3-10-2021

Developing a Cost of Delay (CoD) Framework for the DoD & Analyzing the Current State of Air Force Agile Cost Estimation

James Goljan

Developing a Cost of Delay (CoD) Framework for the DoD & Analyzing the Current State of Air Force Agile Cost Estimation

THESIS

March 2021

James Goljan, 1st Lieutenant, USAF

AFIT-ENV-MS-21-M-231

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-21-M-231

Developing a Cost of Delay (CoD) Framework for the DoD & Analyzing the Current State of Air Force Agile Cost Estimation

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cost Analysis

James Goljan, 1st Lieutenant, USAF

March 2021

AFIT-ENV-MS-21-M-231

Developing a Cost of Delay (CoD) Framework for the DoD & Analyzing the Current State of Air Force Agile Cost Estimation

James Goljan, 1st Lieutenant, USAF

Committee Membership:

Dr. Jonathan D. Ritschel, PhD
Chair

Lieutenant Colonel Scott T. Drylie, PhD
Member

Dr. Edward D. White, PhD
Member

Abstract

The study has two research objectives. The first research objective develops a proof of concept for a Cost of Delay (CoD) framework to evaluate the Air Force's Kessel Run Software Factory. The CoD for a given requirement is defined as the value that could be produced over a length of time if said requirement was available immediately (Arnold & Yuce, 2013). CoD provides a means to *justify* and *prioritize* competing interests regarding budgetary, manpower, and overall strategic decisions.

DoD organizations can employ a cost minimization framework to conduct CoD Analysis. More specifically, this study's CoD Analysis discovers approximately $16.54M annually in CoD associated with failing to modernize the legacy Theater Battle Management Core System. Additionally, the research identifies four high value requirements for reprioritization within the Kessel Run All Domain Operations Suite (KRADOS) portfolio which would save over one thousand dollars per week in opportunity costs. The proof of concept demonstrates the potential for large savings.

The second objective compares the predominant literature on Agile cost estimating techniques against the methods utilized within 11 Air Force Agile Software Factories. The study identifies the 11 Software Factories are in agreement with the extant literature in moving away from SLOC as the means to assess cost. Additionally, statistical analysis demonstrates disparities in cost estimating strategies between the literature and the Agile Air Force organizations. The results demonstrate Software Factories eschew trends in the literature towards more complicated cost models that incorporate machine learning with Data-Based techniques in lieu of simpler models that utilize Engineering Build-up Capacity Based techniques.

Acknowledgments

I'd like to say thank you to my advisor who provided me with countless hours of counsel and feedback. Additionally, I would like to thank the Kessel Run team for enabling me to work alongside them. And lastly, thank you to my family and friends to whom I made a solemn pledge that I'd work my hardest to finish this research.

James Goljan

Table of Contents

List of Figures

List of Tables

Developing a Cost of Delay (CoD) Framework for the DoD & Analyzing the Current State of Air Force Agile Cost Estimation

I. Introduction

Software development in industry has experienced a largescale evolution within the last few decades. Incremental development and Agile processes have become commonplace in the private sector. Despite the commercial sector's rapid advancement, the Department of Defense (DoD) has continued to utilize software development techniques and strategies from the 1970s (Defense Science Board, 2018). The Air Force's Kessel Run Software Factory represents an abrupt departure from these antiquated DoD practices. Kessel Run employs a modern, Agile software development environment. But changing to the Agile development environment has brought new challenges. One challenge Kessel Run leadership currently faces is a search for a new way to *prioritize* and *justify* their various product lines. Kessel Run leaders must make judgment calls regarding the best way to allocate manpower and *prioritize* specific applications. Additionally, leadership must *justify* the existence of their various product lines by demonstrating the value they add to the warfighter. To better understand how to achieve these goals, this study uses Kessel Run as a testbed to create a DoD based Cost of Delay (CoD) framework that quantifies future opportunity costs. The framework will assist with organizing resources in a more cost effective manner and providing a more holistic valuation of a product's operational impact.

The second part of this research investigates the implications for the cost estimator from adopting Agile methods. Agile originated in 2001 when a group of engineers published the Agile Manifesto which created a new way to approach software development (Regan, Lapham, Wrubel, Beck, & Bandor, 2014). Since its inception, Agile practices have become widely

adopted throughout private industry in companies such as IBM (Randall, 2014). Kessel Run is one of the first major DoD Software Factories to fully embrace the Agile approach. With the relatively new development of Agile Software Factories in the DoD, research into the best practices for Agile software cost estimation is lacking. Traditionally, Air Force software cost estimates commonly rely upon Source Lines of Code (SLOC) as a way to measure a program's size (McQuade et al. 2019). But the Agile development environment is unique. The Agile mentality relies on flexibility and working requirements in small iterations. Utilizing SLOC is discouraged as it constrains the team to a pre-conceived work estimate and because it can incentivize the contractor to develop inefficient code (Bhatt, Tarey, & Patel, 2012). As a result, Agile programs require cost analysts to adopt new methods for proper cost estimation. For example, Agile programs may use techniques such as level-of-effort estimates which incorporate the number of team members and the expected duration of time to work on a new requirement (Rosa W. , Madachy, Clark, & Boehm, 2020). Due to the Air Force's lack of experience and familiarity with Agile, the second objective of this research is to investigate the current state of Agile software cost estimation and provide recommendations for cost analysts.

CoD Background

CoD provides an organization with a methodology to optimize their portfolio's structure. Kessel Run is interested in adopting a CoD methodology tailored to the unique nature of a public sector organization. To achieve this, it is first imperative to garner a basic understanding of how the private sector defines CoD and then delineate the elements that comprise CoD. The CoD for a given requirement is defined as the value that could be produced over a length of time if said requirement was available immediately (Arnold & Yuce, 2013). The CoD framework consists of three components: *benefit type*, *urgency profile*, and *development duration* (Arnold & Yuce,

2013). Typically, CoD Analysis identifies *benefits* as either increasing revenue, protecting revenue, reducing cost, or avoiding cost. Additionally, CoD Analysis categorizes *urgency profiles* as Short Life-Cycle Peak Affected by Delay, Long Life-Cycle Peak Unaffected by Delay, Short Life-Cycle Peak Affected by Delay, or Impact of External Deadline. Lastly, CoD Analysis determines the *development duration* which is the amount of time require to complete a requirement from start to finish. Taking a requirement's benefit type and urgency profile over the development duration produces a quantitative CoD score which can be compared to other requirements. The CoD score quantifies a requirement's opportunity cost into one cumulative dollar figure. To prioritize, requirements with the highest opportunity costs should be completed first. However, the described CoD framework has been developed solely in the private sector which utilizes economic profits as the value proposition. Therefore, this research will investigate how the CoD framework can be adapted to the public nature of the DoD.

Problem Statement

There are two purposes for this research. The first is to determine a CoD framework that can be utilized within DoD organizations. The results of this effort provide decision makers with the framework to understand the opportunity cost behind postponing a new solution's implementation. A proof of concept of the constructed framework will be applied to Kessel Run's KRADOS' applications and Theater Battle Management Core System (TBMCS) contract. The research is scoped to the TBMCS contract and two specific application teams (Jigsaw and Chainsaw) within KRADOS. Improving the analysis of CoD can aid decision makers when prioritizing or justifying relevant capabilities.

The Air Force has only recently implemented Agile software development. As a result, research into the impacts to defense cost estimation is scarce. Therefore, the second purpose of

this research is to provide a present state of Agile software cost estimating. The study will analyze the current differences between Agile cost estimation in the private and public sector. A collection of Agile estimation best practices will assist Air Force cost analysts to understand the current trends.

Research Objectives/Questions

1. Which components comprise a conceptual framework for CoD in the public sector?
2. How can CoD be demonstrated in Air Force Software Factories such as Kessel Run?
3. What are the extant cost estimating methodologies employed in Agile software programs? Which of these methods are considered best practices and recommended for Air Force cost analysts?

Methodology

The first objective of the research is to determine the components that comprise the conceptual CoD framework in the public sector. The research will evaluate the differences between the CoD concept in both private industry and the DoD. Unlike private industry, public sector entities like the DoD do not operate under the profit motive. Therefore, an analysis of value in the public sector will be undertaken through a literature review and through discussions with Kessel Run program managers. From these efforts, a conceptual CoD framework germane to the public sector can be developed.

Next, the derived CoD framework will be demonstrated in the Kessel Run Air Force Software Factory. Specifically, the research will analyze the deprecation of the Theater Battle Management Core System (TBMCS) legacy contract in favor of the KRADOS Agile application teams. The analysis requires access to Kessel Run's financial database, Apptio, to extract cost

reports and estimates for the TBMCS contract. Then, development and opportunity costs associated with KRADOS applications are collected from Kessel Run subject matter experts (SME). The total CoD will be derived by integrating the cost of the TBMCS contract, the KRADOS cost estimates, and the potential opportunity cost savings identified by the application teams.

The second research objective requires analysis of the various Agile software estimation techniques utilized across industry and the DoD. Many organizations have developed their own methodologies leading to a variety of different approaches. The methodological approach to this research objective will be answered in two parts. First, through a detailed literature review regarding the *recommended* Agile software cost estimation techniques. Second, through data collected directly from 11 Agile Air Force Software Factories demonstrating what practitioners *actually* do. Comparison of the two sets of data will be accomplished through confidence intervals when possible. The results will provide the Air Force with insight on how to best adopt and implement Agile cost estimation to stay current with the recommended standard.

Scope and Limitation

The scope of this research is to analyze Kessel Run's KRADOS applications and TBMCS contract. As a proof of concept, the study is attempting to find a way forward with data that has not yet been gathered. The first research objective focuses on the theoretical adaptation and construction of an appropriate DoD CoD framework. The unique mission of the DoD to provide military readiness dictates that the needs of the warfighter typically supersede overall cost considerations. Therefore, solutions presented through CoD Analysis should not be considered the optimal and final solution, but rather as a starting point to assist decision makers. The second research objective relies on the published articles in Academia over the last 20 years

to represent the current state of Agile cost estimation. Additionally, the study utilizes Air Force Software Factories as a representative example of the DoD's approach towards Agile cost estimating.

Thesis Overview

The following chapter contains a literature review of the previous research completed on Agile software estimation and CoD concepts in the private and public sector. Following the literature review, Chapter Three outlines the methodologies used to answer the research questions. Chapter Four describes the results of applying the CoD framework to the Kessel Run and software cost estimation data. The closing chapter discusses the implications of the research for decision makers and potential areas for follow-on research.

II. Literature Review

Chapter Overview

The chapter highlights that CoD Analysis offers a way for the government to more effectively fulfill its mission to provide military readiness. However, CoD has been typically applied to commercial industry. There is a significant gap in the literature regarding an application of CoD concepts to the public sector. National defense is a public good that does not attempt to optimize profit. Therefore, the CoD framework must be restructured to accurately reflect the public good nature of national defense. Military readiness provides one way to view the relative strength of national defense. With the continued advancement of technology systems, military readiness is tied to the ability for a nation to develop and implement effective software. Agile's iterative approach offers a number of distinct advantages over traditional software development approach. The DoD has been slow to integrate Agile, but recent endeavors have been made to establish Software Factories that take advantage of its benefits. As such, the cost estimating techniques employed by the Air Force must be adapted to account for the new Agile environment.

Improving Decision-making: Cost of Delay

Leaders typically make tradeoff decisions when organizing a requirement backlog. In Freeform Dynamics 2018 Agile and DevOps report, they recorded 52% of Agility Masters in industry responded that portfolio management is the key role to the success of a program (Freeform Dynamics & CA Technologies, 2018). As an example, First in, First out (FiFo) is a commonly utilized system to organize the sequence and prioritization of work. FiFo methods are frequently used in inventory management systems to ensure that older products are used prior to newer ones (Manohar & Aappaiah, 2017). However, in a software development environment,

FiFo proves inefficient. Certain features are more critical than others and some items added to the product backlog of requirements can become obsolete over the course of time due to changing priorities or product direction. For this reason, software organizations typically utilize alternatively developed methods to assist in prioritizing their product backlog. Two such applied Agile approaches are the Kano and MoSCoW models.

The Kano model, developed in 1980 by Professor Noriaki Kano, relies on teams categorizing features depending on the needs of the customer. The approach classifies requirements into five different groupings: Must-be, One-dimensional, Attractive, Indifferent, and Reverse (Mkpojiogu & Hashim, 2016). The MoSCoW method, developed in 2004 by Dai Clegg, relies on a similar categorization of requirements into groupings: Must Have, Should Have, Could Have, and Won't Have (MoSCoW Prioritization, 2020). Both approaches are similar in that they qualitatively group requirements by degree of need of the customer. Both models rely on the assessment of SMEs to create the groupings; however, reliance on these qualitative judgements is their greatest weakness. For example, Arnold and Yuce (2013) reported that the Maersk line suffered from an issue they called the Highest Paid Person's Effect (HiPPO) effect (Arnold & Yuce, 2013). The HiPPO was typically the most senior individual in the room and would remain adamant about the importance of certain requirements during the prioritization and planning stages. When Arnold and Yuce (2013) collected a sample of 22 requirements, including the HiPPO's selection, and calculated quantitative values for each requirement they found that, in fact, eight other features appeared to be more valuable than the HiPPO's original choice (Arnold & Yuce, 2013). While an over reliance on SMEs qualitative assessments are not ideal, a more quantitative approach can help discretely distinguish between requirements.

An alternative method is the CoD which is a quantitative optimization framework to help prioritize requirements, tasks, or new work from solely a cost perspective. Reinertsen (2009) states, "if you only quantify one thing, quantify the cost of delay" (Reinertsen, 2009). CoD is calculated by assessing the impact of not having something when it is needed. In economic terms, it is the opportunity cost between having some value now versus later. If a feature creates value, then delaying its implementation will create a cost. Therefore, the goal is to quantify the economic impact of the value provided by features and ultimately prioritize the ones that have the highest CoD. In one specific implementation, the world's largest shipping company, Maersk Line, applied this economic framework across their $100M portfolio to understand the value and urgency in delivering certain tasks (Arnold & Yuce, 2013). By implementing this optimized structure they saved costs, increased profits, improved end-to-end cycle times, and reduced the number of defects. Maersk uses the business value, the value of information, and an assessment of how these values decay over time to create what they called the Cost of Delay Divided by Duration (CD3) or simply the CoD score (Arnold & Yuce, 2013). A graphic of their formula can be seen in Figure 1.

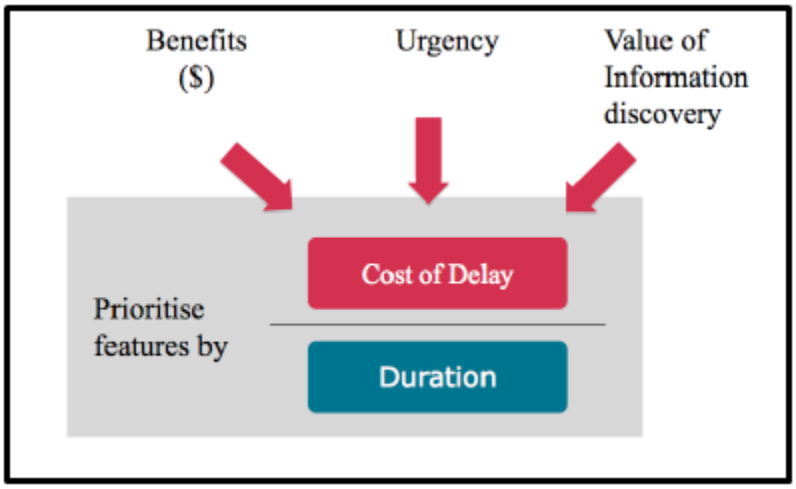

Figure 1: Maersk Line's CoD Calculation

Maersk defined value through four different benefit types: *increase revenue*, *protect revenue*, *reduce costs*, and *avoid costs* (Arnold & Yuce, 2013). As with the qualitative assessments, it may be difficult for SMEs to calculate the value associated with certain features. However, Arnold and Yuce (2013) implemented a strategy to take qualitative assessments and turn them into quantitative values. They used two strategies when meeting with SMEs to calculate quantitative figures; estimate the beneficial effects of the change and make the value of the requirement equal to the cost of alternatives (Arnold & Yuce, 2013). The beneficial effects of the change can be represented by the cost savings associated with a process' improved accuracy or clarity. Alternatively, for example, the value of a requirement to automate a process could be equated to the cost of having to execute that process manually. By continuously asking SME's "why," Arnold and Yuce (2013) were able to boil down requirements to identify their true value (Arnold & Yuce, 2013).

The key benefit of using the CoD approach is that it forces teams to produce the black and white dollar figure estimates which can be directly compared against other requirements (Black Swan Farming, 2017). There is a concern regarding the difficulty in acquiring and subsequent use of quantitative approximations (Black Swan Farming, 2017). However, by identifying assumptions, teams can provide accurate although potentially imprecise estimates (Black Swan Farming, 2017). Quantifying the CoD is essential to improving the ROI delivered with scarce resources, managing demands of stakeholders, making rational economic trade-offs, and altering the focus of the discussion towards value and urgency (Black Swan Farming, 2017).

In the case of the public sector, program offices prioritize requirements that directly assist the warfighter and thus national defense readiness levels. While the use of the CoD score helps with initial prioritization, it is worth noting that not all decisions can always be made entirely by

this score. There are other considerations such as overall business or mission strategy. CoD frames value which decision makers can use to make better informed trade-off decisions. Leaders make decisions based on needs that are tactically driven by the warfighter and strategically defined by the Air Force's required mission (McQuade et al., 2019). Additionally, the DoD, unlike commercial industry, does not make strategic acquisition based decisions pertaining to the pursuit of monetary profits. Therefore, applying a CoD framework to the DoD acquisition environment must account for the inherently unique setting in which the Air Force calculates and ultimately assesses trade-off decisions. The purpose of this research is to understand how to appropriately apply a CoD framework inside a public domain environment.

Public Vs Private Goods

A public sector entity such as the DoD cannot utilize a CoD framework that operates under a profit maximization structure. National defense is a traditional example of a public good. Public goods can be consumed regardless of the number of consumers and whose consumption cannot be reduced due to others usage. In economic terms, these conditions are referred to as non-excludable and non-rivalrous. Robert Higgs articulates, "public goods, if created for anyone, are created for all" (Higgs & Niskanen, 1990). Following this logic, an individual cannot reduce the amount of national defense being supplied, nor prevent another from enjoying its benefits (Hartley, 2012). This naturally causes the free rider problem, which stems from the non-excludability trait of public goods (Miceli, 2011). Free riders benefit from a public good while not having contributed anything to the cost of its creation. The lack of incentive for free riders to pay for a public good results in a market failure characterized by an under provision of the good through private means. As a remedy to this market failure, a common solution is for governments to intervene by taking over creation and management of public goods (Gwartney,

Stroup, Sobel, & Macpherson, 2017). Therefore, public goods rely on the political process and government planners to effectively distribute their distribution and funding (Hartley, 2012).

The private sector businesses rely on market forces to guide the allocation of resources to their most productive purpose (Mises, 1958). Ultimately, private industry's actions and motivations stem from the need to generate profits. The common understanding of economic profit or loss is the difference between total revenue earned by producing and selling a good or service compared to the opportunity cost of producing and selling that good or service (Carden, 2007). From a different perspective, Mises (1979) defines profit not as personal gain or a level of happiness from a successful action but rather the satisfaction a customer experiences (Mises, 1979). The government attempts to satisfy its citizens or customers by supplying the nation with the desired amount of national defense. While it may be an inherent function of government to provide protection of its resources and tax base in the form of territory and citizens respectively, there is no implicit consideration towards the increased generation of profit as a result of military funding. However, this brings up the question; how do governments determine what is the amount of national defense required to placate its citizens?

Measuring National Defense: Readiness

The motivation for conducting a CoD Analysis is to more effectively fulfill the government's duty to provide military readiness. Military readiness assesses the preparedness of a nation to face potential threats. Readiness consists broadly of whether a military contains an adequate amount of properly trained individuals and appropriate supply of equipment in working order (Forrester, O'Hanlon, & Zenko, 2001). The amount of trained individuals and working equipment stands as an arbitrary number until it is compared to the number of soldiers and war machine employed by the enemy. Economically, this means that the marginal productivity of

defense spending is determined by the advantage it provides over a rival nation. Lazear and Rosen (1981) investigated the effects of pay on workers' performance to understand competitive environments (Lazear & Rosen, 1981). They investigated an office environment in which the most successful individuals received the highest salary in an organization. Under this environment, a worker's level of financial success is directly tied to the performance of his or her coworkers. Nations face a similar decision in regards to how much defense spending is required. The more governments invest in equipment and training, the more they can separate themselves from the competition and ultimately increase their chance to achieve victory. However, not all national defense goals are the same. Humanitarian operations are dissimilar to conflicts involving technologically similar adversaries which are dissimilar from missions against potential insurgents (Snyder, Lim, Carrillo, & Hildebrandt, 2012). The result is that there are various metrics to take into account to determine the relative readiness of a nation's military.

While the metrics for assessing Air Force readiness have generally stayed consistent, the means in which to achieve readiness has changed as the fleet continues to advance technologically. The importance of robust software has become essential to maintaining military readiness. The former Chief of Staff of the Air Force, General Goldfein (2019), identified five main metrics that he assesses when evaluating the current state of the United States Air Force: training, flying-hour funding, mission preparations, robust sustainment, and time in air (Everstine, 2019). At their core, these metrics all relate to *availability* and *utilization*. *Availability* refers to the amount of time that a resource is prepared for use while *utilization* is the amount of time that said resource is employed (Defense Technical Information Center, 1980). Traditionally, these metrics are dictated by the quality of the personnel and hardware (Forrester, O'Hanlon, &

Zenko, 2001). However, today, the ability to develop and implement effective software has become the prevailing way in which to successfully achieve military readiness.

The importance of software to military readiness can be seen in the development of the Lockheed Martin F-35. In March 2014, the Government Accountability Office (GAO) conducted a report that found that delays in developmental flight testing of the F-35's critical software would hinder the delivery of the warfighting capabilities and thus impact *availability* (Government Accountability Office, 2014). They determined that challenges to the mission system software continued through development and testing due to initial software delivery delays, limited capability in the software upon delivery, and the need to fix new problems and retest multiple software versions. As a result, the Director of Operational Test and Evaluation estimated a schedule slip of 13 months, while cost estimators forecasted an increase in funds to maintain annual costs at $12.6 billion until 2037 (Government Accountability Office, 2014). A change in the production phase affects *utilization* due to impacts to the operational field. Every time there is a concern with aircraft in production, any fighters in operational status must be grounded, updated, and verified to ensure they maintain concurrency (Maldonado, 2015). The issue expands beyond just the F-35, with the ever increasingly reliance on software to execute missions, integrate and collaborate with allies, and manage the defense enterprise (McQuade et al., 2019).

National governments are experiencing massive demand for more robust and efficient defense based software. The National Research Council (2010) stated that DoD software code in service has been increasing by more than an order of magnitude in every decade equivalent to approximately 25 percent annual growth (National Research Council, 2010). Similarly, the National Aeronautics and Space Administration unmanned space systems have increased their

SLOC by an order of magnitude every year while manned systems SLOC are growing even faster (Dvorak, 2009). The DoD is experiencing increasing demand for software across all branches. Tate (2017) estimates an annual growth rate of 15-25% in the demand for developing and maintaining all defense software (Tate, 2017). An annual growth of 25% indicates the amount of software doubling over three to five years (Tate, 2017). Anticipating this assumption, Figure 2 provides a projection of the unconstrained demand for defense related software. China has additionally recognized the need for increasing its software development capabilities and has begun funding the creation of a $1 trillion dollar artificial intelligence industry (McQuade et al., 2019). Given the expanding demand for defense based software, the DoD's ability to develop, procure, maintain, and continuously improve is essential to providing military readiness.

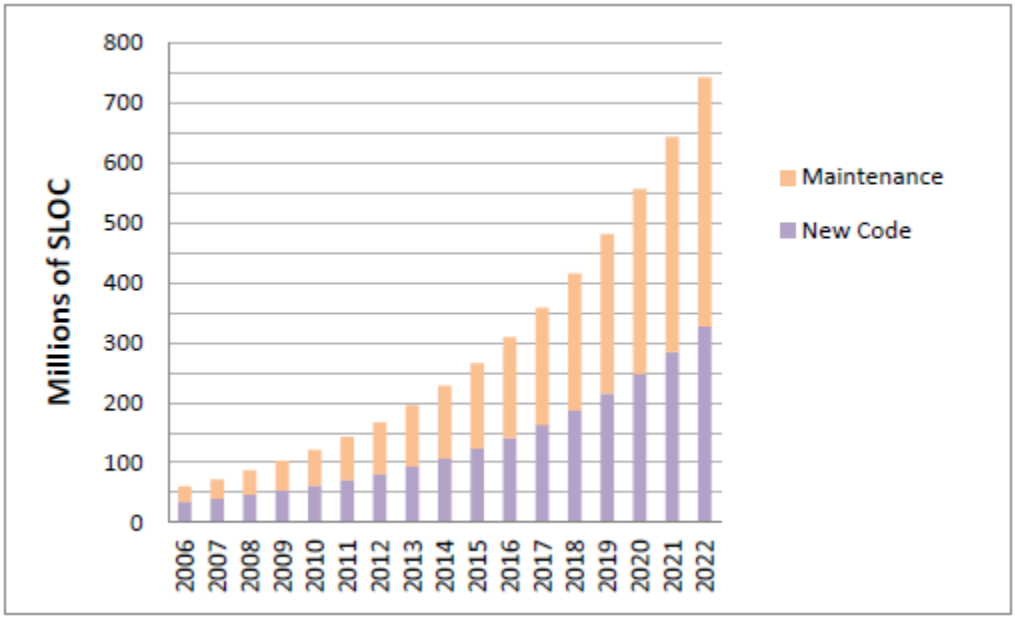

Figure 2: Forecast of DoD Software Demand (Tate, 2017)

Due to the increasing dominance of software in military systems, the DoD's ability to adapt and respond to threats is now determined by its capacity to rapidly develop and deploy effective software (McQuade et al., 2019). Therefore, speed, cycle time, and value have become

the most important metrics to effectively manage software and subsequently impact national defense readiness. Specifically, program offices can affect speed and cycle time by working closely with operators to deliver capabilities based on the most urgent requirements and by accounting for any new requirements as they arise (Cohen, 2019). Additionally, the DoD should measure the ultimate value delivered to the customer rather than simply monitoring the compliance with requirements (McQuade et al., 2019). To accurately address this need for rapid deployment of valuable software capabilities, new acquisition processes are being adopted by the Air Force to achieve the readiness levels necessary to maintain their strategic superiority.

