create a multimedia application with open-source libraries and dependencies (McLeod 2015).

5.2.3 Objectives

5.2.3.1 Research Contributions

- A methodology to quickly post-process and analyze RAW videos into an acceptable format to display in Dash and other similar frameworks.
- A case study for creating meaningful analysis documents by post-processing large RAW video files and accompanying meta-data when in-situ processing restraints are in place.

5.2.3.2 Benefits

- Simplicity: this framework is written in a single language, Python, but the packages adopted wrap and leverage both JavaScript and HTML.

- Well-known open-source community: Python has a strong open-source community with developers actively and openly communicating about features and implementation options for both well-known and obscure packages.

5.2.3.3 Detriments

- Performance: performance has not been a focus of this framework and therefore is an area for forward work and improvement.
- File size: this framework is designed to work for videos up to 500GB. The proposed framework is a template of modules that can be swapped to increase file capacity and storage capacity to the user's discretion as needed.
- Displayed image quality: compressed frames displayed in browser for user input as a 1 channel grayscale image.

5.2.3.4 Limitations

- Use case: the authors were limited to post-process use of the RAW frames only - *in situ* analysis of streaming frames was not available due to video capturing system restraints.

5.2.4 Framework Design

To effectively visualize RAW videos to measure the distance of objects of interest (OOI) by combining location data and frames to attach distance metadata to image objects in multimedia documents, the following system requirements were established:

- The accompanying CSV location data will be uploaded along with the RAW videos, timestamps extracted and correlated

- RAW videos need to be able to be compressed to MP4s for distribution.
- Batch, asynchronous video conversion processing that will not interrupt the creation of manual RAW video distance reports.
- The framework will be written in Python to reduce the barrier to entry for new developers and analysts on the team and provide continuity with existing code within the team.

The following elements are needed to satisfy the above requirements:

- An analytics and visualization driven GUI for loading video and ability to upload through the browser.
- A method of extracting the RAW video frames and timestamps.
- A method of scrolling through frames and selecting coordinates in video to calculate distance between objects.
- A task queue for completing batch asynchronous video conversion and creating distance analysis documentation.

The outline of the modules and elements that make up the framework is detailed in the sections below.

- Plotly Dash: The low-code, largely API and analytics driven GUI has native coordinate selection capabilities and ease of building data tables. Leveraging the ability to load images as a graph with a coordinate system, the RAW frames are loaded as single-channel grayscale images.
- Celery and RabbitMQ: Celery executes tasks as background jobs or daemons and RabbitMQ is a message broker that maintains a list of jobs and their status. Celery is used to process batch video conversions in the background. Celery can support brokers other than RabbitMQ, providing a flexible design framework.

- Flask: A Python web microservices framework. Dash is built on Flask and uses it for all communication between the WSGI server and the client.
- RAW Video Parsing Script: Recorded alongside the MWIR sensor that records the optical data is time metadata stored within the RAW video binary format. The number of frames, bits per pixel, and image matrix size are stored in the header. As depicted in Figure 5.1, the header is read and then the file is parsed into frames stored in a NumPy memory map and timestamps stored in a NumPy array.

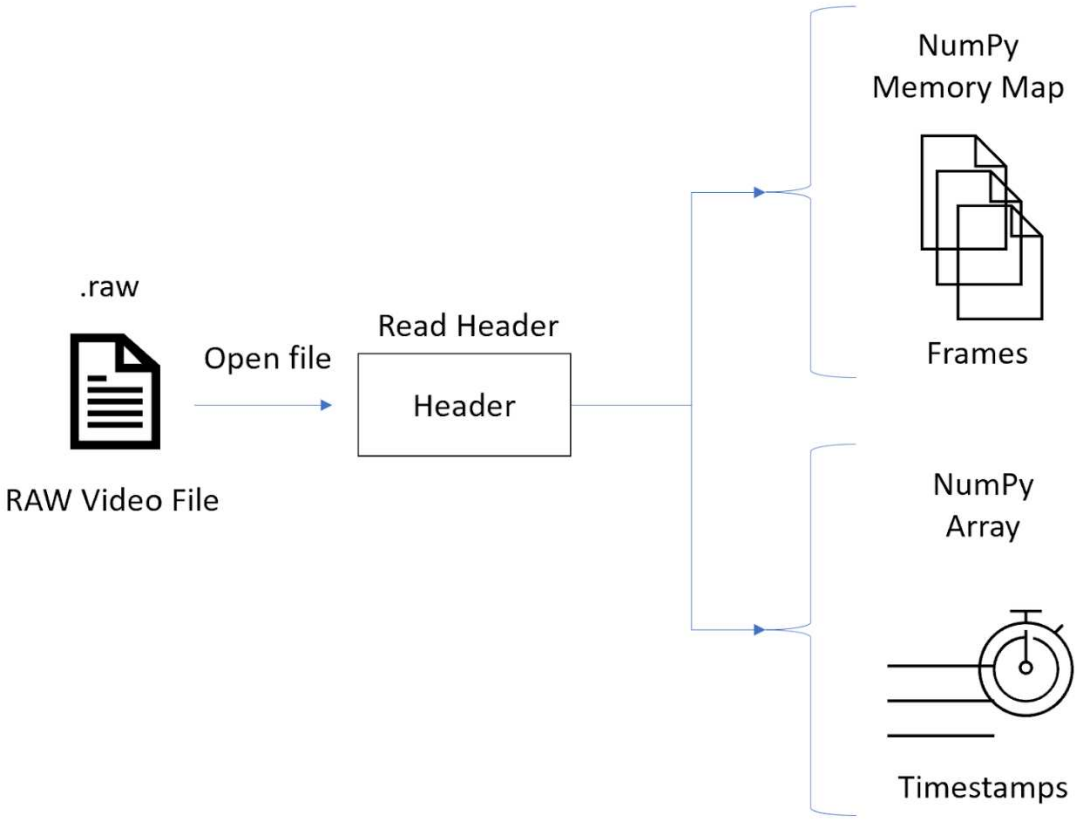

Figure 5.1: RAW Video Parsing Script Flow Diagram

The videos are loaded directly from disk and the memory map is accessed for frames and timestamps. The workflow for the RAW video analysis framework is outlined in the figure below.

