Software Estimating in an Agile Environment

Captain James Goljan

Jonathan D. Ritschel, Ph.D.

Scott Drylie, Ph.D.

Edward D. White, Ph.D.

Introduction

Defense organizations are moving towards agile methodologies as a preferred approach to software development. The desire to implement agile methods is discussed in the 2019 Defense Innovation Board (DIB) report, which identifies speed and cycle time as the most important metrics for software development (McQuade et al., 2019). This movement toward agile methodologies provides a conundrum for defense cost analysts. These cost analysts are proficient in developing software estimates based on commonly accepted defense sizing metrics such as Source Lines of Code (SLOC). But the agile environment is unique. The agile mentality relies on flexibility and working in small iterations. Utilizing metrics like SLOC are often 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 potentially 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, Madachy, Clark, & Boehm, 2020). Due to the DoD's lack of experience and familiarity with agile, the objective of this article is to investigate the current state of agile software cost estimation and provide recommendations for cost analysts.

The DoD has only recently implemented agile software development, but the agile concept itself dates back to 2001 with the publication of the Agile Manifesto (Regan, Lapham, Wrubel, Beck, & Bandor, 2014). Since its inception, agile practices have become widely adopted throughout private industry (Randall, 2014). The private sector's 20 years of experience provides an opportunity to uncover best practices for cost analysts in an agile environment. To study this, we first conduct an extensive literature review regarding the *recommended* agile software cost estimating models and techniques. The question then becomes, "how do the recommended techniques align with the methods defense cost analysts are currently using?" To answer this, we collect data from 11 agile Air Force software factories to determine what practitioners *actually* do. Comparison of the two results will provide defense analysts with insight on differences between current DoD practices and those advocated by the published literature.

The Importance of Software in Military Systems

Software plays a critical role in military systems. The Defense Innovation Board (2019) states that 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). To address this need for rapid deployment of valuable software capabilities, agile software factories such as Kessel Run were initiated. These software factories were designed to move the DoD away from traditional development approaches such as Waterfall or Spiral and towards the more modern agile approach.

The Agile Advantage

There is a debate regarding the merits of agile in comparison to traditional software development approaches. That discussion is outside the scope of this article. Rather, the DoD's shift to agile (rightly or wrongly) necessitates a basic understanding of the potential advantages of an agile approach. What are those advantages? The agile software development method has advantages along three dimensions: 1) ability to rapidly adjust to the immediate needs of the customer 2) delivers viable products sooner and 3) provides more cost effective programs.

The first Agile advantage is that it provides an environment for the customer to communicate constructive feedback to the 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, 2020).

The second Agile advantage is reduced cycle time. Agile methodologies have already been adopted and successfully proven to reduce 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. While this potential benefit of Agile is debatable, there is evidence for the claim. 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). Additionally, 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).

These benefits have resulted in Agile development becoming the leading methodology that private industries use to create software. The commercial successes suggest the DoD may similarly benefit by employing Agile. However, the nascent Agile implementation in defense programs means research into the impacts to defense cost estimation is scarce. The defense cost analyst is left with uncertainty regarding the best techniques and methods for an Agile environment.

Data and Methods

To inform the discussion on how agile cost estimation can be improved in the defense arena, we compare the predominant published literature on recommended agile cost estimation methods to current practices in Air Force Software Factories. This approach necessitated two data sets be collected and analyzed. The first data set comes from a robust literature search that resulted in 1,814 published articles being examined. The second data set comes from practitioner responses at 11 Air Software Factories that are currently employing Agile techniques. Details of each follows:

Published Literature Data

To identify the relevant literature, a four phased search strategy was employed. The first phase involves searching for all articles generated from search strings in two major databases: IEEE Xplore and Science Direct. The primary search string consists of:

"Software Effort Estimation" <AND> "Cost"

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

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

The second phase of the search strategy eliminated 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. Table 1 outlines the inclusion/ exclusion criteria. 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.