Agile Software Acquisitions

The commercial sector has experienced changes in software development towards Agile methods over the last 15 years, while defense systems have continued to utilize techniques developed in the 1970s-1990s (Defense Science Board, 2018). Traditionally, the DoD has utilized the same type of acquisitions strategies to procure software as they would hardware (Modigliani & Chang, 2014). Government program offices have historically utilized the Waterfall development method which breaks down the creation process into structured segmented steps. Waterfall requires the creation of all function specifications prior to development. After development of stated specifications, the software is released for testing. Upon passing all specified tests, the software is released to the user. There have been advancements of the Waterfall method such as the creation and implementation of Spiral acquisitions. The Spiral method is a risk driven model which uses a cyclical approach to development and the use of anchor point milestones to ensure stakeholder commitment (Boehm, 2000). However, those methods are fundamentally different from Agile which works in iterations, often called sprints, to simultaneously combine requirement building, development,

and test to produce smaller workable increments for delivery to the customer (Pinto et al., 2016). A visualization of the comparison can be seen in Figure 3.

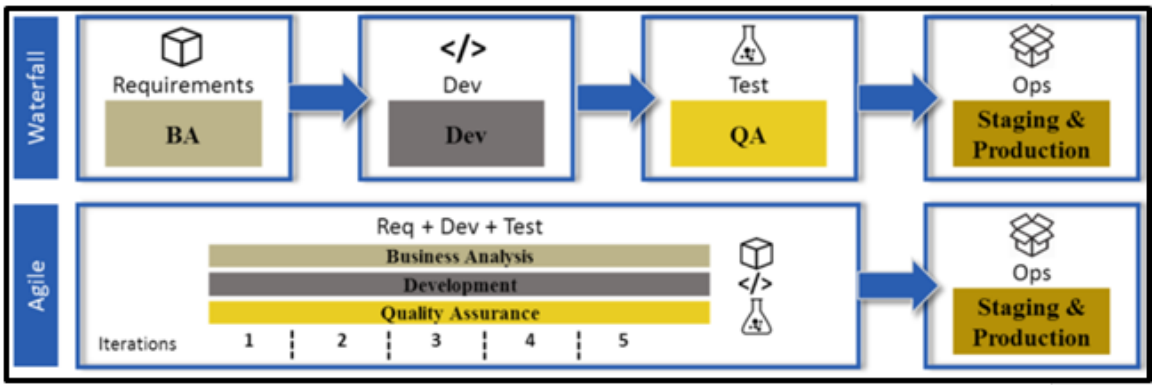

Figure 3: Waterfall & Agile Development (Defense Science Board, 2018)

Agile development has become the leading methodology that private industries use to create software. A 2020 survey conducted by Agile training service, Digitial.ai, reported that 95% of 1129 different company responses indicated that their organizations currently practice Agile development methods (Digital.AI, 2020). Survey respondents also reported the spread of Agile practices towards more areas of their respective organizations such as in the marketing and sales departments. The spread of Agile to additional fields shows its prevalence and importance continues to grow in commercial industry. A separate survey conducted by Freeform Dynamics & CA Technologies (2018) reported that among the almost 1279 Information Technology (IT) business related companies polled, 75% stated that Agile played a vital role in delivering the correct product, accelerating decision making and speed to market, and improving overall customer satisfaction (Freeform Dynamics & CA Technologies, 2018). Amongst the more notable users of Agile are companies such as Intel, Philips, Shell, EMC^2, and Lockheed Martin (Digital.AI, 2020).

Commercial successes suggest the DoD may similarly benefit by employing Agile. Doing so will enable the DoD to move from a capabilities-based acquisition model to a more responsive

Agile threat-based acquisition model which can rapidly adapt to adversarial advancement (Defense Science Board, 2018). Making this transition is necessary if the United States is to maintain technological superiority and counter rapidly growing adversary capabilities.

The Agile Advantage

The Agile software development method has advantages over traditional methods along three dimensions: 1) ability to rapidly adjust to the immediate needs of the warfighter 2) delivers viable products sooner and 3) provides more cost effective programs.

The first advantage of Agile is that it provides an environment for the warfighter to communicate constructive feedback to development team. A distinct advantage of Agile stems from the shorter cycle times to produce useable iterations on a product for the customer. Agile teams produce a Minimum Viable Product (MVP) which has enough features of the end product to meet the basic minimum functionality required by the client (McQuade et al., 2019). The use of an MVP allows Agile teams to get immediate feedback from the end user which developers can utilize to decide the best course of action for future development. Agile development differs from the traditional Waterfall approach since constant feedback loops decrease the risk of implementing the wrong functionality for a product (Perkins & Long, James, 2020). The technique ultimately focuses on creating immediate customer value. The GAO recently reported that the four major software intensive space programs using Waterfall practices are estimated to cost billions of additional funding, resulting in overruns up to three times the original estimate, and have been in development for periods between five to twenty years (Government Accountability Office, 2019). GAO reports that a major issue is that key programs failed to effectively engage the user and provide the necessary feedback required to develop effective software (Government Accountability Office, 2019).

The second advantage of Agile methods is reduced cycle time. Agile methodologies have already been adopted and successfully proven to increase the delivery time of products in several federal government organizations including the Integrated Strategic Planning and Analysis Network (ISPAN), Department of Veteran's Affairs (VA), and National Aeronautics and Space Administration (NASA). For example, the ISPAN program shortened the acquisition cycle duration between initiation and Initial Operational Capability by 45 months (Pinto et al., 2016). Similarly, the VA currently delivers capabilities an average of 4.2 months compared to 3-7 years prior to implementing Agile practices (Pinto et al., 2016). The 14th Annual State of Agile Report showed that the number one reason why commercial companies adopted Agile practices was because it helped them "accelerate software delivery" (Digital.AI, 2020).

The third advantage of Agile methods is that they have the potential to make programs more cost efficient. In the commercial sector, Digital.AI (2020) reports that 26% of companies adopt Agile due to its increased cost savings (Digital.AI, 2020). Similarly, Freeform Dynamics & CA Technologies (2018) found that 29% of IT related companies experienced a reduction to overall costs through the incorporation of Agile methodologies (Freeform Dynamics & CA Technologies, 2018). The Air Force's first dedicated Agile Software Factory, Kessel Run, has already produced positive financial results. Kessel Run developed a tanker planning tool for the Qatar AOC utilizing state of the art software to construct planning routes which immediately saved a reported $214,000 per day in logistics and fuel (Cohen, 2019). The tool was released after only six months of development compared to the previous five year contract which over that time failed to produce a viable product (Cohen, 2019). The flexibility, cost, and schedule benefits from Agile techniques provide one way for the United States to maintain its strategic superiority in the software domain.

Agile Software Factories

While Agile practices can be beneficial to the Air Force, they must be carefully integrated into the proper environment to ensure the best results. Lindvall (2004) documented the difficulties that larger commercial organizations such as Motorolla, Chrysler, and Nokia have experienced in successfully integrating Agile practices (Lindvall, 2004). Lindvall (2004) found that assimilating new practices with existing processes and quality systems regarding the conduct of software development requires tailoring (Lindvall, 2004). The subtext for defense is that existing program offices with established contracts may have difficulty adjusting towards new systems.

Commercial industry has experienced the same issues with companies reporting difficulty in adapting to the new Agile work style. In Digital.AI's report (2020) on commercial based software companies, respondents stated that organizational culture is ranked the number one challenge towards adopting an Agile environment (Digital.AI, 2020). To account for the issues with culture change, the Defense Innovation Board (2018) released best practices for Agile defense software development and recommends that all programs start small, iterative, and build on success (Defense Innovation Board, 2018). McQuade et al., (2019) recommends the government create software development units in each branch of the military that consist of a mix of military and civilian personnel (McQuade et al., 2019). They claim that the ideal state is for each service to retain organic capabilities to develop software. Due to the competition for skilled developers, the DoD can benefit from having established its own force of capable software developers ready to work time sensitive issues without the need for contracting outside sources.

Software Factories: Kessel Run

The Defense Science Board (2018) asserts that the Under Secretary of Defense for Research and Engineering should immediately task each service to establish the creation of internal government Software Factories (Defense Science Board, 2018). Kessel Run is one of those leading Air Force Software Factory located in Boston, Massachusetts. It is comprised of three different product lines: Operational Command and Control Users (Ops C2), Wing Command and Control Users (Wing C2), and Development Teams and Application Users (ADCP). Ops C2 builds the dynamic tasking order, Wing C2 executes the dynamic tasking order, and ADCP provides features and services. There are multiple products under each these product lines. The focus of this research is the Ops C2 product line, looking specifically at KRADOS & 10.1 products. KRADOS consists of 10 user-facing applications, application programming interfaces, and data services. See Table 1 for complete listing of all applications in the KRADOS portfolio.

Table 1: KRADOS Application Teams

Number	**Overall Mission**	**Application Name**
1	Tanker Planning Tool	Jigsaw
2	Standardization of Data	Maitai
3	Mission Planning & Execution	Slapshot
4	Airspace Management Tool	Spacer
5	Air Tasking Orders Tool	Triton
6	Synchronization of Application Data	Skyhook
7	Air Support Request Tool	Gambit
8	Mission Monitoring Tool	Direct
9	Fuel Calculation Tool	Coaxium
10	Prioritized Target List Integration	Jump

Kessel Run utilizes Agile software development in partnership with the Defense Innovation Unit Experimental and Pivotal Labs to efficiently modernize the Air Force's

Targeting community and the Air & Space Operations Center. Through this process, each KRADOS application contains a mix of product managers, designers/developers, and software engineers. Product managers maintain the stakeholder's vision and work towards the users' needs by identifying technical challenges. Designers/Developers interact with the user to determine what solutions will best fulfill their needs. Software engineers create software and ensure that the application is operational and secure. The average team consists of 8-10 members with the typical product team consisting of one product manager, one designer, three software engineers, and three software developers. The KRADOS product of Kessel Run works with AFLCMC/Detachment 12 to build and deliver capabilities in support of the Air tasking order cycle.

While Kessel Run marks the first DoD Agile Software Factory, Agile practices have been structured and implemented since the 2001 development of the Agile Manifesto (Hohl, et al., 2018). Agile frameworks have become a highly prevalent topic in the rapidly advancing world of software leading to the DoD Enterprise DevSecOps Initiative spearheaded by Mr. Nicholas Chaillan, the first Air Force Chief Software Officer (U.S. Air Force Biographies, 2019). DevSecOps is a movement focused on creating new solutions for complex software development processes within an Agile framework (Zaydi & Nasserddine, 2020). Mr. Chaillan presented a status of the effort to attendees at the Washington, DC AFCEA Chapter's Technology Summit Series in June of 2019. He reports that the DoD Enterprise DevSecOps Initiative has 29 organizations working with it, with the goal to expand to 172 in the next six months (Chaillan, 2019). With the increased transition towards Agile acquisitions in the DoD, there is also an increased need for credible and accurate Agile cost and schedule (Rosa et al., 2020). Agile's

unique acquisition approach requires new techniques to accurately calculate a programs' estimated costs leading to a variety of different strategies currently applied in the DoD.

Cost Estimating in the DoD vs. Agile

Cost estimating combines science and art to predict the future cost of something based on known historical data that are adjusted to reflect new materials, technology, software languages, and development teams (GAO, 2009). The DoD's Software Development Estimating Handbook describes SLOC as one of the most prevalent ways to obtain the scope for a software program (NCCA & AFCAA, 2008). SLOC is a metric used to calculate the size of a software program by counting the number of lines in a given program's source code (Bhatt et al., 2012). SLOC provides an easy to understand counting metric that produces an intuitive measurement for understanding the magnitude of a program. However, SLOC inherently has its disadvantages. The problem with SLOC is that it measures an approximation of the quantity of code written without taking into account the quality of the code. Varying levels of efficiency and experience between developers causes a disparity in the amount of SLOC and time required to develop similar functionality (Bhatt et al., 2012). Agile Factories express that they typically do not contract developers to work on a specified amount of code. Rather, they hire individuals based on their ability to develop efficient effective code. Software is never "done" and therefore must be consistently managed as an enduring capability (McQuade et al., 2019). Without a stated end-point, focusing solely on the quantity of software by using SLOC becomes a poor metric with which to estimate a program's cost.

By contrast, most Agile projects use cost estimating strategies based on the relative size and effort of a program (McQuade et al., 2019). From an overall structure standpoint, the cost estimation format does not differ greatly for Agile compared to traditional development

programs. Cost estimates still must consider development, management, acquisition, and maintenance costs associated with a program (Pinto et al., 2016). Nonetheless, Agile projects, in contrast to common Waterfall programs, have a variable scope that fluctuates throughout the program. The distinguishing factor in Agile programs is the manner in which the scope for any given project is calculated. As a result, there are a variety of different methods and strategies to estimate an Agile project's scope. One type of approach relies on using alternative objective size metrics. Function Point Analysis (FPA) provides a defined reproducible unit of measure that remains consistent regardless of the specific user or requirement. FPA is commonly utilized at the project beginning or release (David Consulting Group, 2020). FPA quantifies functionality requested and provided to the customer based on logical design (International Function Point Users Group, 2020). Some of the more subjective methods include advising group effort estimation, subjective expert judgment, estimation by analogy, and planning poker (Schweighofer, Kline, Pavlic, & Hericko, 2016). The methods can be used to develop Story Points which are a relative unit of measurement to describe the complexity, effort, and risk required to develop a given feature (Modigliani & Chang, 2014). A development team's estimated effort and productivity can be used to determine program cost required for the completion of Story Points in a Sprint (Pinto et al., 2016). For example, a budget can be established based on the number of sprints, the number of workers, and the hourly rate required to develop a certain number of Story Points. Following this structure creates the basic equation:

$$Cost\ Estimate = \#\ of\ Sprints * \#\ of\ Workers * Hrly\ Rate\ of\ Workers$$

To calculate the number of Sprints required, Agile teams must determine the amount of work and the rate at which that work can be accomplished. The estimated velocity is a rate measurement that describes the amount of Story Points a team can accomplish during a sprint

(Cohn, 2005). Programs initially measure team velocity using historical values or by making a forecast based on the skill sets of the team and the team's experience with the specific product or technology (Pinto et al., 2016). However, the more a team works the better the fidelity in estimates become over time. Refinement will come as the development team becomes more adept as the Sprints progress (Pinto et al., 2016). For this reason, it is important that the program review the actual costs after completion of the project so that they can utilize the results in validating and revising the estimating factors for future use (Modigliani & Chang, 2014). By adopting effort estimation techniques designed for the unique Agile environment, programs can more efficiently and accurately forecast costs.

Chapter Summary

Chapter 2 provides the analytic foundation undergirding the topics discussed in this research. First, the chapter introduces the concept of CoD as a means to improve organizational efficiency. Then, there is a discussion on the need to adapt the CoD framework due to the public good nature of national defense. The analysis highlights the connection between developing effective software with the ability to maintain proper military readiness. The chapter also describes the foundation, advantages, and application of Agile in the DoD. Lastly, there is an analysis of Agile cost estimation practices utilized within industry and the DoD.

III. Methodology

Chapter Overview

Chapter 3 is divided into two parts. Part 1 establishes the traditional CoD model as well as the calculations needed for CoD Analysis. Part 2 describes the method for collecting data on the current Agile cost estimating strategies. It includes two data sets. Data Set #1 is a literature review of the relevant publications on software effort estimation over the last 20 years. Data Set #2 is a collection of data from Air Force Software Factories regarding their preferred software cost estimating strategies.

Part 1: Cost of Delay (CoD)

Traditional CoD Framework

The purpose of this section is to define the traditional CoD methodology Arnold and Yuce (2013) apply in their analysis of Maersk Shipping Line (Arnold & Yuce, 2013). The unique structure of national defense organizations requires updates to the traditional CoD model. Subsequently, Chapter 4 builds off the extant model and defines a DoD CoD framework for Kessel Run and other government organizations.

The CoD framework consists of three components: *benefit type, urgency profile,* and *development duration* (Arnold & Yuce, 2013). The *benefit type* and *urgency profile* are used in conjunction to calculate the opportunity cost. A CoD score is simply the dollarized opportunity cost per unit of time divided by the development duration (Figure 4).

Figure 4: CoD Score Equation

The opportunity cost is expressed as the dollar value that could be generated or saved per unit of time (days, weeks, months etc). The opportunity cost is calculated after identifying the *benefit type* and *urgency profile*. Each requirement or feature's *benefit type* can be defined in terms of whether it increases revenue, protects revenue, reduces cost, or avoids cost. Features that increase revenue grow sales from existing customers or new customers (Arnold & Yuce, 2013). Features that protect revenue maintain profits currently being received from existing paying customers (Arnold & Yuce, 2013). Features that reduce cost improve efficiency in current operations (Arnold & Yuce, 2013). Features that avoid costs evade potential costs that may be incurred unless an action is taken (Arnold & Yuce, 2013). Each requirement or feature's *urgency profile* can be defined as Short Life-Cycle Peak Affected by Delay, Long Life-Cycle Peak Unaffected by Delay, Long Life-Cycle Peak Affected by Delay, or Impact of External Deadline. In Short Life-Cycle Peak Affected by Delay, benefits ramp to a peak and quickly decline again, as the value-add becomes standard for customers or if the market itself moves on to something different (Arnold & Yuce, 2013). In Long Life-Cycle Peak Unaffected by Delay, benefits ramp up to a peak, and are sustained over a long period (Arnold & Yuce, 2013). In Long Life-Cycle Peak Affected by Delay, the potential realized benefits are lessened over time due to a first mover advantage (Arnold & Yuce, 2013). In Impact of External Deadline, the CoD only ramps up as the 'last responsible moment' approaches (Arnold & Yuce, 2013). The denominator simply consists of the time (days, weeks, months, etc.) measured from initial development to the actual deployment of a feature.

Additionally, the CoD score establishes an initial means to prioritize features; however, leadership is capable of adjusting scores up or down based on business strategy, operational goals, or I.T. strategy. Therefore, after the raw scores are prioritized, government leadership can manually manipulate the order based on the need to deliver specific operational capabilities or other subjective goals which may not be captured by the total CoD. The assumptions and reasons for adjusting scores should be well documented and made visible to everyone to understand the reasoning behind any changes to the prioritization. Additionally, Arnold and Yuce (2013) recognized the benefit of knowledge gained from failure and attempted to include the value of information discovery in any given requirement or feature's opportunity cost calculations (Arnold & Yuce, 2013).

CoD Calculation

The following outlines an approach to prioritize features. Using the Figure 4 equation calculates the Table 2 CoD scores. The team would prioritize the features with the highest CoD scores and therefore work in the following order: B, C, & A.

Table 2: Theoretical CoD Formulation

Feature	Opportunity Cost	Development Duration (weeks)	CoD Score
A	$1/week	5	0.2
B	$4/week	1	4
C	$5/week	2	2.5

The CoD incurred while developing Feature B is calculated by:

$$= (CoDA + CoDB + CoDC) * DurationB$$

$$\$10 = \left(\frac{\$1}{week} + \frac{\$4}{week} + \frac{\$5}{week}\right) * 1week$$

Following this formula, the CoD incurred while working on Feature B is $10. When working on Feature C, since Feature B has already been accomplished, the calculation only takes into account the CoD A and CoD C. Feature C CoD is the addition of CoD of A and CoD of C multiplied by the duration of C to produce the CoD of $12 ({$1/week + $5/week}*2weeks). And lastly, when working on Feature A, the CoD of A is multiplied by the duration of A for the CoD of $5 ({$1/week} * 5weeks). Adding these three CoD values provides a total CoD of $27 which is the lowest cost solution to this particular data set. Alternatively, had the team prioritized the features in a FIFO manner, the total CoD would be $69. Figure 5 shows a visual representation of the CoD prioritization method on the left compared against the FIFO method on the right.

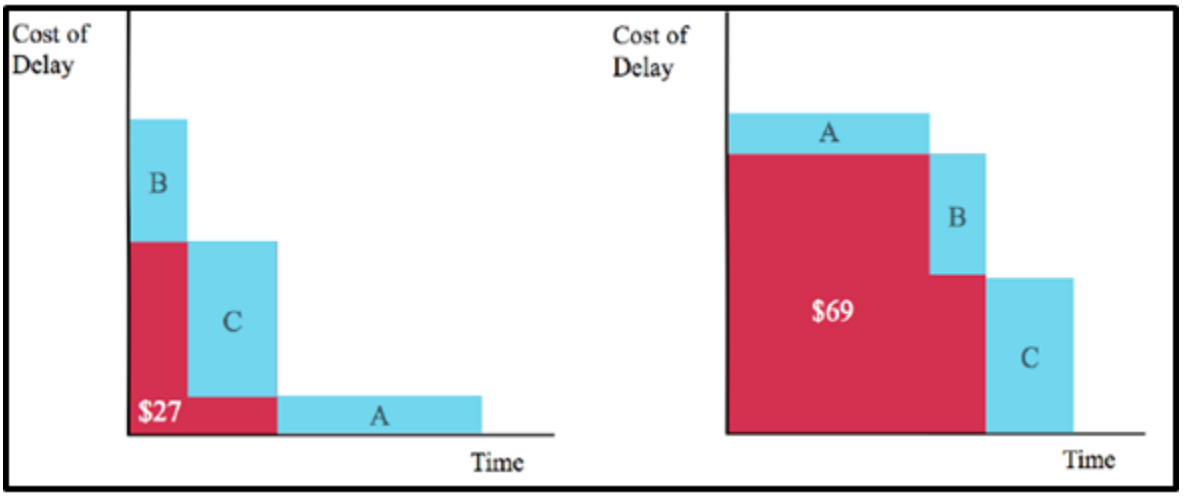

Figure 5: CoD vs FIFO Method

Justification & Prioritization Tool

There is a formulaic distinction between the uses of CoD as a *justification* vs *prioritization* tool. Only the opportunity cost is required to utilize CoD Analysis as means for

justifying decisions. The opportunity costs can help leaders decide why certain programs require additional manpower or funding. As part of an economic analysis, the opportunity costs provide dollarized figures which capture the true loss of value due to a delay. Accounting for the opportunity costs provides leaders with additional information beyond simply the contract costs. By contrast, as seen in the explained example, CoD as a *prioritization* tool requires the development time for its calculations. The development duration is used to for calculation and comparison of CoD scores.

Application of CoD Tools

The adapted DoD CoD model developed in Chapter 4 is applied to Kessel Run to demonstrate the *justification* and *prioritization* capabilities. First, the CoD *justification* tool demonstrates the theoretical benefits with deprecating the legacy TBMCS contract in favor of modern KRADOS applications. Second, the CoD *prioritization* tool demonstrates a proof of concept by organizing Jigsaw and Chainsaw application team features. Appendix A provides greater detail on the TBMCS contract, KRADOS portfolio, Chainsaw application team, and Jigsaw application team.

Part 2: Agile Software Cost Estimating Methods

The purpose of Part 2 is to establish the method ology for analyzing the current state of Agile cost estimation in the Air Force. The analysis will be accomplished by comparing the techniques used in 83 collected sources to the practices of 11 Air Force Software Factories.

Data Set #1

The search strategy consists of four parts. The first phase involves searching for all articles generated from search strings in the databases listed below. The second phase eliminates

all duplicate files and articles that are not published in the English language. In phase three, the articles are analyzed to deduce whether they meet the inclusion and exclusion criteria set for the study. During this phase, the articles' title, abstract, conclusion, and keywords are read to determine if they meet the standards for the research. After this primary reading, the fourth phase consists of a full read through of the article to ensure an article meets the required acceptance criteria. The relevant papers that pass all requirements are presented in Appendix B.

Major Databases:

- IEEE Xplore
- Science Direct

Search Strings:

The main string is supplemented with the use of additional keywords to better refine the search. The main string consists of:

"Software Effort Estimation <AND> "Cost"

Additional keywords:

"Agile" <OR> "Expert Judgment" <OR> "Algorithm" <OR> "Machine Learning" <OR> "Technique" <OR> "Estimate" <AND> language "English"

Inclusion Criteria:

- Provide analysis or recommendation of the techniques, models, & approaches used in Agile software cost estimation
- Published DoD manual/report
- Published in the last 20 years
- Published in peer reviewed journal articles or conference proceedings

Exclusion Criteria:

- Not related to Agile based environments
- Simply defines or explains the types of software estimation techniques

Filtering Data:

Figure 6 outlines the phased approach and the number of articles remaining after the application of inclusion/exclusion factors.

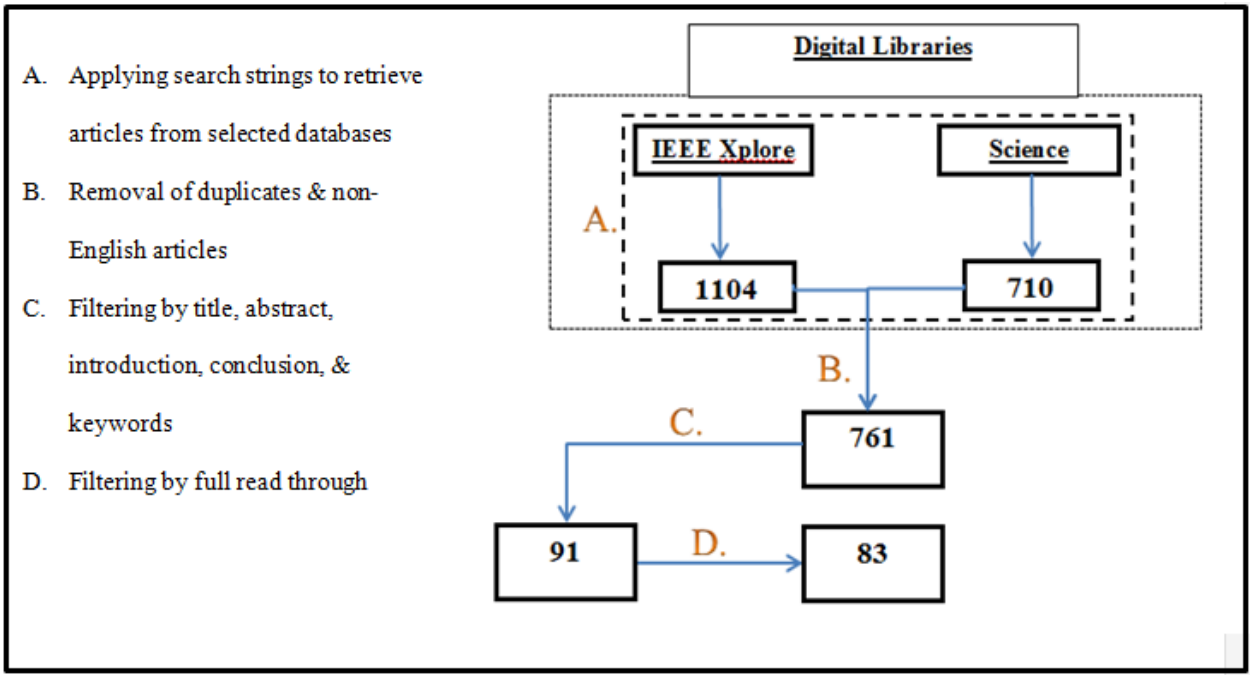

Figure 6: Article Search and Filter Process

Ultimately, from the original 1814 search hits, 83 articles are selected for the analysis of this study. The articles chosen support, advocate, or defend the usage of specific practices when conducting cost estimation in a software environment. 76 of the articles are from an industry perspective while 7 relate to the manner in which the DoD advises or conducts its cost estimation. The 83 articles included in Data Set #1 provide insight into the currently recommended Agile cost estimation best practices. Data Set #1 offers a reference point for the Air Force specific data collected in Data Set #2.