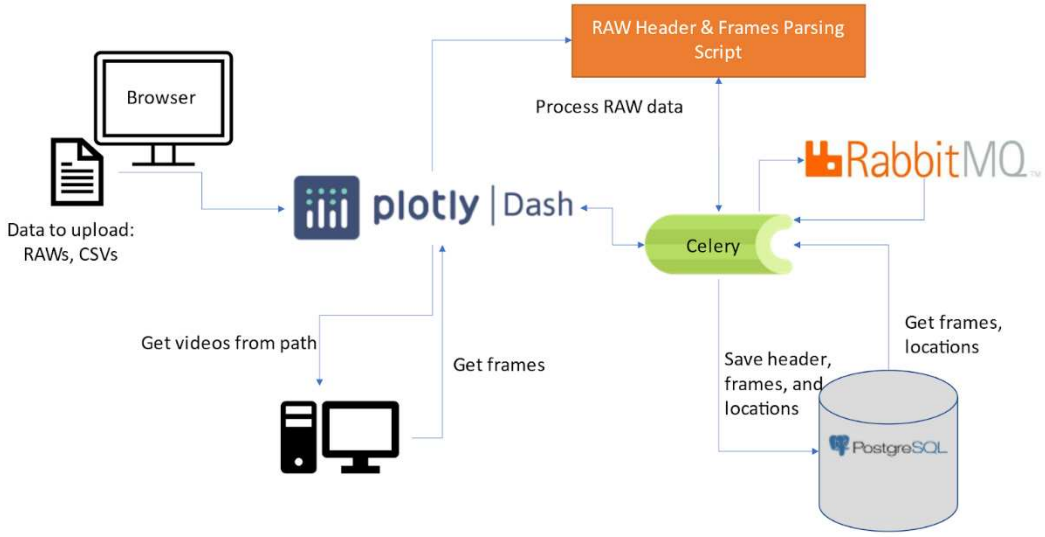

Figure 5.2: The RAW Video Analysis Pipeline Flow Diagram

5.2.5 Transforming RAW Video for Analysis in Common Multimedia Documents

Parsing the RAW video to isolate the frames in an array-type structure and adding relevant metadata is key to transforming it to a browser-acceptable multimedia image object. After the RAW frames are in a list or memory map type that is indexable and subscriptable, the frames are converted into a supported color space, greyscale. Figure 5.3 depicts the layout of the RAW analysis tool where the frames are displayed as a Dash graph with coordinate selection options. The video is loaded asynchronously upon request. The view below shows the multimedia application document before the RAW frames are loaded.

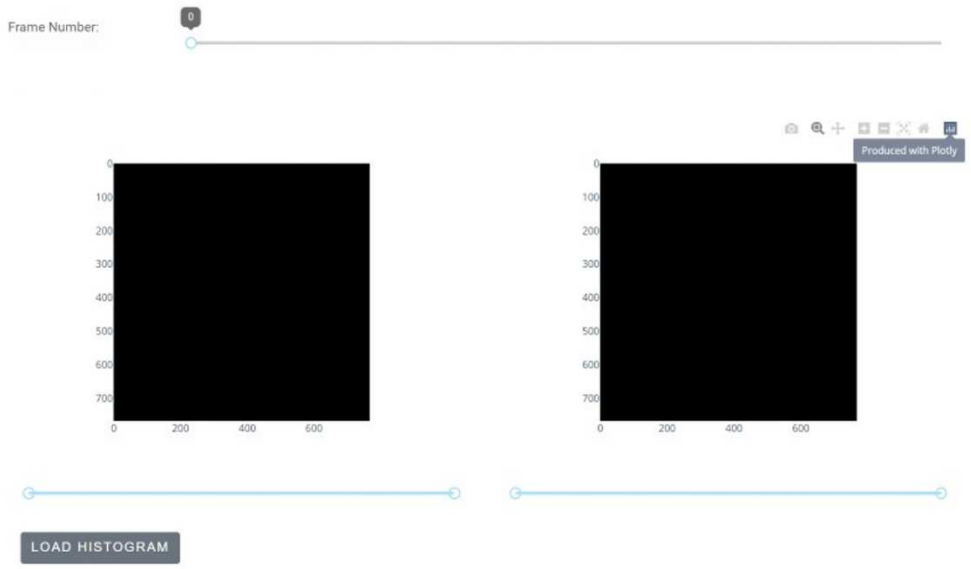

Figure 5.3: Layout of the RAW Distance Analysis GUI

The authors included a slider to change the dynamic range of the video to allow for fine tuning of the automatic thresholding applied to the image. Two camera views are used to attach location metadata to corresponding frames at corresponding times shown in Figure 5.4.

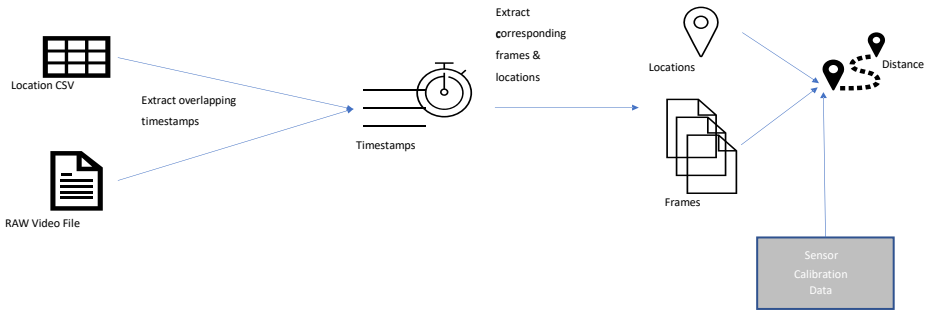

Figure 5.4: A High-level Overview of Attached Metadata to a Frame

The metadata is saved on the client-side in JSON format with the frame, selected points in each view, and calculated distance all available to be output in a CSV report upon request. Each RAW

video format chosen is native to the sensor capturing the video. In the scope of this paper, the steps to obtain the RAW video frames and ready the frames for display in readily available libraries are abstracted to the following:

1. Obtain the RAW header and file format from the manufacturer of the sensor.
2. Read the frames into a format that supports indexing and subscription if the programming language supports it.
3. Convert each frame to a supported color space: RGB, greyscale, etc.
4. Pass an individual frame or stream of frames to the browser as a JSON object, array, or similar type to be loaded as an image object.

Once the frame or set of frames is in a JSON or array like type, encapsulating the frames with additional relevant metadata for any use case is readily accessible.

5.2.6 Use Case

The use case that drove this work is that of measuring object distance in RAW surveillance video. In Figure 5.5, a synthetically generated example of two cameras watching the same object (the shuttle) with an OOI (the block). With the known locations of the cameras, the sensor calibration data (known distances per pixel), the location metadata of the centroid of the matrix, and the distance between the known object in the centroid of the image, the distance between the objects is calculated and attached to the frame image object.

The RAW Video Analysis Tool is made up of the following elements:

- A slider above the pair of frames to allow the user to slide through the video frames.
- The two frames are loaded as Plotly graph objects that allow for selection of XY positions within each frame. This XY position informs the distance calculations using the location

metadata of the object in the centroid of the image, the camera calibration data, and the OOI.

- A slider below each frame to reduce the dynamic range and a histogram of the entire video that can be loaded that informs the frequency of the pixel values of the video.

