Volume 10 | Issue 2 | April 2022

ISSN: 2160-4746

Journal of Cost Analysis and Parametrics

ICEAA
International Cost Estimating & Analysis Association

www.iceaaonline.com

ICEAA

Journal of Cost Analysis and Parametrics

Volume 10 | Issue 2 | April 2022 ISSN: 2160-4746

Editor's Note

David L. Peeler, Jr., CCEA

Welcome back to our relaunched *Journal of Cost Analysis and Parametrics* (JCAP). We were delighted at the response to the journal's resurrection late last year. In keeping with our plan to publish discerning work of significance that speaks to ICEAA members, as well as advances intellectual pursuits within the cost arena *writ large*, we have complied another set of interesting articles. This issue contains three best track papers from our cancelled 2020 Workshop, three student papers, and a pre-*hiatus*, legacy submission with continued relevance.

Your reading pleasure starts with *Modeling Evolutionary Architectural Growth in Major Defense Acquisition Programs*, where **Matthew Dabkowski**, *et al*, examine the assumptions underlying an algorithm to estimate the cost of unanticipated growth. Applying the algorithm to 24 defense programs, the authors found valid application but also the need for additional, alternative and connection mechanisms. Ultimately, they propose a modified, data-driven approach using number theory, network science, simulation, and statistical analysis.

Next, the reader will find a paper based on the 2021 ICEAA Outstanding Air Force Institute of Technology Thesis Award. In this paper, Captain **James Goljan** explores *Software Estimating in an Agile Environment*. He compares the literature around software cost estimation techniques with actual practice at the 11 Air Force software factories. His results illuminate both commonalities and differences between the theory and practice of current software cost estimating approaches.

Three papers that were submitted for the ultimately cancelled 2020 ICEAA Workshop were submitted for review and are contained in the heart of this issue. In *Augustine's Law: Are We Really Headed for the $800 Billion-Dollar Fighter?* **Brent Johnstone** examines the trend in U.S. fighter costs and relates them to generational changes in aircraft design and manufacture. He also examines more recent jet fighter costs to see if Augustine's Law is really unfolding, *to wit* "In the year 2054, the entire defense budget will purchase just one aircraft." **Christian B. Smart**, *et al*, provide an interesting discussion on *Leveraging the Wisdom of Crowds with Modern Regression, Machine Learning, and Ensembles with Application to Army Software Sustainment*. The paper discusses ways to correct weaknesses in the log-transformed ordinary least squares method for development of cost estimating relationships, using modern regression methods, applying machine learning techniques, and combining multiple models. **Katherine Mann** and **Ryan Hoang** address the challenge of early lifecycle software cost estimation in *But Wait, There's More! Using SiSE for your Cost, Schedule & Performance Needs*. Using real examples, Mann and Hoang demonstrate the Simple Software Estimation (SiSE) methodology for tying high level requirements to a standard sizing metric.

We include another very interesting submission from the Air Force Institute of Technology, *Empirical Investigation of Engineering Change*

Order Percentages in Defense Contacts, in which Captain **Kaiana M. Miller**, *et al*, explore the generally accepted rules of thumb associated with engineering change orders (ECO) vis-à-vis contract cost growth. The long recognized 10% for development and 5% for production management reserve rules of thumb serve as the basis for empirical analysis using over a thousand contracts. The authors provide cogent analyses and recommend informed alternatives for future management reserve decisions.

The final article in this JCAP edition is the first of two papers which constitute the principal conclusions of **Danny Polidi**'s doctoral dissertation. The second article will appear in our October issue. The first installment, *Foundation of Structured Architecture, System & Cost Modeling*, describes generalized block diagram use to create a work breakdown structure for both a system model and a cost model. The author introduces cost model options offering direct input into design for system optimization.

We trust you will find something of interest and possible use within these seven articles. Again, we are happy to have the publication restarted. We hope you enjoy the material and find productive ways to apply it in either your professional efforts or personal interests. Thank you for your continued supported and please keep the manuscripts coming.

David Peeler
JCAP Editor

Modeling Evolutionary Architectural Growth in Major Defense Acquisition Programs

Matthew Dabkowski
Arthur Middlebrooks
Ricardo Valerdi

Abstract: One of the critical artifacts in the system design process is the architecture. Represented in a variety of ways, architectures evolve over time as user needs evolve and design limits are reached. The emergent requirement to submit Department of Defense Architecture Framework (DoDAF) models prior to Milestone A has created opportunities for the early life cycle cost estimation of major defense acquisition programs. Although this development could be seen as positive, any cost estimation procedure ultimately needs to be informed and validated by real-world data. In this paper, we examine the assumptions underlying one such method – an algorithm to estimate the cost of unanticipated, evolutionary architectural growth via the SV-3 (or Systems-Systems Matrix) and the Constructive Systems Engineering Cost Model. Specifically, using SV-3s from 24 different defense programs, we found that while the data generally comport with the method, alternative growth and connection mechanisms are needed. As such, we propose a modified, data-driven approach using number theory, network science, simulation, and statistical analysis – one which not only improves the algorithm's fidelity but also reinforces the value of viewing DoDAF models as computational objects.

Introduction and Context

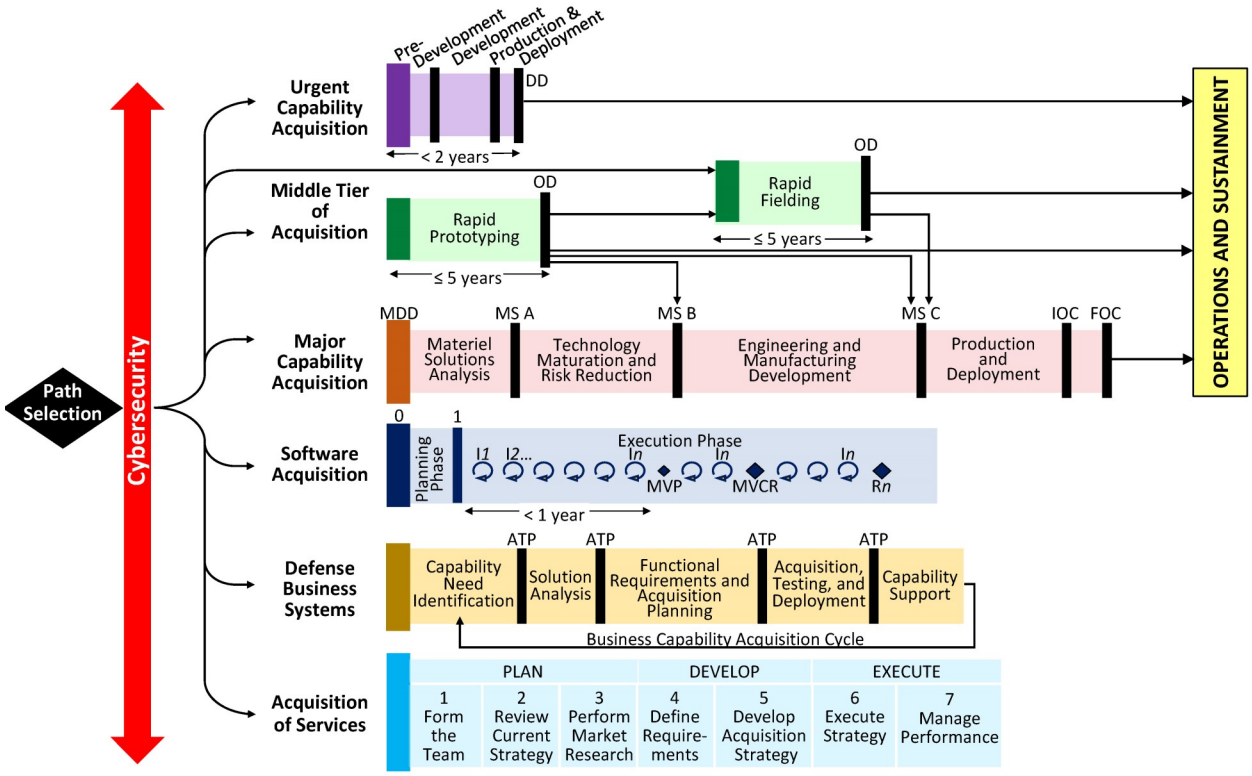
The Budget Control Act (BCA) initiated substantial budget cuts to reduce spending within the federal government (2011). Inside the Department of Defense (DoD), sequestration – the unofficial title given to the BCA – drastically reduced spending for military operations and defense acquisitions. In the years that followed, various pieces of legislation were passed to modernize and streamline the defense acquisition process to facilitate the reduction in defense spending. Much of this legislation focused on increasing the useable lifecycle of defense systems and reducing development costs; however, these changes alone were not substantial enough to create meaningful and lasting change. To address this gap and other modernization efforts, the National Defense Strategy (NDS) was updated for the first time in over ten years.

As a strategic document, the NDS "provides a clear road map for the [DoD] to meet the challenges posed by a re-emergence of long-term strategic competition with China and Russia" (DoD, n.d.). To this end, the NDS outlines several strategic objectives, including "sustaining Joint Force military advantages, both globally and in key regions, . . . defending allies from military aggression and bolstering partners against coercion, fairly sharing responsibilities for common defense; . . . [and] continuously delivering performance with affordability and speed" (DoD, 2018, p. 4). With respect to the equipment service members require to execute their global mission, the latter objective cannot be achieved without changes to how the DoD approaches the design, development, and

acquisition of military technology. In a budget-constrained environment, this imperative becomes increasingly more important.

Notably, even prior to the initiation of sequestration, researchers found that most of a program's life cycle costs are often committed early in the system design process without knowledge of future requirements (e.g., Blanchard & Fabrycky, 1998, p. 37; Dowlatshahi, 1992, p. 1803). Put simply, the architectural choices made by systems engineers today have profound implications for a system's total cost at retirement. The inability to forecast this structural dependence reduces the DoD's ability to be responsive and flexible to future capability gaps. Unfortunately, despite the opportunity to affect substantive change on the resulting cost of the system early in its life cycle, it is difficult to anticipate how requirements will change over time. For example, researchers found that more than 10% of a system's baseline requirements will change during the development phase of a system's life cycle (Peña & Valerdi, 2015, pp. 63-65). Recognizing this uncertainty and its implications, the NDS identified "reforming the Department's business practices for greater performance and affordability" as one of its key lines of effort (DoD, 2018, p. 5).

Within the performance and affordability line of effort, the NDS outlines five competitive approaches to improving the DoD's practices, namely: "[1] deliver performance at the speed of relevance . . . [2] organize for innovation . . . [3] drive budget discipline and affordability to achieve solvency . . . [4] streamline rapid, iterative approaches from development to fielding . . . [5]



Legend:

ATP: Authority to Proceed	DD: Disposition Decision	FOC: Full Operational Capability
I: Iteration	IOC: Initial Operational Capability	MDD: Materiel Development Decision
MS: Milestone	MVCR: Minimum Viable Capability Release	MVP: Minimum Viable Product
OD: Outcome Determination	R: Release	

Figure 1. AAF Pathways (USD(A&S), 2020b, p. 5.)

harness and protect the Nation Security Innovation Base" (DoD, 2018, pp. 10-11). While each of these approaches is worthy of intense pursuit, particularly streamlining development and fielding processes, the appropriate mechanisms must be in place for the right capabilities to reach service members at the right time. Recognizing this imperative, two years following the 2018 update to the NDS, the DoD issued *DoD Instruction (DoDI) 5000.02 – Operation of the Adaptive Acquisition Framework* (AAF).

With the NDS's priorities and focus on improving the DoD's ability to deliver improved technical solutions at a reduced cost, "[t]he AAF supports the [Defense Acquisition System] with the objective of delivering effective, suitable, survivable, sustainable, and affordable solutions to the end user in a timely manner" (USD(A&S), 2020a, p. 3). Figure 1 depicts the AAF's six pathways, each of which is tailored to the desired capability gap. Within the scope of this research, we focus specifically on Major Capability Acquisition efforts, which consist of an initial Material Development Decision (MDD), followed by a series of critical milestones (MS).

Following the MDD, but prior to MS A, there are several key considerations with respect to the system itself, including "technical, cost and schedule risks, and the plans and funding to offset them during the [Technology Maturation and Risk Reduction] TMRR phase" (USD(A&S), 2020b, p. 12). This involves the designated PM conducting an analysis of the "Should Cost" targets, which are directly tied to the requirements specified in the Initial Capabilities Document. These targets are critical, as they establish the foundation for executing the final Request for Proposals and the cost for incorrectly doing so are high. For example, a RAND Project Air Force study found that across 35 mature programs unstable requirements accounted for a 12.9% increase in total costs (Bolten, Leonard, Arena, Younossi, & Sollinger, 2008, p. 72). In FY2005 dollars, this translated to a $23.7 billion increase. Additionally, between 1997 and 2009, the majority of Nunn-McCurdy Breaches (significant cost overruns that must be statutorily reported to Congress) cited engineering/design issues and requirement changes as significant factors contributing to their programs' unexcepted, excessive cost growth (GAO, 2011, p. 5).

Taken together, these findings support the need to improve two dimensions of the system design and acquisition process with respect to cost reduction: requirements development and extending the useable life of systems. The AAF seeks to address the first issue; however, as specified by the NDS, these solutions must not only address today's capability gaps but also be capable of mitigating those in future environments. Additionally, they must do so in a cost-effective manner. In recognition of this necessity, the recently published *DoDI 5000.88 – Engineering of Defense Systems* mandates that all Major Defense Acquisition Programs (MDAP) require a formalized Systems Engineering Plan (SEP) (USD (R&E), 2020b, p. 12). Moreover, within the SEP and specific to extending the useable life of systems, DoDI 5000.88 directed the use a Modular Open Systems Approach (MOSA).

As a framework for addressing capability gaps, MOSA "provides an integrated business and technical strategy for competitive and affordable

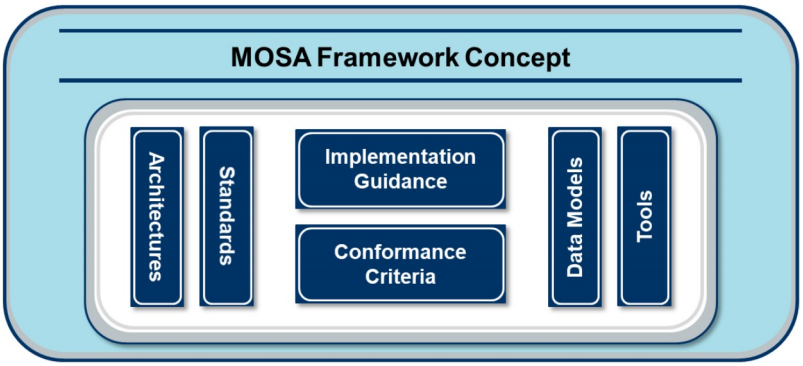


Figure 2. MOSA Framework (USD(R&E), 2020a, p. 4).

acquisition and sustainment of a new or legacy system (or a component within a new or legacy system) over the system life cycle" (USD(R&E), 2020a, p. 2). As seen in Figure 2, this framework includes six elements. Specific to this research, architecture is a representation of the "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution" (ISO, 2011, p. 2).

Within the AAF, during Material Solutions Analysis and prior to MS A, MOSA forces system designers to consider not only the first instantiation of the system but also how it will evolve over time. In doing so, they must consider and communicate the system's architecture, specifying how components interface with one another to better facilitate future changes or upgrades. In support of MOSA, the DoD utilizes the DoD Architecture Framework (DoDAF), a series of viewpoints and models, to communicate information about the system. The requirement to develop these models Pre-MS A not only supports the NDS and AAF, but more practically, provides analysts with rich data regarding the components and interfaces of the system.

For example, the DoDAF's Systems Viewpoint 3 (SV-3 or Systems-Systems Matrix) "provides a tabular summary of the system interactions" (DoD DCIO, 2010, p. 209). Within the context of system requirements and extending the useable life of the system, the SV-3 depicts relationships between the system components that execute the system's functions, providing the analyst with critical information that can be used to explore options for evolving the system over time. This ensures that while we may not be capable of predicting the future operational environment, at the earliest point in the systems engineering process, we can be more reasonably assured that the system is designed to remain responsive to external changes in a budget-constrained environment.

Given the NDS's focus on improving the defense acquisition process for a rapidly changing operational environment, the dire implications of incorrectly estimating and managing system costs, and the adoption of the AAF and MOSA, there exists a tremendous responsibility and opportunity to improve early life-cycle cost estimates.

Exploiting an Untapped Source of Early Life Cycle Information – Mapping DoDAF's Models to COSYSMO's Drivers

In a recent article, Valerdi, Dabkowski, and Dixit (2015) exploited the Pre-MS A availability of DoDAF's models by mapping them to the drivers of the Constructive Systems Engineering Cost Model (COSYSMO), a parametric, open academic cost model with the following cost estimating relationship (CER):

$$PM_{NS} = A \cdot \underbrace{\left(\sum_{i \in \{e,n,d\}} \sum_{k=1}^{4} w_{i_k} \Phi_{i_k} \right)^{E}}_{\text{size}} \cdot \underbrace{\prod_{j=1}^{14} EM_j}_{\text{effort}} \qquad (1)$$

where:

PM_{NS} = systems engineering effort in person months (nominal schedule),

A = calibration constant derived from historical project data (assume as 0.25),

w_{ik} = weight for the ith complexity level of the kth size driver ($i \in \{e$ (easy), n (nominal), d (difficult)$\}$),

Φ_{ik} = quantity of the kth size driver with complexity level i ($k \in \{1$ (requirements), 2 (interfaces), 3 (algorithms), 4 (operational scenarios)$\}$),

E = diseconomies of scale constant (assume as 1.06), and

EM_j = systems engineering effort multiplier for the jth cost driver (assume $\prod_{j=1}^{14} EM_j = 0.89$)) (Valerdi, 2008, p. 34).

Specifically, using a blend of text mining and social network analysis techniques, they analyzed

DoDAF's manual to identify subsets of models that, on a basis of doctrine, should contain useful information for populating COSYSMO's 18 drivers. The results of this analysis are summarized graphically in Figure 3, where: (a) a link between DoDAF model X and COSYSMO driver Y indicates model X should be relevant for rating driver Y, (b) DoDAF models shaded yellow were required Pre-MS A between 2012 and 2015 (CJCS, 2012a; CJCS, 2012b), (c) DoDAF models highlighted with a black star (∗) were required Pre-MS A between 2015 and 2018 (CJCS, 2015), (d) DoDAF models with their names highlighted in green are currently required pre-MS A (CJCS, 2018), and (e) COSYSMO drivers shaded red are not linked to any of the DoDAF models required by the Capability Development Document (CDD).