Inclusion Criteria	Exclusion Criteria
Provides analysis or recommendation of the techniques, models, & approaches used in Agile software estimation	Not related to Agile based environments
Published in peer-reviewed journal articles or conference proceedings	Simply defines or explains the type of software estimation techniques
Published DoD report	
Published in the last 20 years (2000 or later)	

Table 1: Article Inclusion/Exclusion Criteria

It is important to note that simply defining or listing a cost estimating technique resulted in that paper being excluded from the final dataset. Our interest is to discern those models or techniques that are being supported or advocated for by the authors. Including papers that simply define or list a technique would artificially inflate the advocacy for the cost estimating method. Figure 1 outlines the four phased search approach and the number of articles remaining after the application of inclusion/exclusion factors.

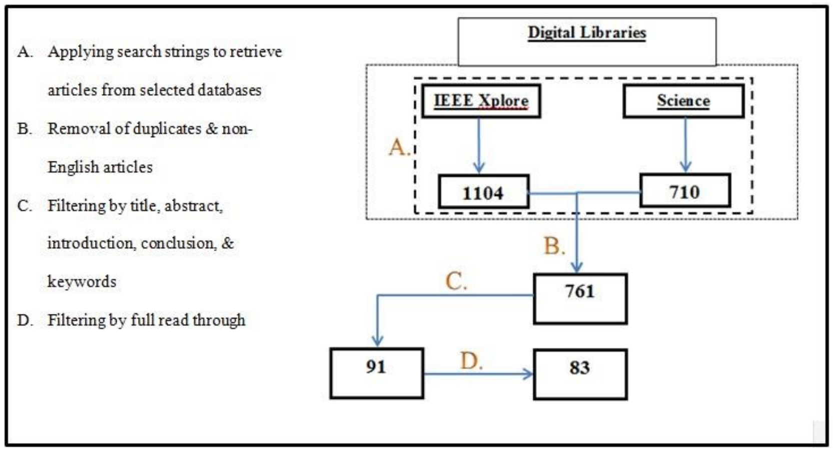
A. Applying search strings to retrieve articles from selected databases
B. Removal of duplicates & non-English articles
C. Filtering by title, abstract, introduction, conclusion, & keywords
D. Filtering by full read through
Digital Libraries
IEEE Xplore
Science
A.
1104
710
B.
C.
761
91
D.
83

Figure 1. Article Search and Filter Process

Ultimately, from the original 1814 search hits, 83 articles are selected for the analysis of this study (see Appendix A for the full article list). The articles chosen support, advocate, or defend the usage of specific practices when conducting cost estimation in a software environment. Seventy-six of the articles are from an industry perspective while seven relate to the manner in which the DoD advises or conducts its cost estimation. The 83 articles provide insight into the currently recommended Agile cost estimation best practices. This information serves as a reference point for the Air Force specific data collected in the second data set.

<u>Air Force Practitioner Data</u>

Practitioner data is obtained from a data call of Agile Air Force Software Factories. This information 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. Eleven 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 will be compared to the sources compiled from the published literature. A direct statistical comparison of certain metrics and techniques will be accomplished using Clopper Pearson binomial confidence intervals. The comparison of the two data sets will provide insight into how the military is conducting its software effort and cost estimation compared to the current literature.

Results

We first examine the results from the published literature. Agile cost estimation methods in industry today can be categorized into three major styles: Algorithmic, Non-Algorithmic, and Data-Based (see Table 2). 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, 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

Technique Style	Techniques
Algorithmic	COCOMO-II
Algorithmic	SLIM
Algorithmic	SEER-SEM
Algorithmic	Parametric Models
Algorithmic	Regression Models
Non-	Expert Judgment
Non-	Planning Poker/disaggregation
Non-	Wideband Delphi
Data-Based	Neural Networks
Data-Based	Regression Using Unsupervised
Data-Based	Fuzzy Models
Data-Based	Genetic Algorithms
Data-Based	Case Based Analogy
Data-Based	Bayesian Networks

Table 2: Technique Style and Techniques

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).