Figure 5.5: RAW Video Analysis Tool with Frames Loaded

The distance data generated from this report is used to output automatically generated CSV, PDF, and three-dimensional Plotly Dash graph HTML files that communicate statistics such as:

- Distance between objects in different coordinate systems, planes of reference, and vectors
- Limitations of the data being reported (such as when OOIs are not in the field of view)
- Animations showing distance events in reference to video time

5.2.7 Process Improvements

The predecessor tool that they data analysis team used before was a desktop application that would accept only one of these tasks at a time:

- Single RAW video conversion to MP4
- Combined RAW video conversion to MP4 (combining two or more videos by time sync into one video)
- Distance reporting

Leveraging open-source libraries to convert videos in the background of the web application, the development team produced several process improvements:

- The time to convert a single RAW video to MP4s reduced by ~50% in the mean
- The ability to continue producing distance reports while processing RAW videos in the background

5.2.8 Conclusion & Future Work

The proposed framework presents a REST software stack that accepts large RAW binary video types and transforms the frame data to be loaded in the browser. The transformation of the RAW frames allows for additional metadata to be attached to the frame and stored in the browser as multimedia data to be used for automated report generation. The approach for loading the RAW frames can be used agnostic of language given the GUI is HTML and JavaScript based – allowing the use of RAW videos in other web-based multimedia applications. Forward work on this framework includes:

- Automating the identification of the OOIs and frames of interest using machine learning methodologies
- Improve streaming of the video frames for the ability for play through the RAW videos in the browser during analysis without converting to an MP4 or MPEG format

5.3 Discussion

Company X implemented FlexOps which matured to FlexDevOps alongside developing and delivering the software system which was detailed in section 5.2. Laporte et al established a model shown in Figure 5.6 named the Systems Engineering Basic Profile that was developed as an ISO standard for implementing both program management and Systems Definition and Realizations (SR) processes into small businesses. Company X implemented a modified version of Systems Engineering Basic Profile in order to effectively deliver continuous improvements to the RAW Video Analysis Framework.

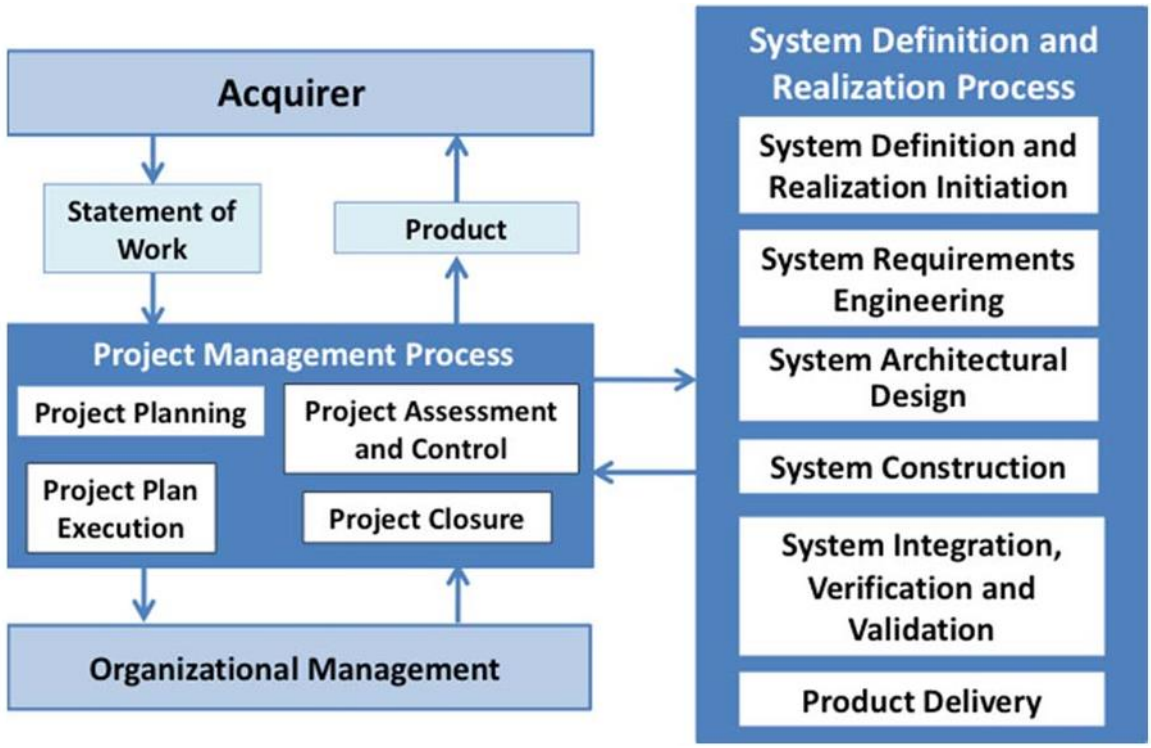

Figure 5.6. The Systems Engineering Basic Profile (Laporte 2014)

This model presents actionable steps to incorporate PM and SE processes into businesses, but doesn't take into account the potential need of a business to implement iterative design and deployment throughout a project. FlexOps is a project execution strategy that can be inserted into

the PM and SR processes that are proposed by Laporte and WG24 of ISO/ISC. FlexOps or FlexDevOps can be inserted into the Project Plan Execution; System Requirements Engineering; System Architectural Design; System Construction; and System Integration, Verification, and Validation steps of the Systems Engineering Basic Profile as a definable project execution plan for software systems implemented by VSEs (Figure 5.7).