As Figure 3 shows, the collection of DoDAF models required Pre-MS A nearly spans COSYSMO's parameters, as they cover 14 of the 18 drivers. Additionally, several DoDAF models that are not explicitly required by the CDD are derivatives of data contained within the mandatory models, suggesting even greater coverage is possible. For example, the SV-3 is simply a more compact, summary representation of the interfaces described in the SV-1 (DoD DCIO, 2010, p. 209).

Exploiting this untapped, relevant source of early life cycle information, Valerdi et al. (2015) provide an algorithm for organizations to improve the measurement reliability of their MDAP cost estimates using COSYSMO (pp. 543-545).

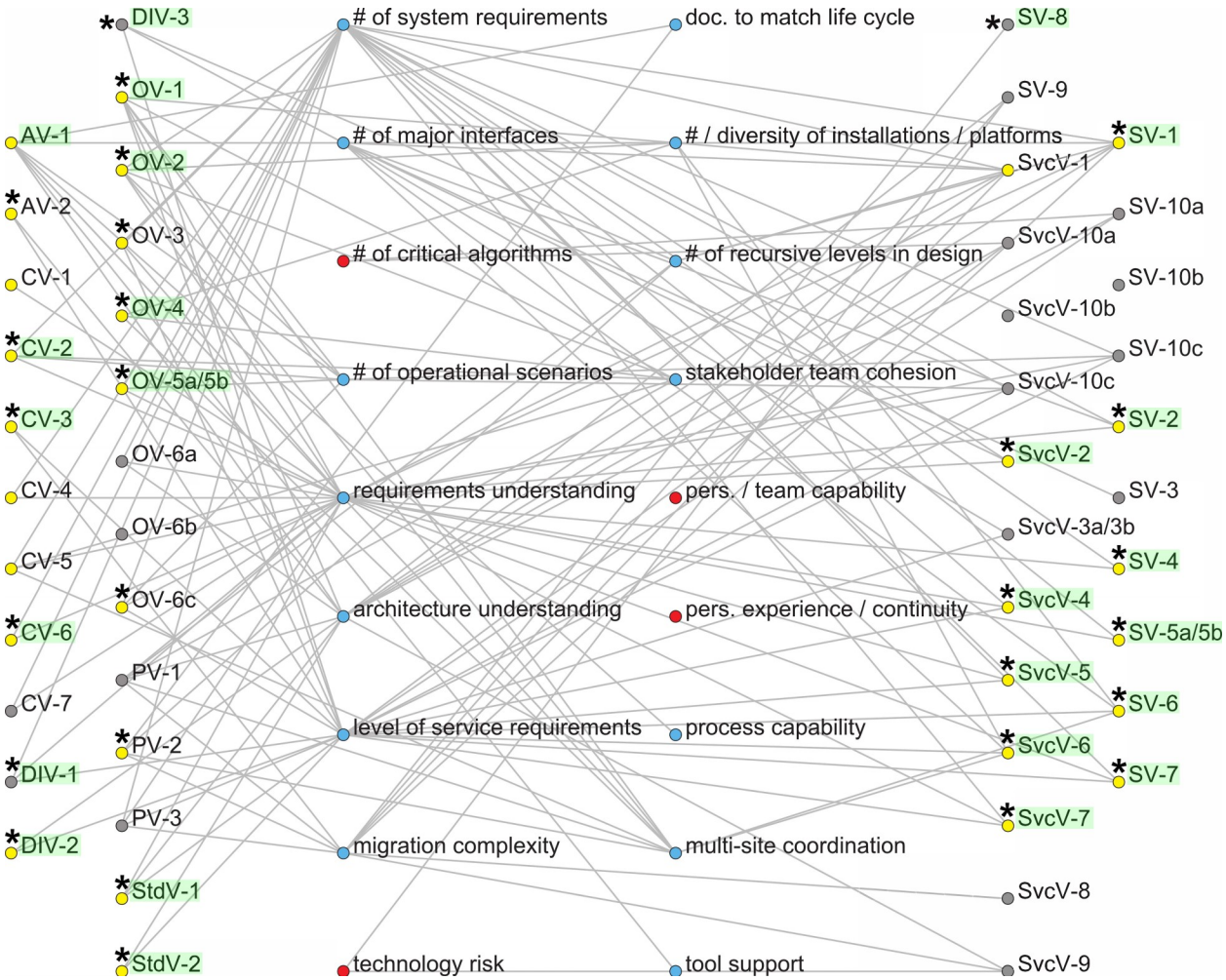


Figure 3. Mapping of DoDAF's models to COSYSMO's drivers.

Additionally, there are other tangential benefits stemming from their work, including opportunities for "data mining techniques . . . to extract knowledge from existing systems engineering concepts" (Valerdi et al., 2015, pp. 546). Given the rapid proliferation of such techniques and their application to new domains, algorithms employing these methods should be accompanied by a detailed explanation and assessment of any underlying assumptions. Consistent with this assertion, in this paper we methodically examine one such algorithm, namely, a method for estimating the cost of unanticipated, evolutionary architectural growth via the SV-3, COSYSMO, and network science (Dabkowski, Valerdi, & Farr, 2014).

Estimating the Cost of Architectural Growth Early in the Life Cycle

In "Exploiting Architectural Communities in Early Life Cycle Cost Estimation," Dabkowski et al. (2014) estimate the cost of adding a future subsystem (**X**) to an existing system architecture when the purpose and function of **X** are unknown. Consisting of 12 steps and seen in Algorithm 1 below, their approach leverages the current SV-3's structural properties to iteratively and randomly attach **X** to architectural modules (or communities), generate interfaces of reasonable complexity, and estimate the marginal cost of attachment.

Algorithm 1 (Dabkowski et al., 2014, p. 101)

For a specified, suitably large number of iterations . . .

Preprocessing

(1) Initialize the system as the current system,

(2) Use the Girvan-Newman (2002) community detection heuristic to identify architectural communities,

(3) Randomly assign **X** to community k,

Intracommunity Growth

(4) Generate a realization for $M_{\mathbf{X},\text{intra}}$ given **X** is assigned to community k (m_{intra}),

(5) Connect **X** to m_{intra} subsystems inside community k using the Barabási-Albert (BA) model (1999),

(6) For each interface established in (5), assign complexity ($w_{i\mathbf{X},\text{intra}}$),

Intercommunity Growth

(7) Generate a realization for $M_{\mathbf{X},\text{inter}}$ given **U** is assigned to community k (m_{inter}),

(8) Connect **U** to m_{inter} communities using the BA model,

(9) For each interface established in (8), assign complexity ($w_{i\mathbf{X},\text{inter}}$),

Cost Estimation

(10) Estimate the cost for the augmented system using COSYSMO (PM_{NS}*),

(11) Calculate the additional cost of adding subsystem **U** (PM_{NS}* − PM_{NS}), and

(12) Store results and return to (3).

For example and without loss of generality, imagine the hypothetical SV-3 in Panel (a) of Figure 4 serves as the **current system** in Step (1) of Algorithm 1. With 8 subsystems and 12 interfaces, this SV-3 graphically summarizes the relationships between existing subsystems, where shading in row i and column j implies subsystem i interfaces with subsystem j, and darker shades indicate greater interface complexity. Moving to Step (2), given the emphasis on MOSA within the DoD (USD(R&E), 2020b), we suspect that the current system's architecture might contain **communities of subsystems** where the density of interfaces within communities is high relative to the density of interfaces between them. To identify and exploit this structure, we apply the popular Girvan-Newman (2002) community detection heuristic, which yields the permuted, isomorphic SV-3 seen in Panel (b) of Figure 4. In Step (3), subsystem **X** is randomly assigned to one the current system's two architectural communities, namely C1 or C2.

Continuing with the example, suppose subsystem **X** is assigned to C1. Using a rich-by-birth effect, the number of subsystems **X** interfaces with in C1 is modeled as a discrete random variable ($M_{\mathbf{X},\text{intra}}$) with a probability mass function (PMF) equal to the observed intracommunity degree distribution of C1's subsystems. By inspection, C1's PMF for $M_{\mathbf{X},\text{intra}}$ is $P(M_{\mathbf{X},\text{intra}} = 2) = 0.5$ and $P(M_{\mathbf{X},\text{intra}} = 3) = 0.5$. As per Step (4), we generate a realization (m_{intra}) from this PMF to represent the number of **X**'s intracommunity interfaces.

Assuming $m_{\text{intra}} = 2$, we connect **X** to 2 subsystems in C1 using the BA model from network science, which mimics a rich-get-richer effect. Observed in a variety of real-world networks (e.g., Redner, 1998; Banavar, Maritan, & Rinaldo, 1999; Newman, 2001; Cancho, Janssen, & Solé, 2001), the BA model specifies that the probability of **X** attaching to an existing subsystem *i* in C1 is proportional to *i*'s number of intracommunity interfaces (Barabási & Albert, 1999). In Panel (b) of Figure 4, C1's subsystems **A**, **D**, **F**, and **H** have degrees 2, 3, 3, and 2, respectively. Accordingly, their corresponding attachment probabilities are 0.2, 0.3, 0.3, and 0.2, and in Step (5) we use these probabilities to determine **X**'s adjacency.

If Step (5) determines that **X** connects to subsystem **A**, then in Step (6) we use the observed interface complexity distribution of subsystem **A** as the PMF for the complexity of the interface between **X** and subsystem **A** ($w_{\mathbf{AX}}$). Specifically, one of **A**'s existing interfaces is rated **easy**, and the other is rated **nominal**. Accordingly, we apply the relative weights for COSYSMO's **number of major interfaces** size driver (Valerdi, 2008, p. 86) to obtain the following PMF for $w_{\mathbf{AX}}$: $P(w_{\mathbf{AX}} = 1.1) = 0.5$ and $P(w_{\mathbf{AX}} = 2.8) = 0.5$, and a Bernoulli trial determines the outcome.

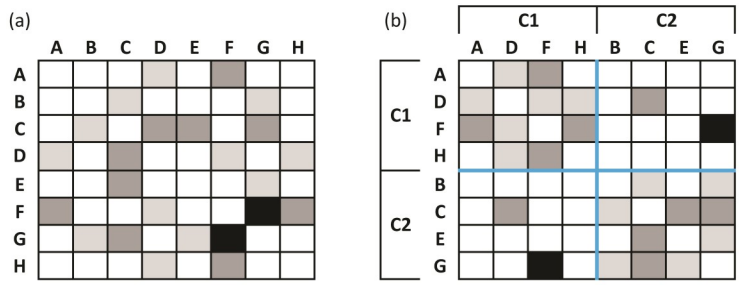


Figure 4. Hypothetical SV-3 with 8 subsystems and 12 interfaces in its original (Panel (a)) and permuted (Panel (b)) forms, where shading indicates interface complexity such that light gray ⇒ easy, medium gray ⇒ nominal, and black ⇒ difficult.

Steps (7) through (9) model intercommunity growth in a manner similar to Steps (4) through (6), and Steps (10) and (11) apply COSYSMO's CER to calculate the marginal cost of attachment. Finally, Step (12) completes the loop and captures the result. Subsequent iteration generates the data necessary to estimate the cumulative distribution functions and statistics for the estimated cost of adding subsystem **X** to C1 and C2.

Algorithm 1's Underlying Assumptions and Associated Hypotheses

Although Algorithm 1 has intuitive appeal, the validity of its underlying assumptions must be assessed. Starting with its principal input, we note that an SV-3 is nothing more than the adjacency matrix representation of a network, where nodes symbolize subsystems and edges denote interfaces. As such, if Dabkowski et al. (2014) have designed the right algorithm (i.e., Algorithm 1 is valid), the networks represented by real-world SV-3s will be similar to the hypothetical example in Figure 4. Methodologically, network similarity is typically assessed by comparing a list of structural statistics (Shore & Lubin, 2015), but selecting which statistics to compare is a matter of debate. With this in mind, study objectives typically drive the selection (e.g., Hunter, Goodreau, & Handcock, 2008), and our primary goal is to determine

whether Algorithm 1 can accommodate real-world SV-3s. Therefore, we restrict our focus to three characteristics, namely: type, density, and community structure.

With respect to type, we note that the hypothetical example reflects the interfaces that exist between a single set of subsystems. Specifically, the subsystems represented by the rows match those identified in the columns. In the parlance of network analysis, this is known as a **one-mode network** (Wasserman & Faust, 2009, p. 36-37). Moreover, upon closer examination, we note that the hypothetical example is symmetric across its main diagonal, implying the interfaces are bidirectional; the network is **undirected**. Finally, as mentioned earlier, the hypothetical example's cells are shaded according to interface complexities, which ultimately map to numerical weights in COSYSMO's CER. Accordingly, the network is **valued**. Taken together, Algorithm 1 accepts **one-mode, undirected, valued** SV-3s as input, and our first hypothesis is as follows: (H1) – Real-world SV-3s are one-mode, undirected, valued networks.

Moving on to density, Algorithm 1 makes use of the Girvan-Newman community detection heuristic, which "was designed with sparse networks in mind, . . . [and] may not perform as well on dense networks" (2002, p. 7826). Formulaically, the sparsity of a one-mode, undirected network with N nodes can be assessed by its density, which is simply the ratio of its observed number of edges (E) to the maximum possible number of edges or $2E / (N(N - 1))$ (Wasserman & Faust, 2009, p. 101). Although there is no definitive standard for characterizing a network as sparse, informally, the adjacency matrix of a sparse network consists primarily of zeros. Adopting this as our standard, this implies that a sparse network will have a density less than 0.5. For example, with $N = 8$ subsystems and $E = 12$ interfaces the hypothetical example in Figure 4 has a density of 0.428. As such, it is sparse, and, as expected, the Girvan-Newman community detection heuristic performs well, correctly identifying its two architectural communities. In sum, by using the Girvan-Newman community detection heuristic in Algorithm 1, Dabkowski et al. (2014) implicitly assume real-world SV-3s are sparse, and this leads us to our second hypothesis: (H2) – The densities of real-world SV-3s are less than 0.5.

As for community structure, recall that the hypothetical example in Figure 4 consists of two, well-defined communities, and Algorithm 1 conforms to this structure. The questions are: "Does the community structure in the hypothetical example warrant this approach," and "If so, do real-world SV-3s exhibit similar behavior?" To address these issues, we use Girvan and Newman's well-known modularity metric, which ranges from its minimum of 0, when "the number of within-community edges is no better than random," to its theoretical maximum of 1, indicating very strong community structure (Newman & Girvan, 2004, p. 7). In practice, however, Girvan and Newman note modularities above 0.7 are uncommon, and values greater than 0.3 suggest strong community structure (Newman & Girvan, 2004, p. 7). Adopting this standard, we used the edge.betweenness.community function from the statistical software R's igraph package (Csardi & Nepusz, 2006) to calculate the modularity for the hypothetical example. With a value of 0.333, its modularity exceeds the 0.3 threshold, and Algorithm 1 rightly abides its strong community structure. Whether or not this also applies to real-world SV-3s needs to be assessed, yielding our third hypothesis: (H3) – The modularities of real-world SV-3s are greater than 0.3.

Beyond validity related to input, the validity of Algorithm 1's growth and connection mechanisms must also be assessed. Looking at the growth mechanism, Dabkowski et al. (2014) modeled an incoming subsystem's number of interfaces (or degree) as a random variable with a PMF equal to the observed degree distribution of the current system. In fact, their approach

differs substantially from the growth step in the standard BA model, where a fixed versus random number of edges (m) are added each time a new node enters the network (Barabási & Albert, 1999). That said, when m is constant the total number of edges in a network of N nodes is fixed, reducing the flexibility of the standard BA model. For example, if a network starts as the smallest possible connected graph prior to the addition of new nodes, it will have m nodes and m - 1 edges. In this case, following the network's eventual growth to N nodes, it will have $E = m(N - m) + (m - 1) = m(N - m + 1) - 1$ total edges. For the hypothetical example in Figure 4, this is problematic. Specifically, if $m = 1$, once the SV-3 grows to its final size of $N = 8$ subsystems it will have $E = 7$ interfaces. This is 5 too few; therefore m must be greater than 1. However, if $m = 2$, it will ultimately have $E = 13$ interfaces, which is 1 too many.

Accordingly, holding an incoming subsystem's number of interfaces constant is overly restrictive, and using the observed degree distribution in Algorithm 1 seems reasonable. As Dorogovtsev and Mendes (2003) note:

> How can we account for the influence of the network on the properties on a newborn vertex? This, our baby, has only one characteristic, namely the number of its connections. Then, let this number not be fixed by God but be distributed with some distribution function which depends on the current state of the network . . . let the degree distribution of the newborn vertex be dependent on the degree distribution of the network (p. 42) .

Nonetheless, as with the standard BA model, the validity of this approach must be assessed. Ultimately, we want a stochastic model capable of growing SV-3s with N subsystems and approximately E interfaces. More precisely, if the mean absolute percentage error (MAPE) in interfaces is no more than 20%, we feel the model is sufficient, and our fourth hypothesis is as follows: (H4) – Using the observed degree distribution of a real-world SV-3 to model its incoming subsystems' number of interfaces grows SV-3s with a MAPE in interfaces less than or equal to 20%.

Finally, for Dabkowski et al.'s (2014) connection mechanism, in Steps (5) and (8) of Algorithm 1 they used linear preferential attachment to connect an incoming subsystem to subsystems already in the architecture. As such, if real-world SV-3s utilize linear preferential attachment, their connection mechanism is valid; otherwise, it is not. Unfortunately, proving the presence of linear preferential attachment in a network's evolution requires longitudinal data or snapshots of its growth over time (Newman, 2001; Jeong, Néda, & Barabási, 2002), and such data is often unavailable.

Nonetheless, even without longitudinal data, we can use a network's degree distribution to examine whether linear preferential attachment played a role in its evolution, as networks grown via linear preferential attachment have a characteristic statistical marker. Specifically, the probability that a node has degree d, $p(d)$, is proportional to $d^{-\omega}$ where $\omega = 3 - p$ and p represents the fraction of edges that are directed (Barabási & Albert, 1999). Moreover, while our earlier discussion demonstrated that m is not fixed and the standard BA model does not apply, this marker still holds. As Barabási and Albert note in their landmark 1999 paper: "For most networks, the connectivity m of the newly added vertices is not constant. However, choosing m randomly will not change the exponent" (p. 512). In short, if real-world SV-3s have been grown via linear preferential attachment, ω should be 3. Formally known as a power law, this scale-free behavior typically presents itself in the heavy, right-hand tail of the network's degree distribution, and following normalization, its

discrete distribution function is:

$$p(d) = \frac{d^{-\omega}}{\sum_{n=0}^{\infty}(n + d_{min})^{-\omega}} \quad (2)$$

where an approximate maximum likelihood estimator for ω is given by:

$$\widehat{\omega} \approx 1 + n\left[\sum_{i=1}^{n} \ln \frac{d_i}{d_{min} - 0.5}\right]^{-1} \quad (3)$$

and:

> $d_{\min}$= the lower bound for the power law behavior (Clauset, Shalizi, & Newman, 2009).