Data Set #2

Data Set #2 consists of a data call from Agile Air Force Software Factories. Data Set #2 is a baseline for how the Air Force and DoD have adapted cost estimation in an Agile environment. As of January 2021, there are 16 identified Air Force Software Factories. 11 of these organizations provided information regarding their software cost estimation process. Organizations provided their preferred sizing metrics and cost estimation techniques. Additionally, Software Factories provided context regarding their thoughts on cost estimation techniques employed in their organization as well as their overall level of satisfaction with the processes.

The information collected from the Software Factories, Data Set #2, will be used as a direct source of information for the use of software cost estimating techniques in the Air Force. In Chapter 4, Data Set #2 will be compared to the sources compiled in Data Set #1. A direct statistical comparison of certain metrics and techniques will accomplished using Clopper Pearson binomial confidence intervals. The comparison of the two data sets will provide insight into how the Air Force is conducting its software effort and cost estimation compared to the current literature.

Chapter Summary

Chapter 3 outlined the methodology and data sets involved in the two parts of the study. Part 1 created the CoD framework and identifies *justification* and *prioritization* aspects. Part 2 outlined the two cost estimating data sets to be analyzed in this study. Chapter 4 provides an analysis for the two parts. First, the CoD framework is explained and applied to a government organization. Second, Data Set #1 is compared to Data Set #2 to understand how the Air Force's Agile cost estimation compares to the prevalent literature.

IV. Results and Analysis

Chapter Overview

Chapter 4 is divided into two parts. Part 1 analyzes the CoD framework established in Chapter 3 when applied to Kessel Run. Part 1 consists of three sections. Section 1 outlines the CoD framework required for opportunity cost data collection. Section 2 demonstrates a theoretical application of the CoD *justification* tool and a proof of concept of the *prioritization* tool. Section 3 reviews the notable takeaways for DoD CoD Analysis. Part 2 analyzes the data gathered on the current state of Agile software cost estimating. Part 2 consists of four sections. Section 1 provides an overview of the sizing metrics and effort estimation techniques. Section 2 displays figures and analysis regarding the literature gathered in Data Set #1. Section 3 displays figures and analysis regarding the data collected from Air Force Software Factories in Data Set #2. Section 4 is a comparison of the two data sets to expound upon how the Air Force has adopted Agile cost estimating techniques compared to the prevailing literature.

Part 1: Cost of Delay Analysis

Section 1: DoD CoD Framework

Kessel Run and other military acquisition offices are not inherently revenue seeking institutions. Therefore, the DoD CoD framework consists of only certain benefit types and urgency profiles. In lieu of accounting for revenue or profit, the model accounts solely for the *reduce cost* and *avoid cost* benefit types. The decision is made after an evaluation of the economic structure of government organizations as well as a discussion with Kessel Run's leadership. The *reduce costs* category covers changes that improve overall efficiency of operations. The *avoid cost* bucket consists of costs that are not currently incurred; however, there is a probability that they will be in the future. Kessel Run's leadership identified *avoid cost* and

reduce cost encompass the types of requirements their organization typically completes. Furthermore, *protect revenue* and *increase revenue* benefit types are outside the scope of government organizations as they revolve around profit generation.

The research identifies the two urgency profiles to be considered in the DoD analysis: *Long Life-Cycle Peak Unaffected by Delay* and I*mpact of External Deadline*. *Short Life-Cycle* features are identified where the benefits are relatively short and dictated quickly by market demand and typically comprised of fast-moving consumer goods (Arnold & Yuce, 2013). The assumption is that DoD's demand for certain capabilities typically will not fluctuate dramatically over short periods of time to warrant the consideration of S*hort Life-Cycle* urgency profile. *Long Life-Cycle, Peak Affected by Delay* identifies features where there is a clear first-mover advantage which penalizes latecomers (Arnold & Yuce, 2013). The profile highlights the benefits and costs associated with falling behind competition. In a macro sense this concept addresses the heart of this research regarding the importance for the DoD to stay ahead of the competition. However, this urgency profile looks at the cost associated amongst rival organizations providing business to the DoD. The intent of the research is to analyze the public good nature of the DoD. Therefore, internal competition does not matter as the assumption is that the government benefits as long as military readiness as a whole is improved.

Long Life-Cycle, Peak Unaffected by Delay is applicable to the DoD. It occurs when there are lifecycle benefits that ramp up to a peak and are sustained over an extended period of time (Arnold & Yuce, 2013). The opportunity cost associated with the feature is the same regardless of whether it is late or not. As a result, the urgency profile is the most common and easiest to compute. Figure 7 shows a visualization of this concept.

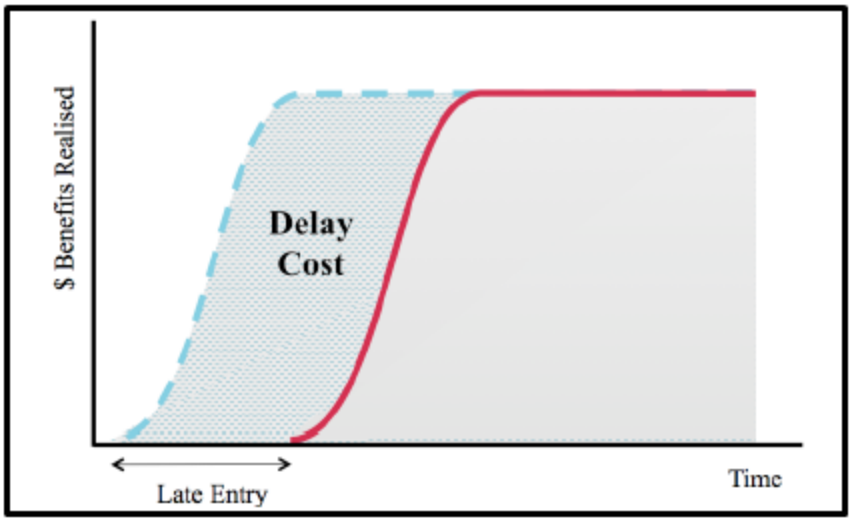

Figure 7: Long Life-Cycle, Peak Unaffected by Delay (Arnold & Yuce, 2013)

I*mpact of External Deadline* is the other potential urgency profile seen in the DoD. In this configuration, there is a specific deadline associated with a feature and the CoD only begins to ramp up as it approaches the "last responsible moment" (Arnold & Yuce, 2013). To effectively compute these profiles, the team has to consider the lead time required to complete a certain feature so as to be certain not to deliver too soon or too late. Effectively, features that fall under this category are tied to a specific delivery date and will have a CoD of zero until the last responsible moment. Figure 8 demonstrates how to represent this urgency profile.

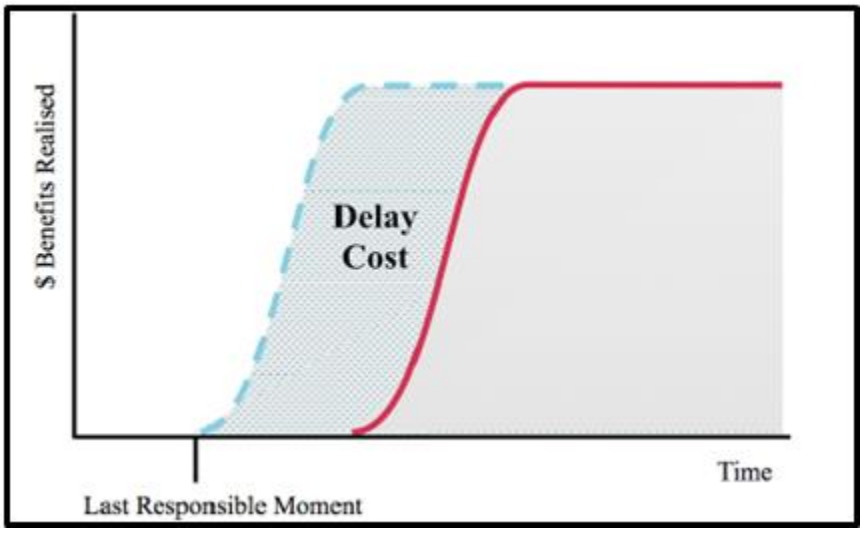

Figure 8: Impact of External Deadline

After limiting the benefit types and urgency profiles, the overall methodology for calculating DoD CoD remains the same as the methodology explained in Chapter 3.

Section 2: CoD Justification and Prioritization

The *justification* tool incorporates the TBMCS contract costs, the 10 KRADOS application teams' costs, and the 10 KRADOS application teams' opportunity costs. However, due to the inability to ascertain the necessary opportunity costs estimates from the 10 KRADOS teams, the *justification* tool will only be discussed theoretically.

The continued operation of the TBMCS contract and KRADOS portfolio applications consist of several cost components. The cost estimate for TBMCS accounts for $16.54M in procurement and O&M costs over FY21. There is an estimated $3.02M in procurement in the form of capital equipment replacement and SPO support. There is an estimated $13.53M in sustainment costs associated with prime contractor costs, field support, materials, and other Government costs. In contrast, the KRADOS application teams utilize a different procedure for estimating costs.

To construct the estimate for the cost of application teams, Kessel Run utilizes a standardized methodology. First, analysts calculate an average rate for a software engineer/developer. Historically, Air Operations Center (AOC) labor rate studies have contracted software workers listed around ~$150/hr for a composite rate. To produce an estimate, that value can be extrapolated to a year and multiplied by 1900 hours (2080 for 52 weeks of work minus time off/holidays) to produce an average value of $285k per worker. For an eight person application, this is approximately $2.28M per team. However, the $2.3M is just the cost to develop the application or the cost of labor necessary, as it does not include the overall platform costs to run the application. For the sake of the exercise and due to limitations in the availability

of data provided, the platform costs are excluded from the consideration of CoD calculations in this example. Given that there are 10 application teams under KRADOS, there is an expected operation and maintenance cost of $22.8M.

There is a total of $39.34M in estimated yearly costs associated with the use of TBMCS and KRADOS application teams. With the full deprecation of TBMCS, this would allow for the total operational costs to reduce to $22.8M. Therefore, the government is effectively paying an additional yearly $16.54M for the TBMCS contract. Alternatively, as a partial CoD calculation, there is approximately a $1.378M cost for every month that KRADOS is not fully stood up. However, this estimate does not incorporate the potential increased operational cost savings received from improved operations achieved with the 10 KRADOS applications. The estimated yearly opportunity cost savings provided by features can be added in addition to the $16.54M. The inclusion of the yearly operational opportunity cost savings experienced by the successful implementation of applications provides a more robust estimate of the true impacts caused by delay. By incorporating application team opportunity costs, the total CoD estimate would reveal far greater cost savings experienced through the implementation of KRADOS over the legacy TBMCS beyond what the contract costs would indicate.

A proof of concept of the *prioritization* tool is demonstrated using data from Kessel Run Jigsaw and Chainsaw application teams. Each application team provided software features from their product backlog. For proprietary reasons, the exact specification and descriptions of the features are removed from the work. However, both teams provided generic details regarding the work to be done as well as the potential cost savings to be gained from successful implementation. The Jigsaw and Chainsaw teams both described two features that could be used as examples for this analysis. The four analyzed features all identify reductions in manpower

hours as a means to determine their cost saving capabilities. The calculations considered are *reduce cost* and take the urgency profile of L*ong Life-Cycle Peak Unaffected by Delay*. No *avoid costs* benefit types nor *Impact of External Deadline* urgency profiles are examined for this assessment. The AFI 65-503 table A33-1 provides the hourly cost rates for active duty military members in FY20 that will be utilized in the CoD calculations. The rates applied represent the cost for one part-time military member without PCS costs, and they exclude the cost of lost productivity due to time spent on leave as well as time spent in activities other than members' primary duties. Table 3 provides a list of all hourly rates according to the specific military rank.

Table 3: Direct Hourly Pay Rate by Military Rank

OFFICER	Direct Hourly Pay Rate w/o PCS	ENLISTED	Direct Hourly Pay Rate w/o PCS
O-10	$174	----------	----------
O-9	$177	E-9	$91
O-8	$174	E-8	$78
O-7	$156	E-7	$70
O-6	$140	E-6	$62
O-5	$119	E-5	$53
O-4	$104	E-4	$45
O-3	$88	E-3	$36
O-2	$74	E-2	$33
O-1	$59	E-1	$29

Table 4 provides the opportunity cost, development time and CoD scores for the four features provided by Jigsaw and Chainsaw.

Table 4: CoD Scores for Jigsaw & Chainsaw

Application Feature	Opportunity Cost	Development Duration (weeks)	CoD Score
Jigsaw Feature 1	$456/week	3	152
Jigsaw Feature 2	$1140/week	1	1140
Chainsaw Feature 1	$483.24/week	3	161.08
Chainsaw Feature 2	$24.814/week	1	24.81

Jigsaw Feature 1 improves the efficiency of a planning system by removing additional manual inputs and saving manpower hours for a team of military members. According to user feedback and subject matter opinion, the integration of the feature would save an estimated 2 hours of work per week. The team working on the exercises consists of one Major, one Lieutenant, and one Senior Airman. Applying the rates from Table 3, the team costs approximately $228/hr (104 + 74 + 50). Therefore, the feature would save $456/week or $23,712/yr in manpower hrs. The software team has provided a 3 week timeframe for the successful development and implementation of the feature. Therefore, the feature has a CoD score of 152 (456 ÷ 3).

Jigsaw Feature 2 is a bug fix that stops a program from crashing in the middle of a calculation saving manpower hours caused by the need for rework. According to user feedback and subject matter experts, the implementation of the fix for the bug would save an estimated 5 hours of work per week. The team that works on the exercises consists of one Major, one Lieutenant, and one Senior Airman. Applying the rates from Table 3, the team costs approximately $228/hr (104 + 74 + 50). The bug fix would save $1140/week or $59,280/yr in manpower hours. The software team has provided a one week estimate of development time required to successfully patch the bug. Therefore, the bug has a CoD score of 1140 (1140 ÷ 1).

Chainsaw Feature 1 is a feature that automates a collateral damage assessment saving a team time and effort. According to user feedback and subject matter experts, the feature would save an estimated 20 minutes of work per week. The team that currently has to work on the process manually consists of 10 officers and 10 enlisted. The composition of the specific Officers and Enlisted members varies from week to week. Therefore, applying the rates from Table 3 for the CoD estimate, the average hourly cost is $95/hr for Officers and $50/hr for Enlisted. The feature would save $483.24/week or $25,128.50/yr in manpower hrs. The software team provided a 3 week estimate to complete development of this feature. Therefore, the feature has a CoD score of 161.08 (483.24 ÷ 3).

Chainsaw Feature 2 is a bug associated with the execution of the collateral damage assessments. The bug causes a delay in the processing time of the tool, leading to 10 minutes of time lost per week. Currently, one Major and one Senior Airman work on providing this damage assessment. Applying the rates from Table 3, the team costs the Air Force $149/hr (104 + 45). The implementation of the feature would therefore save $24.81/week or $1,290.34/yr. The software development team estimates that it would take them 1 week to successfully develop this feature. Therefore, the feature has a CoD score of 24.81 (24.81 ÷ 1).

The data presented can be analyzed to help the decision making process. As described, each application team has its own product backlog of features and bugs to fix. Analyzing the CoD scores would indicate that Jigsaw should work on Feature 2 first while Chainsaw should work on Feature 1 first. Jigsaw Feature 2 presents the greatest opportunity cost to the team by waiting in the queue. To further illustrate the point, under a hypothetical consideration, if Chainsaw and Jigsaw were comprised of the same team members, CoD Analysis would indicate

that the team should work on the features in the following order: Jigsaw Feature 2 (1140), Chainsaw Feature 1 (161.08), Jigsaw Feature 1 (152), and Chainsaw Feature 2 (24.81).

Section 3: CoD Takeaways

The results demonstrate several points and limitations of the research. First, the analysis demonstrates how some features are vastly more significant from an opportunity cost standpoint compared to others. Chainsaw Feature 2 presents the smallest opportunity cost while Jigsaw Feature 1 represents the greatest. The small data sample highlights the disparity that can be seen when considering the importance of a product backlog. Typically a development team would focus on the features that are the most important to the user. The assumption is that the most important features will have the greatest operational opportunity costs. Therefore, CoD Analysis provides a more quantitative and potentially more defendable way to illustrate which features are the most impactful to the user. Second, the CoD assessments show how small inefficiencies can add up to significant cost losses. Once again, the data set only represents a small sample of the potential cost saving. However, with just this initial assessment an increase in manpower efficiency from one feature can save the government thousands of dollars per week. A deeper discussion on the other cost saving capabilities as well as the CoD quantification of the multitude of other features in the backlog could reveal even more significant efficiencies that could be achieved through the successful implementation of features.

One major limitation is that the analysis fails to consider the other potential cost saving capabilities besides manpower hours. The opportunity costs currently only represent the smallest potential cost savings provided by improving manpower efficiency. However, further discussion with application team members would be required to extract the additional opportunity costs such as the reduction in resources or procedures. Additionally, the second limitation is the lack

of features oversimplifies the potential issue. In a two feature game, the decision making is not overly complicated. However, these calculations only highlight four of the many features in the product backlog within each application team. Program managers typically have to prioritize amongst dozens of features and bugs. Applying this analysis upfront provides an immediate and empirical way to demonstrate a potentially cost efficient list of priorities. Third, quantifying the dollar cost savings gives some idea of what features are the most important to providing better overall military readiness; however, it does not provide a full picture. The CoD scoring and analysis must come with additional context. More elaborate requirements require sufficient description to capture the full operational potential. Features that provide significant improvements to warfighting capability or a potential to save lives become challenging to quantify and justify with a simple dollar estimate. For this reason, the DoD CoD assessments are best utilized as a supplemental tool to help prioritize features, but should not be considered a final optimal solution.

Part 2: Agile Software Estimating Methods Analysis

The following chapter is divided into four sections: Section 1 provides a brief description of the types of sizing metrics and effort estimation techniques found in the data. Section 2 displays tables, graphs, and analysis of the cost estimation techniques currently found and advocated for in recent literature. Section 3 displays a report of the common cost estimation techniques in use by the Agile Air Force Software Factories. Section 4 analyzes how the Air Force Agile cost estimation practices compare to the recommended literature.

The most prevalent Agile cost estimation methods in industry today can be divided into three major styles: Algorithmic, Non-Algorithmic, and Data-Based. Algorithmic models use statistical formulation to generate software estimates (Mahmood, Kama, & Azmi, 2019). The

major forms of Algorithmic models include: Use Case Points, Function Points, Story Points, COCOMO-II, Parametric models such as SLIM & SEER-Sim, Case Based Analogy (CBR), and SLOC (Mahmood, Kama, & Azmi, 2019). Use Case Points, Function Points, Story Points, and SLOC can all be utilized as independent variables in Algorithmic models as a means to estimate cost. However, at their core, they are all sizing metrics. Therefore, for the purpose of this study, they will be excluded from the Algorithmic category and included in a separate table tallying sizing metrics. Non-Algorithmic models are typically based on interpretation and comparison to historical data to generate estimates for the future. The major forms of Non-Algorithmic models include: Expert Judgment (Top-down & Bottom-Up buildups), Planning Poker/disaggregation, and Wideband Delphi (Mahmood, Kama, & Azmi, 2019). Data-Based estimates utilize machine learning and artificial intelligence to develop optimization models that develop multifaceted relationships between inputs and outputs. The most common form of Data-Based methods include: Artificial Neural Networks (ANN's), Genetic Algorithms, Fuzzy-Based Models, and Bayesian Networks (Mahmood, Kama, & Azmi, 2019).

Section 1: Description of Sizing Metrics & Estimation Techniques

The following section provides an overview of the various size metrics and cost estimation techniques. As previously noted, for the purpose of this study, Function Points, Use Case Points, Story Points, and SLOC are taken out of the Algorithmic category and defined as sizing metrics. Appendix B provides a more detailed discussion on the definitions of all metrics and techniques described in Section 1. Table 5 highlights the 14 techniques and styles captured in Data Set #1.

Table 5: Technique Styles and Techniques

Technique Style	Techniques
Algorithmic	COCOMO-II
Algorithmic	SLIM
Algorithmic	SEER-SEM
Algorithmic	Parametric Model
Algorithmic	Regression Model
Non-Algorithmic	Expert Judgment (Top-Down, Bottom-Up)
Non-Algorithmic	Planning Poker
Non-Algorithmic	Wideband Delphi
Data-Based	Neural Networks
Data-Based	Regression Using Unsupervised Learning Techniques
Data-Based	Fuzzy Models
Data-Based	Genetic Algorithms
Data-Based	Case Based Analogy
Data-Based	Bayesian Networks

Section 2: Literature Analysis

The following section provides the data and analysis for the literature regarding software effort estimation. The various tables use a number system to reference the articles provided in the *Selected Cost Estimation Techniques Works Cited* in Appendix A. There are 83 sources in the literature review. A variety of the sources incorporate multiple references to techniques in their methodology. A reference indicates that the article advocates for the use of a certain technique, style, or size metric. For example, article 34 is one particular source; however, it references the

use of SLOC, COCOMO-II, and Neural Networks. Section 2 tracks both the number of references and the number of sources for the analysis. All citations in the charts are listed chronologically according to their respective date of publication.

Table 6 and Figure 9 show all of the identified references to techniques utilized in the citations. The results indicate that Neural Networks (44.58%), Regression using Unsupervised Learning Techniques (20.48%), and Expert Judgment (21.69%) are amongst the most prevalent effort estimation strategies referenced in the literature. Additionally, there is only one reference to the use of SLIM or SEER-SEM which comes from the same source (74). Figure 9 shows the raw number of technique references without tracing to the cited literature. The % Use column identifies the percentage of sources that reference a particular Technique Style. Data Based approaches are the most common, appearing in 57.83% of the sources.

Table 6: References to Software Effort Estimation Techniques

Techniques	Statistics of Usage	Cited Literatures
Neural Networks	44.58%	67, 1, 4, 72, 9, 49, 36, 70, 50, 34, 31, 59, 63, 15, 5, 46, 54, 24, 3, 2, 81, 23, 8, 48, 82, 19, 30, 6, 68, 69, 17, 80, 55, 41, 11, 73, 64
Expert Judgment (Top-Down, Bottom-Up)	21.69%	35, 26, 40, 25, 18, 9, 10, 38, 75, 45, 53, 20, 44, 76, 65, 27, 37, 16
Regression Using Unsupervised Learning Techniques	20.48%	67, 43, 36, 32, 51, 31, 5, 24, 62, 48, 82, 19, 6, 68, 69, 55, 64
COCOMO-II	10.84%	4, 34, 29, 52, 66, 20, 30, 27, 73
Regression Model	10.84%	43, 35, 1, 75, 33, 71, 81, 60, 68
Case Based Analogy	9.64%	43, 35, 9, 31, 22, 48, 65, 37
Parametric Model	7.23%	21, 57, 56, 58, 53, 12
Wideband Delphi	7.23%	79, 40, 47, 10, 38, 16
Planning Poker	4.82%	47, 38, 76, 14
Fuzzy Models	4.82%	15, 48, 61, 6
Genetic Algorithms	3.61%	7, 31, 48
Bayesian Networks	3.61%	42, 48, 64
SLIM	1.20%	74
SEER-SEM	1.20%	74

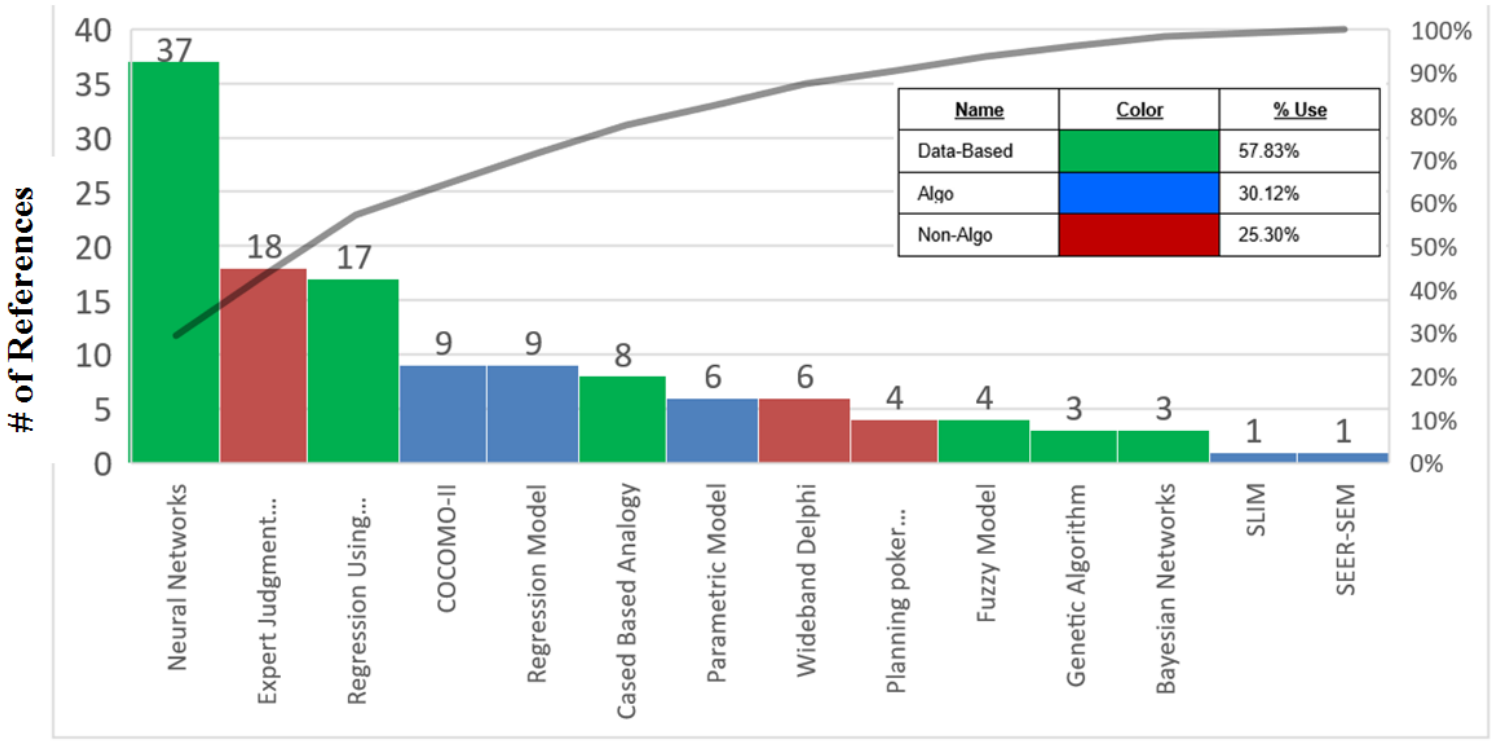

Figure 9: References to Software Effort Estimation Techniques

Table 7 shows the various sizing metrics utilized in the literature. The most obvious conclusion from Table 7 is that more than half of the articles do not directly specify the sizing metric used. Authors may make reference to generic size or effort terminology without directly identifying the specific metric utilized. Use Case Points appear to be the most commonly referenced sizing metric at 15.66%; however, according to Table 7, each sizing metric appears to have a relatively similar number of appearances in the data set as they are all mentioned in the range of 8.43%-15.66%.