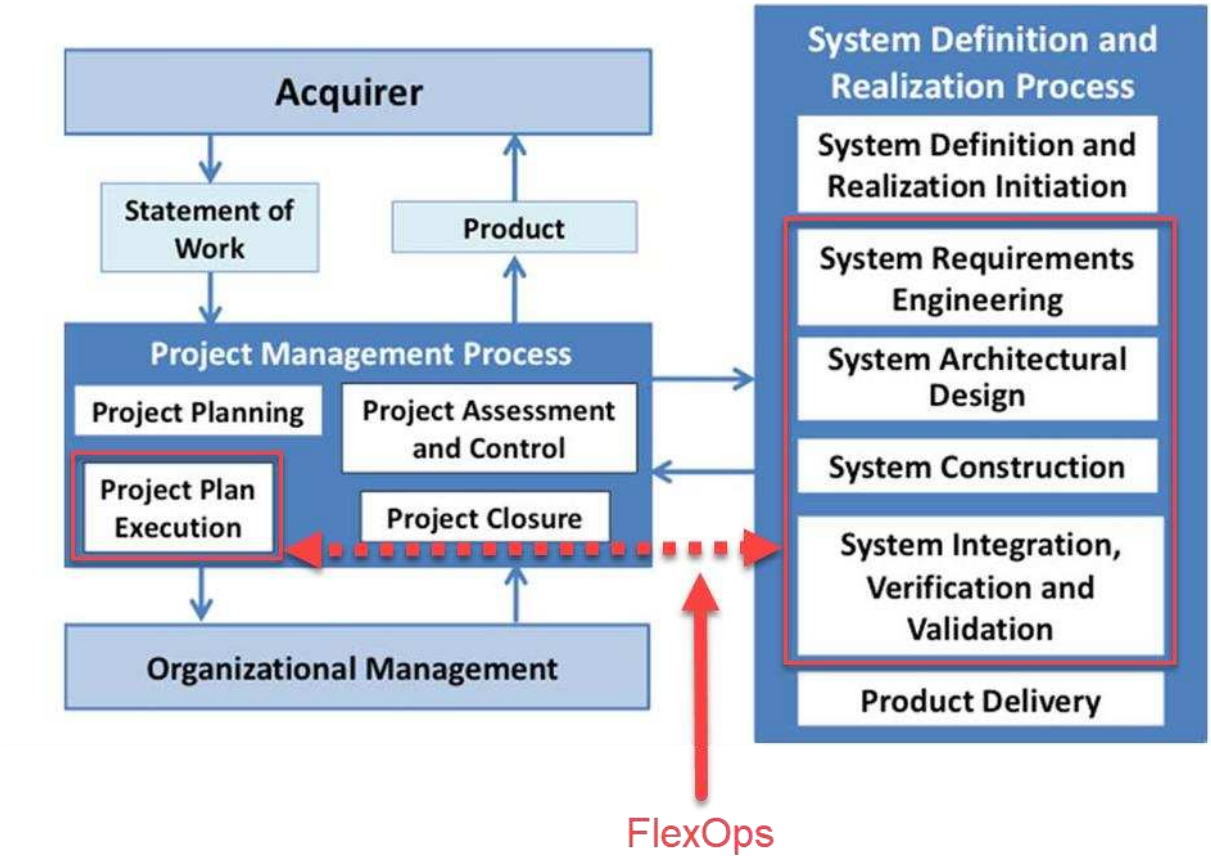

Figure 5.7. FlexOps and FlexDevOps Insertion Points into the Systems Engineering Basic Profile

5.3.1 FlexOps as a Project Plan Execution Strategy

FlexOps (we will be referring to FlexOps and FlexDevOps as one concept for the context of this section) can be implemented as a project execution strategy that directly influences these core elements of the SR process: Systems Requirements Engineering; System Architectural Design;

System Construction; and System Integration, Verification, and Validation. FlexOps can also impact Product Delivery through the implementation of continuous integration, delivery, and deployment (CI/CD): this is the primary design goal for DevOps that is addressed by FlexOps. However, depending on an organizations' maturity of CI/CD, product delivery may be at the end of the project lifecycle as proposed in the SR process of the SE Basic Profile. We will discuss how each FlexOps element maps to the above steps of the SR process and achieves needed elements of a VSE's project execution strategy (Figure 5.8).

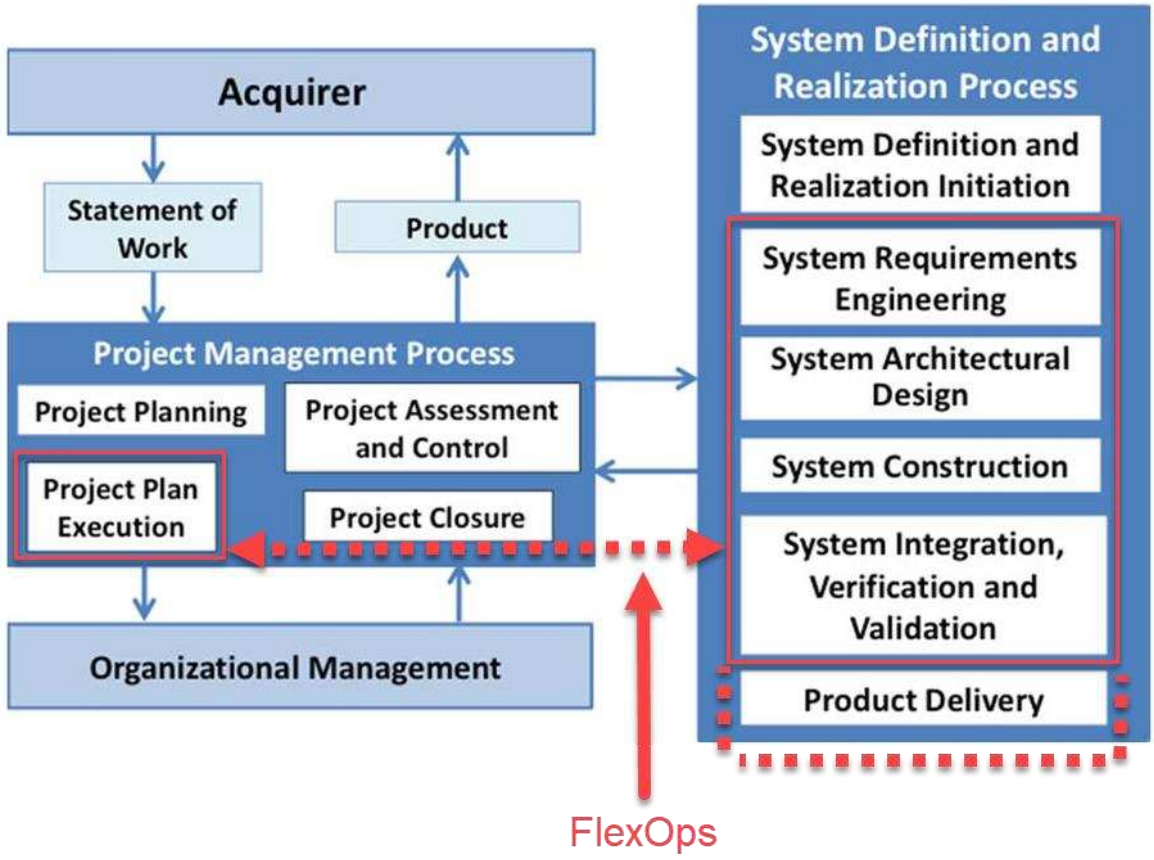

Figure 5.8. FlexOps and FlexDevOps Insertion Points into the Systems Engineering Basic Profile with Optional Product Delivery

5.3.1 Systems Requirements Engineering

The Systems Requirements Engineering step in SR maps to the Planning stage of FlexOps.

5.3.2 System Architectural Design

The System Architectural Design step in SR maps to the Modeling stage of FlexOps.

5.3.3 System Construction

The System Construction step in SR maps to the Code stage of FlexOps.

5.3.4 System Integration, Verification, and Validation

The System Integration, Verification, and Validation steps directly map to the Test and Deploy stages of FlexOps and the Test, Release, Deploy, and Operate stage of FlexOps. Particularly, Test maps to system verification validation and deploy maps to system integration. The Monitor stage maps both to the verification and validation steps in the SR process as well as project assessment and control in the PM process.