The 83 sources from the literature review are mapped to the various techniques (see Appendix B). The table in Appendix B uses a number system that references the 83 specific articles provided in the *Selected Cost Estimation Techniques Work Cited* of Appendix A. Note that a variety of 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. We track both the number of references and the number of sources for the analysis. All citations in the Appendix B table are listed chronologically according to their respective date of publication.

Figure 2 summarizes the data from Appendix B. 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, the table within Figure 2 aggregates the data by the three Technique Styles. 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.

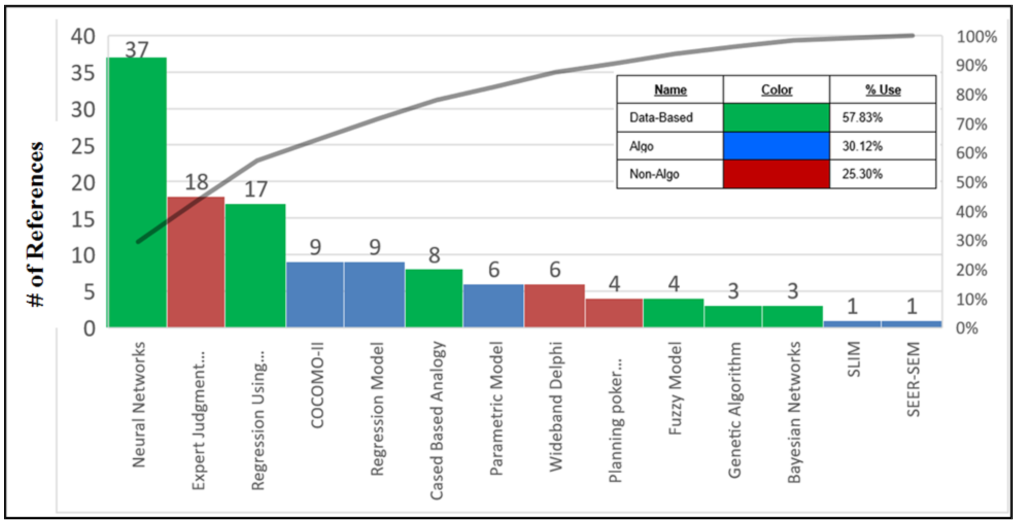

Figure 2. References to Software Effort Estimation Techniques

Sizing Metric	Statistics of Usage	Cited Literature
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 Points	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 Points	8.43%	28, 70, 13, 52, 77, 16, 39

Table 3: Software Size Metrics

The sizing metric used is also an important consideration for cost analysts. The authors are agnostic regarding the best sizing metric. However, many defense cost analysts have strong opinions (for and against) regarding sizing metrics such as SLOC. Therefore, we examine the various sizing metrics identified in the peer-reviewed literature (see Table 3).

The most obvious conclusion from Table 3 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. There are a total of 37 articles that did reference size (note that articles 16, 20, and 52 discuss more than one size metric). Of these articles, Use Case Points appears to be the most commonly referenced sizing metric at 15.66%; however, according to Table 3, 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%.

Figure 3 illustrates the references to technique styles when accounting for the articles that additionally identify the sizing metric utilized. Recall that articles that *only* use a size metric to build their model are not mapped to one of the three technique styles: Algorithmic, Non-