Table 7: Software Size Metrics

Sizing Metric	Statistics of Usage	Cited Literatures
Unidentified Metric	55.42%	67, 79, 7, 43, 35, 26, 40, 25, 18, 42, 47, 21, 4, 72, 9, 49, 32, 10, 38, 75, 31, 59, 74, 63, 45, 15, 22, 24, 3, 44, 2, 81, 23, 8, 48, 76, 19, 12, 68, 69, 17, 80, 55, 41, 11, 64
Use Case Pts	15.66%	51, 50, 83, 78, 71, 29, 52, 5, 20, 62, 6, 37, 16
SLOC	13.25%	1, 36, 34, 66, 20, 54, 60, 30, 27, 73, 16
Story Points	12.05%	33, 57, 56, 58, 53, 46, 61, 82, 14, 65
Function Pts	8.43%	28, 70, 13, 52, 77, 16, 39

Table 8 shows the number of references to techniques when sorting estimating techniques to their respective styles: Algorithmic, Non-Algorithmic, and Data-Based. Data-Based techniques are the most prevalent in the literature appearing in 57.83% of the sources. As mentioned previously, a variety of the literature references the benefits of multiple techniques and styles. There are 22 articles accounting for 26.51% of data set that refer to the benefit of multiple technique styles.

Table 8: Effort Estimation Technique References

Technique Styles	Statistics of Usage	Cited Literatures
Data-Based	57.83%	67, 7, 43, 35, 1, 42, 4, 72, 9, 49, 36, 70, 32, 51, 50, 34, 31, 59, 63, 15, 5, 22, 46, 54, 24, 3, 62, 2, 81, 23, 8, 48, 61, 82, 19, 30, 6, 68, 65, 69, 17, 80, 55, 41, 11, 37, 73, 64
Algorithmic	30.12%	43, 35, 1, 21, 4, 34, 75, 74, 33, 57, 56, 58, 71, 29, 52, 66, 53, 20, 81, 60, 30, 12, 68, 27, 73
Multiple Styles	26.51%	43, 35, 1, 28, 4, 9, 83, 34, 75, 78, 13, 53, 77, 20, 81, 30, 68, 65, 27, 37, 73, 39
Non-Algorithmic	25.30%	79, 35, 26, 40, 25, 18, 47, 9, 10, 38, 75, 45, 53, 20, 44, 76, 14, 65, 27, 37, 16

Figures 10 and 11 illustrate the references to technique styles when accounting for the articles that additionally identify the sizing metric utilized respectively. There are discrepancies seen with the increase in the total number of Algorithmic and Non-Algorithmic references when accounting for the sizing metric. The increase in Algorithmic references occurs due to source 52 discussing the use of the model with either Use Case Points or Function Points and because source 20 references the use of the model with either SLOC or Use Case Points. A similar occurrence appears with the Non-Algorithmic styles. Source 20 once again references both SLOC and Use Case Points while source 16 references the use of SLOC, Use Case Points, and Function Points. It is worth noting that source 16 is an Agile Assessment guide for the DoD and specifies generic guidance for the use of all sizing metrics.

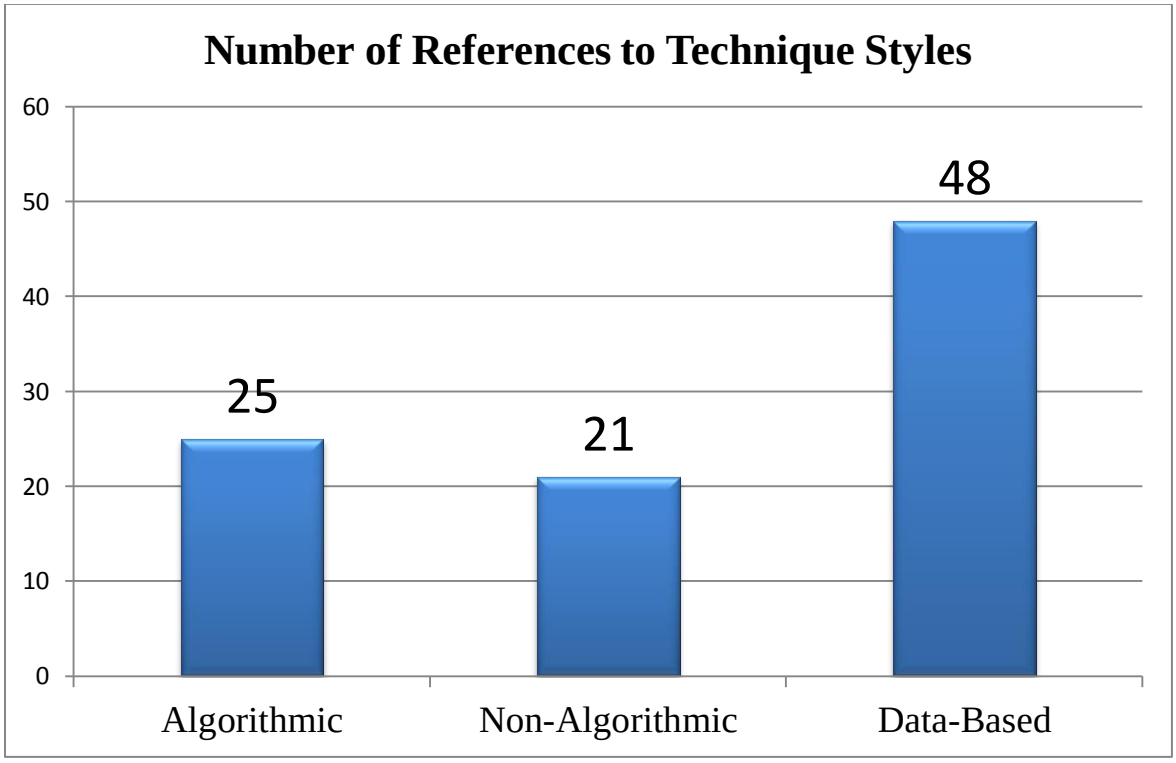

Figure 10: Number of References to Technique Styles

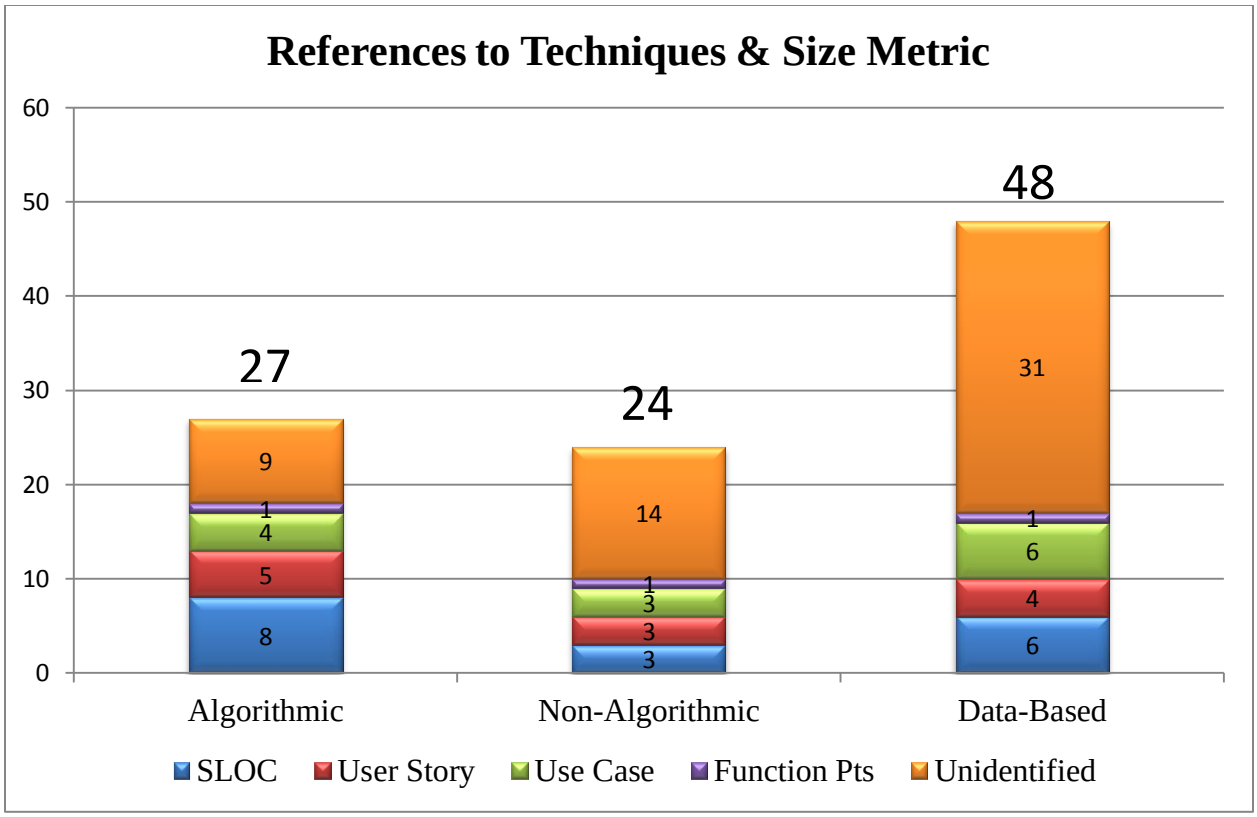

Figure 11: Number of References to Technique Styles Accounting for Size Metrics

There are a total of 37 articles that reference the use of a size metric in calculating their model. Six articles (28, 83, 78, 13, 77, & 39) only reference the use of a size metric to build a model. The six articles utilize either Function Points or Use Case Points to create an overall estimate. As previously discussed in the beginning of Chapter 4, this can be explained as Function Points and Use Case Points can be classified as standalone Algorithmic formulations. 31 articles reference the use of a size metric in combination with another effort estimating technique. In Table 9, the Repeated Articles row highlights the fact that 9 of those 31 articles are repeated more than once across the technique styles. The Repeated Articles row additionally shows that a number of sources make reference to the use of multiple techniques. The results highlight the fact that a few articles advocate for a variety of different sizing metrics and technique combinations.

Table 9: References to Style Techniques & Size Metrics

Technique Styles & Size Metric	Statistics of Usage	Cited Literatures
Size & Data-Based	20.48%	1, 36, 70, 51, 50, 34, 5, 46, 54, 62, 61, 82, 30, 6, 65, 37, 73
Size & Algorithmic	19.28%	1, 34, 33, 57, 56, 58, 71, 29, 52, 66, 53, 20, 60, 30, 27, 73
Repeated Articles	10.84%	65, 37, 73, 30, 34, 1, 53, 20, 27
Size & Non-Algorithmic	8.43%	53, 20, 14, 65, 27, 37, 16

One reason for the repeated reference articles is because a number of sources describe the viability of a hybrid or ensemble model which incorporates multiple techniques into the creation of a new multifaceted one. However, not all articles that mention multiple techniques are advocating for a hybrid model. In Table 10, the 'Indifference Between Techniques' row captures articles which find that different techniques can be equally viable or that certain techniques should only be utilized under specific conditions. 60.24% of the data set only make use of one technique. However, 25 articles, 30.12%, recommend the construction of a hybrid/ensemble model. Additionally, 21 of the 25 articles that mention the use of an ensemble method incorporate a Data-Based approach in that model. Figure 12 shows that when accounting for time, there has been growth in the number of articles that refer to the benefits of ensemble methods over the last decade.

Table 10: References to Hybrid/Ensemble Methods

Model	Statistics of Usage	Cited Literatures
Single Technique	60.24%	79, 7, 26, 25, 18, 42, 21, 28, 72, 49, 70, 32, 50, 83, 59, 74, 78, 33, 63, 57, 56, 58, 13, 45, 71, 29, 52, 66, 22, 53, 46, 77, 54, 3, 62, 44, 2, 23, 8, 60, 61, 12, 14, 68, 17, 80, 55, 41, 11, 39
Hybrid/Ensemble	30.12%	43, 35, 47, 4, 36, 51, 34, 75, 31, 15, 5, 20, 24, 81, 48, 82, 19, 30, 6, 65, 69, 27, 37, 73, 64
Indifference Between Techniques	9.64%	67, 40, 1, 9, 10, 38, 76, 16

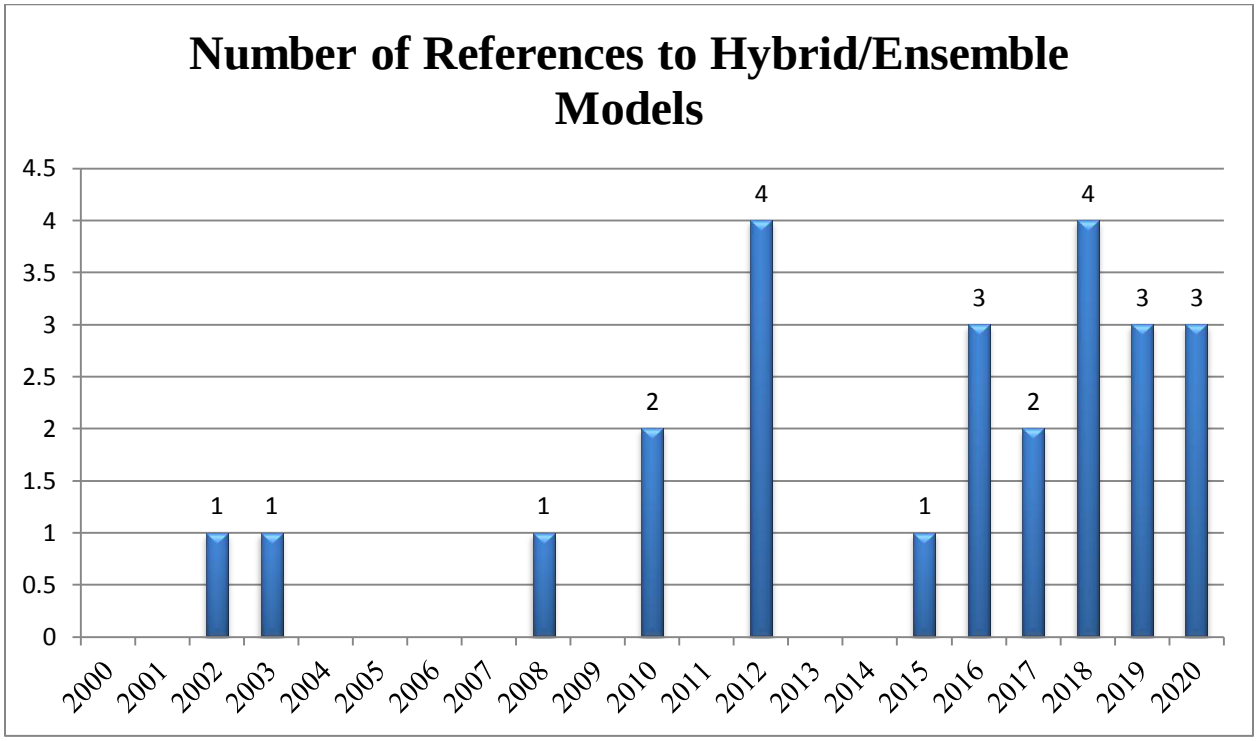

Figure 12: References to Hybrid/Ensemble Methods Over Time

Figure 13 illustrates the number of sources over time. The graph shows that the majority of sources captured in the data set are from 2010-2020. Figure 14 controls for the different technique styles. The figure displays that the sources primarily recommend Data-Based approaches. The surge can be explained by the emergence of Ensemble methods which additionally boost the presence of Data-Based styles.

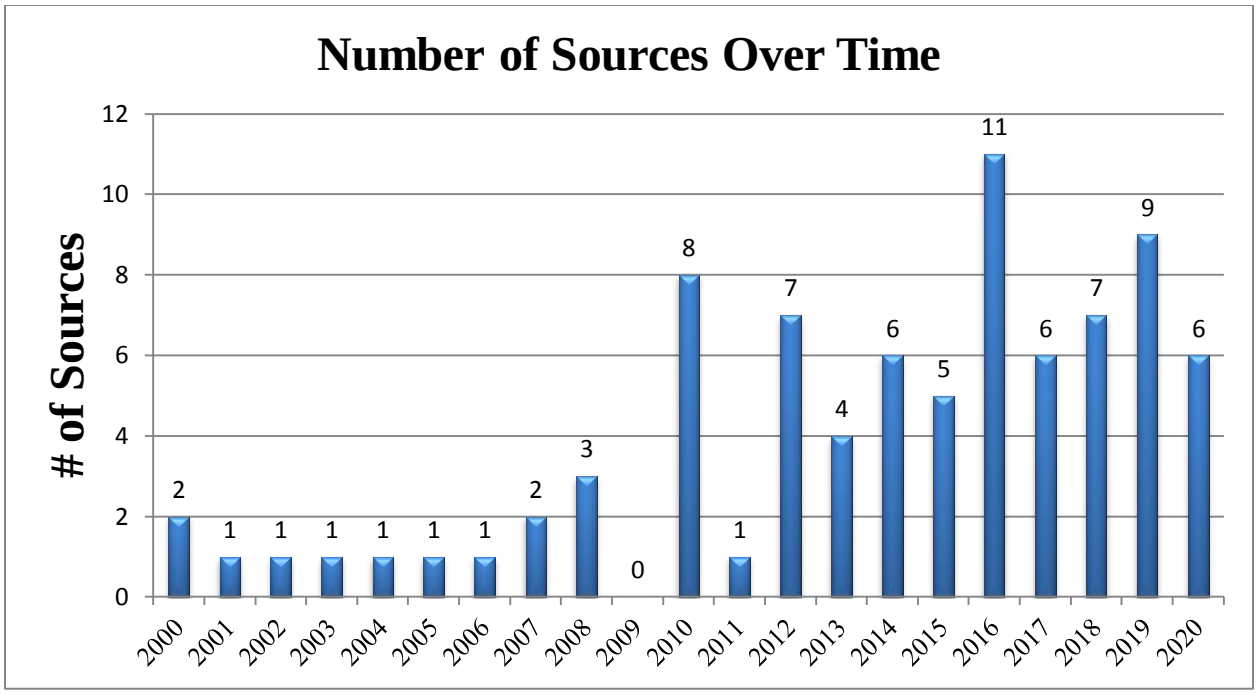

Figure 13: Number of Sources over Time

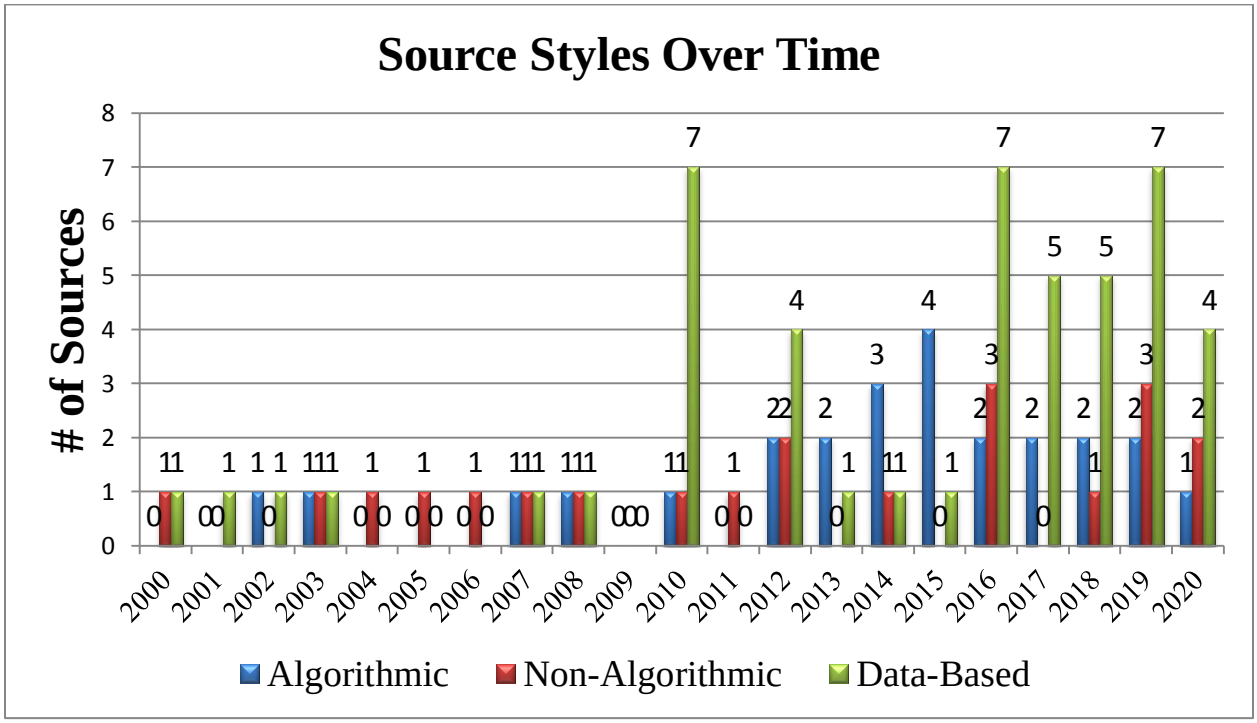

Figure 14: Source Styles Over Time

There are 7 sources found in the literature regarding DoD policy and doctrine on Agile software cost estimation. Due to the limited information, Figure 15 captures the techniques and

sizing metrics specified in the DoD literature in one graphic. There cannot be any conclusive determinations due to the low sample size; however, there is a noticeable lack of discussion regarding the use of Data-Based styles. The previous literature has highlighted the increase in the academic discussion regarding Data-Based styles. Only one DoD article (41) mentions the need for effort estimating to pivot towards using machine learning. Also of note, there is continued discussion and adherence to SLOC (16, 63) as a viable sizing metric as well as the reliance on expert judgment (16, 45) to construct estimates. Section 3 will analyze data collected from Software Factories to understand how the Air Force has been conducting its cost estimation in an Agile environment.

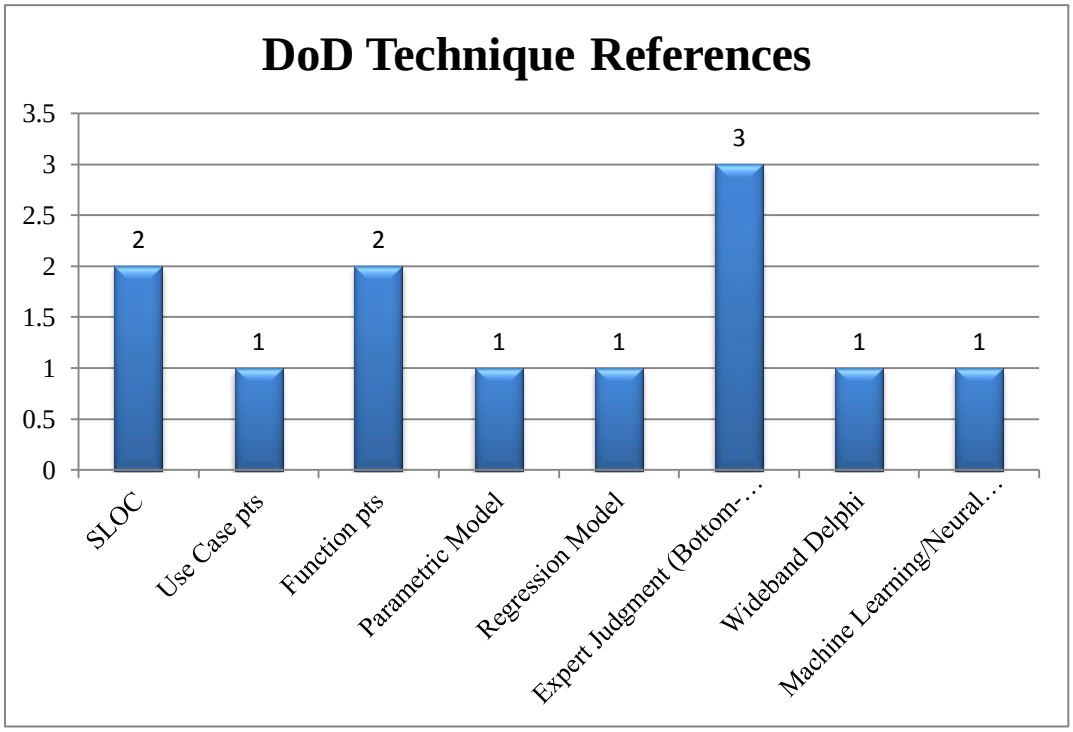

Figure 15: DoD Article References to Techniques

Section 3: Air Force Software Factory Analysis

Section 3 shows results from the data collection of the 11 Agile Air Force Software Factories. For the purpose of the data collection, a Software Factory is defined as any software

development team striving to apply Agile principles to their processes as they support DoD systems. For Data Set #2, Software Factories provide either the name of their organization or the specific program they are working on. Data Set #2 includes information from the Factories listed in Table 11. To maintain the integrity of responses, each of the Factory's specific answers will remain anonymous in the subsequent analysis. Factories are randomly assigned a number from 1 to 11 and any discussion regarding specific responses will refer to the respective sources as Factory #1-11.

Table 11: Data Set #2 Software Factories and Programs

Software Factory/Program Name	**Overall Mission**
Bespin	Delivering Custom Mobile Experiences to Airmen
Kessel Run	Delivering War-Winning Software Capabilities
Platform 1	DoD Enterprise DevSecOps Provider
Unified Platform	Providing DevSecOps/Software Factory Managed Services with Integrated Security
Rogue Blue	Developing & Sustaining STRATCOM Tools
Ski Camp	Employing DevSecOps to Support Embedded Weapon System Software
Space Camp	Software Node of Platform One Deploying Space Mission Capabilities
SMC Forge Program	Delivering a Common Command and Control Network for Satellites
A-10 Operational Flight Program	Delivering Avionics Software for the A-10
Personnel Recovery Command and Control	Delivering Tools & Services for Planning, Collaborating, and Managing Search and Rescue Efforts
F-16 Center Display Unit	Delivering Avionics Software for the F-16 Center Display Unit

The data call from the Software Factories closely mirrors the sizing metrics and technique categories determined in the literature review; however, there are some differences. Data Set #2 covers three main technique styles: Algorithmic, Non-Algorithmic, and Engineering

Build-up. In contrast, the three main styles in Data Set #1 are Algorithmic, Non-Algorithmic, and Data-Based. There are no references to Data-Based techniques at any of the Software Factories, so this technique style is effectively discarded for Data Set #2. Engineering Build-up represents a new categorization of cost estimation for this data set. The Algorithmic technique style has additional changes to its composition. Unlike Data Set #1, in Data Set #2, the various parametric techniques are compiled together under one 'Parametric' category due to the lack of overall responses. The Parametric category includes references to SEER-SEM, SLIM, COCOMO-II, and generic parametric techniques. The Software Factories elaborate on the use of Capacity Based and Analogy estimation which are techniques not previously defined or explored in Data Set #1.