5.3.5 Product Delivery

As mentioned previously, the Product Delivery step maps directly to Deploy in FlexOps and Release, Deploy, and Operate in FlexDevOps. This capability is highly dependent on a VSE's CI/CD pipeline. If a VSE has the maturity to execute the release, deploy, and operate steps to their customer, they may have reached DevOps or ModDevOps maturity.

5.3.2 Limitations and Other Considerations

The limitations to using FlexOps as an execution strategy mirrors those mentioned in Chapter 3 Section 3.2.8:

- FlexOps inherently lacks the benefits of deploying full scale DevOps and MBSE processes independently and combined. It should be used in organizations that are small and are needing a stepping stone to implementing these engineering project frameworks.

- FlexOps is intended for software systems.
- FlexOps is not intended for development of mission critical software.

FlexOps is a potential project execution strategy for implementing elements of requirements and system modeling and iterative design where a VSE believes that a combination of MBSE and DevOps strategies will achieve the customer's desired end product in a way that meets the Statement of Work (SOW) defined by the Acquirer.

Chapter 6. Discussion, Summary, and Conclusion

6.1 Discussion

The FlexOps and FlexDevOps models were formalized by the need and experience of the software project team for Company X. However, the model was implemented in its infancy during the implementation of the tool in the small department described in Chapter 4. The goal is to allow small organizations of any kind to use this methodology: small departments that operate like their own small business, stand alone small businesses, or any applicable group.

The concept of implementing MBSE or DevOps iteratively or in phases is not new. In addition, hybridization of MBSE and DevOps for software projects is not new as shown by Hugues & Yankel's ModDevOps model (Hugues & Yankel 2021). However, these iterative implementations and models may be too mature for VSEs with micro-project teams of 1-5 members. VSEs with software engineering projects should have the opportunity to take advantage of the benefits of MBSE and DevOps and omit elements of either paradigm that is not feasible or cannot serve them at their current size. FlexOps proposes a model that VSEs may use to achieve more successful outcomes from software engineering projects.

6.1.1 Comparing Alternate Models and Strategies

6.1.1.1 Waterfall

In a comparison study of Waterfall and Agile software development methodologies, Andrei et al characterized Waterfall as a structured process in which requirements are defined in the beginning (and will not change), and each step cannot be modified and has to be completed sequentially (Andrei et al 2019). Waterfall has multiple fixed phases requiring the information and analysis

from the previous phase: requirements, design, coding, testing, and operation (Van Casteren 2017). Waterfall works best with fixed, well-defined requirements and scope. FlexOps works best in projects that require rapid development and constant customer feedback.

6.1.1.2 Agile

Abrahamsson et al defined Agile as software development that is incremental, cooperative, straightforward, and adaptive (Abrahamsson et al 2017). There are many agile software development methods such as Scrum, Kanban, and Scaled Agile Framework (SAFe) but they all have the characteristics defined as Agile above at the core of the approach. Based on Abrahmsson et al.'s definition of Agile, FlexOps is an Agile software development methodology. FlexOps meets the Agile characteristics by being (Abrahamsson et al 2017):

- Incremental: FlexOps is designed to develop and deliver software based on customer feedback on a continuous basis at a pace set by the project needs and requirements.
- Cooperative: The success of FlexOps hinges on the teamwork of the project team and communication with the customer.
- Straightforward: The FlexOps model is visually presented to be continuous similar to DevOps and with steps that are easy to understand.
- Adaptive: FlexOps is intended and designed to scale with a VSE's size and resources.

As a result of FlexOps being an Agile project management strategy, it inherits the weaknesses of Agile methods: getting off track due to changing requirements (McCormick 2012) and increased pressure on employees to present progress (Dima & Maassen 2018).

6.1.1.3 Document Based Systems Engineering (DBSE)

DBSE focuses on "written documents based on text as sources of managerial information" (Swanson & Culnan 1978). Although a VSE may use document-based program (Microsoft Word,

PowerPoint) to implement FlexOps to report their models in the implementation's infancy, the model-based approach of making models "central and indispensable artifacts throughout a product's lifecycle" ensures that the project is model-based and not document-based (Feiler & Gluch 2012), (Henderson & Salado 2021).

6.1.1.4 MBSE

FlexOps intentionally pulls from important MBSE concepts like requirements traceability, model-based engineering, and product verification & validation (V&V) through testing. The goal of leveraging these concepts is to allow a VSE to experience perceived benefits of MBSE such as those listed by Henderson and Salado (Henderson & Salado 2021):

- Better communication and information sharing
- Increased traceability
- Better complexity management
- Improved consistency
- Increased capacity for reuse
- Reduced project cost

FlexOps attempts to avoid some perceived difficulties of MBSE adoption were attempted to be avoided such as implementation cost, complexity, and adoption strategy (Chami et al 2018). However, as a result of FlexOps adopting MBSE concepts, it by default inherits some of the disadvantages and advantages of the paradigm.

6.1.1.5 DevOps

FlexOps also pulls from DevOps to take advantage of the continuous integration and continuous deployment (CI/CD) concept. Leite et al define DevOps to be a collaborative and multidisciplinary effort within an organization to automate continuous delivery of new software versions, while

guaranteeing their correctness and reliability (Leite et al 2019). By automating the delivery of software to the customer and continuously improving the product, a VSE can take advantage of the two pillars of DevOps named by Leite et al (Leite et al 2019):

1. Human collaboration across departments (or disciplines)
2. Automation.

As with MBSE, because FlexOps adopts DevOps practices, it inherits its challenges as well. Some of the issues with DevOps are the lack of consensus on how to collaborate and how to structure the organization to effectively implement the paradigm. FlexOps is aimed at VSEs with the goal of minimizing the occurrence of the above issues. As a VSE scales, those issues are bound to emerge and the VSE will have to address the issues in the future: knowingly accepting "project management technical debt".

6.1.2 FlexOps Future Growth

The authors plan on implementing FlexOps in more VSEs and to assess the benefits and detriments of the paradigm across more case studies. FlexOps was designed for software engineering projects, but can be expanded to hardware engineering projects as well. The concept of Agile manufacturing (Gunasekaran 1999) focuses on response to requirements changes and customer needs. A VSE can take advantage of the continuous FlexOps process to rapidly improve manufacturing processes and hardware products by replacing the "Code" step with "Manufacture."