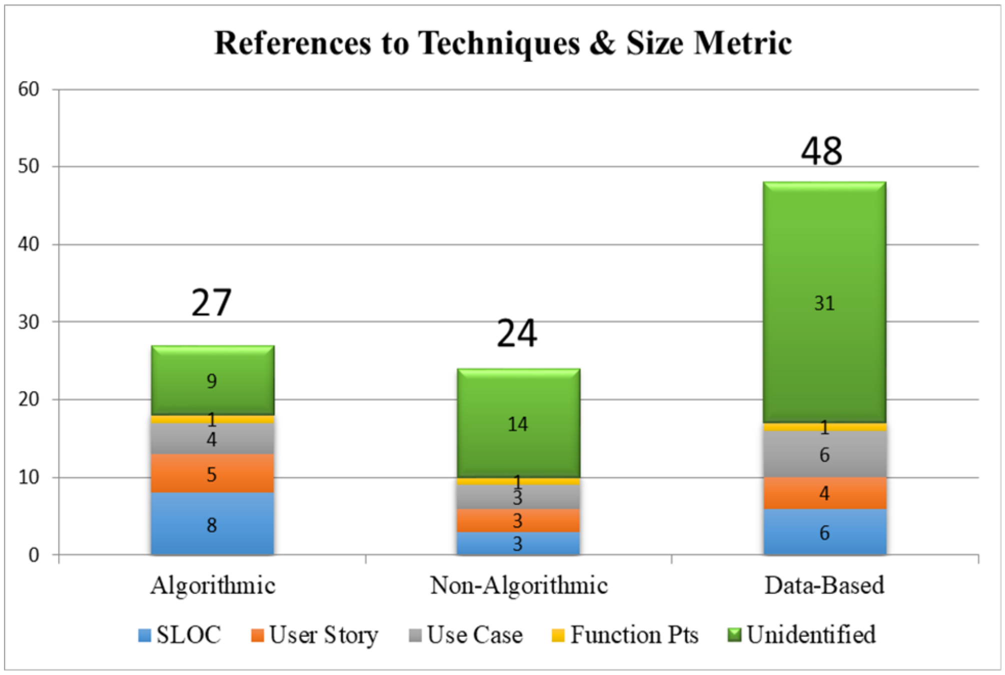

Figure 3: Number of References to Technique Styles Accounting for Size Metrics

Algorithmic, and Data-Based. Using Function Points as an example, Table 3 shows seven instances of the Function Point metric in the literature. But Function Points only appear three times in Figure 3 as being associated with one of the technique styles. Another important consideration in Figure 3 is that an article may discuss multiple techniques. Using SLOC as an example, Table 3 shows 11 instances of the SLOC metric in the literature. Figure 3, however, shows 17 instances of SLOC associated with a technique style. The reason is that six articles (1, 34, 20, 30, 27, 73) include multiple techniques with the SLOC sizing metric.

Further analysis of the published literature reveals a number of sources describing the viability of a hybrid or ensemble model which incorporates multiple techniques into the creation of a new multifaceted one. This is one reason articles appear in the previous tables as repeated references. Table 4 shows 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. However, not all articles that mention multiple techniques are advocating for a hybrid model. 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. Lastly, the largest category comprising 60.24% of the data set only makes use of one technique.

There are seven sources found in the literature regarding DoD policy and doctrine on Agile software cost estimation. Examining these sources separately is important given that we will be comparing the literature to current defense practitioner practices. Due to the limited information, Figure 4 captures the DoD techniques and estimating sizing metrics specified 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 discussion on SLOC (16, 63) as a viable sizing metric as well as the reliance on expert judgment (16, 45) to construct estimates.

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

Table 4. References to Hybrid/Ensemble Methods

Software Factory Results

This section provides results from the data collection of the 11 Agile Air Force Software Factories. We defined a Software Factory as any software development team striving to apply Agile principles to their processes as they support DoD systems. The Software Factories provided either the name of their organization or the specific program they are working on (see Appendix C