Using these facts, we formulate our fifth and final hypothesis as follows: (H5) – The observed degree distributions of real-world SV-3s follow a power-law with $\hat{\omega}$ = 3.

Results and Analysis of Hypothesis Testing

In order to test the hypotheses identified in the previous section, we used the SV-3s from 24 different defense programs. Consisting of weapons systems; combat and transportation vehicles; command, control, and communications suites; and intelligence, surveillance, and reconnaissance platforms, the programs vary considerably in both size and scope, providing a broad sample and facilitating generalization (AIMD, 2014). The results and analysis of this testing are given below.

(H1) – Real-world SV-3s are one-mode, undirected, valued networks.

As seen in the first several columns of Table 1, real-world SV-3s do not match Algorithm 1's expected input; therefore, (H1) is refuted. First, with respect to mode, 6 of the 24 SV-3s only reflect the interfaces that exist between two different sets of subsystems (e.g., internal and external) [Endnote 1]. Known as **two-mode networks** (or bipartite graphs), these SV-3s do not comport with the standard BA model [Endnote 2]. Next, of the 18 remaining SV-3s, 4 are asymmetric across their main diagonal or **directed**. While this does not agree with Algorithm 1's expected input, the standard BA model is extensible to directed networks by using each node's in-degree ($d_{i,\text{in}}$, the number of directional edges pointing to node i) when calculating attachment probabilities. Last, none of the 24 SV-3s are valued, let alone weighted according to interface complexity. Without interface complexities, the validity of using subsystem i's observed interface complexity distribution to estimate future interface complexity cannot be assessed. In sum, although (H1) is refuted, 14 of the 24 real-world SV-3s are one-mode, undirected networks, and these will form the basis for our subsequent analysis.

(H2) – The densities of real-world SV-3s are less than 0.5.

Of the 14 one-mode, undirected SV-3s in Table 1, only one (i.e., System 6) has a density greater than 0.5. Accordingly, in general, (H2) is affirmed. Moreover, the average and median density among these 14 SV-3s are 0.245 and 0.214, respectively, which compare favorably with the hypothetical example's density of 0.428. As such, our sample suggests real-world SV-3s are sparse, and using the Girvan-Newman community detection heuristic in Algorithm 1 is appropriate.

(H3) – The modularities of real-world SV-3s are greater than 0.3.

While Table 1 provides definitive evidence for refuting (H1) and affirming (H2), the data for (H3) is less clear. In particular, 6 of the 14 one-mode, undirected SV-3s have modularities greater than 0.3. Put another way, roughly 50% of the real-world SV-3s in our sample have strong community structure worth exploiting, and Algorithm 1 accommodates this. On the other

	Type			Size			Community Structure	
System	Modes	Undirected?	Valued?	*N*	*E*	Density	Communities	Modularity
1	2	–	–	–	–	–	–	–
2	2	–	–	–	–	–	–	–
3	1	Y	N	19	26	0.152	4	**0.365**
4	1	Y	N	14	19	0.209	4	0.256
5	1	Y	N	21	46	0.219	2	0.072
6	1	Y	N	4	4	0.667	1	0
7	1	Y	N	15	16	0.152	3	**0.498**
8	1,2	Y	N	9	12	0.333	2	0.247
9	2	–	–	–	–	–	–	–
10	1	N	N	–	–	–	–	–
11	1	Y	N	24	19	0.069	8	**0.611**
12	1	N	N	–	–	–	–	–
13	1	N	N	–	–	–	–	–
14	1	Y	N	10	16	0.356	3	0.271
15	1	Y	N	28	23	0.061	8	**0.749**
16	1	Y	N	22	67	0.290	11	0.057
17	1	Y	N	19	18	0.105	6	**0.532**
18	1	Y	N	18	30	0.196	6	0.150
19	1	N	N	–	–	–	–	–
20	1,2	Y	N	9	8	0.222	3	**0.414**
21	2	–	–	–	–	–	–	–
22	2	–	–	–	–	–	–	–
23	2	–	–	–	–	–	–	–
24	1,2	Y	N	6	6	0.400	4	0.042

Table 1. Summary of the type, density, and community structure of real-world SV-3s. Bold entries highlight modularity values which indicate strong community structure. Systems with both one- and two-mode SV-3s are denoted by "1,2" in the "Modes" column. For these systems, values in the size and community structure panels correspond to the one-mode SV-3.

hand, when an SV-3's modularity is less than 0.3, the identified community structure may be spurious, and exploiting it in Algorithm 1 is not recommended. As such, a non-community version of the connection mechanism is needed, and Algorithm 1 should be modified accordingly.

(H4) – Using the observed degree distribution of a real-world SV-3 to model its incoming subsystems' number of interfaces grows SV-3s with a MAPE in interfaces less than or equal to 20%.

To test (H4), we simulated the growth of the 14 one-mode, undirected SV-3s in Table 1 using Algorithm 1's growth and connection mechanisms. In particular, at $t = 0$ we start with an SV-3 consisting of two subsystems connected by a single interface. Accordingly, the PMF for the first incoming subsystem's number of interfaces is simply $P(M = 1) = 1$, and it generates one interface, which is attached to either of the two existing subsystems with equal probability. At $t = 1$, the SV-3 consists of 3 subsystems, where two subsystems have one interface and one subsystem has two interfaces. As such, the PMF for the second incoming subsystem's number of interfaces is $P(M = 1) = 2/3$ and $P(M = 2) = 1/3$. Drawing a random variate from this PMF determines the number of interfaces the second incoming subsystem generates, and these interfaces are connected to the three existing subsystems using linear preferential attachment. At $t = 3$ the process repeats itself, and it continues until the SV-3 consists of N subsystems, at which point the total number of interfaces is recorded. The results of this simulation for 1,000 trials at each value of N are given in Table 2.

System	3	4	5	6	7	8	11	14	15	16	17	18	20	24
N	19	14	21	4	15	9	24	10	28	22	19	18	9	6
E	26	19	46	4	16	12	19	16	23	67	18	30	8	6
Density	0.15	0.21	0.22	0.67	0.15	0.33	0.07	0.36	0.06	0.29	0.11	0.2	0.22	0.4
Rank (by density)	10	8	7	1	10	4	13	3	14	5	12	9	6	2
95% CI on μ(LB)	63	35	76	3	40	15	99	18	131	83	63	56	15	7
95% CI on μ(UB)	65	37	79	3	41	15	103	19	137	86	65	58	15	7
MAPE	146	92	74	17	153	35	431	29	485	40	255	93	86	26
Rank (by MAPE)	10	8	6	1	11	4	13	3	14	5	12	9	7	2

Table 2. Results of growing SV-3s when the observed degree distribution is used to generate an incoming subsystem's number of interfaces. The 95% confidence intervals (CIs) reflect a plausible range of values for the mean number of interfaces (μ), where LB and UB denote the lower and upper bounds, respectively.

As Table 2 shows, (H4) is refuted. Specifically, with the exception of System 6, using the observed degree distribution generated too many interfaces on average, and this is reflected in the magnitude of the lower bounds of the 95% confidence intervals on the mean number of interfaces as well as the MAPEs. Furthermore, as highlighted by the system ranks according to density and MAPE, sparse SV-3s performed worse than dense SV-3s, with the MAPE reaching 485% for the least dense SV -3 (System 15). Accordingly, the observed degree distribution is an inadequate model for an incoming subsystem's number of interfaces, and an alternative growth mechanism is needed.

(H5) – The observed degree distributions of real-world SV-3s follow a power-law with $\hat{\omega} = 3$

As late as 2008, the preferred method for testing (H5) would have been to plot the histogram of the SV-3's degrees on a log-log graph and subsequently perform least-squares regression to estimate ω. However, this method has several statistical shortcomings, and in 2009 Clauset et al. formalized hypothesis tests for assessing power law behavior, along with procedures for estimating ω and $d_{\min}$ in Equation (3) [Endnote 3]. These procedures have been implemented in R's igraph package via the power.law.fit function (Csardi & Nepusz, 2006), and the results of this estimation for the 14 one-mode, undirected SV-3s are given in Table 3.

Power law distribution fitting					H_0 : Data follow a power law distribution with the fitted parameters H_1 : Data do not follow a power law distribution with the fitted parameters									
System	3	4	5	6	7	8	11	14	15	16	17	18	20	24
$\hat{\omega}$	3.06	2.27	2.78	∞	2.8	2.69	2.52	2.26	2.33	2.57	2.06	2.22	2.01	∞
$\hat{d}_{\min}$	3	2	3	–	2	2	1	2	1	4	1	2	1	–
p-value	0.99	0.99	0.58	–	0.99	0.99	1	0.67	1	0.5	0.28	0.99	0.83	–
95% CI on $\hat{\omega}$ (LB)	2.13	1.68	2.09	–	1.95	1.86	1.98	1.67	1.9	2	1.66	1.72	1.51	–
95% CI on $\hat{\omega}$ (UB)	5.11	3.64	4.09	–	4.82	4.76	3.46	3.62	3.05	3.57	2.77	3.16	3.19	–
N_{total}	19	14	21	4	15	9	24	10	28	22	19	18	9	6
N_{fit}	10	9	15	–	9	8	24	9	28	18	19	14	9	–

Table 3. Results of fitting discrete power law distributions to the observed degree distributions of real-world SV-3s. Systems 6 and 24 do not have sufficient data to estimate ω.

Based on these results, the power law distribution is a statistically plausible model for the observed degree distributions of the SV-3s. After all, among the 12 cases with sufficient data to estimate ω and $d_{\min}$, the smallest p-value is 0.28 (System 17), suggesting we unanimously fail to reject the null hypothesis that the data follow a power law distribution with the fitted parameters. Moreover, in all but one case (System 17), the 95% confidence intervals for ω include 3, suggesting there is statistical support for the presence of linear preferential attachment in the SV-3s' evolution. Nevertheless, the high p-values for several of the systems are concerning, and they are likely due to the small sizes of the data sets used to fit the parameters (N_{fit}) [Endnote 4]. As Clauset et al. (2009) caution, "It is possible for small values of n that the empirical distribution will follow a power law closely, and hence that the p-value will be large, even when the power law is the wrong model for the data" (p. 678).

With this in mind, we can test whether the data follows another heavy-tailed distribution more closely, and the discrete exponential distribution is an appropriate choice, as it represents the limiting distribution of the subsystems' degrees when the p_i are uniform and preferential attachment is not present (Barabási, Albert, & Jeong, 1999). Accordingly, we performed a likelihood ratio test to determine whether a discrete power law or a discrete exponential distribution better fits the data (Clauset et al., 2009). Implemented in R's poweRlaw package (Gillespie, 2015), the results of this test are summarized in Table 4, where p-values less than 0.05 indicate statistically significant results and, if significant, a positive (negative) test statistic implies the power law (exponential) distribution is a better fit.

As seen in Table 4, only System 5 produced a statistically significant result, with the power law being favored over the exponential distribution. Additionally, Systems 14 and 20 were nearly significant, and, in both cases, the exponential distribution better fits the data. In sum, our likelihood ratio testing cannot refute the presence of linear preferential attachment in real-world SV-3s; however, due to small n, our conclusion lacks statistical power. Accordingly, (H5) is affirmed, but further investigation is warranted.

Accommodating Reality – A Modified, Data-Driven Approach

As the hypothesis testing in the previous section demonstrated, Algorithm 1 cannot accommodate all real-world SV-3s in its current form, and it must be modified. In particular, evaluating (H1) identified that SV-3s may be two-mode and directed, requiring different growth and connection mechanisms. Moreover, none of the 24 real-world SV-3s we examined are valued; therefore, the validity of using the observed interface complexity distribution to estimate future interface complexity cannot be assessed. Although these shortcomings must be addressed, they are beyond the scope of this work, and we restricted our focus to the 14 one-mode, undirected SV-3s.

For these SV-3s, evaluating (H2) confirmed the appropriateness of using the Girvan-Newman community detection heuristic, and analyzing

Power law versus exponential					H_0 : Both distributions are equally far from the true distribution H_1 : One of the test distributions is closer to the true distribution									
System	3	4	5	6	7	8	11	14	15	16	17	18	20	24
p-value	0.33	0.56	**0.01**	–	0.86	0.28	0.31	0.06	0.65	0.24	0.21	0.53	0.08	–
Test Statistic	-0.9	-0.6	**2.8**	–	-0.2	-1.1	1	-1.9	-0.5	-1.2	-1.2	-0.6	-1.8	–

Table 4. Likelihood ratio test results reflecting whether a discrete power law or a discrete exponential distribution better fits the observed degree distributions of real-world SV-3s. Bold entries indicate statistically significant results.

	$E_{(1)}$	$E_{(2)}$	$E_{(3)}$	$E_{(4)}$	$E_{(5)}$	$E_{(6)}$	$E_{(7)}$	$E_{(8)}$	$E_{(9)}$	$E_{(10)}$	$E_{(11)}$	$E_{(12)}$	$E_{(13)}$	$E_{(14)}$	$E_{(15)}$	$E_{(16)}$	$E_{(17)}$	$E_{(18)}$	Sum
1	1	2	2	2	1	1	1	1	1	1	2	1	1	4	2	1	1	1	26
2	1	1	1	1	1	3	1	6	1	1	1	1	1	1	2	1	1	1	26
3	1	1	2	2	1	1	2	2	2	1	1	1	1	2	1	1	2	2	26

Figure 5. Three possible solutions to Equation (4).

(H3) indicated strong community structure for 6 of the 14 SV-3s. To accommodate SV-3s without strong community structure, Algorithm 1 needs an additional, non-community connection mechanism. Furthermore, investigating (H4) clearly indicated that using the observed degree distribution to model an incoming subsystem's number of interfaces generates too many interfaces on average, and an alternative growth mechanism is needed. Finally, although examining (H5) provided statistical support for the presence of linear preferential attachment in the SV-3s' evolution, the results lack statistical power.

While these shortcomings may seem damning, we see them as an opportunity to develop a modified, data-driven approach. Specifically, real-world SV-3s hold the key to finding models of evolutionary architectural growth, starting with the PMF for an incoming subsystem's number of interfaces.

Establishing Feasible PMFs for an Incoming Subsystem's Number of Interfaces (P($M = m$))

Consider System 3 from Table 1. With 19 subsystems and 26 interfaces, if m is modeled by the observed degree distribution, Table 2 indicates that the resulting SV-3s will have 38 too many interfaces on average. Therefore, our first objective is to find a distribution for m (P($M = m$)) that grows SV-3s of size (N = 19, E = 26).

To do this, we return to System 3 and make the following observation. If the SV-3 starts as a single unconnected subsystem and each incoming subsystem must generate at least one interface, the remaining N - 1 = 18 subsystems must produce exactly E = 26 interfaces, and the following relation holds:

$$E_{(1)} + E_{(2)} + \cdots + E_{(N-1)} = E \tag{4}$$

Where $E_{(i)}$ represents the number of interfaces the ith oldest subsystem generates and

$1 \le E_{(i)} \le \min\{i, E - (N - 2)\}$[Endnote 5]. For example, consider Figure 5, which gives three possible solutions to Equation (4) for System 3.

Examining the first solution, we note that $1 \le E_{(i)} \le \min\{i, 9\}$ for each subsystem and $\sum_{i=1}^{18} E_{(i)} = 26$; it is feasible. However, given the distribution of the $E_{(i)}$'s, the first solution has multiple feasible permutations. To show this directly, note that there are 12 subsystems with 1 interface, and one of these subsystems must fill the first position ($E_{(i)} = 1$). Following this assignment, 16 subsystems with less than 3 interfaces are available to fill the second and third positions, which yields four cases, namely:

(a) $\{E_{(1)} = 1, E_{(2)} = 1, E_{(3)} = 1\}$,

(b) $\{E_{(1)} = 1, E_{(2)} = 1, E_{(3)} = 2\}$,

(c) $\{E_{(1)} = 1, E_{(2)} = 2, E_{(3)} = 1\}$, and

(d) $\{E_{(1)} = 1, E_{(2)} = 2, E_{(3)} = 2\}$.

At this point, the single subsystem with four interfaces becomes available for assignment, and any of the 15 remaining subsystems can fill positions 4 through 18, where the distribution of the remaining E_i's is dependent on the case. In particular, if we denote the number of remaining subsystems with 1, 2, or 4 interfaces as n_1, n_2, and n_4, respectively, we note:

(a) $\{n_1 = 9, n_2 = 5, n_4 = 1\}$,

(b) $\{n_1 = 10, n_2 = 4, n_4 = 1\}$,

(c) $\{n_1 = 10, n_2 = 4, n_4 = 1\}$, and

(d) $\{n_1 = 11, n_2 = 4, n_4 = 1\}$.

Armed with this information and applying the well-known formula for the permutation of indistinguishable objects with repetition, the number of ways to feasibly permute the first solution in Figure 5 is:

$$\underbrace{\frac{15!}{(9!\,5!\,1!)}}_{\text{Case (a)}} + \underbrace{\frac{15!}{(10!\,4!\,1!)}}_{\text{Case (b)}} + \underbrace{\frac{15!}{(10!\,4!\,1!)}}_{\text{Case (c)}} + \underbrace{\frac{15!}{(11!\,3!\,1!)}}_{\text{Case (d)}} = 65{,}520 \qquad (5)$$

Of course, similar logic applies to the second and third solutions in Figure 5, as well as every other non-permutationally equivalent solution. Simply put, the total number of feasible solutions is large.

Fortunately, while the sequencing of the subsystems affects feasibility and must be checked, it does not affect the distribution of m. Specifically, each of the 65,520 feasible permutations of the first solution has 12 subsystems with 1 interface, 5 subsystems with 2 interfaces, and 1 subsystem with 4 interfaces, implying P($M = m$) = {0.667, m = 1; 0.278, m = 2; 0.55, m = 4}. Recalling our objective is to find a PMF for m that consistently grows SV-3s of size (N, E), we do not need to find every feasible sequence for m. In the context of Equation (4), we need to find the unordered sets of $E_{(i)}$, $\{E_1, E_2, \cdots, E_{N\text{-}1}\}$, such that $\sum_{i=1}^{N-1} E_i = E$ where $1 \le E_i$ for all i.

In fact, this is equivalent to solving a restricted partition problem from number theory, where E (a positive integer) is decomposed into the sum of exactly N - 1 positive integers. Using the generating function approach (Gupta, 1970), the total number of restricted partitions, $p(E, N - 1)$, is equivalent to the coefficient of the x^E term after expanding the polynomial:

$$x^{N-1} \prod_{r=1}^{N-1} (1 + x^r + x^{2r} + \cdots) \qquad (6)$$

where each of the infinite order polynomials of the form $(1 + x^r + x^{2r} + \cdots)$ can be truncated just prior to its order exceeding N - 1. Applying this result to System 3 yields:

$$x^{18} \prod_{r=1}^{18} (1 + x^r + x^{2r} + \cdots) = \cdots 30x^{27} + 22x^{26} + 15x^{25} + \cdots$$

Therefore, p(26, 18) = 22, and these 22 restricted partitions (r) capture the possible PMFs for m, which can be enumerated using the restrictedparts function from R's partitions package (Hankin, 2006). Calling this function for E = 26 and N - 1 = 18 returns the result in Figure 6.