Estimation by Analogy is a technique utilized by Factories. 4 of the 11 Factories (1, 6, 7, & 10) reference the use of Analogy based estimation. For the intent of the study, estimation by Analogy can be classified as a Non-Algorithmic technique style. The technique attempts to estimate cost for a new project by comparing it to previous projects. The method includes choosing the correct analogy, determining similarities and differences, examining analogy quality, and calculating an estimate (El Bajta, 2015). Capacity Based estimation is another prevalent technique, used in 7 of the 11 Factories (2, 3, 4, 5, 7, 8, & 10). For the purpose of this research, Capacity Based estimation will be treated as a completely separate technique style and will be classified as an 'Engineering Build-up.' Contrary to the previous techniques which focus on correlating the software produced to the cost, Capacity Based focuses on estimating the labor required to develop an expected amount of software. Elements of a contract are reviewed individually to assess the number of full time employees required to satisfy the requirements. The different perspective associated with Capacity Based estimation dictates its own technique

style categorization. After establishing the criteria for Data Set #2, Table 12 displays a summary of the technique styles and the respective Factory references.

Table 12: Data Set #2 Technique Styles and References

Technique Style	Techniques	Statistics of Usage	Factory References
Non-Algorithmic	Planning Poker	81.82%	3, 4, 5, 6, 7, 8, 9, 10, 11
Non-Algorithmic	Expert Judgment (Top-Down, Bottom-Up)	63.64%	1, 3, 4, 5, 6, 7, 10
Engineering Build-up	Capacity Based	63.64%	2, 3, 4, 5, 7, 8, 10
Non-Algorithmic	Analogy	36.36%	1, 6, 7, 10
Algorithmic	Parametric (COCOMO-II/SLIM/SEER-SEM/Generic Parametric)	27.27%	4, 6, 10
Non-Algorithmic	Wideband Delphi	18.18%	5, 10
Algorithmic	Regression	9.09%	6

Figure 16 shows the number of references to the three established technique styles. 10 of the 11 Factories articulate the use of Non-Algorithmic techniques in the creation of their cost estimates. By contrast, only 3 of the 11 Factories (4, 6, & 10) reference the use of Algorithmic technique styles for cost estimation. Additionally, all 3 of these Factories also utilize Non-Algorithmic techniques either in conjunction or as an alternative. The other notable takeaway is the prevalence of Engineering Build-up and Capacity Based estimation. 7 of the 11 (2, 3, 4, 5, 7, 8, &10) Factories utilize Engineering Build-up.

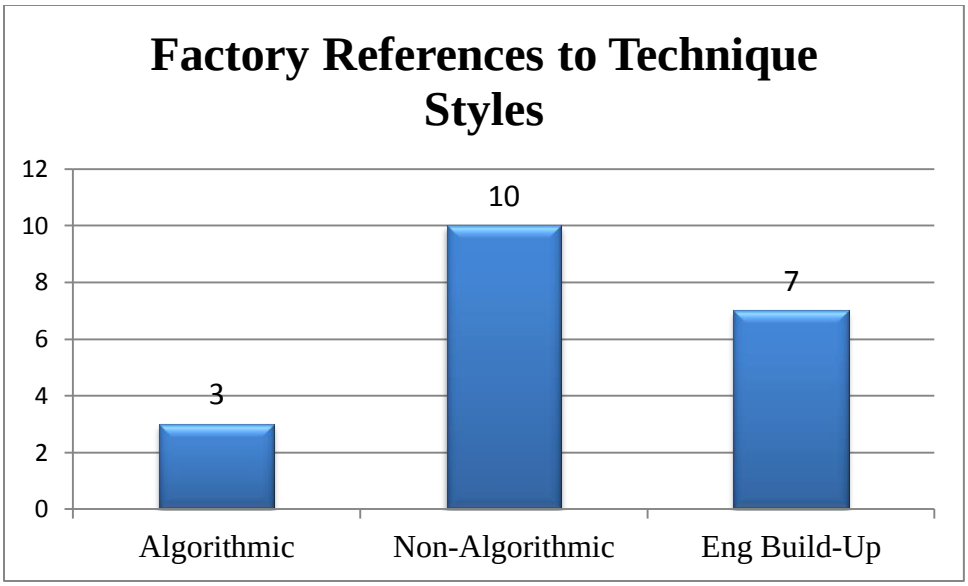

Figure 16: Factory References to Technique Styles

Figure 17 displays the references when accounting for the specific techniques utilized at Factories. The results help illustrate how Planning Poker and Subject Matter experts are the dominant Non-Algorithmic technique styles. Planning Poker is the most prevalent technique and is used in 9 of the 11 Factories (3, 4, 5, 6, 7, 8, 9, 10, & 11). Additionally, Subject Matter Experts are a predominant technique in 7 of the 11 Factories (1, 3, 4, 5, 6, 7, & 10). Wideband Delphi is only present in 2 Factories (5 and 11) while Analogy is present in 4 Factories (1, 6, 7, & 10).

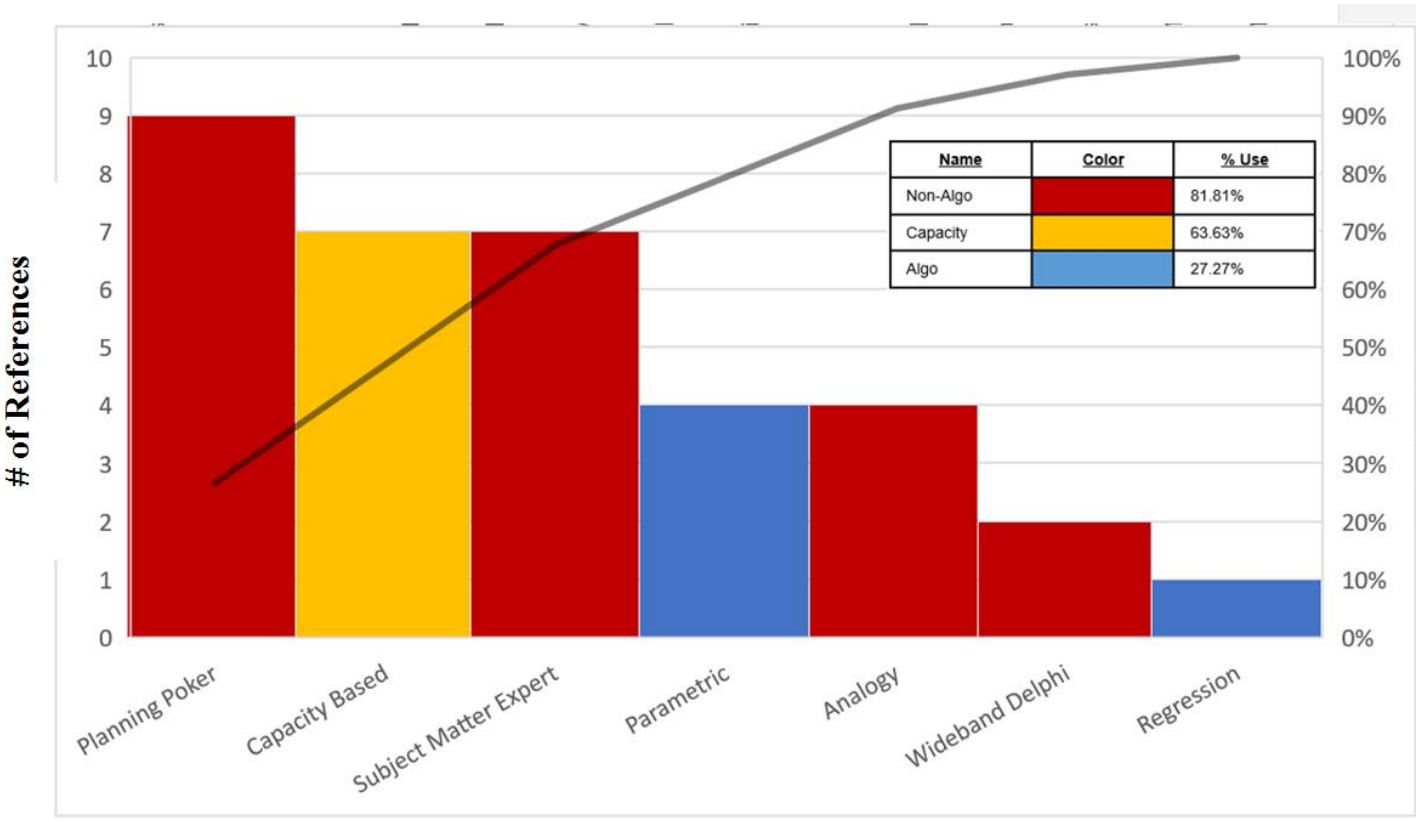

Figure 17: Factory References to Techniques

In regards to Algorithmic, there are 3 Factories that identify the usage of Algorithmic style techniques. While there are 3 Factory references (4, 6, & 10) to Parametric techniques, these references include caveats. As explained previously, all 3 Factories that specify the use of Algorithmic technique styles additionally utilize Non-Algorithmic techniques. Factory #6 articulates that Parametric techniques are typically only utilized by contractors or when mandated cost estimating databases do not have analogous projects. Additionally, there is one reference to Regression techniques at Factory #6; however, the team highlights that only some of the Parametric models include a Regression based approach. Furthermore, Factory #10 states that they rarely utilize Parametric techniques. Specifically, the Factories articulate that none of their organizations utilize the COCOMO-II model. The results contrast directly with Data Set #1

which has 9 of the 83 sources touting the use of the COCOMO-II model. Overall, the results highlight a predominant presence and preference towards Non-Algorithmic technique styles.

In addition to Non-Algorithmic, Factories also express their preference for Capacity Based estimates. 7 of the 11 Factories (2, 3, 4, 5, 7, 8, 10) highlight the use of Capacity Based estimation. Factory #2 articulates that since they are putting positions on contract instead of the product itself, it makes sense to directly estimate the capacity. They argue that the use of Capacity Based estimation has far more fidelity than the traditional use of any type of traditional Parametric techniques. Furthermore, Factories #3 and #4 support the notion that cost estimates should be constructed based on equipment, licenses, and full time employees. Factory #7 estimates the effort according to the number of overall Story Points to be accomplished over the course of the year and then determines the number of full time employees required to accomplish that established goal. Factory #8 identifies that cost estimation is independent of software size and is rather a function of personnel, equipment, contracting, and other direct costs. The results illustrate that Capacity Based is a widely utilized and supported technique for cost estimation at the Air Force Software Factories.

Table 13 shows specific Factory usage of sizing metrics. The data does not present a clear dominance of any one metric. Even the most prevalent metric, Story Points, is only incorporated in 5 of the 11 Factories (7, 8, 9, 10, & 11). However, there are notable takeaways. Only one Factory reports using Function Points (11) while 4 Factories (6, 7, 10, & 11) utilize Use Case Points. The data additionally highlights the fact that only 2 Factories (6 and 10) utilize SLOC. Factory #6 states they are not satisfied with the results of SLOC estimates, and that they typically transform SLOC values into Use Case Points. Factory #10 caveats that their usage of SLOC is only to support other program's metrics. Additionally, Factory #5 reports that they have removed

the use of SLOC in estimates as they do not believe it to be an accurate or relevant metric. Factory #7 clarifies that they have only recently transitioned from using SLOC to Use Case Points and Story Points. The results demonstrate that SLOC is generally not considered a viable metric at the Air Force Software Factories.

Table 13: Factory Sizing Metrics

Sizing Metric	Factory References
SLOC	6, 10
Function Points	11
Use Case Points	6, 7, 10, 11
Story Points	7, 8, 9, 10, 11

Overall, the results from the data present three major findings. First, Non-Algorithmic techniques are prevalent in almost the entirety of Data Set #2 while Algorithmic styles are almost non-existent. Second, Capacity Based estimating is highly prevalent in Factories and represents a form of software effort cost estimation that is not seen in Data Set #1. Third, almost all Factories do not support SLOC as metric in the Agile environment. Section 4 synthesizes the results from Data Sets #1 and #2.

Section 4: Comparison of Data Set #1 & #2

There are three main conclusions derived from comparing Data Set #1 and #2. First, the Air Force is lagging in terms of adaptation and adoption of Data-Based models. However, secondly, the Air Force is synchronized with the findings of the prevailing literature which shows that SLOC is typically not used as a metric in Agile environments. Lastly, despite the literature favoring Algorithmic and Data-Based techniques, the Air Force predominantly follows the use of Non-Algorithmic and Capacity Based cost estimation models.

One of the most noticeable differences is that there are no recorded instances of Data-Based techniques in Data Set #2. While perhaps surprising given the large quantity of Data-Based solutions in Data Set #1, the results can be explained by a number of reasons. The Air Force Agile Software Factories have only been recently established within the last several years. As of 2021, 6 out of the 11 Factories respond that they are either not happy or uncertain regarding their current cost estimation process. Data-Based solutions offer a much more advanced methodology for conducting cost estimates as an optimization on existing techniques. Air Force Software Factories are still trying to establish themselves and their overall framework. Therefore, as of 2021, the relative infancy of the Software Factories may help explain the lack of adopting more complicated cost models.

Furthermore, Data Set #1 shows the techniques that academics are perpetuating as the most preferred methodologies. It is worth noting, that while the case studies and data can mathematically justify the empirical advantage of using more refined techniques, it does not speak toward the level of difficulty in successfully adopting such practices. The Data-Based techniques may offer superior solutions; however, those solutions may only be minutely superior to a far simpler alternative. In economic terms, the marginal benefit experienced by the improved results may not outweigh the marginal costs required to adapt the model. Therefore, it makes sense that a less complicated and more easily adoptable cost model could provide Factories with a superior solution in the meantime.

Additionally, the absence of Data-Based techniques can be justified from a mathematical perspective. There appears to be a connection between Algorithmic and Data-Based Solutions. 9 of the 48 sources (43, 35, 1, 4, 34, 81, 30, 68, & 73) which recommended Data-Based solutions in Data Set #1 are optimizations of Algorithmic models. Data Set #1 demonstrates that there

appears to be a connection between the other Data-Based solutions as a means to optimize Algorithmic techniques. A decrease in the number of Algorithmic references could be correlated to a decrease in the number of Data-Based references. Therefore, the relatively low use of Algorithmic techniques in Data-Set #2 can help further explain the relative absence of Data-Based solutions.

Despite the prevalence of Non-Algorithmic and the lack of Data-Based techniques in current practice, the Air Force should still be advised to investigate the benefits and possible integration of machine learning concepts. Data-Based techniques can provide a refined optimization on models. The occurrence and preference for Data-Based solutions in Data Set #1 highlights a trend which the Air Force should not ignore. Source 41 in Data Set #1 is a DoD specific cost estimation guide which highlights the need for the Air Force to adopt machine learning Data-Based cost estimation techniques. However, there may be difficulties in assimilating Data-Based techniques especially in an Air Force Agile environment which predominantly favors Capacity Based and Non-Algorithmic estimation. There could be an issue as there appears to be a correlation between Algorithmic and Data-Based techniques; however, there are ways to integrate Non-Algorithmic techniques. 4 of the 48 Data-Based sources in Data Set #1 (35, 9, 65, & 37) are optimizations on Non-Algorithmic approaches. It is worth noting that all 4 of the sources utilize Case Based Analogy Data-Based techniques to optimize Expert Judgment Non-Algorithmic techniques. Therefore, Case Based Analogy estimates may offer a way for the Air Force to potentially integrate Data-Based techniques into their current estimation process. Regardless of the specific techniques, Factories should study the inclusion of Data-Based solutions at their organizations as a means to keep pace with the prevailing literature.

Data Set #1 does not account for Analogy or Capacity Based software effort estimation. The lack of Analogy Based estimation can be explained because the established criterion for identifying references in Data Set #1 effectively hides their usage. Although Analogy provides a method for conducting effort estimation, the goal of Data Set #1 is to capture the source's recommended methodologies. In general, sources in Data Set #1 highlight the superiority of Data-Based optimizations on techniques, including Analogy. Therefore, while there are Analogy examples (Data Set #1: 9, 22, & 48) present in Data Set #1, their inferiority to Data-Based techniques hides their usage. As a result, the preference for Analogy techniques is effectively non-existent.

Data Set #1 additionally does not include any sources of Capacity Based software effort estimation. The lack of Capacity Based references in Data Set #1 appears to simply be due to the lack of discussion in the published literature. To ensure the fidelity of Data Set #1, additional searches are conducted to expand the initial search made in Chapter 3 Part 2. Searches on IEEE and Science Direct that include strings "Capacity" and "Engineering build up" along with the previous search criteria fail to capture any additional viable sources. However, a more robust study on the appropriate search criteria needs to be conducted before definitely claiming the absence of Capacity Based estimation in literature of the last two decades. While there is a lack of information in Academia, the use of Capacity Based estimates is not unfounded in the Air Force. As explained by Factory #2, the methodology comes from the standard Air Force adherence to utilizing actuals, analogies, engineering build-up, or Parametric models for its traditional cost estimation. Capacity Based, as defined, is a derivation of the engineering build-up model. Factories adopt the model as it appears to be the most logical and viable practice.

However, additional research and accuracy measures would have to be applied to understand if Capacity Based estimation offers not only the easiest but the optimal cost estimation strategy.

According to the DoD's Software Development Estimating Handbook SLOC is one of the most widely used methods to obtain the scope for a software program (NCCA & AFCAA, 2008). However, typical Agile proponents argue against its use as the level of efficiency and experience between developers causes a disparity in the amount of SLOC and time required to develop similar functionality (Bhatt et al., 2012). The research appears to support the prevailing sentiment that SLOC is not widely used in Agile environments. Data Set #1 has 11 out of 83 references to SLOC as a metric while Data Set #2 has 2 out of 11 references. A comparison of confidence intervals can be utilized to understand if the two sets of data have statistically equivalent proportions in regards to the use of SLOC. A Clopper Pearson interval can be constructed to provide a 95% binomial confidence interval for the responses for SLOC usage in each data set. The null hypothesis is that there is not a significant difference between the data sets' use of SLOC. The alternative is that there is a significant difference in the way each data set uses SLOC. Applying Clopper Pearson to Data Set #1 provides a confidence interval of the proportion between 0.0681 and 0.2248. The Clopper Pearson interval for Data Set #2 is between 0.0228 and 0.5178. Figure 18 displays the two confidence intervals overlaid on the same graph, with the interval for Data Set #1 on the bottom in red and the interval for Data Set #2 on the top in blue. When comparing the confidence intervals, because there is an overlap, this results in the failure to reject the null. Therefore, the conclusion is that there is not a significant difference between the ways each data set uses SLOC as a metric.

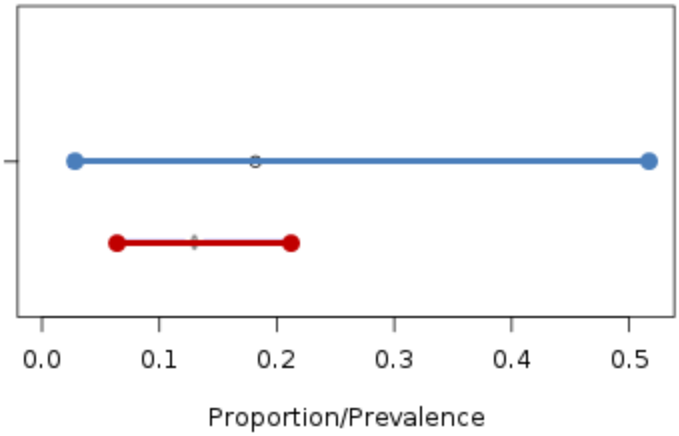

Figure 18: SLOC Use in Data Set #1 & #2

Furthermore, there are two major caveats to the 11 references to SLOC in Data Set #1. First, there is correlation between SLOC and the COCOMO-II model. The COCOMO-II model is known to work primarily with SLOC based inputs. 6 of the 11 sources (34, 66, 20, 30, 27, & 73) in Data Set #1 that reference SLOC additionally recommend the COCOMO-II model. By contrast, none of the Factories in Data Set #2 make use of the COCOMO-II model. Therefore, it is not surprising to see a lack of support for both SLOC and the COCOMO-II model in Data Set #2. The contrast highlights the fact that the COCOMO-II model may be more prevalent in the world of academic research rather than in regular industry practice. Therefore, under this assumption, when controlling for the COCOMO-II specific sources, there are only 5 references to SLOC in Data Set #1. Second, 2 of those remaining 5 references (16 and 60) are from DoD sources regarding cost estimation in an Agile environment. Therefore, when additionally controlling for those DoD sources, there are actually only 3 references (1, 36, and 54) from the literature that recommend the use of SLOC. The analysis further supports that the Air Force's Agile cost estimation practices, as demonstrated by Data Set #2, coincide with the majority of the sources in Data Set #1 which also do not incorporate SLOC into their cost estimation models. Tthe low proportions in both data sets show the low prevalence of SLOC in Agile.

Data Set #2 shows a far greater reliance on Non-Algorithmic models in comparison to Data Set #1. Data Set #1 highlights the fact that 18 of 83 sources (21.69%) compared to 7 of the 11 sources (63.63%) in Data Set #2 recommend the use of Subject Matter Judgment for cost estimates. Additionally, Data Set #1 shows that 4 of 83 sources (4.82%) compared to 9 of the 11 sources (81.81%) recommend Planning Poker as a technique for cost estimation. Overall, there are 21 of the 83 sources (25.30%) compared to 10 of the 11 Factories (90.90%) that recommend Non-Algorithmic techniques. Once again, a Clopper Pearson interval can be utilized to construct a 95% confidence interval for each data set's proportion of references to Non-Algorithmic styles. The analysis will evaluate if there is a statistical difference between the ways each data set treats the use of Non-Algorithmic styles. The null hypothesis is that there is not a significant difference between the data sets' use of Non-Algorithmic styles. The alternative is that there is a significant difference between the ways each data set addresses the use of Non-Algorithmic styles. When applying Clopper Pearson to Data Set #1, the confidence interval of the proportion is between 0.1639 and 0.3604. When applied to Data Set #2, the confidence interval for the proportion is between 0.5872 and 0.9977. Figure 19 displays the two confidence intervals overlaid on the same image, with the interval for Data Set #1 on the bottom in red and the interval for Data Set #2 on the top in blue. When comparing the confidence intervals, because there is not an overlap this results in the rejection of the null in favor of the alternative hypothesis. Therefore, the conclusion is that there is a significant difference between the ways each data set uses Non-Algorithmic styles.

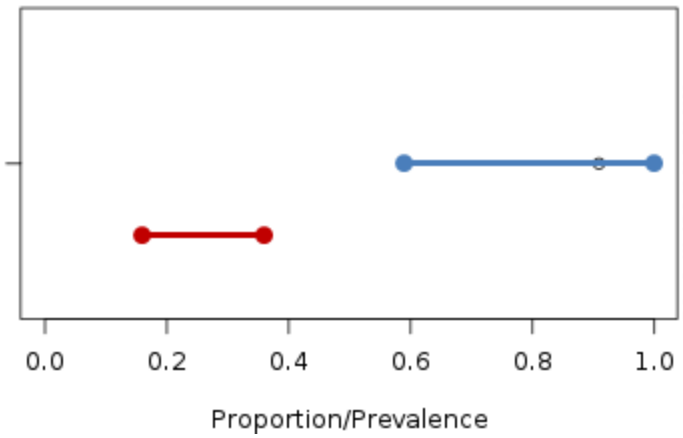

Figure 19: Non-Algorithmic Styles in Data Set #1 & #2

Lastly, there appears to be equivalent use of Algorithmic styles in both data sets. There are 25 of 83 sources (30.12%) in Data Set #1 compared to 3 of 11 Factories (27.27%) in Data Set #2 that recommend the use of Algorithmic models. Once again, a test of each proportion's 95% confidence interval can help verify if there is a statistical difference between the ways each data set uses Algorithmic styles. The null hypothesis is that there is not a statistical difference in the way each data set uses Algorithmic styles. The alternative hypothesis is that there is a statistical difference in the way each data set makes use of Algorithmic styles. When applying Clopper Pearson, the confidence interval for the proportion for Data Set #1 is between 0.2003 and 0.4118. When applied for Data Set #2, the confidence interval for the proportion is between 0.0602 and 0.6097. Figure 20 displays the two confidence intervals overlaid on the same image, with the interval for Data Set #1 on the bottom in red and the interval for Data Set #2 on the top in blue. When comparing the confidence intervals, because there is an overlap this results in the failure to reject the null. Therefore, the conclusion is that there is not a significant difference between the ways each data set uses Algorithmic styles.

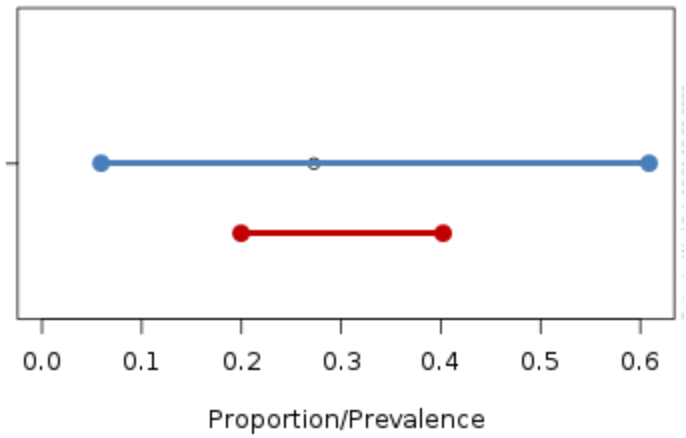

Figure 20: Algorithmic Styles in Data Set #1 & #2

However, there are caveats to the findings regarding Algorithmic techniques. As described in Section 3, although there are three instances of the use of Algorithmic styles, the Factories express their hesitancy to utilize Parametric techniques. All 3 Factories (4, 6, & 10) that specify the use of Algorithmic technique styles additionally utilize Non-Algorithmic techniques. Factory #6 specifies that Algorithmic techniques are typically only utilized by contractors or when mandated cost estimating databases do not have analogous projects. Furthermore, Factory #10 states that they rarely utilize Parametric techniques in favor of Non-Algorithmic techniques. Therefore, the real usage and reliance on Algorithmic methods may in fact be lower at the Factories compared to the literature. However, without further analysis or a larger Factory sample size, no claims regarding the statistical difference between the two data sets can be verified.