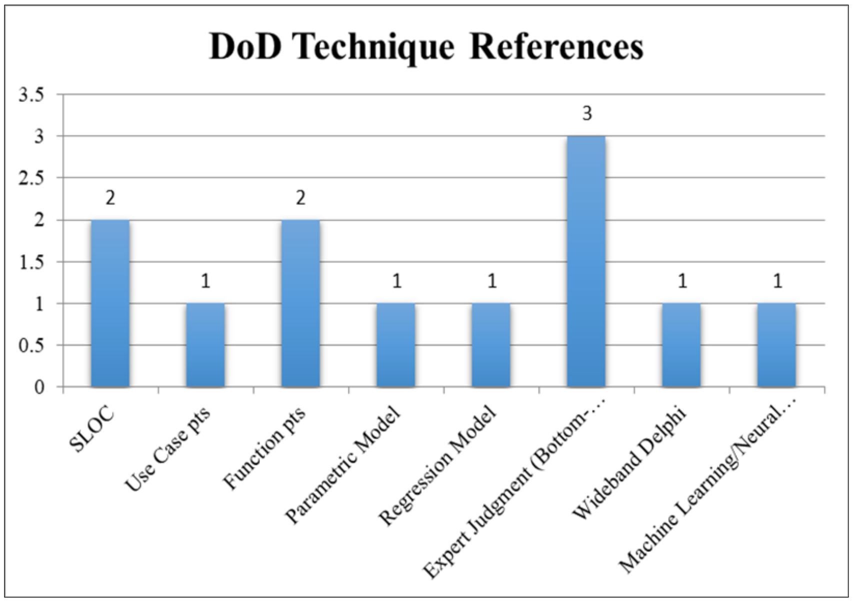

Figure 4. DoD Article References to Techniques

for the full list). 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.

The data call from the Software Factories closely mirrored the sizing metrics and technique categories determined in the literature review; however, there are some differences. The software factory data covers three main technique styles: Algorithmic, Non-Algorithmic, and Engineering Build-up. In contrast, the three main styles from the literature are Algorithmic, Non-Algorithmic, and Data-Based. There are no references to Data-Based techniques in any Software Factory response, so this technique style is effectively eliminated from the data. Instead, Engineering Build-up represents a new categorization of cost estimation for this data set. Also, the Algorithmic technique style has a change to its composition. For the software factory data, 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. Lastly, the Software Factories elaborate on the use of Capacity Based and Analogy estimation which are techniques not previously defined or explored in the published literature data.

Figure 5 depicts the techniques used by the Software Factories. The Non-Algorithmic category is the largest with usage by 9 of the 11 Factories. The dominant Non-Algorithmic techniques are planning poker in nine Factories (3, 4, 5, 6, 7, 8, 9, 10, & 11) and subject matter expert in seven Factories (1, 3, 4, 5, 6, 7, & 10).

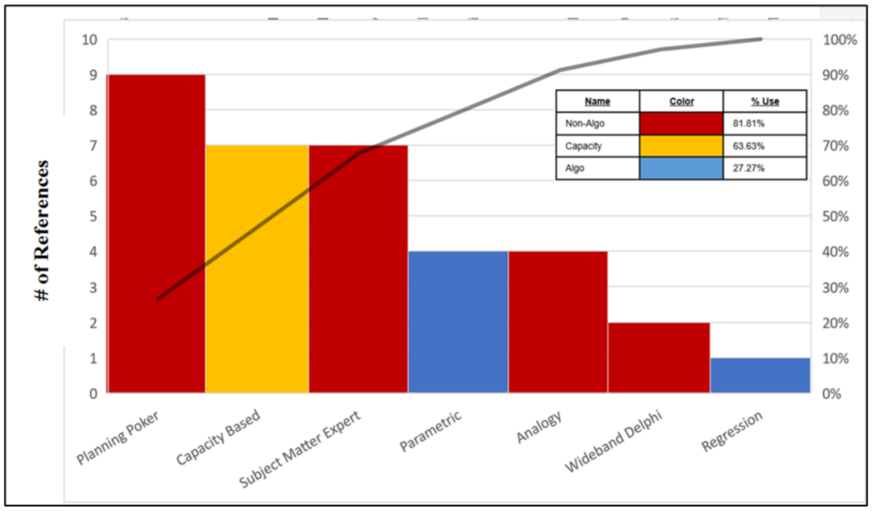

Figure 5. Software Factory References to Techniques

In addition to Non-Algorithmic, Factories also expressed a preference for Capacity Based estimates. Capacity based estimating, which falls under the new Engineering Build-up category, is used by 7 of 11 Factories (2, 3, 4, 5, 7, 8, & 10). Capacity based estimating examines contract elements to individually assess the number of full -time employees required to satisfy the requirement. 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 technique. 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 agile cost estimation at the Software Factories.