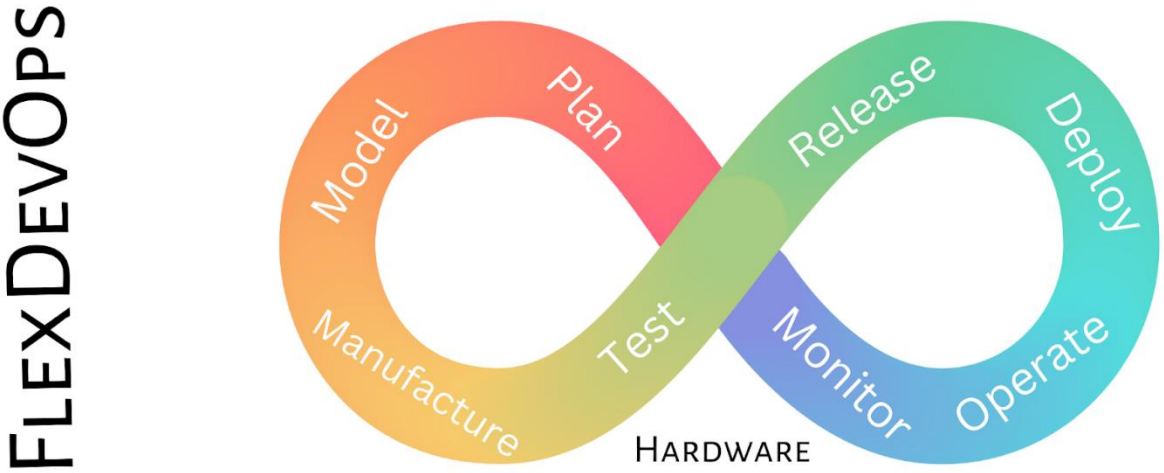

Figure 6.1 FlexDevOps as a Hardware Engineering Project Execution Strategy

Moreover, a VSE implementing a software-hardware system can implement manufacturing and software methodologies in tandem with feedback in between each process.

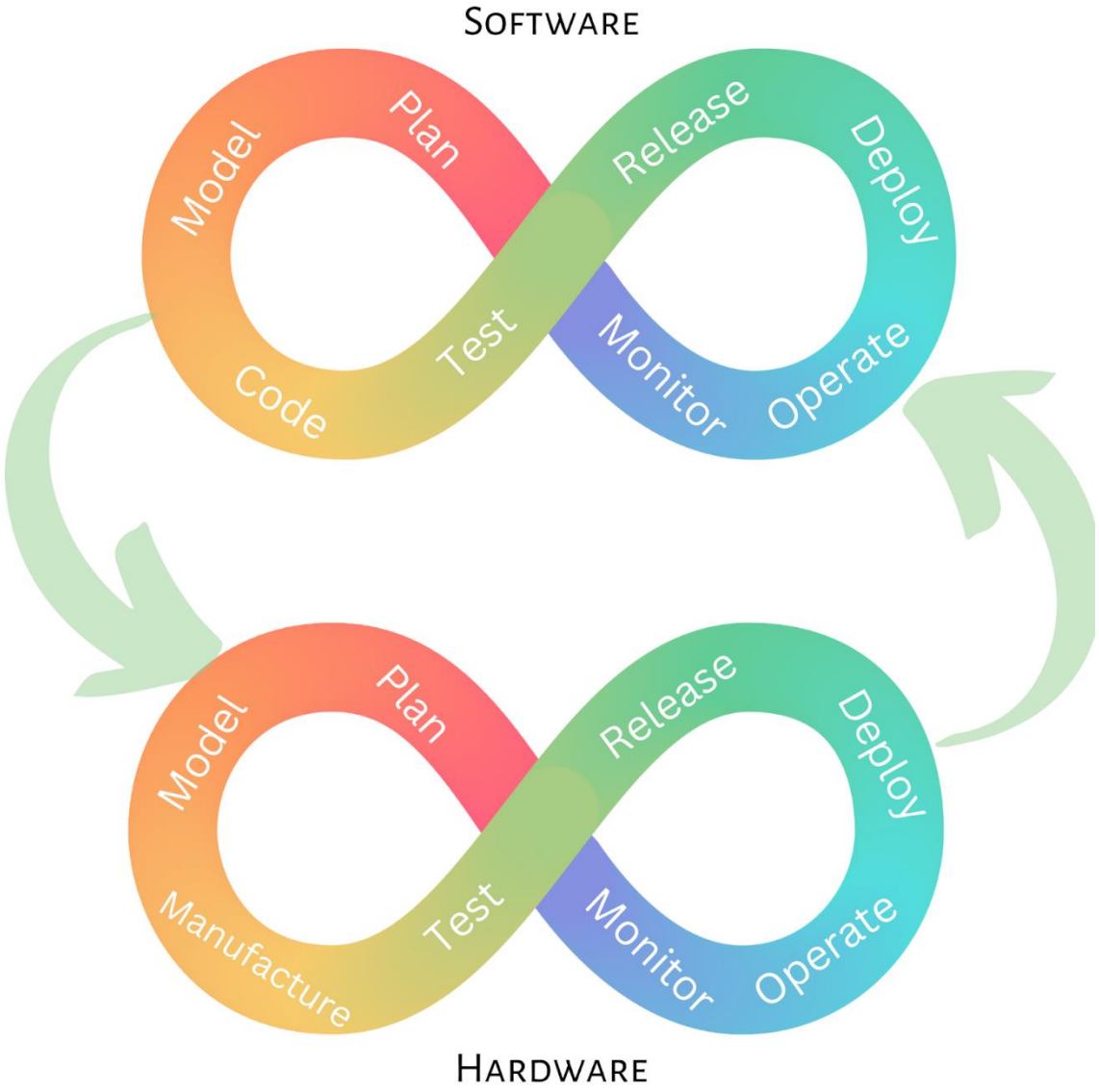

Figure 6.2 Parallel Software & Hardware FlexOps Processes

FlexOps ventures to provide actionable, continuous steps that give VSEs the opportunity to implement Agile and systems engineering practices. INCOSE's Vision 2035 puts forth that one of the challenges of the future of systems engineering are "Tools and Data Integration" and "Software Complexity, Agility, and Scale" (INCOSE 2022). Dove et al described eight core aspects of agile systems engineering to address the concerns of INCOSE's Vision 2035 (Dove et al 2023):

1. *Product Line Architectures*

2. *Iterative Incremental Development*
3. *Attentive Situational Awareness*
4. *Attentive Decision Making*
5. *Common-Mission Teaming*
6. *Shared-Knowledge Management*
7. *Continual Integration & Test*
8. *Being Agile: Operations Concept*

FlexOps maps to the eight core aspects of agile systems engineering to allow VSEs to lean forward to the systems of 2035 in the following ways:

1. *Product Line Architectures*: FlexOps inherently is designed to allow a VSE to experiment with tooling, products, and processes that can support their product design, engineering, and delivery needs.
2. *Iterative Incremental Development*: FlexOps takes on the continuous nature of DevOps and can be incrementally phased to scale to mature project execution strategies. Allowing for development of not only engineering products but also internal business processes.
3. *Attentive Situational Awareness:* FlexOps is a model, therefore it inherently lacks the ability to "teach" how to adjust to emergent risks and opportunities. However, it provides a continuous process on how to incorporate and potentially address risks and opportunities as they arise. It is on the implementers to assess and address risks and opportunities in the plan, model, and test steps of FlexOps.
4. *Attentive Decision Making*: Reflecting the above comment, FlexOps isn't meant to teach how to adjust and assess actions, but provides a continuous process on how to incorporate new information into system development.