As Figure 6 shows, although each of the 22 restricted partitions sum to 26 and have a minimum value of 1, the maximum value ranges from a high of 9 to a low of 2. Accordingly, while

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}	E_{12}	E_{13}	E_{14}	E_{15}	E_{16}	E_{17}	E_{18}	Sum
1	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
2	8	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
3	7	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
4	6	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
5	5	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
6	7	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
7	6	3	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
8	5	4	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
9	5	3	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
10	4	4	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
11	6	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
12	5	3	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
13	4	4	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
14	4	3	3	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
15	3	3	3	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	26
16	5	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	26
17	4	3	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	26
18	3	3	3	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	26
19	4	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	26
20	3	3	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	26
21	3	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	26
22	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	26

Figure 6. Restricted partitions of System 3's interfaces.

these PMFs for m are all feasible, they are different, and some may better replicate the degree distribution of System 3 when we use them to grow SV-3s of size (N = 19, E = 26) .

Parameterizing the Strength of Preferential Attachment (β)

Armed with feasible PMFs for m, our next task is to determine the role linear preferential attachment plays in the formation of System 3's SV-3. Looking at Table 3, we failed to reject the null hypothesis that System 3's degree distribution followed a discrete power law distribution with ω = 3, implying linear preferential attachment is present. However, as seen in Table 4, subsequent likelihood ratio testing failed to reject that a discrete exponential distribution fits the data equally well, suggesting linear preferential attachment might be absent.

These two cases represent poles along a continuum, and another possibility is that preferential attachment is present but it is not linear. For example, the probability of an incoming subsystem attaching to an existing subsystem i could be given by $p_i = {d_i}^{\beta} / \sum_{j=1}^{N} {d_j}^{\beta}$ where $0 < \beta < 1$ (Barabási, 2015). In this case, the incoming subsystem is more likely to connect with highly connected subsystems but the attachment probability is a sublinear function of degree – it is more muted. Moreover, even if likelihood ratio testing concludes that a discrete power law distribution fits the data better, it does not address the possibility that the preferential attachment could be a superlinear function of degree ($\beta < 1$). As with the feasible PMFs for m, settling on a best value for β necessitates that we grow SV-3s of size (N = 19, E = 26) for various values of β and measure how close their degree distributions are to System 3's. The question is:

"How should the difference between the degree distributions of two SV-3s of size (N, E) be measured?"

Assessing the Difference Between Degree Distributions (z^*)

Although there are several well-known methods for calculating the difference between probability distributions (e.g., the two-sample Kolmogorov-Smirnov test statistic (Darling, 1957) and the 1st Wasserstein metric or Earth Mover's Distance (Rubner, Tomasi, & Guibas, 1998)), the SV-3s' equivalent size invites an intuitive approach based on their observed degrees. Drawing on a frequently used example from combinatorial optimization, imagine we have N skiers with heights (h_i) and N sets of skis with lengths (l_j), and we want to match the skis to the skiers such that the sum of the absolute pairwise differences between the heights of the skiers and the lengths of the skis ($z = \Sigma|h_i - l_j|$) is minimized. As simple as it sounds, a globally optimal solution to this problem can be obtained by ordering the skis/ skiers by length/height and assigning skis to skiers on a basis of their relative size (i.e., $z^* = \Sigma|h_{(k)} - l_{(k)}|$, where $_{(k)}$ denotes the kth height/length of smallest skier/ski) (Lawler, 1976, p. 208).

By analogy, if we consider the N subsystems of SV -3s A and B as the skis and skiers, respectively, and match their subsystems on a basis of relative degree, the sum of the absolute pairwise differences between their degrees ($z = \Sigma|d_i - d_j|$) is minimized. If the degrees of the two SV-3s are identical, their degree distributions are the same, and the value of the minimum (z^*) equals zero. On the other hand, as the degrees become dissimilar, their degree distributions begin to diverge, and z^* increases. Consequently, z^* provides an intuitive measure of the difference between the degree distributions. With this in mind, when growing SV-3s of size (N, E) , with the goal of replicating the degree distribution of a real-world SV-3, smaller values of z^* are preferred.

Simulating Growth to Identify Optimal P(*M* = *m*) and Values for *β*

At this point, we have generated a set of feasible PMFs for an incoming subsystem's number of interfaces $\{P(M = m)\}$, and we have parameterized the attachment probabilities to vary the strength of preferential attachment (β). Moreover, we have developed an intuitive measure z^* to assess the difference between the degree distributions of two SV-3s of size (N, E). Ultimately, for a real-world SV-3, we would like to find the member(s) of $\{P(M = m)\}$ and value(s) of β that minimize z^*. Given the stochastic nature of the connection mechanism, as well as the need to feasibly permute the restricted partitions indicated by $\{P(M = m)\}$, z^* is a random variable (Z^*). Accordingly, we designed an experiment to simulate the growth of SV-3s of size (N, E) for all restricted partitions and various values of β to identify which, if any, combination(s) of the parameters produces the smallest population mean for Z^* (μ_{z^*}). The pseudocode for this procedure is as follows:

(1) Calculate, sort, and store the degrees of a real-world SV-3 of size (N, E)

(2) Generate and store the restricted partitions for SV-3s of size ($N - 1, E$) ,

(3) Determine a representative set of values for β (i.e., $\beta = \{0, 0.1, 0.2, \cdots, 1\}$), and

(4) For each restricted partition (r) and value of β, for a specified, suitably large number of iterations, on each iteration l . . .

Restricted Partition Number (r)	β = 0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	Mean
1	11.68	11.70	11.49	11.43	11.20	11.65	11.52	12.00	11.92	12.59	13.21	11.85
2	10.34	10.24	10.02	10.09	10.05	10.07	10.22	10.41	10.92	11.39	11.90	10.51
3	8.61	8.41	8.56	8.45	8.42	8.60	8.81	9.04	9.65	10.10	10.62	9.02
4	7.45	7.37	7.29	7.32	7.27	7.47	7.90	8.31	8.67	9.34	10.06	8.04
5	7.22	7.07	7.046	7.13	7.22	7.41	7.69	8.02	8.59	9.22	9.67	7.84
6	9.08	8.88	8.93	8.74	8.62	8.88	9.12	9.45	9.77	10.35	10.66	9.32
7	7.56	7.50	7.38	7.50	7.52	7.72	7.94	8.13	8.61	9.19	9.76	8.07
8	7.72	7.60	7.36	7.45	7.58	7.51	7.72	8.03	8.46	9.09	9.64	8.02
9	7.42	7.23	7.23	7.18	7.054	7.24	7.55	7.73	8.18	8.58	9.31	7.70
10	7.93	7.84	7.68	7.55	7.61	7.71	7.80	8.20	8.42	8.95	9.51	8.11
11	8.06	8.04	7.84	7.97	7.80	8.16	8.20	8.37	8.80	9.38	9.75	8.40
12	7.95	7.71	7.59	7.52	7.58	7.55	7.81	8.16	8.43	8.74	9.37	8.04
13	8.38	8.15	7.90	7.97	7.92	7.98	8.07	8.25	8.64	8.97	9.74	8.36
14	7.98	7.91	7.63	7.57	7.70	7.54	7.74	8.08	8.29	8.65	9.19	8.02
15	8.06	7.73	7.55	7.20	7.32	7.20	7.36	7.67	7.97	8.22	8.85	7.74
16	8.26	8.29	8.00	8.17	8.06	7.97	8.31	8.57	8.72	9.15	9.69	8.47
17	8.45	8.34	8.22	8.14	8.07	8.10	8.35	8.53	8.53	8.90	9.40	8.46
18	8.44	8.14	8.13	7.77	7.72	7.85	7.87	7.96	8.43	8.60	9.25	8.20
19	9.13	8.94	8.88	8.59	8.56	8.59	8.76	9.04	9.16	9.46	10.06	9.02
20	8.96	8.73	8.52	8.60	8.44	8.43	8.36	8.65	9.01	9.08	9.65	8.77
21	9.60	9.38	9.15	8.92	9.06	9.02	9.07	9.31	9.53	9.92	10.38	9.39
22	10.09	9.87	9.72	9.64	9.64	9.73	10.07	10.05	10.46	10.77	11.30	10.12
Mean	8.56	8.41	8.28	8.22	8.20	8.29	8.47	8.73	9.05	9.48	10.04	

Figure 7. Sample means of the minimum absolute pairwise differences between the degrees of System 3's SV-3 and the degrees of 1,000 simulated SV-3s for each (r, β) pair. Yellow shading denotes the five smallest sample means, and blue shading reflects the relative magnitude of the sample means across the 242 (r, β) pairs, where darker shades reflect larger means.

(a) Randomly permute the restricted partition until feasible ($E_{(i)} \leq i$ for $i \in \{1, 2, \cdots, N\text{-}1\}$),

(b) Initialize the simulated SV-3 as a single subsystem,

(c) Sequentially add N - 1 subsystems to the simulated SV-3, where the ith incoming subsystem generates $E_{(i)}$ interfaces and attaches to existing subsystem k with probability $p_k \propto d_k^{\beta}$,

(d) Calculate and sort the degrees of the simulated SV-3, and

(e) Sum the absolute pairwise differences between the sorted degrees of the simulated SV-3 and the real-world SV-3; store the result as $z^*_{r,\beta,l}$.

Increasing β from 0 to 1 in increments of 0.1 and running 1,000 iterations for each (r, β) pair produce 242,000 realizations of Z^* for System 3. The sample means of this experiment are summarized in Figure 7.

As seen in Figure 7, the sample means range from a minimum of 7.046 for the (r = 5, β = 0.2) pair to a maximum of 13.21 for the (r = 1, β = 0.1) pair. Additionally, the poorest performing (r, β) pairs seem to be concentrated in r = {1, 2, 21, 22} and β = {0.9, 1}. Based on these observations, parameter settings affect the fit, and this is highlighted in Figure 8, where the sample means have been converted to ascending ranks by row and column and shaded accordingly.

In particular, on the left side of Figure 8 it appears that β = {0.2, 0.3, 0.4, 0.5} produce SV-3s with degrees that more closely match the degrees of System 3's SV-3, immaterial of r. Similarly, regardless of the value of β, on the right side of Figure 8 it seems that r = {5, 9, 15} perform relatively well. Based on these observations, we conclude that the underlying population means depend on the restricted partition and the value of β and that a best fitting (or better fitting) (r, β) pair(s) may exist.

To test this conclusion formally, consider (a) we have two factors (the restricted partitions and

Restricted Partition Number (r)	β											β											Restricted Partition Number (r)
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	
1	6	7	3	2	1	5	4	9	8	10	11	22	22	22	22	22	22	22	22	22	22	22	1
2	7	6	1	4	2	3	5	8	9	10	11	21	21	21	21	21	21	21	21	21	21	21	2
3	6	1	4	3	2	5	7	8	9	10	11	15	15	16	15	15	17	17	16	18	18	18	3
4	5	4	2	3	1	6	7	8	9	10	11	3	3	3	4	3	4	9	11	12	14	16	4
5	4	2	1	3	5	6	7	8	9	10	11	1	1	1	1	2	3	3	4	9	13	10	5
6	6	3	5	2	1	3	7	8	9	10	11	17	17	18	18	18	18	19	19	19	19	19	6
7	5	3	1	2	4	6	7	8	9	10	11	4	4	5	6	5	9	10	7	10	12	14	7
8	6	5	1	2	4	3	7	8	9	10	11	5	5	4	5	6	5	4	5	7	10	8	8
9	6	4	3	2	1	5	7	8	9	10	11	2	2	2	2	1	2	2	2	2	2	4	9
10	7	6	3	1	2	4	5	8	9	10	11	6	8	9	8	8	8	6	9	4	7	7	10
11	5	4	2	3	1	6	7	8	9	10	11	9	10	10	12	11	14	12	12	14	15	13	11
12	7	5	4	1	3	2	6	8	9	10	11	7	6	7	7	7	7	7	8	5	5	5	12
13	8	6	1	3	2	4	5	7	9	10	11	12	12	11	11	12	12	11	10	11	8	12	13
14	7	6	3	2	4	1	5	8	9	10	11	8	9	8	9	9	6	5	6	3	4	2	14
15	9	7	5	1	3	1	4	6	8	10	11	9	7	6	3	4	1	1	1	1	1	1	15
16	5	6	2	4	3	1	7	8	9	10	11	11	13	12	14	13	11	13	14	13	11	11	16
17	7	5	4	3	1	2	6	9	8	10	11	14	14	14	13	14	13	14	13	8	6	6	17
18	9	7	6	2	1	3	4	5	8	10	11	13	11	13	10	10	10	8	3	6	3	3	18
19	8	6	5	3	1	2	4	7	9	10	11	18	18	17	16	17	16	16	17	16	16	15	19
20	8	7	4	5	3	2	1	6	9	10	11	16	16	15	17	16	15	15	15	15	9	9	20
21	9	7	5	1	3	2	4	6	8	10	11	19	19	19	19	19	19	18	18	17	17	17	21
22	8	5	3	1	2	4	7	6	9	10	11	20	20	20	20	20	20	20	20	20	20	20	22

Figure 8. Ascending ranks of the sample means from Figure 7 by restricted partition number (left side) and β (right side). Yellow shading denotes the seemingly best performing levels of β and r by rank, and blue shading reflects the relative magnitude of the ranks across the rows (left side) and columns (right side), where darker shades reflect higher ranks.

the set of βs); (b) we are evaluating each factor at every level of interest; (c) we are interested in which, if any, combination(s) of the factors produce the minimum mean; and (d) the number of iterations at each combination of the factors is the same. Accordingly, our simulation experiment is a balanced, fixed effects two-factor factorial design, and the corresponding effects model for our simulation experiment is given by:

$$z^*_{r,\beta,l} = \mu + \tau_r + \gamma_\beta + (\tau\gamma)_{r,\beta} + \epsilon_{r,\beta,l} \quad \begin{cases} r = 1, 2, \cdots, p(E, N-1) \\ \beta = 0, 0.1, \cdots, 1 \\ l = 1, 2, \cdots, 1{,}000 \end{cases} \tag{7}$$

where μ is the overall mean effect; τ_r is the effect of the rth restricted partition; γ_β is the effect of β at its indicated level; $(\tau\gamma)_{r,\beta}$ is the effect of the interaction between τ_r and γ_β; and $\in_{r,\beta,i}$ is a random error component (Montgomery, 2005, p. 165). Furthermore, if we assume the $\in_{r,\beta,l}$ are independently and normally distributed with mean 0 and variance σ^2 for each (r, β) pair, we can use two-way analysis of variance (ANOVA) to perform an omnibus F test of the following hypotheses (Montgomery, 2005, p. 166):

$$\begin{aligned} &H_0: \tau_1 = \tau_2 = \cdots = \tau_{p(E,N-1)} = 0 \\ &H_1: \text{at least one } \tau_r \neq 0 \end{aligned} \tag{8}$$

$$\begin{aligned} &H_0: \gamma_0 = \gamma_{0.1} = \cdots = \gamma_1 = 0 \\ &H_1: \text{at least one } \gamma_\beta \neq 0 \end{aligned} \tag{9}$$

$$\begin{aligned} &H_0: (\tau\gamma)_{r,\beta} = 0 \text{ for all } r, \beta \\ &H_1: \text{at least one } (\tau\gamma)_{r,\beta} \neq 0\,. \end{aligned} \tag{10}$$

Unfortunately, the $Z^*_{r,\beta,l}$ constitute a finite set of positive integers; therefore, the $\in_{r,\beta,l}$ will not be normally distributed. Nonetheless, with respect to the Type I error rate, the F test is robust against violations of the normality assumption (Donaldson, 1968), even when the dependent variable assumes a very small number of discrete values (Bevan, Denton, & Myers, 1974). Additionally, although the F test assumes equality of variance in the $\in_{r,\beta,l}$, when this assumption is violated in balanced designs numerous studies have shown that the actual probability of committing a Type I error closely matches the nominal level of significance (e.g., Glass, Peckham, & Sanders, 1972). That said, these studies typically employ experimentation versus analytical derivation, and generalizing their conclusions to our specific situation seems questionable. With this in mind, we can alleviate any issues by halving the nominal level of significance (i.e., from α = 0.05 to α = 0.025), as Keppel and Wickens (2004) note this is "the fastest and simplest way to eliminate concerns about heterogeneity" (p. 152). Using this approach, we performed a two-way ANOVA on the output of our simulation experiment, and the results are summarized in Table 5.

As seen in Table 5, both of the main effects and the interaction effect are significant at α = 0.025,

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F_0	p-value
r	243083	21	11575	1607.38	0
β	77610	10	7761	1077.71	0
Interaction	9043	210	43	5.98	0
Error	1740994	241758	7		

Table 5. Results of two-way ANOVA.

and we reject the null hypotheses captured in Equations (8), (9), and (10) in favor of their alternatives [Endnote 6]. Additionally, given the significant interaction effect, the focus of our post -hoc testing is on the individual cell means in Figure 7 versus the row or column means.

With this in mind, we can treat each of the 242 (r, β) pairs as separate levels of a single factor, thereby reducing our two-way ANOVA problem to a one-way problem. Moreover, as we are interested in whether there is an (r, β) pair with the smallest population mean, Hsu's multiple comparisons with the best (MCB) procedure (Hsu, 1984) is an appropriate post-hoc test. In particular, if we denote the population mean of the ith (r, β) pair as μ_i, Hsu's MCB constructs simultaneous, two-sided confidence intervals for $\mu_i - \min_{j \neq i} \mu_j$, $i, j = 1, 2, \cdots, 242$ such that the family

-wise error rate (FWER) is controlled at a specified level (typically 0.05). If a confidence interval contains 0, then the population means of the corresponding (r, β) pairs are deemed equivalent and optimal. Otherwise, a statistically significant difference exists, and the sign of the confidence interval's bounds determines which (r, β) pair is best. With this in mind, we set the FWER at 0.025 to account for the known heterogeneity, and we ran Hsu's MCB in the statistical software Minitab (2015) [Endnote 7]. The results of this post-hoc testing are given Figure 9.