Figure 5 also shows that Factories identified the usage of Algorithmic style techniques. While there are four Factory references (4, 6, & 10) to Parametric techniques, these references include caveats. More specifically, three 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. These results directly contrast the literature results which had 9 of the 83 sources touting the use of the COCOMO-II model. Overall, these results highlight a predominant presence and preference towards Non-Algorithmic technique styles.

In addition to techniques, we are also interested in the sizing metrics used by the Software Factories. The data (see Table 5) 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 four Factories (6, 7, 10, & 11) utilize Use Case Points. The data additionally highlights the fact that only two Factories (6 & 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 not generally considered a viable metric at the Software Factories.

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

Table 5: Software Factory Sizing Metrics

In summary, the results from the software factories present three major findings. First, Non-Algorithmic techniques are prevalent in almost the entirety of Software Factory responses 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 the literature. Third, almost all Factories reject SLOC as a metric due to accuracy concerns in the Agile environment.

Comparison of Literature and Factory Data

There are three main conclusions derived from comparing the literature with the practitioner data. 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 the Software Factory data. While perhaps surprising given the large quantity of Data-Based solutions in the literature, the results can be explained by a number of reasons. The Air Force Agile Software Factories have only been 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, the published literature shows the techniques that academics are perpetuating as the most preferred methodologies. It is worth noting, 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 is intuitive that a less complicated and more easily adoptable cost model could provide Factories with a superior solution in the meantime.

Second, The sizing metric, and in particular SLOC, is a flashpoint for software estimators. 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, many 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. The literature only has 11 out of 83 references to SLOC as a metric while the Software Factories had 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. Figure 6 displays the two confidence intervals overlaid on the same graph, with the interval for the literature on the bottom in red and the interval for the Software Factories 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.

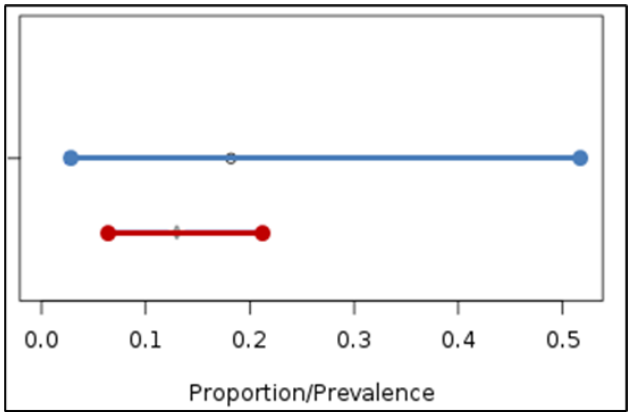

Figure 6. SLOC Usage

Furthermore, there are two major caveats to the 11 references to SLOC in the literature. First, there is correlation between SLOC and the COCOMO-II model. The COCOMO-II model is known to work primarily with SLOC based inputs. Six of the 11 sources (34, 66, 20, 30, 27, & 73) in the literature that reference SLOC additionally recommend the COCOMO-II model. By contrast, none of the Factories use the COCOMO-II model. Therefore, it is not surprising to see a lack of support for both SLOC and the COCOMO-II model in the software factories. 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 five references to SLOC in the literature. Second, two of those remaining five 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 three 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 the Software Factory data, coincide with the majority of the published literature sources which also do not incorporate SLOC into their cost estimation models. The low proportions in both data sets show the low prevalence of SLOC in Agile.

Third, the Software Factories shows a far greater reliance on Non-Algorithmic models in comparison to the published literature. 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 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. Figure 7 displays the two confidence intervals overlaid on the same image, with the interval for the published literature on the bottom in red and the interval for Data the Software Factories on the top in blue. When comparing the confidence intervals, because there is not an overlap this results in the rejection of the null hypothesis. Therefore, the conclusion is that there is a significant difference between the ways each data set uses Non-Algorithmic styles.