5. *Common-Mission Teaming*: FlexOps has stakeholder feedback monitoring and assessment built into the model, but it is up to the VSE to incorporate and facilitate open lines of communication amongst the product team and stakeholders to ensure common-mission teaming.
6. *Shared-Knowledge Management*: FlexOps is intended to be implemented to include a single source of truth, however this is up to the VSE to implement. One single source of truth also facilitates the common-mission teaming mentioned above.
7. *Continual Integration & Test*: FlexOps is intended to be implemented with an integrated test suite, however implementation of this is up to the VSE and it's maturity, resources, and technical know-how to do so.
8. *Being Agile: Operations Concept*: FlexOps is designed to be Agile, prioritizing continuous integration and improvement in a dynamic small business environment. FlexOps allows scalability as a VSE's knowledge evolves.

FlexOps attempts to provide an avenue to INCOSE's *Systems Engineering Vision 2035*, but implementation of all of the Agile Systems Engineering concepts is up to the VSE's maturity, resources, and technical knowledge (INCOSE 2022).

6.2 Summary

6.2.1 FlexOps Benefits

6.2.1.1 A Stepping Stone

FlexOps provides a flexible, scalable model to learning MBSE and DevOps practices. This allows the VSE to pick the level of project management complexity needed for the size and scope of their

current projects. This allows a VSE to implement DE practices to lean forward to meet industry directives like the DoD's *Digital Engineering Workforce Plan* (DoD 2022) or INCOSE's *Systems Engineering Vision 2035* (INCOSE 2022).

6.2.1.2 Agile, MBSE, and DevOps Strategies in One

FlexOps was developed to aid VSEs in utilizing DE practices. *Systems Engineering Vision 2035* and the *DoD Engineering Workforce Plan* both highlight the importance of DE practices, specifically model-based engineering practices. However they do not lay out an exact "how" roadmap. FlexOps provides a stepping stone that incorporates proven DE strategies for VSEs to lean forward to meet industry standards and future directives.

6.2.2 FlexOps Detriments and Limitations

6.2.2.1 Not Intended for Mission Critical Software

FlexOps is not intended to be implemented for use in mission critical software such as the examples given illustrating how ModDevOps was implemented in several mission critical systems (Carnegie Mellon University 2021).

6.2.2.2 Continuous Improvement, but Maybe a Lack of Continuous Integration

The case studies presented did not showcase an integrated model to the software products. This can be implemented as the technical aptitude and resources within a VSE increase. Free modeling tools like Eclipse Papyrus and Modelio, mentioned in Chapter 3.3 can be used to model the system during early implementations of FlexOps. However, for integrated models, the VSE will likely need to implement tools like Cameo.

6.2.2.3 Not a Full Scale Implementation of Either DevOps or MBSE

A lack of full scale implementation of either MBSE nor DevOps inherently lacks some of the benefits of either paradigm, let alone both. The goal of the FlexOps methodology is to give VSEs a stepping stone to realized process improvements by incorporating continuous integration, continuous improvement, and modeling practices.

6.3 Conclusion

This work introduces the FlexOps and FlexDevOps models, which were developed based on the needs and experiences of the software project team at Company X and the small R&D department mentioned in Chapter 6. The models were initially implemented in a small department, and their aim is to provide small organizations, including standalone small businesses or small departments operating independently, with a methodology that combines Agile, MBSE, and DevOps practices. While the iterative implementation of MBSE and DevOps is not new, existing models may be too mature for VSEs with micro-project teams of 1-5 members. To address this, FlexOps proposes a model that allows VSEs to benefit from both MBSE and DevOps while omitting elements that are not feasible for their size.

When comparing FlexOps to other software development methodologies like Waterfall and MBSE, FlexOps's suitability for projects requiring rapid development and continuous customer feedback is emphasized. FlexOps is categorized as an Agile software development methodology with the characteristics of being incremental, cooperative, straightforward, and adaptive. The FlexOps model adopts and integrates MBSE and DevOps principles, inheriting both their

advantages and disadvantages. FlexOps is intended to allow VSEs to experiment with tools, products, and processes that suit their engineering needs. Furthermore, the authors discuss FlexOps's potential future growth, including expansion to hardware engineering projects and integration of manufacturing processes with software methodologies.

This work highlights the benefits of FlexOps as a flexible and scalable model, combining Agile, MBSE, and DevOps practices, while acknowledging its limitations, such as not being suitable for mission-critical software and not representing a full-scale implementation of either MBSE or DevOps. Despite these limitations, FlexOps offers VSEs a stepping stone towards realizing process improvements and leveraging continuous integration, continuous improvement, and modeling practices. FlexOps is presented as a tool to make DE practices accessible to organizations of all sizes.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J., 2017. Agile software development methods: Review and analysis. arXiv preprint arXiv:1709.08439.

Andrei, B.A., Casu-Pop, A.C., Gheorghe, S.C. and Boiangiu, C.A., 2019. A study on using waterfall and agile methods in software project management. Journal of Information Systems & Operations Management, pp.125-135.

Anon n.d., "DevOps," Office of the Chief Software Officer, U.S Air Force, viewed 11 December, 2022, <https://software.af.mil/training/devops/>

Bonnet, S., Voirin, J.-L., Normand, V., & Exertier, D. (2015). Implementing the MBSE Cultural Change: Organization, Coaching and Lessons Learned. *INCOSE International Symposium*, *25*(1), 508–523. https://doi.org/10.1002/j.2334-5837.2015.00078.x

Bucena, I. and Kirikova, M., 2017, August. Simplifying the DevOps Adoption Process. In BIR Workshops (pp. 1-15).

Bunday, B. D., Bishop, M., McCormack, Jr., D. W., Villarrubia, J. S., Vladar, A. E., Dixson, R., Vorburger, T. V., Orji, N. G., & Allgair, J. A. (2004). Determination of optimal parameters for CD-SEM measurement of line-edge roughness. *Metrology, Inspection, and Process Control for Microlithography XVIII*, *5375*, 515. https://doi.org/10.1117/12.535926.

Carnegie-Mellon University, 2021. Model-Based Engineering with AADL.