Overall, Data Set #1 emphasizes Data-Based approaches, Data Set #2 primarily uses Non-Algorithmic methodologies, and both data sets utilize Algorithmic styles to the same degree. As previously discussed, the lack of advanced Data-Based models in Data Set #1 makes intuitive sense given the potential difficulty and payoff associated with incorporating them. By contrast, the Factories express a greater reliance on Capacity Based and Non-Algorithmic

estimation. Factory #2 articulates that Parametric models involve numerous steps which all incorporate a certain amount of uncertainty. The compounding amounts of uncertainty only serve to distort the estimate making it far less accurate than their Capacity Based and Non-Algorithmic approaches. While Data Set # 1 appears to favor the outcomes of Data-Based solutions, the Air Force adheres to a Non-Algorithmic and Capacity Based approach. There can be no definitive claims regarding the accuracy of the current Air Force cost estimation practices; however, the research indicates that further study should be conducted to determine whether alternative cost estimation techniques could provide superior solutions.

The comparison between Data Set #1 and #2 highlights three main points. First, the Air Force needs to continue to research ways to consider incorporating Data-Based techniques into their Factories. Second, despite DoD literature, the Air Force agrees with the predominant majority of sources and does not utilize SLOC as a metric within its Agile organizations. Third, the Air Force adheres to Non-Algorithmic and Capacity Based estimation which contradicts the prevailing literature that favors Data-Based models.

Chapter Summary

Chapter 4 analyzes the results for Part #1 and #2 of the research. Part 1 demonstrates the CoD Analysis framework applied to Kessel Run data. There are three sections in Part 1. Section 1 describes the CoD framework for opportunity cost data collection. Section 2 estimates KRADOS application costs and TBMCS contract costs. Section 3 provides overall takeaways to the DoD CoD framework. Part 2 analyzes data on the current state of Agile software cost estimating. There are four sections in Part 2. Section 1 is an overview of the sizing metrics and effort estimation techniques. Section 2 displays figures and analysis regarding the literature gathered in Data Set #1. Section 3 displays figures and analysis regarding the data collected from

Air Force Software Factories in Data Set #2. Section 4 synthesizes Data Set #1 and #2 to provide insight into how the Air Force has been conducting its Agile cost estimating compared to the prevailing literature. Chapter 5 will discuss the overall conclusions, limitations, and recommendations for future research as a result of this study.

V. Conclusions and Recommendations

Notable Takeaways on CoD Analysis

The research formulates a method in which DoD entities could beneficially conduct CoD Analysis. The study illustrates that a traditional CoD framework cannot be readily applied to a DoD public sector environment. The public good nature of military readiness prohibits the use of a profit maximization model. However, it is possible to reframe the CoD perspective towards the public sector. The focus pivots towards cost *reduction* and *avoidance* rather than revenue *generation* and *protection* opportunities. By emphasizing an optimization of cost reductions and avoidances, the CoD model can successfully represent DoD programs. Additionally, DoD requirements primarily take the form of either *Long Life-Cycle Peak Unaffected by Delay* or *Impact of External Deadline*. The DoD model focuses primarily on requirements with fixed opportunity costs that do not diminish over time. The DoD CoD framework provides insight into how the government could operate in a more efficacious manner.

The results highlight the fact that certain features have significantly higher opportunity costs compared to others. Features with high levels of value can potentially get lost in the product backlog. CoD prioritization provides leaders a way to understand the potential operational value incurred by the implementation of a requirement. CoD framework establishes a defined structure to ascertain the requirements that would have the greatest impact and to define their value in a more quantifiable form. The features with the greatest opportunity costs should, in theory, relate directly to the features that the users most desire. Even within the small sample of this study, the evidence shows that teams are highly encouraged to take the time to robustly analyze the end impacts gained by features.

Second, CoD Analysis can be used as a *prioritization* and *justification* tool. While the two points are intertwined, there is a distinction between them. *Justification* relates to understanding the overall importance of the mission while *prioritization* attempts to make decisions by directly comparing competing entities. CoD can act as a *justification tool* to articulate the overall importance of contracts, missions, or features as a means to either leverage greater budgets or manpower. Calculations pivot the conversation away from only considering the development costs or abstract qualitative benefits in favor of quantitative opportunity costs associated with operational impact. Understanding the opportunity cost behind improving or failing to improve operational capabilities provides a more robust view of the true cost of delay. Elaborating further, CoD has been shown as a means of *prioritization*. Every decision made carries a number of cost considerations. Leadership must decide the best course of action amongst a variety of different possible solutions. CoD Analysis provides a means of empirically quantifying the opportunity costs associated with the implementation of new features. As a result, executing a CoD Analysis offers leaders a supplemental decision making tool.

Third, the CoD assessments illustrate how small inefficiencies add up to significant opportunity cost losses. The limited data set only represents a small sample of the potential cost savings. However, with just the initial assessment, Jigsaw Feature #2 shows that an increase in manpower efficiency from one feature saves the government thousands of dollars per week. Inefficiencies can often be overlooked due to the relatively small impact they have on the operation. Jigsaw Feature #2 demonstrates how the failure to implement fixes can accumulate over time into large opportunity cost losses.

CoD Limitations and Future Research

The analysis provides a minimal understanding of the true scope capable with CoD Analysis as the research includes a variety of limitations. First, there is insufficient data from the application teams to fully demonstrate the power of a CoD Analysis. The theoretical structure defines how a concept could be applied; however, lack of data from Kessel Run prohibited a more in depth representation of the CoD methodology in practice. Furthermore, the data only highlights the cost savings experienced with reduction in manpower hours. In addition, the data only makes use of the *Long Life-Cycle, Peak Unaffected by Delay* model of urgency profile and does not illustrate an example of *Impact of External Deadline*. Further identification of opportunity costs in requirements could reveal even more significant efficiencies to be gained with the implementation of a requirement.

The small number of features also fails to illustrate the benefit of a CoD prioritization. With only two features, the decision making is simple. The calculations only illustrate four of the many features in an application team's product backlog. However, while quantifying the dollar cost savings gives some idea of what features are the most important to providing better overall military readiness, it does not provide a full picture. The research concedes that because military readiness is a public good that certain operational capabilities become difficult if not impossible to quantify with a dollar assessment. For example, there are challenges in fully dollarizing the benefits or opportunity cost savings associated with being able to locate and eliminate enemy targets. However, as a leader in the DoD, improving military capabilities represent the major consideration when making decisions.

Further research can be conducted into potentially expanding the extant model into a more robust form that accounts for the benefits provided by the improvement of military

readiness. The ability to better frame and quantify the operational impacts and benefits would ultimately lead to a more valuable CoD metric than the current iteration. However, as a direct follow-on to the developed framework, the next step requires additional data collection. Future research should involve using the developed DoD CoD framework in a more data focused analysis. A rigorous collection of data as documented in Table 4 in Chapter 4 would provide further empirical evidence towards the benefits of a CoD Analysis. Further data collection could also allow for statistical comparisons to other prioritization methods. There are a variety of different methods discussed in the research on how organizations can prioritize a backlog. For example, a product backlog prioritized under a CoD framework could be compared to a list prioritized by which features are the most important to the user. The comparison could indicate the linkage between opportunity cost and customer preference. The proof of concept also only illustrates point estimates for many of the time savings. Advanced models should incorporate risk and uncertainty profiles to the point estimates for a more robust range of potential opportunity cost savings. Moving forward, Kessel Run and other Agile oriented organizations are an excellent test bed for such a study to be conducted.

Notable Takeaways for Agile Software Cost Estimation Techniques

The research examines the extant literature regarding Agile cost estimation techniques. The study utilizes a data set of 83 sources which provide recommendations on the best strategies for cost estimation in an Agile environment. To summarize, there has been a strong push in literature in the last 10 years towards more mathematically sophisticated Data-Based cost estimation models. The results indicate that 48 of the 83 sources (57.83%) recommend the application of Data-Based solutions. The literature additionally highlights the ability for Data-Based solutions to work in conjunction with other techniques as a way to optimize accuracy. The

ability for Data-Based models to incorporate additional techniques has led to the growth of hybrid/ensemble models in literature. 25 of the 83 sources (30.12%) recommend the construction of a hybrid/ensemble method. Furthermore, 21 of the 25 sources (84%) that recommend an ensemble model incorporate Data-Based techniques. Continued advances in machine learning over the last decade have enabled further development of Data-Based cost models.

The research on the DoD's use of Agile cost estimation analyzes the practices implemented in 11 of the Air Force's 16 Software Factories. The findings reveal that the Factories predominantly favor the use of Non-Algorithmic techniques especially in comparison to Algorithmic techniques which are only used when mandated. 10 of the 11 Factories utilize Non-Algorithmic styles while only 3 of the 10 incorporate Algorithmic techniques. Of the known Parametric techniques, Factories clarify that none of their organizations utilize the COCOMO-II model. The research also identifies the presence of Capacity Based estimating as a principal strategy employed at the Factories. Capacity Based estimating is a form of engineering build-up that is not seen within the literature gathered in Data Set #1. Further analysis is required to verify the applicability and usage of Capacity Based estimates in an Agile software environment. Lastly, there are 2 out of the 11 Factories that utilize SLOC as a metric. However, the Factories specify that they do not trust SLOC as an accurate or relevant metric inside an Agile environment. The Factories articulate that they incorporate SLOC estimates when mandated by certain contracts. Typically SLOC estimates are translated into different formats such as Use Case Points. As a result, the preference for using SLOC is virtually non-existent within the Factories.

Data Set #2 provides further insight on the Air Force's Agile cost estimation practices, especially when compared to Data Set #1. Having previously acknowledged the widespread use

of Capacity Based estimating in the Factories, there are three other main takeaways from the comparison of data sets. First, the Air Force has an overall preference to utilize Non-Algorithmic and Capacity Based styles for cost estimation which is statistically higher than that discussed in the literature. Second, the Air Force should continue to investigate the potential benefits of Data-Based cost estimation techniques. Third, despite the DoD literature, Factories are moving with the majority of the cited literature and not utilizing SLOC as a metric for estimating code.

Contrary to the predominant literature supporting the advancement of more complicated cost models, Air Force Agile Software Factories favor simplified Engineering Build-up capacity based estimates supplemented by Non-Algorithmic techniques. Overall, 10 out of the 11 Factories utilize some form of Non-Algorithmic techniques. By contrast, 21 out of the 83 sources in Data Set #1 recommend the benefits of Non-Algorithmic techniques. A statistical test of the respective proportions' confidence intervals utilizing Clopper Pearson shows that there is a significant difference between these proportions. Therefore, Data Set #2 utilizes Non-Algorithmic techniques at a higher proportion than Data Set #1.

Data-Based models are absent from Data Set #1 for a variety of potential reasons. First, while empirical case studies can mathematically justify the numerical advantage associated with more refined techniques, they do not address the level of difficulty and effort required to successfully adopt such practices. Although a Data-Based technique may provide a superior solution, that solution may only be incrementally advantageous compared to a far simpler alternative. In economic terms, the marginal benefit provided by the improved results may not outweigh the marginal costs needed to adapt the model. As of 2021, Factories which have only been established within the last few years may not have the technical or economic incentive to adopt the more rigorous mathematical Data-Based solutions. Second, there appears to be a

correlation between Algorithmic and Data-Based Solutions. 9 of the 48 sources (43, 35, 1, 4, 34, 81, 30, 68, & 73) which recommended Data-Based solutions in Data Set #1 are improvements on Algorithmic models. A decrease in the number of Algorithmic references could be correlated to a decrease in the number of Data-Based references. Therefore, the relatively low number of Algorithmic techniques employed in Data-Set #2 can help further justify the relative absence of Data-Based solutions.

The prevalence and touted superiority of Data-Based techniques coupled with the lack of their utilization in the Factories only further justifies the Air Force's need to further investigate their implementation. The DoD acknowledges the need to embrace machine learning in their cost estimation practices (McQuade, Murray, Louie, Medin, Pahlka, & Stephens, 2019). As the literature appears to demonstrate the correlation between Algorithmic and Data-Based solutions, Factories who rely on Non-Algorithmic techniques may have a difficult time adopting Data-Based styles. However, there are 4 sources (35, 9, 65, & 37) in Data Set #1 which refer to Case Based Analogy as an optimization of Subject Matter Experts. Therefore, a possible solution could be the integration of a Case Based Analogy Data-Based technique to optimize Subject Matter Experts estimates. The Air Force should continue to study the possible inclusion of Data-Based techniques to ensure they keep pace with the prevailing best practices for cost estimation.

Machine learning practices are not without flaw. Artificial intelligence boosted Data-Based models are essentially "black-boxes." Models can automatically map complex relations between input and output variables without the need for an analyst to make any manual calculations. Some data scientists worry about over reliance on big data models. Coke spent millions to develop Data-Based models which demonstrate the market effectiveness of creating a "Cherry-Sprite" flavor (Vyas, Jain, Choudhary, & Chaudhary, 2019). Critics argue that Coke

could have easily learned from 7Up that released a cherry flavor variant 30 years prior (Nolis, 2018). The danger of over reliance on Data-Based solutions is the removal of human intuition and critical thinking. Data-Based models are still based on inputted assumptions, which if wrong, can provide a bad result. Ultimately, critics argue that data cannot replace human intuition and it cannot remove risk. Therefore, Data-Based methods and solutions should be treated, like all other models, with sufficient skepticism.

There are two more notable comparisons between the data sets. 25 of the 83 sources in Data Set #1 recommend Algorithmic techniques compared to only 3 of the 11 Factories. Although the Factories may potentially utilize Algorithmic methods less than the literature, the statistical analysis fails to reject the hypothesis that both data sets use them at the same proportion. Additionally, there is no statistical difference in the ways the data sets treat the use of SLOC. 11 out of 83 sources in Data Set #1 compared to 2 out of 11 Factories recommend the use of SLOC. A statistical test of the two proportions' confidence intervals fails to reject the hypothesis that the data sets utilize SLOC to the same degree. However, the low proportions seem to indicate that neither data set utilize SLOC.

There are caveats to each data set which additionally justify the low usage of SLOC conclusion. In Data Set #1, there are several articles which describe the use of the COCOMO-II model. 6 of the 11 sources (34, 66, 20, 30, 27, & 73) that mention SLOC additionally utilize COCOMO-II. Although the COCOMO-II model provides a useful research tool, its presence in the industry practice appears to be limited as none of the Factories in Data Set #2 utilize COCOMO-II. Further research would need to be conducted to understand the true overall usage and prevalence of COCOMO-II in industry. However, the high correlation between COCOMO-II models and the use of SLOC may potentially provide an overly high representation of its true

usage. Furthermore, 2 of the remaining 6 sources are from the DoD, and therefore can be excluded to have a representation of the private sector's perspective on SLOC. As a result, the actual prevalence of SLOC in both data sets is low.

Agile Cost Estimation Techniques Limitations and Future Research

The results from Data Set #1 also highlight limitations. The purpose of the data set is to provide a standard barometer on the currently recommended Agile cost estimation techniques. However, there are likely differences between academic literatures when compared to actual industry practices as demonstrated by the discussion on COCOMO-II. The literature review provides a benchmark for comparison in place of data collection with commercial industries. However, as an academic source, any Data Set #1 results or comparisons must be carefully examined.

Data Set #1 fails to account for Capacity Based and Analogy estimation techniques. Future research should expand to data bases beyond just IEEE and Science Direct. Preliminary research failed to identify additional estimation methods. For future research, search strings should attempt to be more inclusive as a means to find additional effort estimation techniques. The initial findings do not record any sources describing Capacity Based software estimation; however, expanded searches could derive additional sources.

Data Set #1 tracks sources which promote the accuracy or superiority of certain techniques or styles. It documents the frequency of techniques mentioned by sources as a means of justifying the advantage of techniques. Future research should expand the scope of the research and attempt to track empirical accuracy measures as a means of further justifying the overall superiority of techniques.

The research attempts to determine the best practices for the Air Force. However, a limitation of Data Set #2 is that it is only capable of recording the existing cost estimating techniques in use at the Factories. Although Factories provided their level of satisfaction with the current practices, the data set does not provide measures of accuracy that can empirically prove the superiority of techniques. Future research should incorporate a more involved approach to compare levels of accuracy with each of the Factories' cost estimates. Furthermore, the data call with the 11 Factories is used as a representative sample for the way the Air Force handles Agile cost estimation. The Agile Software Factory initiative is still growing. The data from only 11 Factories leads to the inability to do a robust statistical comparison to Data Set #1. The use of Clopper Pearson Confidence interval tests provides some level of comparison; however, the small sample size do not allow for precise hypothesis testing. Future data sets could pull from a larger sample size and take advantage of additional years of costing experience at the already established Factories. Future research can focus on collecting more detailed assessments of Factories' cost estimating practices.

Appendix A: Additional Kessel Run Application and Contract Background

The TBMCS is an integrated air command and control (C2) system that performs standardized, secure, automated air battle planning and execution management for Air Force, multi-service, and allied commanders in theaters of operation worldwide. TBMCS provides a manner in which to plan, direct, and control theater air operations in coordination with land, maritime, and special operations elements. It is deployed at C2 nodes at national, force, and wing/unit-level elements. TBMCS operates in support of planners and decision makers at the level of Joint Force Air Component Commander. The system is modular and scalable for air, land, or sea transport and the deployed configurations can be tailored to meet a particular contingency.

The Kessel Run KRADOS portfolio consists of ten applications that upon full implementation and standup will effectively remove the need to rely on the continued use of TBMCS. However, there is currently an interim period where KRADOS is not fully operational and as a result there is a continued reliance and funding of both TBMCS and KRADOS. The ten application teams that currently make up KRADOS are: Jigsaw, Slapshot, Gambit, Triton, Coaxium, Maitai, Skyhook, Jump, Direct, and Spacer. Effective CoD information for these 10 applications was unobtainable, and therefore the study will rely on the examination of a small sample of data provided by Jigsaw and Chainsaw as a means of constructing a proof of concept for the overall model.

Chainsaw is a Kessel Run application team whose mission is to move dynamic targeting beyond checklist management to near instant strike prosecution. Chainsaw currently supports the Combat Operation Division specifically the Dynamic Targeting Cell at the 609th. The application works as a way to quickly filter and sort potential targets by prosecution status, priority, or date.

The application automatically generates a 10-line mission order to be sent over tactical chat. Among the updates, it provides a variety of benefits over the previous dynamic targeting capabilities provides through TBMCS. Among the benefits, it presents a more streamlined user experience, the inclusion of multiple users on a single entity, and reduced prosecution time.

Jigsaw is a Kessel Run Application team whose mission is to provide an intuitive and reactive interface that enables tanker planners to save time, fuel, and lives. Jigsaw provides a digital whiteboard to tanker planners allowing for the manipulation of flight plan strategies. The application provides a dashboard for users to track efficiency metrics, takeoff, transit, and refueling times. Jigsaw currently works with other Kessel Run applications Spacer, Slapshot, and Mai Tai as a means to automate certain processes, reduce manual calculations, and prevent data corruption. Jigsaw provides improved visualization, calculation, and interfaces that were not previously incorporated under TBMCS.

Appendix B: Selected Cost Estimation Techniques Works Cited

1) Abrahamsson, P., Moser, R., Pedrycz, W., Sillitti, A., & Succi, G. (2007). Effort Prediction in Iterative Software Development Processes – Incremental Versus Global Prediction Models. First International Symposium on Empirical Software Engineering and Measurement (pp. 344-353). Madrid, Spain: ESEM.

2) Adnan, M., & Afzal, M. (2017). Ontology Based Multiagent Effort Estimation System for Scrum Agile Method. IEEE Access Volume: 5, 25993-26005.

3) Amasaki, S., & Lokan, C. (2016). On Applicability of Fixed-Size Moving Windows for ANN-Based Effort Estimation. Joint Conference of the International Workshop on Software Measurement and the International Conference on Software (pp. 213-218). Berlin, Germany: IEEE.

4) Attarzadeh, I., & Ow, S. H. (2010). Proposing a New Software Cost Estimation Model Based on Artificial Neural Networks. 2nd International Conference on Computer Engineering and Technology (pp. 487-491). Chengdu, China: IEEE.

5) Azzeh, M., & Nassif, A. B. (2016). A hybrid model for estimating software project effort from Use Case Points. Applied Soft Computing 49, 981-989.

6) Azzeh, M., Nassif, A. B., Banitaan, S., & Lopez-Martin, C. (2018). Ensemble of Learning Project Productivity in Software Effort Based on Use Case Points. 17th IEEE International Conference on Machine Learning and Applications (pp. 1427-1431). Orlando, Florida: IEEE.

7) Burgess, C. J., & Lefley, M. (2001). Can Genetic Programming Improve Software Effort Estimation? A Comparartive Evaluation. Information and Software Technology, 863-873.

8) Conde, P. P., & Carrillo, I. S. (2017). Comparison of classifiers based on neural networks and support vector machines. 5th International Conference in Software Engineering Research and Innovation (pp. 107-115). Merida, Mexico: IEEE.

9) Cuadrado-Gallego, J. J., Rodriguez-Soria, P., & Martin-Herrera, B. (2010). Analogies and differences between Machine Learning and Expert based Software Project Effort Estimation. 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (pp. 269-276). Madrid, Spain: IEEE.

10) Cunha, J. C., Costa, M., Cruz, S., Vieira, M., & Rodrigues, A. (2011). Implementing Software Effort Estimation in a Medium-sized Company. 34th IEEE Software Engineering Workshop (pp. 92-96). Limerick, Ireland: IEEE.

11) Dan, I., Catalin, R., & Oliver, O. (2020). An NLP Approach to Estimating Effort in a Work Environment. International Conference on Software, Telecommunications and Computer Networks (SoftCOM) (pp. 1-6). Split, Croatia: IEEE.

12) Defense Science Board. (2018). Design and Acquisition of Software for Defense Systems. Department of Defense.

13) Dumke, R. R., Neumann, R., & Schmietendorf, A. (2014). Empirical-Based Extension of the COSMIC FP Method. Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Produt Measurement (pp. 5-10). Rotterdam, Netherlands: IEEE.

14) Gandomani, T. J., Faraji, H., & Radnejad, M. (2019). Planning Poker in Cost Estimation in Agile Methods: Averaging vs. Consensus. 5th Conference on Knowledge Based Engineering and Innovation (pp. 66-71). Tehran, Iran: IEEE.

15) Garcia-Diaz, N., Garcia-Virgen, J., Farias-Mendoza, N., Veruzco-Ramirez, A., Martinez-Bonilla, R., Chavez-Valdez, E., et al. (2015). Software development time estimation based on a new Neuro-fuzzy approach. 10th Iberian Conference on Information Systems and Technologies (pp. 1-7). Aveiro, Portugal: IEEE.

16) Government Accountability Office. (2020). Agile Assessment Guide Best Practices for Agile Adoption & Implementation. U.S. Government Accountability Office.

17) Goyal, S., & Bhatia, P. K. (2019). A Non-Linear Technique for Effective Software Effort Estimation using Multi-Layer Perceptrons. International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (Com-IT-Con) (pp. 1-4). Faridabad, India: IEEE.

18) Grimstad, S., & Jorgensen, M. (2007). Inconsistency of Expert Judgment-Based Estimates of Software Development Effort. The Journal of Systems and Software 80, 1770-1777.

19) Hammad, M., & Alqaddoumi, A. (2018). Features-Level Software Effort Estimation Using Machine Learning Algorithms. International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (pp. 1-3). Sakhier, Bahrain: IEEE.

20) Hira, A., & Boehm, B. (2016). Combatting Use Case Points' Limitations with COCOMO(R) II and Relative Difficulty. 23rd Asia-Pacific Software Engineering Conference (pp. 353-356). Hamilton, New Zealand: IEEE.

21) Hooi, T. C., Yusoff, Y., & Hassan, Z. (2008). Comparative Study on Applicability of WEBMO in Web Application Cost Estimation within Klang Valley in Malaysia. IEEE 8th International Conference on Computer and Information Technology Workshops (pp. 116-121). Sydney, Australia: IEEE.

22) Idri, A., Hosni, M., & Abran, A. (2016). Improved estimation of software development effort using Classical and Fuzzy Analogy ensembles. Applied Soft Computing 49, 990-1019.

23) Ionescu, V.-S. (2017). An approach to software development effort estimation using machine learning. 13th IEEE International Conference on Intelligent Computer Communication and Processing (pp. 197-203). Cluj-Napoca, Romania: IEEE.

24) Iwata, K., Nakashima, T., Anan, Y., & Ishii, N. (2016). Effort Estimation for Embedded Software Development Projects by Combining Machine Learning with Classification. 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence (pp. 265-270). Las Vegas, USA: IEEE.

25) Jorgensen, M. (2004). Top-Down and Bottom-Up Expert Estimation of Software Development Effort. Information and Software Technology 46, 3-16.

26) Jorgensen, M. (2005). Practical Guidelines for Expert-Judgment-Based Software Effort Estimation. IEEE Software, 57-63.

27) Kadir, N. F., Sarkan, H. B., Azmi, A. B., Yusop, O. B., & Karma, M. N. (2019). Specification of a Hybrid Effort Estimation System using UML. 6th International Conference on Research and Innovation in Information Systems (pp. 1-7). Johor Bahru: IEEE.

28) Kang, S., Choi, O., & Baik, J. (2010). Model-based Dynamic Cost Estimation and Tracking Method for Agile Software Develpoment. 9th IEEE/ACIS International Conference on Computer and Information Science (pp. 743-748). Yamagata, Japan: IEEE.

29) Kchaou, D., Bouassida, N., & Ben-Abdallah, H. (2015). Change Effort Estimation based on UML Diagrams Application in UCP and COCOMO-II. 10th International Joint Conference on Software Technologies (pp. 1-8). Colmar, France: IEEE.

30) Khan, M. S., Ul Hassan, A., Shah, M. A., & Shamim, A. (2018). Software Cost and Effort Estimation using a New Optimization Algorithm Inspired by Strawberry Plant. 24th International Conference on Automation and Computing (ICAC) (pp. 1-6). Newcastle Upon Tyne, United Kingdom: IEEE.

31) Kocaguneli, E., Menzies, T., & Keung, J. W. (2012). On the Value of Ensemble Effort Estimation. Transactions on Software Engineering, 1403-1416.

32) Kocaguneli, E., Tosum, A., & Bener, A. (2010). AI-Based Models for Software Effort Estimation. 36th EUROMICRO Conference on Software Engineering and Advanced Applications (pp. 323-326). Lille, France: IEEE.

33) Kompella, L. (2013). Advancement of decision making in Agile Projects by Applying Logsitic Regression on Estimates. 8th International Conference on Global Software Engineering Workshops (pp. 11-17). Bari, Italy: IEEE.

34) Litoriya, R., Sharma, N., & Kothari, A. (2012). Incorporating Cost driver substitution to improve the Effort using Agile COCOMO II. CSI Sixth International Conference on SOftware Engineering. Indore, India: IEEE.

35) MacDonell, S. G., & Shepperd, M. (2003). Combining Techniques to Optimize Effort Predictions in Software Project Management. The Journal of Systems and Software 66, 91-98.