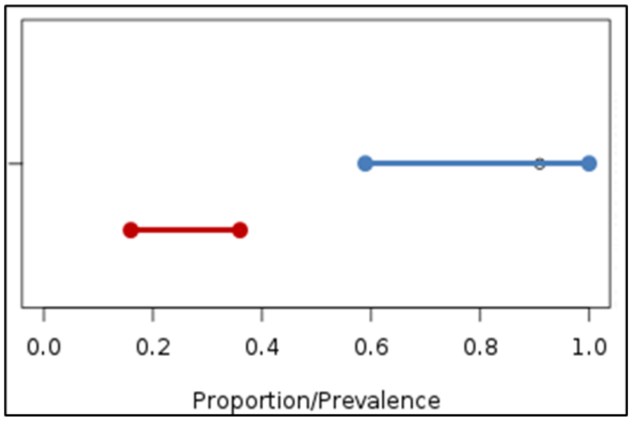

Figure 7. Non-Algorithmic Comparison

Conclusion

The purpose of this article was to identify the differences or commonalities between the recommended published literature on agile software cost estimating in comparison to current practices in the DoD. That comparison illuminated 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 the literature and does not utilize SLOC as a preferred 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.

The finding regarding Data-Based models prevalence in the literature merits further discussion. Recall that data-based models include things such as neural networks or machine learning. These techniques became popular in recent years in many other fields, and as such, their prevalence in the Agile estimating literature may be an artifact of this larger trend. Additionally, it is important to note that many of these models are "black boxes" which mask the relationship between input and output variables. In other words, there may be legitimate concerns in adopting this type of methodology. Regardless, the prudent approach would be for future research to investigate the merits of these models in a DoD environment.

It is an exciting time to be a cost analyst. The adoption of agile software development in the DoD is necessitating new ways of thinking about software cost estimation. Understanding the recommended methods in comparison to current practices is a key step to illuminating a future path where the best possible estimating methods are employed.

Appendix A: 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 Conforence 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.

Appendix B: 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

Appendix C: 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

References:

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.

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/

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

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

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.

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

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.

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/

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.

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.

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.

*Captain **James Goljan**, is a cost analyst at the Space Systems Command, Los Angeles AFB, CA. He holds a BS in Operations Research from the United States Air Force Academy and a MS in Cost Analysis from the Air Force Institute of Technology (AFIT). His primary research interests include agile software development, optimization models, and cost analysis. (Email address: James.Goljan.1@spaceforce.mil)*

*Dr. **Jonathan D. Ritschel** is an Associate Professor of Cost Analysis in the Department of Systems Engineering and Management at AFIT. He received his BBA in Accountancy from the University of Notre Dame, his MS in Cost Analysis from AFIT, and his Ph.D. in Economics from George Mason University. Dr. Ritschel's research interests include public choice, cost analysis, and economic institutional analysis. (E-mail address: Jonathan.Ritschel@aft.edu)*

***Scott Drylie**, Ph.D., is an Assistant Professor in the Department of Systems Engineering and Management at AFIT. He holds a BS in Economics from Montana State University, a M.Ed in Education from University of Nevada, a MS in Cost Analysis from AFIT, and a Ph.D. in Economics from George Mason University. Lt Col Drylie's research interests include Smithian political economy, organizational behavior, public choice, and cost analysis (E-mail address: Scott.Drylie@afit.edu)*

*Dr. **Edward D. White** is a Professor of Statistics in the Department of Mathematics and Statistics at AFIT. He received his BS in Mathematics from the University of Tampa, MAS from The Ohio State University, and Ph.D. in Statistics from Texas A&M University. His primary research interests include statistical modeling, simulation, and data analytics. (E-mail address: Edward.White@aft.edu)*

ICEAA

The International Cost Estimating and Analysis Association is a 501(c)(6) international non-profit organization dedicated to advancing, encouraging, promoting and enhancing the profession of cost estimating and analysis, through the use of parametrics and other data-driven techniques.

www.iceaaonline.com

Submissions:

Prior to writing or sending your manuscripts to us, please reference the JCAP submission guidelines found at

www.iceaaonline.com/publications/jcap-submission

Kindly send your submissions and/or any correspondence to JCAP.Editor@gmail.com

International Cost Estimating & Analysis Association

4115 Annandale Road, Suite 306 | Annandale, VA 22003

703-642-3090 | iceaa@iceaaonline.org