As Figure 9 indicates, although the ($r = 5$, $\beta = 0.02$)pair has the smallest sample mean, Hsu's MCB suggests that 23 additional (r, β) pairs are also optimal. Interestingly, the optimal values of β range from 0 to 0.6, with the majority falling on 0.2 and 0.3. This suggests that the preferential attachment mechanism is sublinear, and this fits with our previous likelihood ratio testing (see Table 4). Specifically, although we failed to reject the null hypothesis that the discrete power law and exponential distributions are equally far from System 3's true degree distribution, the test statistic is slightly negative. As such, the evidence (albeit not statistically significant) favors the exponential distribution, and it hints that the preferential attachment mechanism is more uniform than linear. Nonetheless, in the absence of additional evidence, the optimal P($M = m$) and values for β in Table 6 constitute a set of equally compelling conditions for generating an incoming subsystem's interfaces and preferentially attaching them to System 3.

Restricted Partition Number (r)	β											
	1	0.9	0.8	0.7	0	0.6	0.1	0.5	0.2	0.3	0.4	Mean
1	13.21	12.59	11.92	12.00	11.68	11.52	11.70	11.65	11.49	11.43	11.20	11.85
2	11.90	11.39	10.92	10.41	10.34	10.22	10.24	10.07	10.02	10.09	10.05	10.51
22	11.30	10.77	10.46	10.05	10.09	10.07	9.87	9.73	9.72	9.64	9.64	10.12
21	10.38	9.92	9.53	9.31	9.60	9.07	9.38	9.02	9.15	8.92	9.06	9.39
6	10.66	10.35	9.77	9.45	9.08	9.12	8.88	8.88	8.93	8.74	8.62	9.32
3	10.62	10.10	9.65	9.04	8.61	8.81	8.41	8.60	8.56	8.45	8.42	9.02
19	10.06	9.46	9.16	9.04	9.13	8.76	8.94	8.59	8.88	8.59	8.56	9.02
20	9.65	9.08	9.01	8.65	8.96	8.36	8.73	8.43	8.52	8.60	8.44	8.77
16	9.69	9.15	8.72	8.57	8.26	8.31	8.29	7.97	8.00	8.17	8.06	8.47
17	9.40	8.90	8.53	8.53	8.45	8.35	8.34	8.10	8.22	8.14	8.07	8.46
11	9.75	9.38	8.80	8.37	8.06	8.20	8.04	8.16	7.84	7.97	7.80	8.40
13	9.74	8.97	8.64	8.25	8.38	8.07	8.15	7.98	7.90	7.97	7.92	8.36
18	9.25	8.60	8.43	7.96	8.44	7.87	8.14	7.85	8.13	7.77	7.72	8.20
10	9.51	8.95	8.42	8.20	7.93	7.80	7.84	7.71	7.68	7.55	7.61	8.11
7	9.76	9.19	8.61	8.13	7.56	7.94	7.50	7.72	7.38	7.50	7.52	8.07
4	10.06	9.34	8.67	8.31	7.45	7.90	7.37	7.47	7.29	7.32	7.27	8.04
12	9.37	8.74	8.43	8.16	7.95	7.81	7.71	7.55	7.59	7.52	7.58	8.04
14	9.19	8.65	8.29	8.08	7.98	7.74	7.91	7.54	7.63	7.57	7.70	8.03
8	9.64	9.09	8.46	8.03	7.72	7.72	7.60	7.51	7.36	7.45	7.58	8.02
5	9.67	9.22	8.59	8.02	7.22	7.69	7.07	7.41	7.046	7.13	7.22	7.85
15	8.85	8.22	7.97	7.67	8.06	7.36	7.73	7.20	7.55	7.20	7.32	7.74
9	9.31	8.58	8.18	7.73	7.42	7.55	7.23	7.24	7.23	7.18	7.054	7.70
Mean	10.04	9.49	9.05	8.73	8.56	8.47	8.41	8.29	8.28	8.22	8.20	

Figure 9. Results of Hsu's MCB with FWER 0.025, where the rows and columns of Figure 7 have been permuted by descending sample means. Yellow shading denotes the (r, β) pair with the smallest (optimal) sample / population mean ($\mu_{r=5,\beta=0.2}$); green shading indicates additional (r, β) pairs with population means equal to ($\mu_{r=5,\beta=0.2}$); and blue shading reflects the relative magnitude of the sample means across the remaining, suboptimal (r, β) pairs, where darker shades reflect larger means.

	Optimal P($M = m$)						Optimal β						
	m						β						
r	1	2	3	4	5	6	0	0.1	0.2	0.3	0.4	0.5	0.6
4	0.889			0.056		0.056	1	1	1	1	1		
5	0.889				0.111		1	1	1	1	1	1	
7	0.833	0.056	0.056			0.056			1				
8	0.833	0.056		0.056	0.056				1	1			
9	0.833		0.111		0.056		1	1	1	1	1	1	
15	0.778		0.222							1	1	1	1

Table 6. Optimal P(M = m) and values for β. The restricted partitions (r) listed in the leftmost column appear at least once in an optimal (r, β) pair (i.e., the corresponding rows of yellow or green-shaded cells in Figure 9), and these have been transformed into their corresponding PMFs in the panel titled "Optimal P(M = m)." Similarly, in the panel titled "Optimal β," 1s indicate that the corresponding values of β are part of an associated optimal (r, β) pair (i.e., the corresponding columns of yellow or green-shaded cells in Figure 9).

Incorporating Findings into a Modified, Data-Driven Approach

As seen above, using linear preferential attachment to connect an incoming subsystem to System 3's existing architecture is ill-advised. Moreover, as seen in Table 7, ANOVA and Hsu's MCB analysis of Systems 4, 6, 7, 8, 11, 14, 15, 17, 18, 20, and 24 reinforce the appropriateness and importance of using a data-driven approach.

In particular, while Brown-Forsythe testing found statistically significant heterogeneity in each system's data except System 24, the maximum to minimum variance ratios were less than 4 to 1, allowing us to safely proceed with ANOVA using half the nominal level of significance. As with System 3, ANOVA identified significant main effects in every system expect System 24, and for systems with multiple restricted partitions, the interaction effect was also significant. Subsequent Hsu's MCB analysis for the ten systems with significant effects revealed optimal β ranging from 0 to 1, with some systems favoring uniform attachment (i.e., Systems 14, 15, and 17) and others leaning towards linear attachment (i.e., Systems 4, 6, and 11). Additionally, with the exception of System 15, every system had multiple optimal (r, β) pairs, yet the number of optimal pairs was a fraction of the total number of pairs, especially for systems with multiple restricted partitions.

Simply put, real-world SV-3s suggest a one-size-fits-all approach is overly simplistic, and this also applies to the statistical methods we employed. For example, Systems 5 and 16 are not represented in Table 7, and this is a deliberate omission. Specifically, Systems 5 and 16 have 2,417 and 98,222 restricted partitions, respectively, and, assuming a significant interaction effect and with 11 levels of β, this implies the simultaneous testing of 26,587 and 1,080,442 hypotheses via Hsu's MCB. This is well over 20 times the number of hypotheses tested in the next closest system in Table 7, and it falls into the domain of large-scale simultaneous inference, where minor deviations from the theoretical null hypothesis can substantially affect the results (Efron, 2012). In short, more advanced methods are necessary to analyze these systems.

Beyond statistical methods, our approach is limited to simultaneously finding PMFs for an incoming subsystem's number of interfaces and estimating the strength of preferential attachment. It does not address architectural communities. For example, during the growth of System 3 using the ($r = 5$, $\beta = 0.02$) pair, we applied the Girvan-Newman community detection heuristic to each of the 1,000 simulated SV-3s, producing the histogram and kernel density plot seen in Figure 10.

Preprocessing					ANOVA Source of Variation p-value (α = 0.025)			Hsu's MCB Analysis (FWER = 0.025)		
System #	# of restricted partitions	# of feasible (r, β) pairs	Brown-Forsythe p-value	Max to Min Variance Ratio	r	β	Interaction	# of optimal (r, β) pairs	r	β
4	11	121	0	2.2	0	0	0	6	1	{0.5,…,1}
6	1	11	0	1.89	NA	0	NA	5	1	{0.6,…,1}
7	2	22	0	2.08	0	0	0.002	6	2	{0,…,0.5}
8	5	55	0	1.47	0	0	0	5	5	{0,…,0.4}
11	1	11	0	1.27	NA	0	NA	2	1	{0.9, 1}
14	15	165	0	2.45	0	0	0	10	7	{0, 0.1, 0.2}
									10	{0.2}
									12	{0,…,0.3}
									14	{0}
15	1	11	0	1.44	NA	0	NA	1	1	{0}
17	1	11	0	1.61	NA	0	NA	3	1	{0, 0.1, 0.2}
18	101	1111	0	3.62	0	0	0	31	5	{0.3,…,0.8}
									6	{0.3, 0.5}
									10	{0.5,…,0.8}
									11	{0.4, 0.5, 0.7}
									14	{0.7}
									23	{0.6, 0.7, 0.8}
									40	{0.6, 0.7, 0.8}
									41	{0.7}
									58	{0.6,…,0.9}
									72	{0.7, 0.8, 0.9}
									95	{1}
20	1	11	0	1.72	NA	0	NA	3	1	{0, 0.1, 0.2}
24	1	11	0.8716	1.15	NA	0.25	NA	NA	NA	NA

Table 7. ANOVA and Hsu's MCB analysis of remaining one-mode, undirected SV-3s. Systems with a single restricted partition only have effects due to β*; thus, their interaction effects are undefined.*

Based on Figure 10 and recalling Table 1, these results are encouraging. After all, System 3's SV-3 displayed strong community structure with four communities and a modularity of 0.365. In Figure 10, the plurality of the 1,000 simulated SV-3's contained four communities, and, when this occurred, the mean modularity was 0.361 – a difference of just 1%. When we consider that our simulation does not control for community structure, this miniscule difference in modularity is remarkable. Nonetheless, a majority of the 1,000 simulated SV-3s did not have four communities; therefore, we cannot claim our growth mechanism reliably replicates System 3's community structure.

With this in mind, we see the PMF for an incoming subsystem's number of interfaces as a system-level property. Thus, unlike Algorithm 1, we no longer recommend calculating separate PMFs for an incoming subsystem's number of

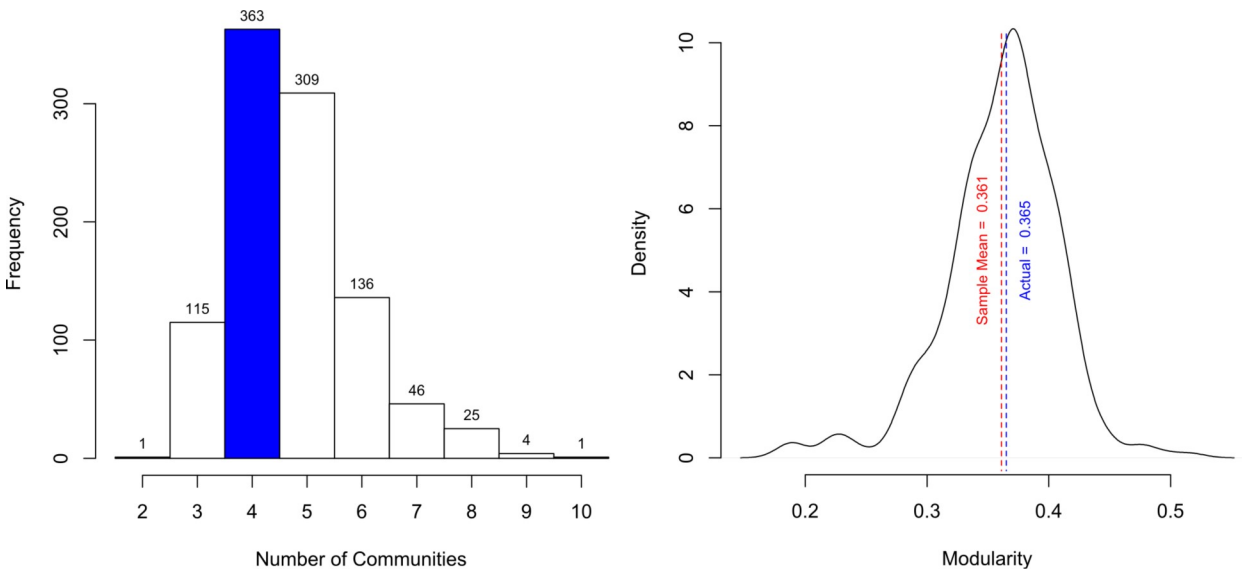


Figure 10. Histogram and kernel density plot reflecting the community structure of System 3's simulated growth using the r=5,β=0.02 pair.

intra and intercommunity interfaces, and we adopt a different approach. Specifically, if an incoming subsystem generates m interfaces and it is subsequently assigned to community k, we can view the intracommunity designation of the m interfaces as a sequence of m Bernoulli trials, where the probability of success (p) is given by:

$$\frac{\text{number of community } k\text{'s intracommunity interfaces}}{\text{number of community } k\text{'s intra and intercommunity interfaces}}$$

When viewed in this way, an incoming subsystem's number of intracommunity interfaces, M_{intra}, is simply a Binomial (m, p) random variable. Of course, it is possible that M_{intra} could generate a realization (m_{intra}) that exceeds the size of community k (N_k). Accordingly, we can set the number of intra and intercommunity interfaces as m'_{intra} = min $\{m_{intra}, N_k\}$ and $m_{inter} = m - m'$ respectively.

Using this approach and in light of our previous findings, we recommend the following, modified version of Algorithm 1:

Algorithm 2

For a specified, suitably large number of iterations . . .

Preprocessing

(1) Initialize the system as the current system,

(2) Build an optimal set of {P(M = m), β} pairs,

(3) Use Girvan-Newman to identify architectural communities and calculate modularity,

Growth

(4) Randomly select a member from the optimal set of {P(M = m), β} pairs,

(5) Generate a realization for the incoming subsystem's (**X**'s) number of interfaces using P(M = m); if the modularity suggests strong community structure, use **Connection Option A**; otherwise, use **Connection Option B**,

Connection Option A

(6a) Randomly assign **X** to community *k,*

(6b) Model M_{intra} as a Binomial (m, p) random variable; generate a realization for M_{intra} ; and set the number of intra and intercommunity interfaces as $m'_{intra} = \min\{m_{intra}, N_k\}$ and $m'_{inter} = m - \{m'_{intra}\}$ respectively,

(6c) Attach **X** to m'_{intra} subsystems inside community k and m_{inter} subsystems outside community k using attachment probabilities $p_i = d_i^{\beta} / \sum_{j=1}^{N} d_j^{\beta}$

(6d) For each intracommunity interface established in (6c), assign complexity ($w_{i\mathbf{X},\text{intra}}$),

(6e) For each intercommunity interface established in (6c), assign complexity ($w_{i\mathbf{X},\text{inter}}$),

Connection Option B

(6a) Attach **X** to m subsystems using attachment probabilities,

$$p_i = d_i^{\beta} / \sum_{j=1}^{N} d_j^{\beta}$$

(6b) For each interface established in (6a), assign complexity ($w_{i\mathbf{X}}$),

Cost Estimation

(7) Estimate the cost for the augmented system using COSYSMO (PM_{NS}*),

(8) Calculate the additional cost of adding subsystem **X** (PM_{NS}* – PM_{NS}), and

(9) Store results and return to (4).

Limitations and Directions for Future Research

Although Algorithm 2 addresses several of Algorithm 1's shortcomings, it still has limitations. First, as indicated earlier, SV-3s are not currently weighted by interface complexity. Thus, the validity of using the observed interface complexity distribution to estimate future interface complexity in Steps (6b), (6d), and (6e) above cannot be assessed, and further research is necessary. Second, while Algorithm 2 accepts one -mode, undirected SV-3s as input, real-world SV-3s can be two-mode or directed. With this in mind, methods that accommodate these SV-3 types could yield additional, valuable information and should be explored. Third, Algorithm 2 employs the Girvan-Newman community detection heuristic, and, despite its appropriateness, better performing heuristics exist (see Danon, Diaz-Guilera, Duch, & Arenas, 2005). Nonetheless, any community detection method, regardless of its performance, may ignore other, more compelling marcostructures within the architecture. For example, subsystems may partition into a hierarchy of clusters, where subsystems in lower ranking clusters not only have a high density of interfaces with subsystems inside their clusters but also have a high density of interfaces with subsystems inside higher ranking clusters. To identify this and other hidden macrostructure, one can apply the network analysis technique known as blockmodeling, and this represents an intriguing way to generalize the current approach.

Conclusions

The requirement to submit DoD component-approved DoDAF models prior to MS A has created interesting, new possibilities for the early life cycle cost estimation of MDAPs. In particular, Valerdi et al. (2015) demonstrate that the DoDAF models required Pre-MS A nearly span COSYSMO's parameters, and Dabkowski et al. (2014) exploited this mapping in Algorithm 1 by estimating the cost of unanticipated, evolutionary architectural growth via the SV-3 and COSYSMO. Although this development could be seen as positive, any cost estimation procedure ultimately needs to be informed and validated by real-world data. Accordingly, in this paper, we

examined the assumptions underlying Algorithm 1 using the SV-3s from 24 different defense programs.

The results were mixed. Specifically, while the type, density, and community structure of real-world SV-3s generally comport with Algorithm 1, modifications and extensions are necessary to accommodate two-mode or directed SV-3s. Moreover, using the observed degree distribution to model an incoming subsystem's interfaces generates too many interfaces, and an alternative growth mechanism is needed. Finally, although there is statistical support for linear preferential attachment as a method of connecting subsystems, the SV-3s are small, and our statistical results inevitably lack power.

With this in mind, we developed a modified, data-driven approach that addresses several of these concerns. In particular, using number theory, network science, simulation, and statistical analysis, we were able to find optimal sets of PMFs and strengths of preferential attachment for 12 of the 14 one-mode, undirected SV-3s. Integrated into Algorithm 2, these optimal sets better represent a system's evolutionary growth, and they improve the fidelity of the algorithm. Nonetheless, as noted earlier, our approach has several limitations, and these represent opportunities for future research.

Aside from developing Algorithm 2, this paper also makes several tangential contributions to network science. First, to the best of our knowledge, this is the first attempt at simultaneously estimating a growing network's incoming edge distribution and detecting the strength of preferential attachment. To date, these efforts have been disconnected, and linking them is not only natural but also necessary in light of their significant interaction effect. Second, assessing the presence of linear preferential attachment has traditionally been confined to large networks with longitudinal data. By modeling the set of feasible edge sequences via restricted partitions, we have provided researchers with a way to accommodate small networks with a single realization. Last, while other metrics exist, our use of the minimum sum of the absolute pairwise differences between the degrees of two identically-sized networks provides analysts with an intuitive measure of the similarity between their degree distributions. Taken together, these contributions highlight the benefits of applying network science to new domains, and they reinforce the value of viewing DoDAF models as computational objects.

Acknowledgements:

This material is based upon work supported by the Naval Postgraduate School Acquisition Research Program under Grant No. N00244-13-1-0032 and the Office of the Secretary of Defense.

Disclaimer:

The views expressed herein are those of the authors and do not reflect the position of the United States Military Academy, the Department of the Army, or the Department of Defense.