Chami, M., Aleksandraviciene, A., Morkevicius, A., & Bruel, J.-M. (2018b). Towards Solving MBSE Adoption Challenges: The D3 MBSE Adoption Toolbox. *INCOSE International Symposium*, *28*(1), 1463–1477. https://doi.org/10.1002/j.2334-5837.2018.00561.x

Crawford, M. (2023, June 23). *6 Top Challenges Facing Engineering Firms in 2023*. The American Society of Mechanical Engineers.

Department of Defense. (2018). *Digital Engineering Strategy*. www.acq.osd.mil/se

Department of Defense. (2022). *Digital Engineering Workforce Plan*. www.cto.mil

Dima, A.M. and Maassen, M.A., 2018. From Waterfall to Agile software: Development models in the IT sector, 2006 to 2018. Impacts on company management. Journal of International Studies (2071-8330), 11(2).

Dove, R., Orosz, M., Lunney, K., & Yokell, M. (2023). *Agile Systems Engineering-Eight Core Aspects*. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=653267

Faustino, J., Adriano, D., Amaro, R., Pereira, R., & da Silva, M. M. (2022). DevOps benefits: A systematic literature review. *Software: Practice and Experience*, *52*(9), 1905–1926. https://doi.org/10.1002/spe.3096

Feiler, P., & Gluch, D. (2012). *Model-based engineering with AADL: an introduction to the SAE architecture analysis & design language*.

Gunasekaran, A., 1999. Agile manufacturing: a framework for research and development. International journal of production economics, 62(1-2), pp.87-105.

Hale, J., & Hobeb, A. (2020). *INCOSE Model-Based Capabilities Matrix and User's Guide*.

Hamunen, J., 2016. Challenges in adopting a Devops approach to software development and operations.

Henderson, K. and Salado, A., 2021. Value and benefits of model-based systems engineering (MBSE): Evidence from the literature. Systems Engineering, 24(1), pp.51-66.

Houde, R., Laporte, C. Y., & Blondelle, G. (2016). ISO/IEC 29110 Deployment Packages and Case Study for Systems Engineering: The "Not-So-Secret" Ingredients That Power the Standard. *INCOSE International Symposium*, *26*(1), 1276–1292. https://doi.org/10.1002/j.2334-5837.2016.00226.x

Hugues, J. and Yankel, J., 2021. From Model-Based Systems and Software Engineering to ModDevOps. CMU Research Review.

Hugues, J., Yankel, J., Hudak, J., & Hristozov, A. (2022). *TwinOps: Digital Twins Meets DevOps*. http://www.sei.cmu.edu

INCOSE. *(2021). Systems Engineering Vision 2035. International Council on Systems Engineering.*

ISO/IEC: ISO/IEC TR 29110-5-1-1:2012 Software engineering – Lifecycle profiles for Very Small Entities (VSEs) Part 5-1-1: Management and engineering guide: Generic profile group: Entry profile, Geneva (2012)

Laporte, C. Y., Alexandre, S., Renault, A., & Crowder, K. V. (2008). The development of international standards for very small enterprises. *18th Annual International Symposium of*

the International Council on Systems Engineering, INCOSE 2008, *5*, 2544–2555. https://doi.org/10.1002/j.2334-5837.2008.tb00831.x

Laporte, C. Y., Munoz, M., Mejia Miranda, J., & O'Connor, R. V. (2017). Applying Software Engineering Standards in Very Small Entities: From Startups to Grownups. *IEEE Software*, *35*(1), 99–103. https://doi.org/10.1109/MS.2017.4541041

Larrucea, X., O'Connor, R. V., Colomo-Palacios, R., & Laporte, C. Y. (2016). Software Process Improvement in Very Small Organizations. *IEEE Software*, *33*(2), 85–89. https://doi.org/10.1109/MS.2016.42

Lazuardi, M., Raharjo, T., Hardian, B., & Simanungkalit, T. (2021). Perceived Benefits of DevOps Implementation in Organization: A Systematic Literature Review. *2021 10th International Conference on Software and Information Engineering (ICSIE)*, 10–16. https://doi.org/10.1145/3512716.3512718

Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A Survey of DevOps Concepts and Challenges. *ACM Computing Surveys (CSUR)*, *52*(6). https://doi.org/10.1145/3359981

Luxem, K., Sun, J. J., Bradley, S. P., Krishnan, K., Yttri, E., Zimmermann, J., Pereira, T. D., & Laubach, M. (2023). Open-source tools for behavioral video analysis: Setup, methods, and best practices. *ELife*, *12*. https://doi.org/10.7554/eLife.79305

McCormick, Mike. "Waterfall vs. Agile methodology." MPCS, N/A 3 (2012).

McDermott, T.A., Hutchison, N., Clifford, M., Van Aken, E., Salado, A. and Henderson, K., 2020. *Benchmarking the benefits and current maturity of model-based systems engineering across the enterprise.* Systems Engineering Research Center (SERC).

McLeod, C. (2015). *A Framework for Distributed Deep Learning Layer Design in Python*. http://arxiv.org/abs/1510.07303

Ploeg, C., Lai, K., & Olechowski, A. (2022). *Prioritization of Best Practices in the Implementation of Model-Based Systems Engineering*. https://doi.org/10.1002/iis2.12975

Revell, D., Gross, D. C., Zhou, G., & Hernandez, A. (2023). Applying a MBSE Methodology in Small Scale Technology Development [1]. *2023 IEEE International Systems Conference (SysCon)*, 1–6. https://doi.org/10.1109/SysCon53073.2023.10131099

Rising, L. and Janoff, N.S., 2000. The Scrum software development process for small teams. IEEE software, 17(4), pp.26-32.

Simpson, C., Vinson, K., Hooper, R. J., & Simske, S. (2019). Fiber Morphology Analysis for Directed-Energy Deposition Manufacturing Process. *NIP & Digital Fabrication Conference*, *35*(1), 75–78. https://doi.org/10.2352/ISSN.2169-4451.2019.35.75

Simpson, C. and Simske, S., 2023, July. Scalable, Flexible Implementation of MBSE and DevOps in VSEs: Design Considerations and a Case Study. In INCOSE International Symposium (Vol. 33).

Swanson, E.B. and Culnan, M.J., 1978. Document-based systems for management planning and control: A classification, survey, and assessment. MIS Quarterly, pp.31-46.

Van Casteren, W., 2017. The Waterfall Model and the Agile Methodologies: A comparison by project characteristics. Research Gate, 2, pp.1-6.

Villarrubia, J. S. (2005). Issues in Line Edge and Linewidth Roughness Metrology. *AIP Conference Proceedings*, 386–393. https://doi.org/10.1063/1.2062992

Wymore, A.W., 2018. Model-based systems engineering (Vol. 3). CRC press.

Zarour, M., Alhammad, N., Alenezi, M., & Alsarayrah, K. (2019). A research on DevOps maturity models. *International Journal of Recent Technology and Engineering*, *8*(3), 4854–4862. https://doi.org/10.35940/ijrte.C6888.098319