36) Machine learning methods and asymmetric cost function to estimate execution effort of software testing. (2010). Third International Conference on Software Testing, Verification and Validation (pp. 275-284). Campinas, Brazil: IEEE.

37) Mahmood, Y., Kama, N., Azmi, A., & Ali, M. (2020). Improving Estimation Accuracy Prediction of Software Development Effort: A Proposed Ensemble Model. The 2nd International Conference on Electrical, Communication and Computer Engineering (ICECCE) (pp. 1-6). Istanbul, Turkey: IEEE.

38) Mahnic, V., & Hovelja, T. (2012). On Using PLanning Poker for Estimating User Stories. The Journal of Systems and Software 85, 2086-2095.

39) Mann, K., & Hoang, R. (2020). But Wait, There's More! Using SFPA for your Cost, Schedule, and Performance Needs. Department of Homeland Security.

40) McConnell, S. (2006). Software Estimation: Demystifying the Black Art. Redmond, Washington: Microsoft Press.

41) McQuade, J. M., Murray, R. M., Louie, G., Medin, M., Pahlka, J., & Stephens, T. (2019). Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage. Department of Defense Office of Prepublication and Security Review.

42) Mendes, E., & Mosley, N. (2008). Bayesian Network Models for Web Effort Prediction: A Comparative Study. IEEE Computer Society, 723-737.

43) Mendes, E., Watson, I., Triggs, C., Mosley, N., & Counsell, S. (2002). A Comparison of Development Effort Estimation Techniques for Web Hypermedia Applications. Eighth IEEE Symposium on Software Metrics. IEEE Computer Society.

44) MITRE. (2016). Federal Aviation Administration Agile Acquisition Principles and Practices. Federal Aviation Administration.

45) Modigliani, P., & Chang, S. (2014). Defense Agile Acquisition Guide. Mitre.

46) Moharreri, K., Sapre, A. V., Ramanathan, J., & Ramnath, R. (2016). Cost-Effective Supervised Learning Models for Software Effort Estimation in Agile Environments. 40th Annual Computer Software and Applications Conference (pp. 136-140). Atlanta, USA: IEEE.

47) Molokken-Ostvold, K., Haugen, N. C., & Benestad, H. C. (2008). Using planning poker for combining expert estimates in software projects. The Journal of Systems and Software 81, 2106-2117.

48) Monika, & Sangwan, O. P. (2017). Software Effort Estimation Using Machine Learning Techniques. 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence (pp. 92-98). Noida, India: IEEE.

49) Nadgeri, S., Hulsure, V. P., & Gawande, A. D. (2010). Comparative Study of Various Regression Methods for Software Effort Estimation. 3rd International Conference on Emerging Trends in Engineering and Technology (pp. 642-645). Goa, India: IEEE.

50) Nassif, A. B., Capretz, L. F., & Ho, D. (2012). Estimating Software Effort Using an ANN Model Based on Use Case Points. 11th International Conference on Machine Learning and Applications (pp. 42-47). Boca Raton, Florida: IEEE.

51) Nassif, A. B., Capretz, L. F., Ho, D., & Azzeh, M. (2012). A Treeboost Model for Software Effort Estimation Based on Use Case Points. 11th International Conference on Machine Learning and Applications (pp. 314-319). Boca Raton, Florida: IEEE.

52) Nathaneal, E. H., Hendradjaya, B., & Sunindyo, W. D. (2015). Study of Algorithmic Method and Model for Effort Estimation in Big Data Software Development Case Study: Geodatabase. The 5th International Conference on Electrical Engineering and Informatics (pp. 427-432). Bali, Indonesia: IEEE.

53) Owais, M., & Ramakishore, R. (2016). Effort, Duration and Cost Estimation in Agile Software Development. Ninth International Conference on Contemporary Computing (pp. 1-5). Noida, India: IEEE.

54) Phan, V. P., Chau, N. P., & Nguyen, M. L. (2016). Exploiting Tree Structures for Classifying Programs by Functionalities. Eighth International Conference on Knowledge and Systems Engineering (pp. 85-90). Hanoi, Vietnam: IEEE.

55) Polkowski, Z., Vora, J., Tanwar, S., Tyagi, S., Singh, P. K., & Singh, Y. (2019). Machine Learning-based Software Effort Estimation: An Analysis. 11th International Conference on Electronics, Computers and Artificial Intelligence (pp. 1-6). Pitesti, Romania: IEEE.

56) Popli, R., & Chauhan, N. (2014). Agile Estimation Using People and Project Related Factors. International Conference on Computing for Sustainable Global Development (pp. 564-569). New Delhi, India: IEEE.

57) Popli, R., & Chauhan, N. (2014). Cost and Effort Estimation in Agile Software Development. International Conference on Reliability, Optimization and Information Technology (pp. 57-61). Faridabad, India: IEEE.

58) Popli, R., & Chauhan, N. (2014). Estimation in Agile Environment using Resistance Factors. International Conference on Information Systems and Computer Networks (pp. 60-65). Mathura, India: IEEE.

59) Prabhakar, V., & Dutta, M. (2013). Prediction of Software Effort Using Artificial Neural Network and Support Vector Machine. Computer Science.

60) Rosa, W., Madachy, R., Clark, B., & Boehm, B. (2017). Early Phase Cost Models for Agile Software Processes in the US DoD. ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 30-37). Toronto, Canada: IEEE.

61) Saini, A., Ahuja, L., & Khatri, S. K. (2018). Effort Estimation of Agile Development using Fuzzy Logic. 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO) (pp. 779-783). Noida, India: IEEE.

62) Satapathy, S. M., Acharya, B. P., & Rath, S. K. (2016). Early Stage software Effort Estimation Using Random Forest Technique Based on Use Case Points. IET Software Vol: 10, Issue: 1, 10-17.

63) Sehra, S. K., Kaur, J., Brar, Y. S., & Kaur, N. (2014). Analysis of Data Mining Techniques for Software Effort Estimation. 11th International Conference on Information Technology: New Generations (pp. 633-638). Las Vegas, Nevada: IEEE.

64) Servadei, L., Mosca, E., Zennaro, E., Devarajegowda, K., Werner, M., Ecker, W., et al. (2020). Accurate Cost Estimation of Memory Systems Utilizing Machine Learning and Solutions from Computer Vision for Design Automation. Transactions on Computers Volume: 69, Issue: 6, 856-867.

65) Shams, A., Bohm, S., Winzer, P., & Dorner, R. (2019). App Cost Estimation-Evaluating Agile Environments. IEEE 21st Conference on Business Informatics (pp. 383-390). Moscow, Russia: IEEE.

66) Sharma, H. K., Tomar, R., Dumka, A., & Aswal, M. S. (2015). OpenECOCOMO: The Algorithms and Implementaion of Extended Cost Costructive Model (E-COCOMO). 1st International Conference on Next Generation Computing Technologies (pp. 773-778). Dehradun, India: IEEE.

67) Shin, M., & Goel, A. L. (2000). Empirical data modeling in software engineering using radial basis functions. Transactions on Software Engineering Vol: 26, Issue: 6, 567-576.

68) Shukla, S., & Kumar, S. (2019). Applicability of Neural Network based Models for Software Effort Estimation. IEEE World Congress on Services (pp. 339-342). Milan, Italy: IEEE.

69) Shukla, S., Kumar, S., & Ranjan Bal, P. (2019). Analyzing Effect of Ensemble Models on Multi-Layer Perceptron Network for Software Effort Estimation. IEEE World Congress on Services (pp. 386-387). Milan, Italy: IEEE.

70) Sikka, G., Kaur, A., & Uddin, M. (2010). Estimating Function Points: Using Machine Learning and Regression Models. 2nd International Conference on Education Technology and Computer (ICETC) (pp. 52-56). Shanghai, China: IEEE.

71) Silhavy, R., Silhavy, P., & Prokopova, Z. (Applied Least Square Regression in Use Case Estimation Precision Tuning). Applied Least Square Regression in Use Case Estimation Precision Tuning. Software Engineering in Intelligent Systems. Advances in Intelligent Systems and Computing, vol 349, 11-17.

72) Smith, A. E., & Mason, A. K. (2010). COST ESTIMATION PREDICTIVE MODELING: Regression Versus Neural Network. The Engineering Economist 42, 137-161.

73) Suherman, I. C., Sarno, R., & Sholiq. (2020). Implementation of Random Forest Regression for COCOMO II Effort Estimation. International Seminar on Application for Technology of Information and Communication (iSemantic) (pp. 476-481). Semarang, Indonesia: IEEE.

74) Toka, D., & Tretken, O. (2013). Accuracy of Contemporary Parametric Software Estimation Models: A Comparative Analysis. 39th Euromicro Conference Series on Software Engineering and Advanced Applications (pp. 313-316). Santander, Spain: IEEE.

75) Tsunoda, M., Monden, A., Keung, J., & Matsumoto, K. (2012). Incorporating Expert Judgment into Regression Models of Software Effort Estimation. 19th Asia-Pacific Software Engineering Conference (pp. 374-379). Hong Kong, China: IEEE.

76) Usman, M., Petersen, K., Borstler, J., & Neto, P. S. (2018). Developing and using checklists to improve software effort estimation: A multi-case study. The Journal of Systems and Software 146, 286-309.

77) Valdes-Souto, F. (2016). Creating a Historical Database for Estimation Using the EPCU Approximation Approach for COSMIC (ISO 19761). 4th International Conference in Software Engineering Research and Innovation (pp. 159-166). Puebla, Mexico: IEEE.

78) Wahid, A., & Masud, P. (2013). Efficiency Factor and Risk Factor B ased User Case Point Test Effort Estimation Model Compatible with Agile Software Development. International Conference on Information Technology and Electrical Engineering (pp. 113-118). Yogyakarta, Indonesia: IEEE.

79) Wiegers, K. E. (2000). Stop Promising Miracles. Software Development.

80) Wright, I., & Ziegler, A. (2019). The standard coder: a machine learning approach to measuring the Effort Required to Produce Source Code Change. IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (pp. 1-7). Montreal, Canada: IEEE.

81) Yazdani-Chamzini, A., Zavadskas, E. K., Antucheviciene, J., & Bausys, R. (2017). A Model for Shovel Capital Cost Estimation, Using a Hybrid Model of Multivariate Regression and Neural Networks. Symmetry Volume: 9, 1-14.

82) Zakrani, A., Najm, A., & Marzak, A. (2018). Support Vector Regression Based on Grid-Search Method For Agile Software Effort Prediction. IEEE 5th International Congress on Information Science and Technology (pp. 492-497). Marrakech, Morocco: IEEE.

83) Ziauddin, Tipu, S. K., & Zia, S. (2012). An Effort Estimation Model for Agile Software Development. Advances in Computer Science and its Applications, 314-324.

Appendix C: Sizing Metrics & Cost Estimating Technique Descriptions

Sizing Metrics

SLOC

Source Lines of Code also known as SLOC is a metric that specifies the number of lines of source code present in a piece of software (Albrecht & Gaffney, 1983). The number of lines of code provides a manner to predict the amount of work required to develop software. SLOC measurements can be integrated into functional models as a means to provide an estimate of cost.

Function Points

Function Points measure the functionality of software from the user's point of view on a basis of what the user requests and receives. Originally, Albrecht and Gaffney (1983) articulated a model that categorizes requirements into one of five different buckets: outputs, inquiries, inputs, internal logical files, and external interface files (Albrecht & Gaffney, 1983). Outputs are information given to the user such as reports (Sikka, Kaur, & Uddin, 2010). Inquiries are queries which provide an online output (Sikka, Kaur, & Uddin, 2010). Inputs are any user fed information (Sikka, Kaur, & Uddin, 2010). Internal logical files are files or databases (Sikka, Kaur, & Uddin, 2010). External interface files are information held by other systems used by the system being analyzed (Sikka, Kaur, & Uddin, 2010). Once the function is identified and categorized into a type, it is assessed for complexity and assigned a subjective function point number (Albrecht & Gaffney, 1983). Function Points can be developed relatively easily in discussions between the user and developer at an early stage of development (Albrecht & Gaffney, 1983). Ultimately, the number provides a unit of measure for the amount of effort required for a certain amount of work which can be integrated into a function to determine cost.

Use Case Points

Proposed by Karner in 1993, Use Case Points is a model based on use case diagrams (Karner, 1993). Software size is calculated based on the number of actors and use cases present in a use case diagram multiplied by a complexity weight factor (Karner, 1993). Software size is calculated through two stages incorporating the Unadjusted Use Case Points (UUCP) and Adjusted Use Case Points (UCP). UUCP is calculated with summation of the Unadjusted Use Case Weight and Unadjusted Actor Weight (Nassif A. B., Capretz, Ho, & Azzeh, 2012). UCP is determined by multiplying UUCP by Technical and Environmental Factors (Nassif A. B., Capretz, Ho, & Azzeh, 2012). Karner originally proposed a 20 person hour to develop each UCP. (Karner, 1993)

Story Points

A story point is a metric used in Agile environments to estimate the difficulty of implementing a given user story which is a relative measure of effort required to implement (Owais & Ramakishore, 2016). Difficulty can be related to complexity, risk, or the effort involved. Typically, groups adopt Fibonacci Numbers (3,5,8,13,21, etc.) as a means of bracketing the point values experts can utilize when assigning points to a story (Visual Paradigm, 2020). A story assigned a value of 2 should be twice as important as a story assigned a value of 1, and only 2/3 of a story valued at 3 points. Story Points can be implemented with various functions as a means to estimate total cost of a specific effort.

Algorithmic

Generic Parametric & Regression & COCOMO II/SLIM/SEER-SIM

Algorithmic models utilize derived formulas from historical data to determine costs (Popli & Chauhan, 2014). Algorithmic models can additionally incorporate the use of regression based strategies. Regression is the statistical establishment of relationships between dependent variables with one or more independent variables (Kok, Kitchenham, & Kirawkowski, 1990). Linear regression is one the more commonly utilized methods and attempts to find a straight line relationship between predictor variables and the dependent variable (MacDonell & Shepperd, 2003). The subsequent analysis of data can be utilized to create a standard formula as seen below in Figure 21 where Y is the estimated variable, X's are the independent variables, and β's are the associated coefficients. The constructed formula in turn can be used with new data to predict and forecast future costs.

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n$$

Figure 21: Parametric Formula

Some articles report the development of new tailor made Parametric models. However, other articles describe the use of COnstructuve COst MOdel (COCOMO-II), Software Life-Cycle Model (SLIM), and System Evaluation and Estimation of Resources (SEER-SEM) which are all examples of well-known established Algorithmic models that utilize historical project data to predict cost (Toka & Tretken, 2013). Boehm (1983) defined the original COCOMO model as a way to estimate effort, cost, and schedule for software projects (Boehm, Software Engineering Economics, 1981). The model requires the size, product, and personnel as inputs to provide an output (Hira & Boehm, 2016). SEER-SEM is a proprietary estimation model owned by Galorath established in 1979 (Hira & Boehm, 2016). SLIM is a proprietary estimation model developed in

the late 1970s by Larry Putnam (Hira & Boehm, 2016). Both software packages are highly configurable and can be adjusted to the appropriate work environment.

Non-Algorithmic

Expert Judgment

Subject matter experts can be a reliable source to construct estimates as their years of industry experience provide them with vast background knowledge on their area of expertise. Published surveys suggest that expert estimation is the dominant strategy for estimating software development effort (Jorgensen M. , 2002). When applying top-down estimation, the total effort of a software project is estimated without a decomposition of the project into activities or other types of project parts. A possible top-down expert estimation strategy is to compare the current project, as a whole, with previously completed similar, projects. The estimated total effort is then distributed over activities, applying for example guidelines about the activities' typical proportion of the total effort. When applying bottom-up estimation, the project work is typically divided into activities and the effort of each activity is estimated. The project's estimate of the total effort is then the sum of the effort estimates of each project activity, possibly with the addition of an effort budget to cover unexpected activities and events.

Planning Poker

Planning Poker is a technique that utilizes multiple experts to determine the Story Points associated with a certain User Story (Gandomani, Faraji, & Radnejad, 2019). Each expert receives a special deck of cards with set values. Teams analyze one User Story at a time ensuring that everyone understands the requirements carefully. Each expert then estimates the size of the selected User Story by showing a card indicating a value. The values are selected with

consideration to factors such as complexity and risk. Teams discuss User Story size estimates until a consensus is reached. In the case of disagreement, the experts who chose higher or lower values need to explain their reasoning to the group (Gandomani, Faraji, & Radnejad, 2019). To facilitate the analysis, teams determine the User Story's specifications, requirements, and limitations. The associated scores can then be utilized with the associated rates to estimate the cost to produce a Story.

Wideband Delphi Method

Wideband Delphi is a process originally popularized by Boehm in 1981 (Boehm, 1981). Delphi is similar to Planning Poker in that there is a moderator who supervises the process of experts providing their estimates regarding the level of effort required to complete a certain portion of work. However, unlike Planning Poker, Delphi incorporates anonymous forms from experts (Molokken-Ostvold, Haugen, & Benestad, 2008). Ultimately, the team must reach a consensus on the time to complete a task which in turn can be utilized to determine the associated cost (Molokken-Ostvold, Haugen, & Benestad, 2008). Wideband Delphi offers an alternative way to leverage expert opinion in a structured format to predict the effort, time, or complexity involved in software development.

Data-Based

Neural Networks/Linear Regression/Bayesian

ANN's are a form of machine learning artificial intelligence that can be used to identify highly complex relationships between input and output includes three main steps. The steps are applied to three layers: input, hidden, and output. The input layer provides input variables to the network in the form of a vector with the dimension equal to the number of neurons (Arabzadeh,

Niaki, & Arabzadeh, 2017). The hidden layer represents the main computational aspect of the network. Based on the inputs received, a corresponding output value is generated using an assigned activation function in Figure 22 (Arabzadeh, Niaki, & Arabzadeh, 2017).

$$y = f\left(\sum_i w_i x_i - \theta\right)$$

Figure 22: Activation Function

The functional notation in the formula above for the symbols (f), w_i, x_i, θ, and y are the activation functions, weighting factor, input of each node, bias, and the output respectively. The most common activation functions, (f), are seen in Figure 23 (Arabzadeh, Niaki, & Arabzadeh, 2017).

$$f(x) = \text{Sigmoid}(x) = {}^{1}/_{(1+e^{x})}$$

$$f(x) = \text{Signum}(x) = \begin{cases} 1 & \text{if } (x) > 0 \\ 0 & \text{if } (x) = 0 \\ -1 & \text{if } (x) < 0 \end{cases}$$

$$f(x) = \text{Step}(x) = \begin{cases} 1 & \text{if } (x) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Figure 23: Activation Functions

ANN's were first developed and applied during the 1990s to estimate construction costs. ANN's utilize the activation function that operates the weighted inputs to provide the final output of the system (Bilgaiayan, Sagnika, Mishra, & Das, 2017). They are purely data driven models which through iterative training transition from a random state to a final model (Smith & Mason, 1997). ANN's have been applied throughout Agile businesses primarily related to engineering industry. Fast (2009) developed an ANN model for an industrial gas turbine (Fast, Assadi, & De,

2009). They utilized the operational data with a multilayer feed-forward network to construct a model. Mesroghli (2009) applied an ANN cost model to estimate the calorific value from coal (Mesroghli, Jorjani, & Chehreh Chelgani, 2009). Additionally, Zhang (1996) developed a feature-based ANN to model cost estimation for the packaging of products (Zhang, Fuh, & Chan, 1996). While computationally complex, ANN's provide an advanced method to develop responsive quantitative cost models which can be used in a variety of different fields.

Case-Based Analogy

CBR is an AI methodology combined with a database of cases related to the topics under consideration for re-using past experience (Cuadrado-Gallego, Rodriguez-Soria, & Martin-Herrera, 2010). The approach relates previous cases similar to the current target project. By finding similar projects with known effort values, these can then be utilized to predict effort for the target project (MacDonell & Shepperd, 2003).

Regression Utilizing Unsupervised Learning Techniques

There are many regression models that additionally incorporate machine learning unsupervised methodologies. Stochastic gradient boosting, treeboost models, support vector regression, and various clustering techniques can all be categorized under this grouping (Nassif A. B., Capretz, Ho, & Azzeh, 2012). Unsupervised learning models differ from supervised ones by looking for patterns in a data set with a minimum use of human supervision.

Works Cited

National Research Council. (2010). *Critical Code: Software Producibility for Defense.* Washington, DC: The National Academies Press.

Abrahamsson, P., Moser, R., Pedrycz, W., Sillitti, A., & Succi, G. (2007). Effort Prediction in Iterative Software Development Processes – Incremental Versus Global Prediction Models. *First International Symposium on Empirical Software Engineering and Measurement* (pp. 344-353). Madrid, Spain: ESEM.

Adnan, M., & Afzal, M. (2017). Ontology Based Multiagent Effort Estimation System for Scrum Agile Method. *IEEE Access Volume: 5*, 25993-26005.

Air Force Logistics Managment Agency. (2009). *Maintenance Metrics U.S. Air Force.* Maxwell: Air Force Logistics Management Agency.

Albrecht, A., & Gaffney, J. E. (1983). Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering Vol:9, Issue:6*, 639-648.

Amasaki, S., & Lokan, C. (2016). On Applicability of Fixed-Size Moving Windows for ANN-Based Effort Estimation. *Joint Conference of the International Workshop on Software Measurement and the International Conference on Software* (pp. 213-218). Berlin, Germany: IEEE.

Arabzadeh, V., Niaki, S., & Arabzadeh, V. (2017). Constructioncost estimation of spherical storage tanks: artificial neural networks and hybrid regression- GA algorithms. *Industrial Engineering International*, 747-756.

Arnold, J., & Yuce, O. (2013). *Black Swan Farming using Cost of Delay.* Nashville: Black Swan Farming.

Attarzadeh, I., & Ow, S. H. (2010). Proposing a New Software Cost Estimation Model Based on Artificial Neural Networks. *2nd International Conference on Computer Engineering and Technology* (pp. 487-491). Chengdu, China: IEEE.

Azzeh, M., & Nassif, A. B. (2016). A hybrid model for estimating software project effort from Use Case Points. *Applied Soft Computing 49*, 981-989.

Azzeh, M., Nassif, A. B., Banitaan, S., & Lopez-Martin, C. (2018). Ensemble of Learning Project Productivity in Software Effort Based on Use Case Points. *17th IEEE International Conference on Machine Learning and Applications* (pp. 1427-1431). Orlando, Florida: IEEE.

Bhatt, K., Tarey, V., & Patel, P. (2012). Analysis of Source Lines of Code (SLOC) Metric. *International Journal of Emerging Technology and Advanced Engineering*, 150-153.

Bilgaiayan, S., Sagnika, S., Mishra, S., & Das, M. (2017). A Systematic Review on Software Cost Estimation in Agile Software Development. *Engineering Science and Technology Review*, 51-64.

Black Swan Farming. (2017). *Estimation Is Hard-But Worthwhile*. Retrieved August 23, 2020, from Black Swan Farming: https://blackswanfarming.com/why-making-value-estimates-is-hard-but-worthwhile/

Black Swan Farming. (2017). *Why Bother *quantifying* the Cost of Delay?* Retrieved August 20, 2020, from Black Swan Farming: https://blackswanfarming.com/why-bother-quantifying-the-cost-of-delay/

Boehm, B. (1981). *Software Engineering Economics.* Prentice Hall.

Boehm, B. (2000). *Spiral Development: Experience, Principles, and Refinements.* Pittsburgh: Carnegie Mellon Software Engineering Institute.

Burgess, C. J., & Lefley, M. (2001). Can Genetic Programming Improve Software Effort Estimation? A Comparartive Evaluation. *Information and Software Technology*, 863-873.

Butler, E. (2010). *Ludwig von Mises - A Primer.* London: The Institute of Economic Affairs.

Carden, A. (2007). Profit and Production. *Quarterly Journal of Austrian Economics*.

Chaillan, N. (2019). *DoD Enterprise DevSecOps Initiative (Software Factory).* DoD Enterprise DevSecOps Initiative.

Cohen, R. (2019, September 1). *The Air Force Software Revolution*. Retrieved August 13, 2020, from Air Force Magazine: https://www.airforcemag.com/article/the-air-force-software-revolution/

Cohen, R. (2019, September 1). *The Air Force Software Revolution*. Retrieved August 24, 2020, from Air Force Magazine: https://www.airforcemag.com/article/the-air-force-software-revolution/

Cohn, M. (2005). *Agile Estimating and Planning.* Upper Saddle River: Prentice Hall.

Conde, P. P., & Carrillo, I. S. (2017). Comparison of classifiers based on neural networks and support vector machines. *5th International Conference in Software Engineering Research and Innovation* (pp. 107-115). Merida, Mexico: IEEE.

Cuadrado-Gallego, J. J., Rodriguez-Soria, P., & Martin-Herrera, B. (2010). Analogies and differences between Machine Learning and Expert based Software Project Effort Estimation. *11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing* (pp. 269-276). Madrid, Spain: IEEE.

Cunha, J. C., Costa, M., Cruz, S., Vieira, M., & Rodrigues, A. (2011). Implementing Software Effort Estimation in a Medium-sized Company. *34th IEEE Software Engineering Workshop* (pp. 92-96). Limerick, Ireland: IEEE.

Dan, I., Catalin, R., & Oliver, O. (2020). An NLP Approach to Estimating Effort in a Work Environment. *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (pp. 1-6). Split, Croatia: IEEE.

David Consulting Group. (2020). *What's the Story(Points) with Function Points?* David Consulting Group.

Defense Innovation Board. (2018). *Ten Commandments of Software.* Washington, DC: Defense Innovation Board.

Defense Science Board. (2018). *Design and Acquisition of Software for Defense Systems.* Washington, DC: Defense Science Board.

Defense Science Board. (2018). *Design and Acquisition of Software for Defense Systems.* Department of Defense.

Defense Technical Information Center. (1980). *Advanced Technology Incorporated*. Retrieved August 20, 2020, from Sensitivity of System Readiness to Resource Allocation: https://doi.org/10.21236/ADA090290

Digital.AI. (2020). *14th Annual State of Agile Report.* Digital.ai Software, Inc.

Dumke, R. R., Neumann, R., & Schmietendorf, A. (2014). Empirical-Based Extension of the COSMIC FP Method. *Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Produt Measurement* (pp. 5-10). Rotterdam, Netherlands: IEEE.

Dvorak, D. (2009). *NASA study on flight software complexity.* Pasadena: California Institute of Technology.

El Bajta, M. (2015). Analogy-Based Software Development Efort Estimation in Global Software Development. *10th International Conference on Global Software Engineering Workshops* (pp. 51-54). Cidad Real: IEEE.

Everstine, B. (2019, September 25). *USAF Wants to Find New Ways to Discuss Fleet Readiness*. Retrieved August 10, 2020, from Air Force Magazine: https://www.airforcemag.com/usaf-wants-to-find-new-ways-to-discuss-fleet-readiness/

Fast, M., Assadi, M., & De, S. (2009). Development and multi-utility of an ANN model for an industrial gas turbine. *Appl. Energy*, 9-17.