Contributions:

Author contributions are as follows: M.D. and R.V. designed research; M.D. analyzed data; M.D. and A.M. performed research and wrote the paper.

Endnotes

1. For those familiar with DoDAF, this seems to violate the generally prescribed structure of the SV-3. However, during DoDAF's most recent, major revision, models shifted from a template-driven paradigm to a "fit for purpose" construct (DoD DCIO, 2010, p. 3).

2. In practice, two-mode networks are often transformed into one-mode networks to facilitate analysis (Borgatti, 2009). For example, imagine a two-mode network **X**, where rows represent professors, columns denote institutional committees, and a 1 in cell (*i, j*) implies professor *i* is a member of committee *j*. Under the assumption that co-memberships in committees imply meaningful connections between professors, $\mathbf{XX}^{\mathrm{T}}$ yields a useful one-mode, valued network **A**, where rows and columns represent the professors and values provide the number of co-memberships for each pair of professors. Further binarizing **A** (such that cells greater than 1 are set to 1) yields a simple one-mode network. That said, using this approach for a two-mode SV-3 is ill-advised, as internal subsystems that interface with common external subsystem(s) do not necessarily interface with one another.

3. While the details of estimating $d_{\min}$ are beyond the scope of this work, Clauset et al. (2009) note that $\hat{d}_{\min}$ is selected such that its value "makes the probability distributions of the measured data and the best-fit power-law model as similar as possible above $\hat{d}_{\min}$ " (p. 671).

4. Data to the left of d_{min} is discarded prior to estimating ω; therefore, $N_{\text{fit}} \leq N_{\text{total}}$.

5. To ascertain the validity of the final inequality, note: (a) prior to the ith oldest subsystem entering the SV-3 there are *i* subsystems in it, which implies the ith oldest subsystem can connect to at most i subsystems and (b) if $E_i \geq 1$, E_j is maximized when $E_i = 1$ for all $i \neq j$, which implies $E_j = E - (N - 2)$.

6. As stated previously, halving the nominal level of significance provides a reasonable hedge against unequal variance in the $\epsilon_{r,\beta,I}$; however, if the variances are equal, it is unnecessary. Accordingly, given the non-normality of the data, we evaluated the homogeneity of the residuals using the Brown-Forsythe test, and this returned a *p*-value of 0, firmly rejecting the variances of the $\epsilon_{r,\beta,I}$ are equal. That said, unequal variances "typically cause Type I error rates to be slightly inflated . . . less than 0.02 at the 0.05 level . . . provided the ratio of the largest to the smallest variance is no more than 4 to 1, and *n* is at least 5" (Myers, Well, & Lorch, 2010, p. 191). In our case, the maximum and minimum variances are 14.02 (for ($r = 22$, $\beta = 0.9$) and 4.35 (for ($r = 5$, $\beta = 0.1$)), yielding a ratio of 3.22. Accordingly, halving the nominal level of significance is appropriate.

7. In fact, Hsu's MCB is equivalent to Dunnett's multiple comparisons with a control (MCC) procedure, where the control condition is selected as the factor setting (i.e., (r, β) pair) with the minimum observed mean (Lawson, 2010, p. 47). This has favorable implications for the robustness of Hsu's MCB. Specifically, when the design is balanced, Dunnett's MCC is known to be robust against non-normality and unequal variance, provided the maximum to minimum variance ratio is less than 4:1 (Toothaker, 1993). As Toothaker (1993) notes, you can use Dunnett's MCC at $\alpha = 0.05$ "with little consequence of unequal variances if the maximum true α you would tolerate would be 0.075" (p. 61). Thus, as with our earlier two-way ANOVA, we halved our nominal FWER to 0.025.

References:

Architecture Integration and Management Directorate (AIMD), Army Capabilities Integration Center (2014). *SV-3s from select defense programs 2011-2014*. [Data file]. Retrieved from the Marine Collaborative Architecture Environment (MCAE).

Banavar, J. R., Maritan, A., & Rinaldo, A. (1999). Size and form in efficient transportation networks. *Nature*, 399(6732), 130-132. doi:10.1038/20144

Barabási, A. L. (2015). Non-linear preferential attachment. In *Network Science* (Chapter 5, Section 7). Retrieved from http://barabasi.com/networksciencebook/content/book_chapter_5.pdf

Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439), 509-512. doi:10.1126/science.286.5439.509

Barabási, A. L., Albert, R., & Jeong, H. (1999). Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications*, 272(1), 173-187. doi:10.1016/S0378-4371(99)00291-5

Bevan, M. F., Denton, J. Q., & Myers, J. L. (1974). The robustness of the *F* test to violations of continuity and form of treatment population. *British Journal of Mathematical and Statistical Psychology*, 27(2), 199-204. doi:10.1111/j.2044-8317.1974.tb00540.x

Blanchard, B., & Fabrycky, W. (1998). *Systems engineering and analysis* (3rd ed.). Upper Saddle River, NJ: Prentice Hall.

Bolten, J. G., Leonard, R. S., Arena M. V., Younossi, O., & Sollinger, J. M. (2008). *Sources of Weapon System Cost Growth - Analysis of 35 Major Defense Acquisition Programs*. Santa Monica, CA: RAND Corporation. Retrieved from http://www.rand.org/content/dam/rand/pubs/monographs/2008/RAND_MG670.pdf

Borgatti, S. P. (2009). Two-Mode Concepts in Social Network Analysis. In R. A. Meyers (Ed.), *Encyclopedia of complexity and systems science* (pp. 8281-8291). New York, NY: Springer.

Budget Control Act of 2011, Pub. L. No. 112–25. 125 Stat. 240 (BCA). (2011). Retrieved from http://www.gpo.gov/fdsys/pkg/PLAW-112publ25/html/PLAW-112publ25.htm

Cancho, R. F., Janssen, C., & Solé, R. V. (2001). Topology of technology graphs: Small world patterns in electronic circuits. *Physical Review E*, 64(4), 046119. doi:0.1103/PhysRevE.64.046119

Chairman of the Joint Chiefs of Staff (CJCS) (2012a, January 19). *Manual for the Operation of the Joint Capabilities Integration and Development System*. Retrieved from https://dap.dau.mil/policy/Documents/2012/JCIDS%20Manual%2019%20Jan%202012.pdf

Chairman of the Joint Chiefs of Staff (CJCS) (2012b, September 20). *JCIDS Manual Errata*. Retrieved from https://dap.dau.mil/policy/Documents/2012/JCIDS%20Manual%20Errata%20-%2020%20Sept%202012.pdf

Chairman of the Joint Chiefs of Staff (CJCS) (2015, February 12). *Manual for the Operation of the Joint Capabilities Integration and Development System*. Retrieved from https://dap.dau.mil/policy/Documents/2015/JCIDS_Manual_-_Release_version_20150212.pdf

Chairman of the Joint Chiefs of Staff (CJCS) (2018, August 31). *Manual for the Operation of the Joint Capabilities Integration and Development System*.

Clauset, A., Shalizi, C. R., & Newman, M. E. (2009). Power-law distributions in empirical data. *SIAM Review*, 51(4), 661-703. doi:10.1137/070710111

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal - Complex Systems*, 1695. Retrieved from http://igraph.org

Dabkowski, M., Valerdi, R., & Farr, J. (2014). Exploiting Architectural Communities in Early Life Cycle Cost Estimation. *Procedia Computer Science*, 28, 95-102. doi:10.1016/j.procs.2014.03.013

Danon, L., Diaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09), P09008. doi:10.1088/1742-5468/2005/09/P09008

Darling, D. A. (1957). The Kolmogorov-Smirnov, Cramer-von Mises Tests. *The Annals of Mathematical Statistics*, 28(4), 823–838. Retrieved from http://www.jstor.org/stable/2237048

Department of Defense (DoD). (n.d.). *Spotlight - National Defense Strategy*. Retrieved October 18, 2021, from https://www.defense.gov/Spotlights/National-Defense-Strategy/

Department of Defense (DoD). (2018). *Summary of the 2018 National Defense Strategy of the United States of America*. https://dod.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf

Department of Defense Deputy Chief Information Officer (DoD DCIO) (2010, August). *The DoDAF Architecture Framework Version 2.02*. Retrieved from http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf

Donaldson, T. S. (1968). Robustness of the F-test to errors of both kinds and the correlation between the numerator and denominator of the F-ratio. *Journal of the American Statistical Association*, 63(322), 660-676. doi:10.1080/01621459.1968.11009285

Dorogovtsev, S. N., & Mendes, J. F. (2003). Non-equilibrium networks. In *Evolution of networks: From biological nets to the Internet and WWW* (Chapter 5). New York, NY: Oxford University Press. doi:10.1093/acprof:oso/9780198515906.003.0006

Dowlatshahi, S. (1992). Product design in a concurrent engineering environment: an optimization approach. *International Journal of Production Research*, 30(8), 1803-1818. doi:10.1080/00207549208948123

Efron, B. (2012). Large-scale simultaneous hypothesis testing. *Journal of the American Statistical Association*, 99(465), 96-104. doi:10.1198/016214504000000089

Gillespie, C. S. (2015). Fitting Heavy Tailed Distributions: The poweRlaw Package. *Journal of Statistical Software*, 64(2), 1-16. Retrieved from http://www.jstatsoft.org/v64/i02/

Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12), 7821-7826. doi:10.1073/pnas.122653799

Glass, G. V., Peckham, P. D., & Sanders, J. R. (1972). Consequences of failure to meet assumptions underlying the fixed effects analyses of variance and covariance. *Review of educational research*, 237-288. Retrieved from http://www.jstor.org/stable/1169991

Government Accountability Office (GAO) (2011). *DoD Cost Overruns: Trends in Nunn-McCurdy Breaches and Tools to Manage Weapon Systems Acquisition Costs*. Retrieved from http://www.gao.gov/assets/130/125861.pdf

Gupta, H. (1970). Partitions–a survey. *Journal of Research of the National Bureau of Standards*, 74B(1), 1-29.

Hankin, R. K. S. (2006). Additive integer partitions in R. *Journal of Statistical Software - Code Snippets*, 16(1), 1-3.

Hsu, J. C. (1984). Constrained simultaneous confidence intervals for multiple comparisons with the best. *The Annals of Statistics*, 12(3), 1136-1144. doi:10.1214/aos/1176346732

Hunter, D. R., Goodreau, S. M., & Handcock, M. S. (2008). Goodness of fit of social network models. *Journal of the American Statistical Association*, 103(481), 248-258. doi:10.1198/016214507000000446

International Organization for Standardization (ISO). (2011). *Systems and software engineering – Architecture description* (ISO/IEC/IEEE Standard No. 42010). Retrieved from https://www.iso.org/standard/50508.html

Jeong, H., Néda, Z., & Barabási, A. L. (2002). Measuring preferential attachment in evolving networks. *Europhysics Letters*, 61(4), 567–572.

Keppel, G., & Wickens, T. D. (2004). *Design and analysis: A researcher's handbook* (4th ed.). Upper Saddle River, NJ: Pearson Prentice Hall.

Lawler, E. L. (1976). *Combinatorial optimization: networks and matroids*. Mineola, NY: Courier Corporation.

Lawson, J. (2010). *Design and Analysis of Experiments with SAS*. New York, NY: CRC Press.

Minitab 17.2.1 Statistical Software (2015). [Computer software]. State College, PA: Minitab, Inc. (www.minitab.com)

Montgomery, D. C. (2005). *Design and Analysis of Experiments* (6th ed.). New York, NY: John Wiley and Sons.

Myers, J. L., Well, A., & Lorch, R. F. (2010). *Research design and statistical analysis*. New York, NY: Routledge.

Newman, M. E. (2001). Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2), 025102. doi:10.1103/PhysRevE.64.025102

Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 026113. doi:10.1103/PhysRevE.69.026113

Peña, M., & Valerdi, R. (2015). Characterizing the Impact of Requirements Volatility on Systems Engineering Effort. *Systems Engineering*, 18(1), 59-70. doi:10.1111/sys.21288

Redner, S. (1998). How popular is your paper? An empirical study of the citation distribution. *The European Physical Journal B-Condensed Matter and Complex Systems*, 4(2), 131-134. http://dx.doi.org/10.1007/s100510050359.

Rubner, Y., Tomasi, C., & Guibas, L. J. (1998, January). A metric for distributions with applications to image databases. In *Proceedings from Sixth International Conference on Computer Vision*, 59-66. Bombay, India: IEEE.

Shore, J., & Lubin, B. (2015). Spectral goodness of fit for network models. *Social Networks*, 43, 16-27. http://dx.doi.org/10.1016/j.socnet.2015.04.004

Toothaker, L. E. (1993). *Multiple comparison procedures* (No. 89). Newbury Park, CA: Sage.

Under Secretary of Defense for Acquisition and Sustainment (USD(A&S)) (2020, January 23). *Operation of the Adaptive Acquisition Framework* (DoD Instruction 5000.02). Retrieved from https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500002p.pdf

Under Secretary of Defense for Acquisition and Sustainment (USD(A&S)) (2020, August 6). *Major Capability Acquisition* (DoD Instruction 5000.85). Retrieved from https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500085p.pdf

Under Secretary of Defense for Research and Engineering (USD(R&E)) (2020, May). *Modular Open Systems Approach (MOSA) Reference Frameworks in Defense Acquisition Programs*. Retrieved from https://ac.cto.mil/wp-content/uploads/2020/06/MOSA-Ref-Frame-May2020.pdf

Under Secretary of Defense for Research and Engineering (USD(R&E)) (2020, November 18). *Engineering of Defense Systems* (DoD Instruction 5000.88). Retrieved from https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500088p.pdf

Valerdi, R. (2008). *The Constructive Systems Engineering Cost Model (COSYSMO): Quantifying the Costs of Systems Engineering Effort in Complex Systems*. Saarbrücken, Germany: VDM Verlag.

Valerdi, R., Dabkowski, M., & Dixit, I. (2015). Reliability Improvement of Major Defense Acquisition Program Cost Estimates – Mapping DoDAF to COSYSMO. *Systems Engineering*. 18(5), 530-547. doi:10.1002/sys.21327

Wasserman, S., & Faust, K. (2009). *Social Network Analysis: Methods and Applications*. New York, NY: Cambridge University Press.

Matthew Dabkowski *is a Colonel in the US Army and an Associate Professor in the Department of Systems Engineering at the U.S. Military Academy. He has served in the United States Army for 24 years as an Infantry Officer, Operations Research Analyst, and Academy Professor. He is a graduate of West Point (B.S. in Operations Research) and the University of Arizona (M.S. in Systems Engineering and Ph.D. in Systems and Industrial Engineering). His research interests include applied statistics and simulation modeling.*

Arthur Middlebrooks *is a Major in the US Army and an Assistant Professor in the Department of Systems Engineering at the U.S. Military Academy. He has served in the United States Army for 11 years as an Armor Officer and Operations Research Analyst. He is a graduate of West Point (B.S. in Systems Engineering) and the Massachusetts Institute of Technology (M.S. in Engineering and Management). His research interests include value-based decision analysis and system dynamics simulation.*

Ricardo Valerdi *is a Distinguished Outreach Professor at the University of Arizona in the Department of Systems and Industrial Engineering. His research focuses on systems engineering, cost estimation, and sports analytics. He was the Founder and Co-Editor-in-Chief of the Journal of Enterprise Transformation and was the Editor-in-Chief of the Journal of Cost Analysis and Parametrics. He served on the Board of Directors of the International Council on Systems Engineering (INCOSE) and is a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE) and a Fellow of INCOSE. He won the Frank Freiman Award from ICEAA for his contributions to cost model development, is a recipient of the USC Center for Systems & Software Engineering Lifetime Achievement Award, and was elected as a foreign member to the Mexican National Academy of Engineering. He received a Ph.D. in Industrial and Systems Engineering from the University of Southern California.*

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:

<u>Published Literature Data</u>

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.

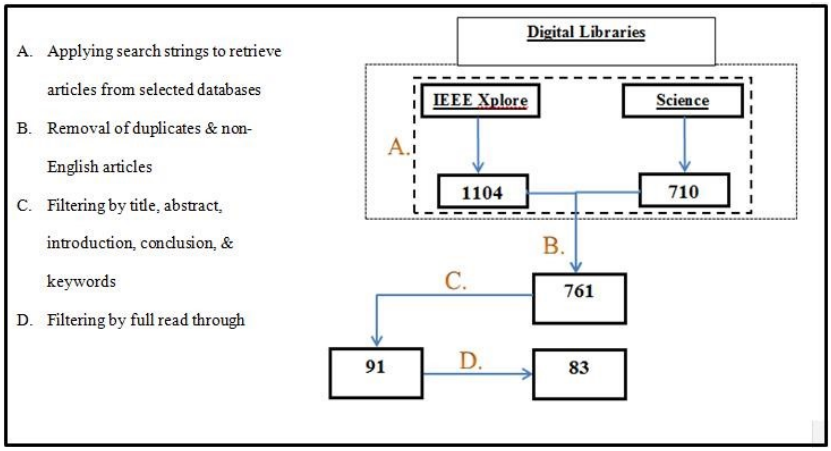


Figure 1. Article Search and Filter Process

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.

Air Force Practitioner Data

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.

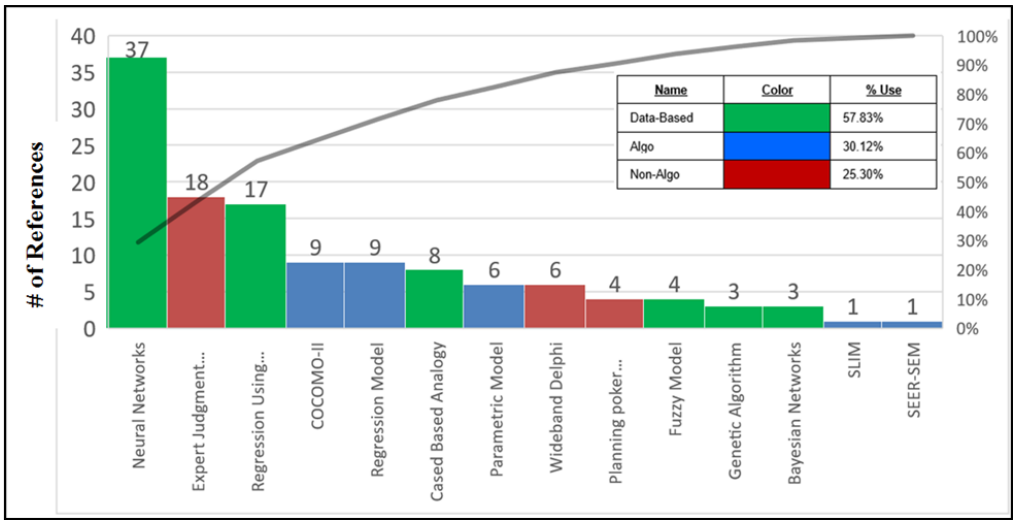


Figure 2. References to Software Effort Estimation Techniques

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-

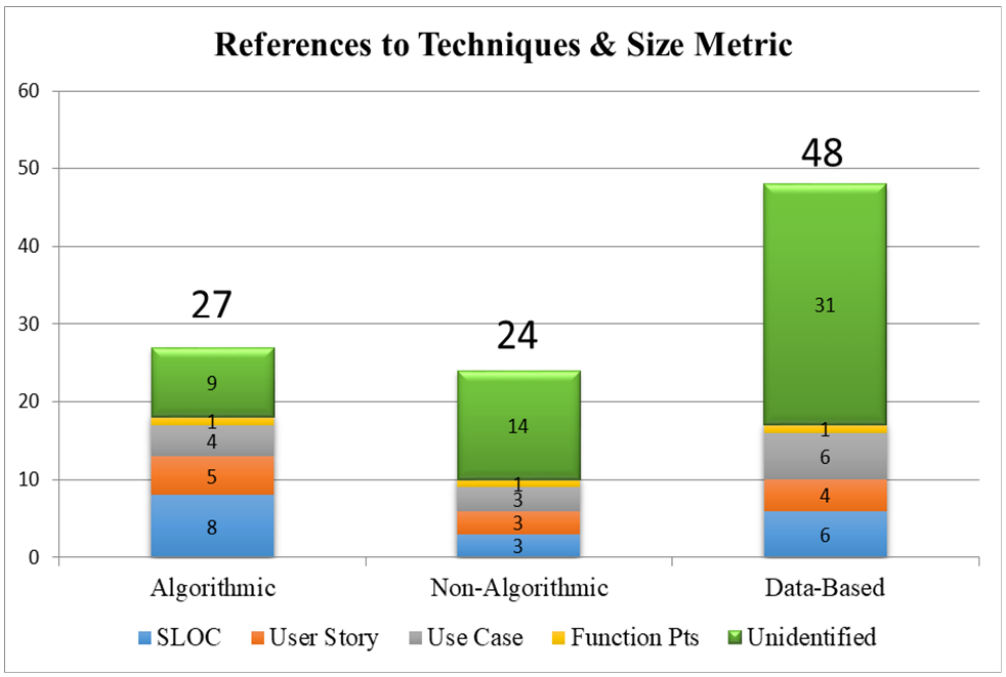


Figure 3: Number of References to Technique Styles Accounting for Size Metrics

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

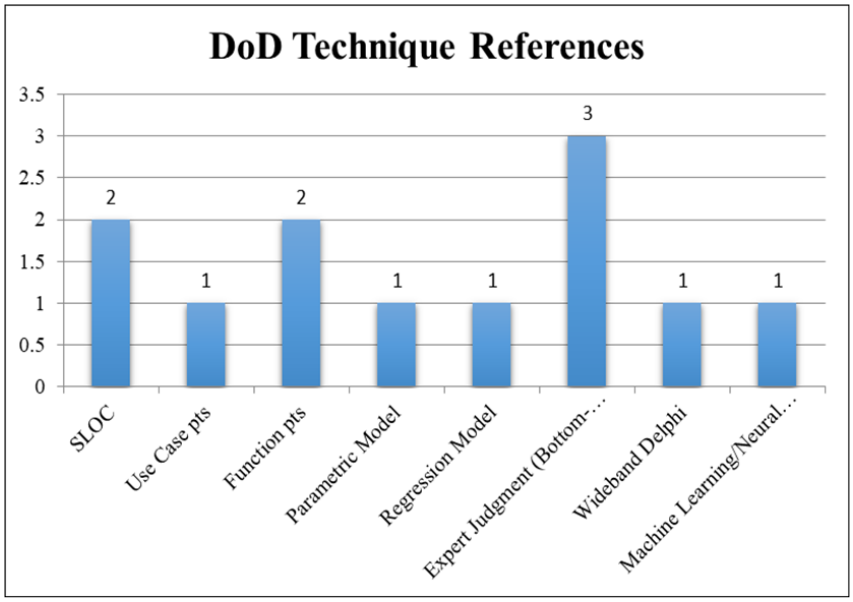


Figure 4. DoD Article References to Techniques

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).

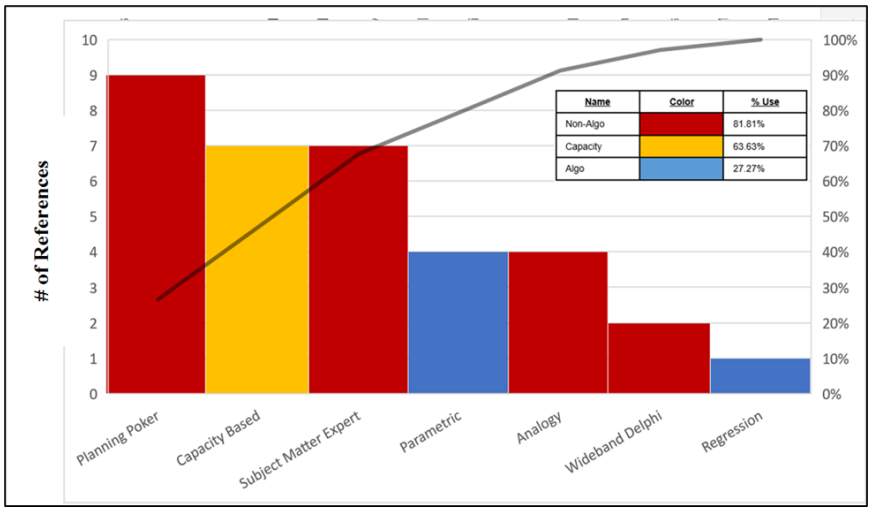


Figure 5. Software Factory References to Techniques

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.

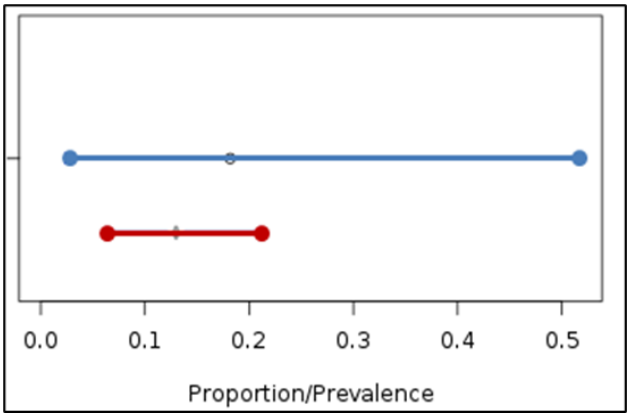


Figure 6. SLOC Usage

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.

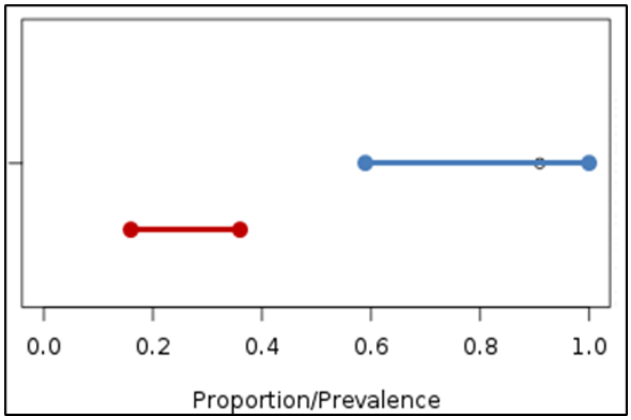


Figure 7. Non-Algorithmic Comparison

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 Compararitive 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.

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)*

Augustine's Law:
Are We Really Headed for the $800 Billion-Dollar Fighter?

Brent M. Johnstone

Abstract: Augustine's Law famously proposed fighter aircraft costs are growing so rapidly that by 2054 buying a single tactical aircraft will consume the entire defense budget. Is the situation really so dire? This paper examines the trend in U.S. fighter costs and relates them to generational changes in aircraft design and manufacture. It also examines the new jet fighters of the 2000s to see if Augustine's Law is really unfolding as its author originally thought.

Key words: Augustine's Law, cost analysis, fighter aircraft, government procurement, cost effectiveness, cost estimates, mathematical models, planning

"In the year 2054, the entire defense budget will purchase just one aircraft. This aircraft will have to be shared by the Air Force and Navy 3 ½ days each per week except for leap year, when it will be made available to the Marines for the extra day." – Augustine's Law XVI.

Introduction

In his book, *Augustine's Laws*, former aerospace executive Norman Augustine proposed a series of "laws" – more accurately, a series of tongue-in-cheek empirical observations -- about management behavior in the spirit of the late C. Northcote Parkinson. The mostly widely quoted of these is Augustine's assertion (quoted above) that the rate of increase in military tactical aircraft costs over time would eventually exceed the cost of the overall defense budget and even (past 2100) the gross national product (GNP) of the United States.

This seemingly absurd conclusion nonetheless had empirical justification, which Augustine produced in his book. Beginning with the Wright Brother's Model A in 1910, Augustine tracked the cost of U.S. tactical aircraft through the release of the F-18A/B in the early 1980's and observed an exponential growth over time. Figure 1, reproduced from Augustine's book, shows the then-year average unit cost increasing by a factor of four every decade.

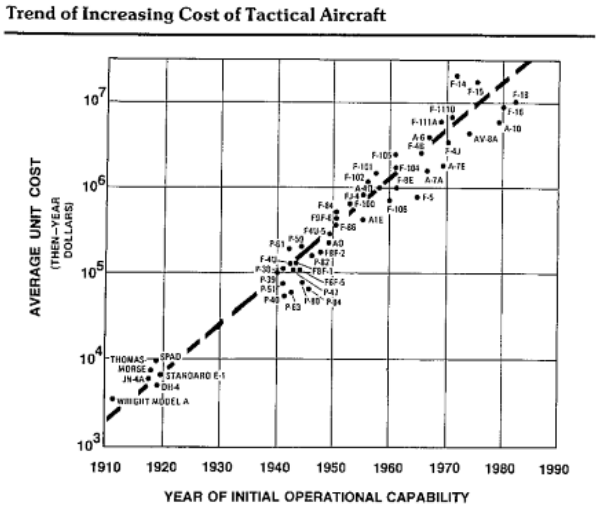


Figure 1: Reprinted with Permission of Norman Augustine

Extrapolating this trend into the future – as well as extrapolating the projected growth in the Defense Department budget and the nation's GNP – Augustine calculated that the price of a single tactical aircraft would equal the entire projected defense budget by 2054. This is shown in Figure 2, as well as the astonishing conclusion that the price of our single aircraft would eventually exceed the GNP sometime around the year 2150:

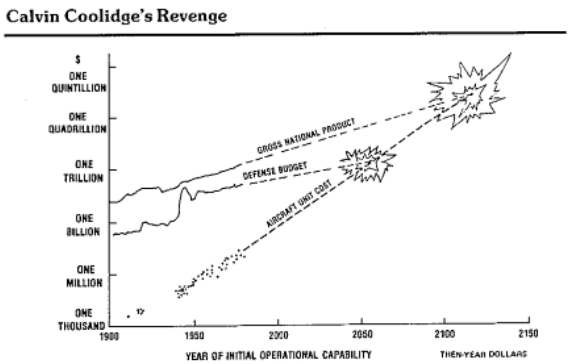


Figure 2: Reprinted with Permission of Norman Augustine

Augustine's book does not provide absolute numbers, but useful approximations can be made from reading the charts themselves. Cost increases for tactical aircraft by a factor of four every decade translates into a 15% increase per year. Likewise, from the chart, it is apparent that the defense budget is assumed to increase by 2.5% per year and the nominal GNP by 5.5% per year. This produces an intersection by 2054 where the defense budget and the cost of a single aircraft meet at approximately $800 billion. Likewise, by 2114 the GNP and the cost of a single aircraft meet at approximately $3.6 quadrillion.

Augustine's prediction first appeared in written form (Augustine, 1979) several years earlier as an article in the *Defense Systems Management Review*. When asked in 2016 about his prediction, Augustine not only confirmed his original prediction but gave it an exact date: "It was 2054. I've refined it actually to July 23, 2054. The *Economist* just came out with [an] update to my law and I'm sorry to say we're right on track." (Aitoro, 2016.)

Along the same vein, another "law" authored by Augustine relates time against months to first flight (reference Figure 3) and concludes that there has been no change in aircraft design and development spans despite the increasing complexity of military and commercial aircraft design and build over time:

> The duration of the design and build phase of aircraft development programs has remained virtually unchanged for 40 years. This period is approximately the same for government projects, commercial projects, and, for that matter, projects undertaken in the Soviet Union. (Augustine, 1983.)

Augustine's ironic projections have been seized upon by critics of the defense establishment as proof of wasteful spending, "gold plate" requirements and an industry which has little to no concern about controlling costs. Summarizing much of the published criticism, Franck writes: "The most commonly held belief (the 'conventional wisdom') regarding quality versus quantity choices is that the major weapon systems are laden with technological bells and whistles that add much to cost but little, if anything, to military effectiveness." (Franck, 1992). In

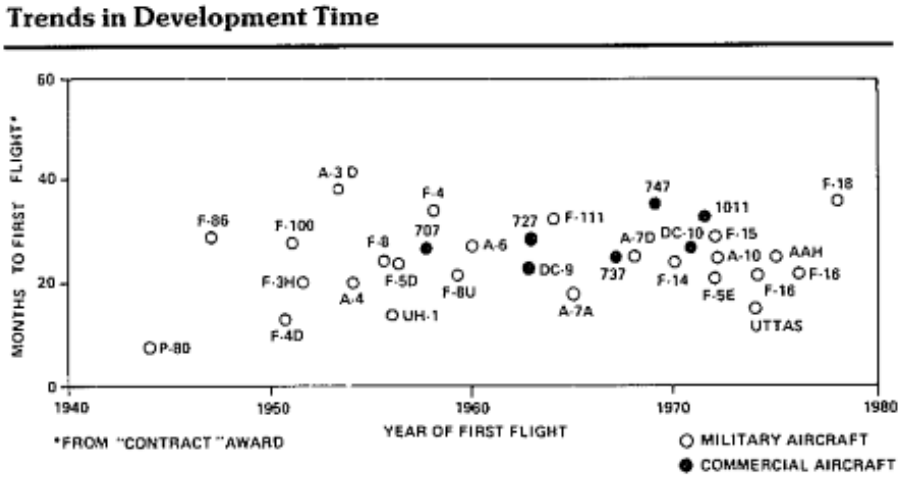


Figure 3: Reprinted with Permission of Norman Augustine

Augustine's view, the observed cost growth over time is not adequately explained by improvements in aircraft performance but is more closely linked to an engineering mindset that values technology for technology's sake – a mindset that has created a cost growth that is unsustainable over time:

> This rate of growth seems to be an inherent characteristic of such systems, with the unit cost being most closely correlated with the passage of time rather than with changes in maneuverability, speed, weight, or other technical parameters. The same inexorable trend can be shown to apply to commercial aircraft, helicopters, and even ships and tanks, although in the last two somewhat less technologically sophisticated instances, the rate of growth is a factor of two every ten years. Automobiles, houses, and certain other commercial products more nearly approximate this latter case. The point is not, of course, that new technology is inevitably more expensive than old technology; the opposite is often the case. But what happens is that…new technology opens vast new capability vistas which are then crammed into each new generation of a product. (Augustine, 1983.)

While Augustine's laws have been widely (and approvingly) quoted over the subsequent three decades, there have been criticisms of this analysis. Eskew (2000) pointed out three methodological issues in Augustine's analysis:

- His projection is based on then-year (inflated) dollars instead of adjusting all values back to a constant dollar base. A substantial portion of the cost growth is therefore due to the larger trend in monetary inflation over time. The Bureau of Labor Statistics on-line Consumer Price Index (CPI) inflation calculator shows that $10,000 in January 1913 would now be worth $261,370 in June 2019 – an increase of 2,514%. (BLS, 2019.)
- The analysis does not consider the total number of aircraft procured. All else being equal, the larger the production buy, the lower the average unit cost – the familiar learning curve effect. If the size of production buys has decreased over time, this could produce a systemic bias toward higher aircraft unit prices over time.
- Likewise, the analysis does not consider the number of aircraft produced in a single year. Larger lot buys are typically associated with lower costs due to the overhead savings due to larger business bases and the allocation of fixed support labor costs across larger quantities. Once again, smaller lot quantities over time could produce a systemic bias toward higher aircraft unit prices.

To Eskew's list, we can add:

- The definition of "average unit cost" as used by Augustine is a nebulous one. This could represent unit recurring flyaway (URF), or production average unit cost (PAUC); or average procurement cost (AUPC) – each of which would reveal a substantially different answer. It is unclear what definition is used by Augustine, or if it is applied consistently over time.

We come then to the purpose of this paper, which is to explore the following questions:

- If fighter aircraft are compared over time using a standardized definition of unit cost, after normalization for inflation and learning curve impacts, are the trends Augustine observed still apparent?
- Since Augustine's initial publication, three major fighter programs have been introduced (F-18E/F, F-22 and F-35). If we introduce this new data, does it change or confirm Augustine's projections?
- Fighter jet aircraft are substantially more complex than their post-World War II predecessors. What is the relationship between cost and each successive generation of fighter aircraft? How much do advances in capability cost historically?
- Regarding Augustine's assertion regarding the unchanging length of development programs, what does history for the most recent fighter programs (F-18E/F, F-22 and F-35) tell us?

Method of Analysis

To examine these issues more closely, a cost database from public domain sources was assembled. The primary source for cost data is the *U.S. Military Aircraft Cost Handbook* (DePuy, 1983). The *Handbook* database represents Total Obligational Authority (TOA) requested by the services to procure attack, bomber and fighter aircraft. Values are reported in then-year dollars and normalized to FY1981 dollars. The *Handbook* uses FY buy average costs to calculate airframe, airframe and engine, and total flyaway unit cost curves, allowing the calculation of theoretical T-100 values. The *Handbook* presents data for aircraft in the active U.S. inventory during FY1960-FY1980 period. This eliminates some of the first-generation fighters such as the P-80. The most recently introduced fighter aircraft in the MCR database is the F-18A/B series. In this analysis, the FY1981 cost data was escalated to FY2018 dollars using December 2018 U.S. DoD aircraft procurement escalation indices.

To include more recent aircraft introduced into the fleet, a supplemental database was created from public domain reports such as U.S. DoD budget documents (USAF, 2010, USN, 1999, USN, 2000, USN, 2001, USN, 2002, USN, 2003, USN, 2004, USN, 2005, USN, 2006, USN, 2007, USN, 2008, USN, 2009, USN, 2010, USN, 2011, DoD, 2012a, DoD, 2013, DoD, 2014) as well as General Accounting Office (GAO) reports (GAO, 2019). This provides then-year flyaway cost data by FY buy for the most recent aircraft introduced to the fleet: F-18E/F, F-22, and F-35. After normalization to FY2018 dollars using the same DoD escalation indices, unit curves were drawn from unit flyaway data and T100 theoretical values calculated.