Forrester, J., O'Hanlon, M., & Zenko, M. (2001). Measuring U.S. Military Readiness. *National Security Studies Quarterly*, 99-117.

Freeform Dynamics & CA Technologies. (2018). *How Agile and DevOps enable digital readiness and transformation.* Freeform Dynamics & CA Technologies.

Fry, F. G. (2010). *Optimizing Aircraft Availability: Where to Spend Your Next O&M Dollar.* Dayton: Air Force Institute of Technology.

Gandomani, T. J., Faraji, H., & Radnejad, M. (2019). Planning Poker in Cost Estimation in Agile Methods: Averaging vs. Consensus. *5th Conference on Knowledge Based Engineering and Innovation* (pp. 66-71). Tehran, Iran: IEEE.

GAO. (2009). *GAO Cost Estimating and Assessment Guide.* Washington D.C: GAO.

Garcia-Diaz, N., Garcia-Virgen, J., Farias-Mendoza, N., Veruzco-Ramirez, A., Martinez-Bonilla, R., Chavez-Valdez, E., et al. (2015). Software development time estimation based on a new Neuro-fuzzy approach. *10th Iberian Conference on Information Systems and Technologies* (pp. 1-7). Aveiro, Portugal: IEEE.

Government Accountability Office. (2014). *F-35 Joint Strike Fighter Problems Completing Software Testing May Hinder Delivery of Expectd Warfighting Capabilities.* Government Accountability Office.

Government Accountability Office. (2014). *F-35 Joint Strike Fighter: Problems Completing Software Testing May Hinder Delivery of Expected Warfighting Capabilities.* GAO.

Government Accountability Office. (2019). *Including Users Early and Often in Software Devlopment Could Benefit Programs.* GAO.

Government Accountability Office. (2020). *Agile Assessment Guide Best Practices for Agile Adoption & Implementation.* U.S. Government Accountability Office.

Goyal, S., & Bhatia, P. K. (2019). A Non-Linear Technique for Effective Software Effort Estimation using Multi-Layer Perceptrons. *International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (Com-IT-Con)* (pp. 1-4). Faridabad, India: IEEE.

Grimstad, S., & Jorgensen, M. (2007). Inconsistency of Expert Judgment-Based Estimates of Software Development Effort. *The Journal of Systems and Software 80*, 1770-1777.

Gwartney, J. D., Stroup, R. L., Sobel, R. S., & Macpherson, D. A. (2017). *Economics: Private and Public Choice.* Cengage Learning; 16th edition.

Hammad, M., & Alqaddoumi, A. (2018). Features-Level Software Effort Estimation Using Machine Learning Algorithms. *International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies* (pp. 1-3). Sakhier, Bahrain: IEEE.

Hartley, K. (2012). Conflict and Defence Output: An Economic Perspective. *Revue d'econmie politique, 122(2)*, 171-195.

Hartley, K., & Solomon, B. (2016). Special Issue: Defence Inflation. *Defence and Peace Economics, 27(2)*, 172-175.

Higgs, R., & Niskanen, W. A. (1990). *Arms, Politics, and the Economy: Historical and Contemporary Perspectives.* Independent Institute.

Hira, A., & Boehm, B. (2016). Combatting Use Case Points' Limitations with COCOMO(R) II and Relative Difficulty. *23rd Asia-Pacific Software Engineering Conference* (pp. 353-356). Hamilton, New Zealand: IEEE.

Hohl, P., Klunder, J., van Bennekum, A., Lockard, R., Gifford, J., Munch, J., et al. (2018). Back to the Future: Origins and Directions of the "Agile Manifesto" - Views of the Originators. *Jounral of Software Engineering Research and Development, 6(1)*, 1-27.

Holcombe, G. R. (2008). Why Does Government Produce National Defense? *Public Choice, 137(1/2)*, 11-19.

Hooi, T. C., Yusoff, Y., & Hassan, Z. (2008). Comparative Study on Applicability of WEBMO in Web Application Cost Estimation within Klang Valley in Malaysia. *IEEE 8th International Conference on Computer and Information Technology Workshops* (pp. 116-121). Sydney, Australia: IEEE.

Hove, K. H., & Lillekvelland, T. (2019). On Growing Operating Costs in the Armed Forces. *Defense and Peace Economics, 30(4)*, 438-453.

Idri, A., Hosni, M., & Abran, A. (2016). Improved estimation of software development effort using Classical and Fuzzy Analogy ensembles. *Applied Soft Computing 49*, 990-1019.

International Function Point Users Group. (2020). *About Functional Point Analysis*. Retrieved August 29, 2020, from International Function Point Users Group: https://www.ifpug.org/about-function-point-analysis/

Ionescu, V.-S. (2017). An approach to software development effort estimation using machine learning. *13th IEEE International Conference on Intelligent Computer Communication and Processing* (pp. 197-203). Cluj-Napoca, Romania: IEEE.

Iwata, K., Nakashima, T., Anan, Y., & Ishii, N. (2016). Effort Estimation for Embedded Software Development Projects by Combining Machine Learning with Classification. *4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence* (pp. 265-270). Las Vegas, USA: IEEE.

Jorgensen, M. (2002). A review of studies on expert estimation of software development effort. *Systems and Software*, 37-60.

Jorgensen, M. (2004). Top-Down and Bottom-Up Expert Estimation of Software Development Effort. *Information and Software Technology 46*, 3-16.

Jorgensen, M. (2005). Practical Guidelines for Expert-Judgment-Based Software Effort Estimation. *IEEE Software*, 57-63.

Kadir, N. F., Sarkan, H. B., Azmi, A. B., Yusop, O. B., & Karma, M. N. (2019). Specification of a Hybrid Effort Estimation System using UML. *6th International Conference on Research and Innovation in Information Systems* (pp. 1-7). Johor Bahru: IEEE.

Kang, S., Choi, O., & Baik, J. (2010). Model-based Dynamic Cost Estimation and Tracking Method for Agile Software Develpoment. *9th IEEE/ACIS International Conference on Computer and Information Science* (pp. 743-748). Yamagata, Japan: IEEE.

Karner, G. (1993). *Resource Estimation for Objectory Projects.* Objective Systems SF AB.

Kchaou, D., Bouassida, N., & Ben-Abdallah, H. (2015). Change Effort Estimation based on UML Diagrams Application in UCP and COCOMO-II. *10th International Joint Conference on Software Technologies* (pp. 1-8). Colmar, France: IEEE.

Keaveney, S., & Conboy, K. (2006). Cost Estimation in Agile Development Projects. *European Conference on Information Systems.* European Conference on Information Systems.

Kellner, D. (2014). Reliability for Operational Assets: A Recipe for Cost Avoidance. *Reliability and Maintainability Symposium* (pp. 1-6). Colorado Springs: IEEE.

Khan, M. S., Ul Hassan, A., Shah, M. A., & Shamim, A. (2018). Software Cost and Effort Estimation using a New Optimization Algorithm Inspired by Strawberry Plant. *24th International Conference on Automation and Computing (ICAC)* (pp. 1-6). Newcastle Upon Tyne, United Kingdom: IEEE.

Kocaguneli, E., Menzies, T., & Keung, J. W. (2012). On the Value of Ensemble Effort Estimation. *Transactions on Software Engineering*, 1403-1416.

Kocaguneli, E., Tosum, A., & Bener, A. (2010). AI-Based Models for Software Effort Estimation. *36th EUROMICRO Conference on Software Engineering and Advanced Applications* (pp. 323-326). Lille, France: IEEE.

Kok, P., Kitchenham, B. A., & Kirawkowski, J. (1990). The MERMAID Approach to software cost estimation. *Commission Of The European Communities Directorate-General Telecommunications, Information Industries And Innovation (eds) ESPRIT*, 296-314.

Kompella, L. (2013). Advancement of decision making in Agile Projects by Applying Logsitic Regression on Estimates. *8th International Conference on Global Software Engineering Workshops* (pp. 11-17). Bari, Italy: IEEE.

Lazear, E. P., & Rosen, S. (1981). Rank-Order Tournaments as Optimum Labor Contracts. *The Journal of Political Economy, 89(5)*, 841-864.

Lindvall, M. (2004). Agile Software Development in Large Organizations. *Computer*, 26-34.

Litoriya, R., Sharma, N., & Kothari, A. (2012). Incorporating Cost driver substitution to improve the Effort using Agile COCOMO II. *CSI Sixth International Conference on SOftware Engineering.* Indore, India: IEEE.

MacDonell, S. G., & Shepperd, M. (2003). Combining Techniques to Optimize Effort Predictions in Software Project Management. *The Journal of Systems and Software 66*, 91-98.

Machine learning methods and asymmetric cost function to estimate execution effort of software testing. (2010). *Third International Conference on Software Testing, Verification and Validation* (pp. 275-284). Campinas, Brazil: IEEE.

Mahmood, Y., Kama, N., & Azmi, A. (2019). A systematic review of studies on use case points and expert based estimation of software development effort. *Software: Evolution and Process*, 1-20.

Mahmood, Y., Kama, N., Azmi, A., & Ali, M. (2020). Improving Estimation Accuracy Prediction of Software Development Effort: A Proposed Ensemble Model. *The 2nd International Conference on Electrical, Communication and Computer Engineering (ICECCE)* (pp. 1-6). Istanbul, Turkey: IEEE.

Mahnic, V., & Hovelja, T. (2012). On Using PLanning Poker for Estimating User Stories. *The Journal of Systems and Software 85*, 2086-2095.

Maldonado, M. (2015). *Qualitative Case Study on F-35 Figher Production Delays Affecting National Security Guidance.* Minneapolis: Walden University.

Mann, K., & Hoang, R. (2020). *But Wait, There's More! Using SFPA for your Cost, Schedule, and Performance Needs.* Department of Homeland Security.

Manohar, H. M., & Aappaiah, S. (2017). Stabilization of FIFO system and Inventory Management. *International Research Journal of Engineering and Technology*, 5631-5634.

McConnell, S. (2006). *Software Estimation: Demystifying the Black Art.* Redmond, Washington: Microsoft Press.

McQuade, J. M., Murray, R., Louie, G., Medin, M., Pahlka, J., & Stephens, T. (2019). *Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage.* Defense Innovation Board.

Mendes, E., & Mosley, N. (2008). Bayesian Network Models for Web Effort Prediction: A Comparative Study. *IEEE Computer Society*, 723-737.

Mendes, E., Watson, I., Triggs, C., Mosley, N., & Counsell, S. (2002). A Comparison of Development Effort Estimation Techniques for Web Hypermedia Applications. *Eighth IEEE Symposium on Software Metrics.* IEEE Computer Society.

Mesroghli, S., Jorjani, E., & Chehreh Chelgani, S. (2009). Estimation of Gross Calorific value based on coal analysis using regression and artifical neural networks. *International Journal of Coal Geology*, 49-54.

Miceli, J. T. (2011). Free Riders, Holdouts, and Public Use: A Tale of Two Externalities. *Public Choice v. 148, n. 1/2*, 105.

Mises, L. v. (1958). *Liberty & Property.* Ludwig von Mises Institute.

Mises, L. v. (1979). *Economic Policy: Thoughts for Today and Tomorrow.* Chicago: Regnery Gateway.

MITRE. (2016). *Federal Aviation Administration Agile Acquisition Principles and Practices.* Federal Aviation Administration.

Mkpojiogu, E., & Hashim, N. (2016). Understanding the relationship between Kano model's customer satisfaction scores and self-stated requirements importance. *SpringerPlus*.

Modigliani, P., & Chang, S. (2014). *Defense Agile Acquisition Guide: Tailoring DoD IT Acquisition Program Structures and Processes to Rapidly Deliver Capabilities.* The MITRE Corporation.

Moharreri, K., Sapre, A. V., Ramanathan, J., & Ramnath, R. (2016). Cost-Effective Supervised Learning Models for Software Effort Estimation in Agile Environments. *40th Annual Computer Software and Applications Conference* (pp. 136-140). Atlanta, USA: IEEE.

Molokken-Ostvold, K., & Jorgensen, M. (2005). A Comparison of SOftware Project Overruns- Flexible Versus Sequential Development Models. *IEEE Transactions on Software Engineering, 31(9)*, 754-766.

Molokken-Ostvold, K., Haugen, N. C., & Benestad, H. C. (2008). Using planning poker for combining expert estimates in software projects. *The Journal of Systems and Software 81*, 2106-2117.

Monika, & Sangwan, O. P. (2017). Software Effort Estimation Using Machine Learning Techniques. *7th International Conference on Cloud Computing, Data Science & Engineering - Confluence* (pp. 92-98). Noida, India: IEEE.

MoSCoW Prioritization. (2020). Retrieved August 28, 2020, from ProductPlan: https://www.productplan.com/glossary/moscow-prioritization/

Nadgeri, S., Hulsure, V. P., & Gawande, A. D. (2010). Comparative Study of Various Regression Methods for Software Effort Estimation. *3rd International Conference on Emerging Trends in Engineering and Technology* (pp. 642-645). Goa, India: IEEE.

Nassif, A. B., Capretz, L. F., & Ho, D. (2012). Estimating Software Effort Using an ANN Model Based on Use Case Points. *11th International Conference on Machine Learning and Applications* (pp. 42-47). Boca Raton, Florida: IEEE.

Nassif, A. B., Capretz, L. F., Ho, D., & Azzeh, M. (2012). A Treeboost Model for Software Effort Estimation Based on Use Case Points. *11th International Conference on Machine Learning and Applications* (pp. 314-319). Boca Raton, Florida: IEEE.

Nathaneal, E. H., Hendradjaya, B., & Sunindyo, W. D. (2015). Study of Algorithmic Method and Model for Effort Estimation in Big Data Software Development Case Study: Geodatabase. *The 5th International Conference on Electrical Engineering and Informatics* (pp. 427-432). Bali, Indonesia: IEEE.

NCCA & AFCAA. (2008). *Software Development Cost Estimating Handbook.* Software Technology Support Center.

Nguyen, V., Deeds-Rubin, S., Tan, T., & Boehm, B. (2007). *A SLOC Counting Standard.* Los Angeles: University of Southern California.

Nolis, J. (2018, May 18). *You're Relying on Data Too Much*. Retrieved March 4, 2021, from Towards Data Science: https://towardsdatascience.com/youre-relying-on-data-too-much-250d4edc70c3

Owais, M., & Ramakishore, R. (2016). Effort, Duration and Cost Estimation in Agile Software Development. *Ninth International Conference on Contemporary Computing* (pp. 1-5). Noida, India: IEEE.

Perkins, J., & Long, James. (2020, January 17). *SOFTWARE WINS MODERN WARS: WHAT THE AIR FORCE LEARNED FROM DOING THE KESSEL RUN*. Retrieved August 20, 2020, from Modern War Institute: https://mwi.usma.edu/software-wins-modern-wars-air-force-learned-kessel-run/

Phan, V. P., Chau, N. P., & Nguyen, M. L. (2016). Exploiting Tree Structures for Classifying Programs by Functionalities. *Eighth International Conference on Knowledge and Systems Engineering* (pp. 85-90). Hanoi, Vietnam: IEEE.

Pinto, A., Liggan, M. E., Subowo, N. K., Goodwin, H. G., Sekhabal-Tafti, S., & Staley, A. M. (2016). *Agile Acquisition Principles and Practices.* Federal Aviation Administration.

Pinto, A., Liggan, M., Subowo, N., Goodwin, H., Skhavat-Tafti, S., & Staley, A. (2016). *Agile Acquisition Principles and Practices.* Federal Aviation Administration & MITRE.

Polkowski, Z., Vora, J., Tanwar, S., Tyagi, S., Singh, P. K., & Singh, Y. (2019). Machine Learning-based Software Effort Estimation: An Analysis. *11th International Conference on Electronics, Computers and Artificial Intelligence* (pp. 1-6). Pitesti, Romania: IEEE.

Popli, R., & Chauhan, N. (2014). Agile Estimation Using People and Project Related Factors. *International Conference on Computing for Sustainable Global Development* (pp. 564-569). New Delhi, India: IEEE.

Popli, R., & Chauhan, N. (2014). Cost and Effort Estimation in Agile Software Development. *International Conference on Reliability, Optimization and Information Technology* (pp. 57-61). Faridabad, India: IEEE.

Popli, R., & Chauhan, N. (2014). Estimation in Agile Environment using Resistance Factors. *International Conference on Information Systems and Computer Networks* (pp. 60-65). Mathura, India: IEEE.

Prabhakar, V., & Dutta, M. (2013). Prediction of Software Effort Using Artificial Neural Network and Support Vector Machine. *Computer Science*.

Pullen, C. (2019). Data Requirements for Agile Software Programs. *CADE Focus Group* (pp. 3-19). Air Force Cost Analysis Agency.

Randall, R. M. (2014). Agile at IBM: Software Developers Teach a New Dance Step to Management. *Strategy and Leadership, 42(2),* 26-29.

Regan, C., Lapham, M. A., Wrubel, E., Beck, S., & Bandor, M. (2014). *Agile Methods in Air Force Sustainment: Status and Outlook.* Software Engineering Institute.

Reinertsen, D. (2009). *Principles of Product Development Flow: Second generation Lean product development.* Redondo Beach: Celeritas.

Rosa, W., Madachy, R., Clark, B. K., & Boehm, B. W. (2020). Empirical Effort and Schedule Estimation Models for Agile Processes in the US DoD. *IEEE*, 1-13.

Rosa, W., Madachy, R., Clark, B., & Boehm, B. (2017). Early Phase Cost Models for Agile Software Processes in the US DoD. *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 30-37). Toronto, Canada: IEEE.

Saini, A., Ahuja, L., & Khatri, S. K. (2018). Effort Estimation of Agile Development using Fuzzy Logic. *7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)* (pp. 779-783). Noida, India: IEEE.

Satapathy, S. M., Acharya, B. P., & Rath, S. K. (2016). Early Stage software Effort Estimation Using Random Forest Technique Based on Use Case Points. *IET Software Vol: 10, Issue: 1*, 10-17.

Schweighofer, T., Kline, A., Pavlic, L., & Hericko, M. (2016). *How is Effort Estimatd in Agile Software Development Projects?* SQAMIA.

Sehra, S. K., Kaur, J., Brar, Y. S., & Kaur, N. (2014). Analysis of Data Mining Techniques for Software Effort Estimation. *11th International Conference on Information Technology: New Generations* (pp. 633-638). Las Vegas, Nevada: IEEE.

Servadei, L., Mosca, E., Zennaro, E., Devarajegowda, K., Werner, M., Ecker, W., et al. (2020). Accurate Cost Estimation of Memory Systems Utilizing Machine Learning and Solutions from Computer Vision for Design Automation. *Transactions on Computers Volume: 69, Issue: 6*, 856-867.

Shams, A., Bohm, S., Winzer, P., & Dorner, R. (2019). App Cost Estimation- Evaluating Agile Environments. *IEEE 21st Conference on Business Informatics* (pp. 383-390). Moscow, Russia: IEEE.

Sharma, H. K., Tomar, R., Dumka, A., & Aswal, M. S. (2015). OpenECOCOMO: The Algorithms and Implementaion of Extended Cost Costructive Model (E-COCOMO). *1st International Conference on Next Generation Computing Technologies* (pp. 773-778). Dehradun, India: IEEE.

Shin, M., & Goel, A. L. (2000). Empirical data modeling in software engineering using radial basis functions. *Transactions on Software Engineering Vol: 26, Issue: 6*, 567-576.

Shukla, S., & Kumar, S. (2019). Applicability of Neural Network based Models for Software Effort Estimation. *IEEE World Congress on Services* (pp. 339-342). Milan, Italy: IEEE.

Shukla, S., Kumar, S., & Ranjan Bal, P. (2019). Analyzing Effect of Ensemble Models on Multi-Layer Perceptron Network for Software Effort Estimation. *IEEE World Congress on Services* (pp. 386-387). Milan, Italy: IEEE.

Sikka, G., Kaur, A., & Uddin, M. (2010). Estimating Function points: Using Machine Learning and Regression Models. *2nd International Conference on Education Technology and Computer (ICETC)* (pp. 52-56). Shanghai, China: IEEE.

Silhavy, R., Silhavy, P., & Prokopova, Z. (Applied Least Square Regression in Use Case Estimation Precision Tuning). Applied Least Square Regression in Use Case Estimation Precision Tuning. *Software Engineering in Intelligent Systems. Advances in Intelligent Systems and Computing, vol 349*, 11-17.

Smith, A. E., & Mason, A. K. (1997). Cost Estimation Predictive Modeling: Regression versus Neural Network. *The Engineering Economist*, 137-161.

Snyder, D., Lim, J., Carrillo, M. J., & Hildebrandt, G. G. (2012). *Improving Air Force Depot Programming by Linking Resources to Capabilities.* RAND Corporation.

Stillion, J., & Orletsky, D. T. (1999). *Airbase Vulnerability to Conventional Cruise-Missile and Ballistic Attacks.* Rand Corporation.

Suherman, I. C., Sarno, R., & Sholiq. (2020). Implementation of Random Forest Regression for COCOMO II Effort Estimation. *International Seminar on Application for Technology of Information and Communication (iSemantic)* (pp. 476-481). Semarang, Indonesia: IEEE.

Tate, D. (2017). *Software Productivity Trends and Issues.* Alexandria: Institute for Defense Analyses.

Toka, D., & Tretken, O. (2013). Accuracy of Contemporary Parametric Software Estimation Models: A Comparative Analysis. *39th Euromicro Conference Series on Software Engineering and Advanced Applications* (pp. 313-316). Santander, Spain: IEEE.

Tsunoda, M., Monden, A., Keung, J., & Matsumoto, K. (2012). Incorporating Expert Judgment into Regression Models of Software Effort Estimation. *19th Asia-Pacific Software Engineering Conference* (pp. 374-379). Hong Kong, China: IEEE.

U.S. Air Force. (2020). *Kessel Run News*. Retrieved July 18, 2020, from Kessel Run: https://kesselrun.af.mil/news/

U.S. Air Force Biographies. (2019, September). *US Air Force Biographies*. Retrieved May 25, 2020, from U.S. Air Force: https://www.af.mil/About-Us/Biographies/Display/Article/1926281/nicolas-m-chaillan/

Usman, M., Petersen, K., Borstler, J., & Neto, P. S. (2018). Developing and using checklists to improve software effort estimation: A multi-case study. *The Journal of Systems and Software 146*, 286-309.

Valdes-Souto, F. (2016). Creating a Historical Database for Estimation Using the EPCU Approximation Approach for COSMIC (ISO 19761). *4th International Conference in Software Engineering Research and Innovation* (pp. 159-166). Puebla, Mexico: IEEE.

Visual Paradigm. (2020). *What is Story Point in Agile? How to Estimate a User Story?* Retrieved October 22, 2020, from Visual Paradigm: https://www.visual-paradigm.com/scrum/what-is-story-point-in-agile/

Vyas, S., Jain, S., Choudhary, I., & Chaudhary, A. (2019). Study on Use of AI and Big Data for Commercial System. *Amity International Conference on Artificial Intelligence* (pp. 737-739). Dubai, United Arab Emirates: IEEE.

Wahid, A., & Masud, P. (2013). Efficiency Factor and Risk Factor B ased User Case Point Test Effort Estimation Model Compatible with Agile Software Development. *International Conference on Information Technology and Electrical Engineering* (pp. 113-118). Yogyakarta, Indonesia: IEEE.

Wiegers, K. E. (2000). Stop Promising Miracles. *Software Development*.

Wright, I., & Ziegler, A. (2019). The standard coder: a machine learning approach to measuring the Effort Required to Produce Source Code Change. *IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering* (pp. 1-7). Montreal, Canada: IEEE.

Yazdani-Chamzini, A., Zavadskas, E. K., Antucheviciene, J., & Bausys, R. (2017). A Model for Shovel Capital Cost Estimation, Using a Hybrid Model of Multivariate Regression and Neural Networks. *Symmetry Volume: 9*, 1-14.

Zakrani, A., Najm, A., & Marzak, A. (2018). Support Vector Regression Based on Grid-Search Method For Agile Software Effort Prediction. *IEEE 5th International Congress on Information Science and Technology* (pp. 492-497). Marrakech, Morocco: IEEE.

Zaydi, M., & Nasserddine, B. (2020). DevSecOps Practices for an Agile and Secure IT Service Management. *Journal of Management Information & Decision Sciences, 23(2)*, 1-16.

Zhang, Y., Fuh, J., & Chan, W. (1996). Feature-based cost estimation for packaging products using neural networks. *Computers in Industry*, 95-113.

Ziauddin, Tipu, S. K., & Zia, S. (2012). *An Effort Estimation Model for Agile Software Development.* World Science.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
25-03-2021	Master's Thesis	Sept 2019 – March 2021

4. TITLE AND SUBTITLE
Developing a Cost of Delay (CoD) Framework for the DoD & Analyzing the Current State of Air Force Agile Cost Estimation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Goljan, James, 1st Lt

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENV-MS-21-M-231

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Agency: Norwegian Defence Research Establishment
Addres Norwegian Defence Research Establishment (FFI) PB 25, 2027 Kieller
Phone and Email 63 80 77 82 Helene.Berg@ff.no
ATTN: POC
Helene Berg

10. SPONSOR/MONITOR'S ACRONYM(S)
NATO

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

12. DISTRIBUTION/AVAILABILITY STATEMENT
A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The study has two research objectives. The first research objective develops a proof of concept for a Cost of Delay (CoD) framework to evaluate the Air Force's Kessel Run Software Factory. The CoD for a given requirement is defined as the value that could be produced over a length of time if said requirement was available immediately (Arnold & Yuce, 2013). CoD provides a means to *justify* and *prioritize* competing interests regarding budgetary, manpower, and overall strategic decisions.

DoD organizations can employ a cost minimization framework to conduct CoD Analysis. More specifically, this study's CoD Analysis discovers approximately $16.54M annually in CoD associated with failing to modernize the legacy Theater Battle Management Core System. Additionally, the research identifies four high value requirements for reprioritization within the Kessel Run All Domain Operations Suite (KRADOS) portfolio which would save over one thousand dollars per week in opportunity costs. The proof of concept demonstrates the potential for large savings.

The second objective compares the predominant literature on Agile cost estimating techniques against the methods utilized within 11 Air Force Agile Software Factories. The study identifies the 11 Software Factories are in agreement with the extant literature in moving away from SLOC as the means to assess cost. Additionally, statistical analysis demonstrates disparities in cost estimating strategies between the literature and the Agile Air Force organizations. The results demonstrate Software Factories eschew trends in the literature towards more complicated cost models that incorporate machine learning with Data-Based techniques in lieu of simpler models that utilize Engineering Build-up Capacity Based techniques.

15. SUBJECT TERMS
Agile, Cost Estimation, Opportunity Cost, Cost of Delay, Prioritization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Daniel Ritschel, AFIT/ENV
U	U	U	UU	123	**19b. TELEPHONE NUMBER** *(Include area code)* 937-255-3636 X4484