In addition, aircraft empty weight information was pulled from public domain sources, most coming from RAND studies (Hess, 1987) supplemented by Selected Acquisition Report (SAR) data or Air Force and/or industry press releases (DoD, 2012b, USAF, 2015, Lockheed Martin, 2019). In all, cost and weight data were found for 23 U.S. fighter aircraft with Initial Operational Capability (IOC) dates ranging from 1949 to 2016.

A popular way to review military aircraft history is to talk of "generations" of fighter jet aircraft. F-22 and F-35 are commonly cited as "fifth generation" fighters, and their eventual replacements as "sixth generation." In truth there is no fully accepted definition of jet fighter generations. However, there is rough agreement on the timelines and characteristics associated with each fighter generation, although there may be disagreement on whether an individual aircraft should be classified as, say, second or third generation. Yoon (2004) suggests the following timeline to assess fighter development:

- First Generation Jet Fighters (circa 1945 to 1955) – Powered by the first turbojet engines, these post-World War II aircraft have capability like their piston engine predecessors. These aircraft are subsonic, usually do not carry radar and carry conventional weaponry such as machine guns and bombs but not guided missiles. First generation aircraft used in the sample were the North American F-86 and Northrop F-89. Unfortunately, other early examples from the U.S. inventory such as the Lockheed P-80 and F-94, Republic F-84, and the North American F-96 were eliminated because reliable T100 flyaway cost or weight data was not available.
- Second Generation Jet Fighters (circa 1955 to 1960) – This generation introduces the first supersonic combat aircraft. They also introduce radar and the use of guided missiles. Second generation aircraft used in the sample were the Douglas A-3; McDonnell A-4, F3H, and F-101; Convair F-102 and F-106; Lockheed F-104; and the Republic F-105.
- Third Generation Jet Fighters (circa 1960 to 1970) – This generation introduces multi-role fighters which combine air defense and ground attack missions in a single configuration. Third generation aircraft used in the sample were the McDonnell F-4, Grumman A-6, Vought A-7 and the General Dynamics F-111.

- Fourth Generation Jet Fighters (circa 1970 to 1990) – This generation continues the trend towards multi-role aircraft but improves capability with more advanced avionics and weapons systems. Greater emphasis is placed on maneuverability versus pure speed and incorporation of lessons learned from the Vietnam air war. Fourth generation aircraft used in the sample were the Grumman F-14, McDonnell Douglas F-15, F-18 and AV-8B, Fairchild A-10 and the General Dynamics F-16.
- Generation 4.5 Jet Fighters (circa 1990 to 2000) – This generation represents an upgrade to existing fourth generation aircraft but incorporates more advanced electronics and, to some degree, a reduced radar cross section (RCS) through the incorporation of limited stealth characteristics. The singular example of Generation 4.5 in the sample was the Boeing F-18E/F.
- Fifth Generation Jet Fighters (circa 2000 to Today) – This generation introduces low observable stealth, more powerful engines, and advanced integrated avionics. The F-35 also introduces shared battlefield awareness and the ability to work with a broad array of networked systems. Fifth generation aircraft used in the sample were the Lockheed Martin F-22 and F-35A. For analysis purposes, only the Conventional Takeoff and Landing (CTOL) version of F-35 was used since it is the most commonly produced variant in lieu of the F-35B and F-35C versions.

Analysis – Military Fighter Aircraft

If we plot the T100 flyaway FY2018 dollars per unit – without performing any adjustment for aircraft weight -- against the initial fielding date of the aircraft, we get the result shown in Figure 4.

Based on the best fit line, the T100 dollars per unit have increased from $2M per unit in 1949 to $134M per unit in 2019 – an annualized increase of 6.6% in real (constant year) dollars. However, this may be slightly misleading since U.S. fighter aircraft have increased in size over time. Because we know from numerous prior studies (Hess, 1987, Resetar, 1991, Younossi, 2001, et al.) that aircraft flyaway cost is positively correlated with

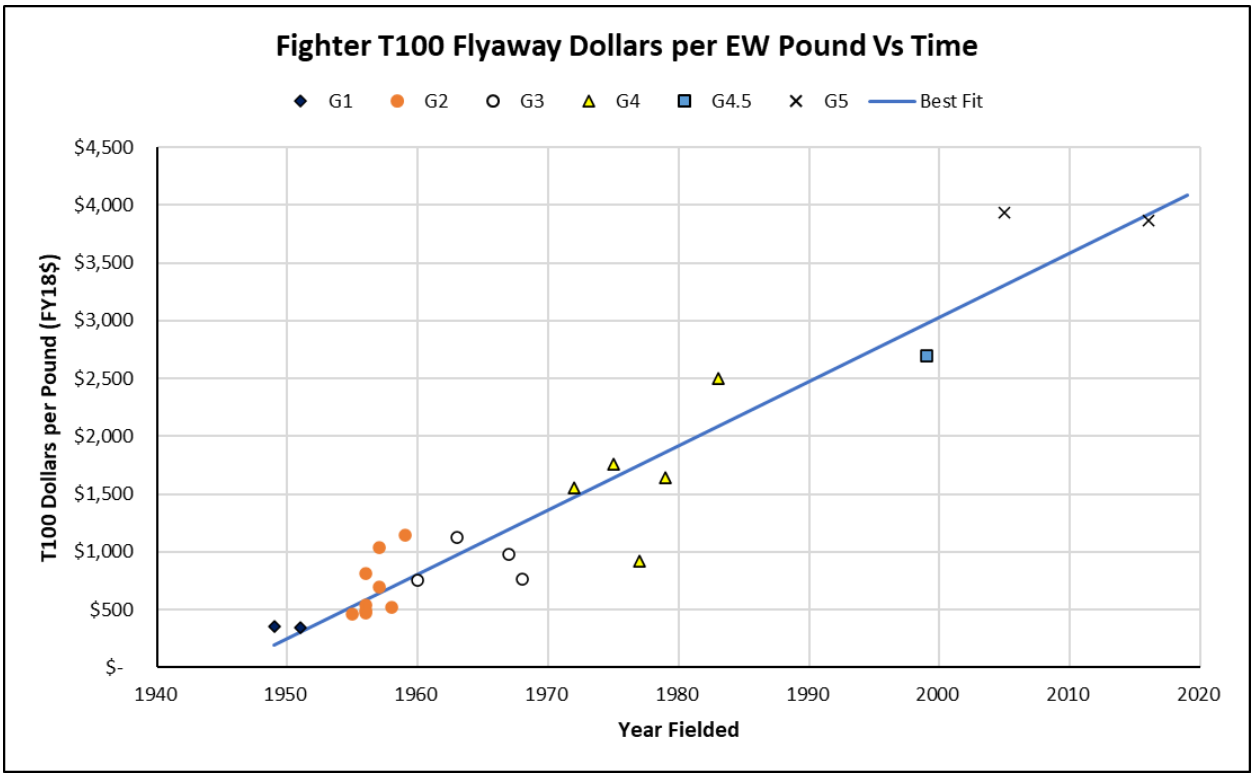


Figure 4. Fighter T100 Flyaway Dollars (FY18$) per Unit Over Time

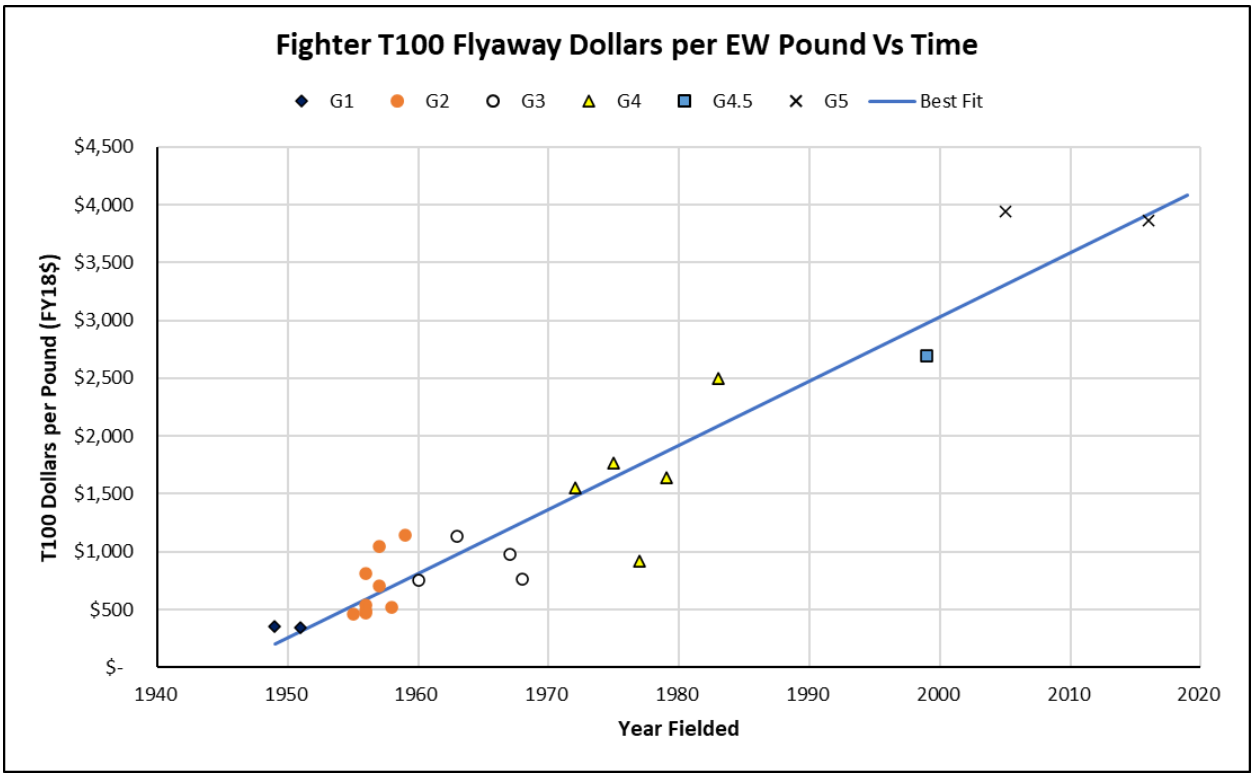


Figure 5. Fighter T100 Flyaway Dollars (FY18$) per Empty Weight Pound Over Time

aircraft weight (all else equal, the heavier the aircraft, the more it costs to build), it is more illuminating to plot the T100 flyaway dollars per pound against the initial fielding date of the aircraft, as shown in Figure 5.

The plot confirms Augustine's general thrust: that cost of fighter aircraft has increased significantly over time even after normalizing for inflation, position on the learning curve and overall aircraft weight. It is also apparent that each generation of aircraft has increased in cost over the prior one. Based on the best fit line, the T100 dollars per pound have increased from $198 per pound in 1949 to $4,087 per pound in 2019 – an annualized increase of 4.4% in real (constant year) dollars.

The plot also reveals the time between each fighter generation has been progressively increasing since the jet age began. Fueled by "hot" wars in Korea and Vietnam, the pace of innovation in the 1950's and 1960's was especially quick with the introduction of supersonic flight and the capability to carry radar and guided missiles (second generation) and the ability to perform multi-role missions (third generation). The pace of innovation has slowed substantially since the end of the Cold War with 34 years between the fourth and fifth generations:

	Average Year of IOC	Years Between Generation
Generation 1	1950	N/A
Generation 2	1957	7
Generation 3	1965	8
Generation 4	1977	12
Generation 4.5	1999	22
Generation 5	2011	12

Table 1. Average Year of IOC by Fighter Generation

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.965
R Square	0.932
Adjusted R Square	0.907
Standard Error	0.306
Observations	23

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	6	20.62	3.44	36.60	0.00
Residual	16	1.50	0.09		
Total	22	22.13			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	(9.81)	1.74	(5.65)	0.00	(13.49)	(6.13)	(13.49)	(6.13)
Empty Weight	1.19	0.18	6.67	0.00	0.81	1.57	0.81	1.57
Generation 2	0.57	0.24	2.34	0.03	0.05	1.08	0.05	1.08
Generation 3	0.83	0.28	2.93	0.01	0.23	1.43	0.23	1.43
Generation 4	1.44	0.26	5.44	0.00	0.88	2.00	0.88	2.00
Generation 4.5	1.90	0.40	4.81	0.00	1.06	2.74	1.06	2.74
Generation 5	2.25	0.34	6.59	0.00	1.52	2.97	1.52	2.97

Table 2. Multiple Regression Outputs

To get a better understanding of the impact that the introduction of each fighter aircraft generation has had on cost multiple variable regression model was constructed using T100 flyaway dollars as the dependent variable and empty weight as an independent variable. In addition, five dummy variables were assigned for second, third, fourth, fifth and Generation 4.5 aircraft. These dummy variables were set at either one (aircraft model included in that generation) or zero (aircraft model not included in that generation). The results of that regression (performed in Microsoft Excel) were as shown in Table 2.

The resulting equation is:

$$T100 = 0.0001 * EW^{1.19} * 0.765^{G2} * 1.285^{G3} * 3.227^{G4} * 5.701^{G4.5} * 8.443^{G5}$$

Where:

T100 = T100 unit flyaway cost (FY2018$ millions)

EW = Empty weight (pounds)

G2 = Second generation aircraft (1 if yes, 0 if no)

G3 = Third generation aircraft (1 if yes, 0 if no)

G4 = Fourth generation aircraft (1 if yes, 0 if no)

G4.5 = Generation 4.5 aircraft (1 if yes, 0 if no)

G5 = Fifth generation aircraft (1 if yes, 0 if no)

The regression demonstrated a R-square value of 93.2%, suggesting the combination of empty weight and aircraft generation explains greater than 90% of the variation observed in the data. All the independent variables were positively correlated with T100 hours per pound as

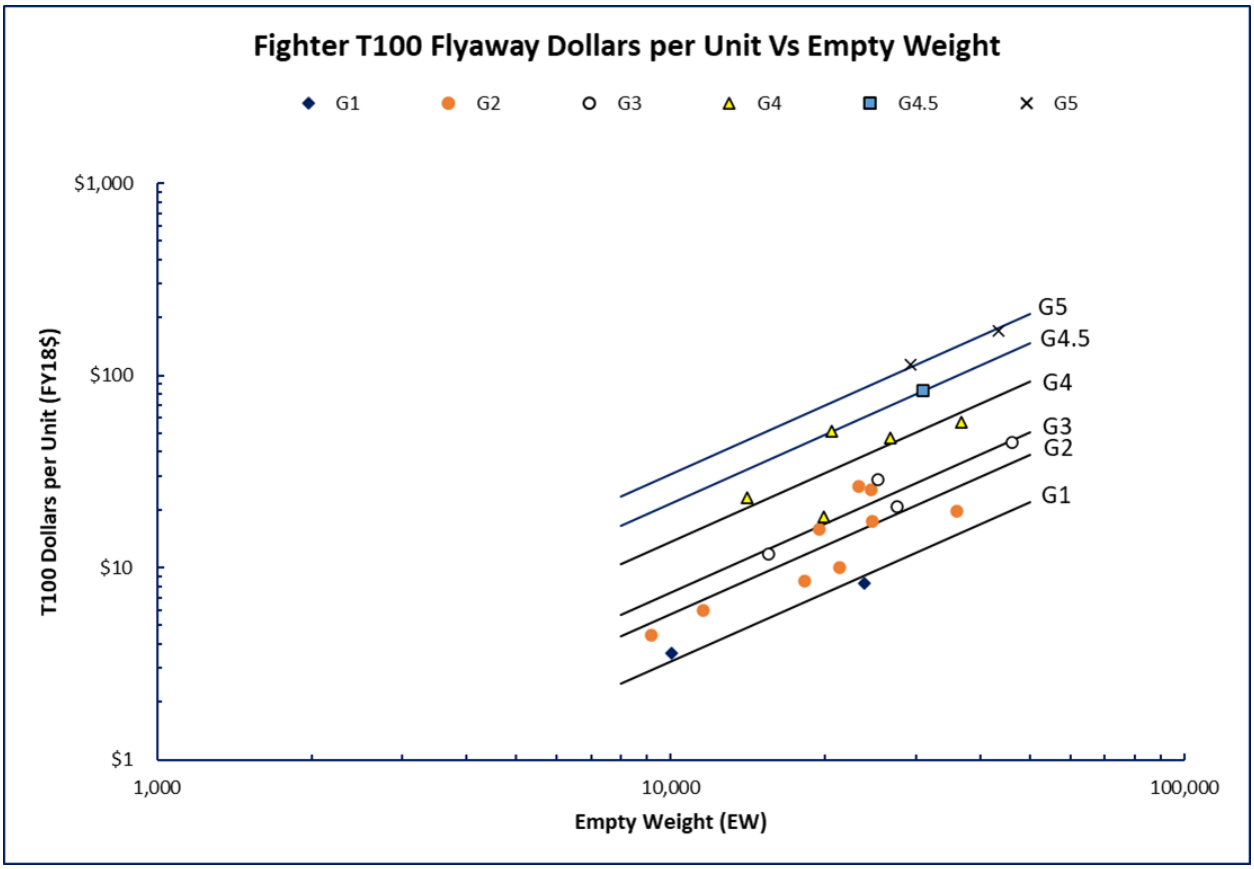


Figure 6. Fighter T100 Flyaway Dollars (FY18$) per Unit v. Empty Weight

expected, and all the resulting coefficients were statistically significant at the 5 percent level of error. Visually, this is shown in Figure 6.

The graph shows us the clear stairstep pattern of increasing cost from generation to generation of fighter aircraft. It also allows us to quantitatively measure that growth. Assuming an aircraft with an empty weight of 30,000 pounds:

If a 30,000 lb fighter was….	T100 FY2018$ Flyaway ($M per Unit)
Generation 1	$12.00
Generation 2	$21.20
Generation 3	$27.40
Generation 4	$50.70
Generation 4.5	$80.30
Generation 5	$113.20

Table 3. Estimated T100 Flyaway Dollars (FY18$) per Unit for 30,000 Pound Fighter

Overall, the growth in cost has been 57% between each generation.

We can see that tactical fighter aircraft have grown in cost over each generation. Now the question is: are they more capable? Intuitively, the answer is "yes" -- clearly, today's fourth and fifth generation fighter aircraft could easily defeat its first-generation counterparts, laboring as they did at subsonic speeds without radars or missiles. Is there any way to measure this increased capability and relate it to the cost?

As it turns out, the *U.S. Military Aircraft Cost Handbook* was developed as a companion volume to a larger study by TASC to develop relative measures of U.S. tactical aircraft capability (TASC, 1980.) One of the results was an Aircraft System Performance (ASP) metric, which considers a variety of factors including payload, range, maneuverability, speed, target acquisition/

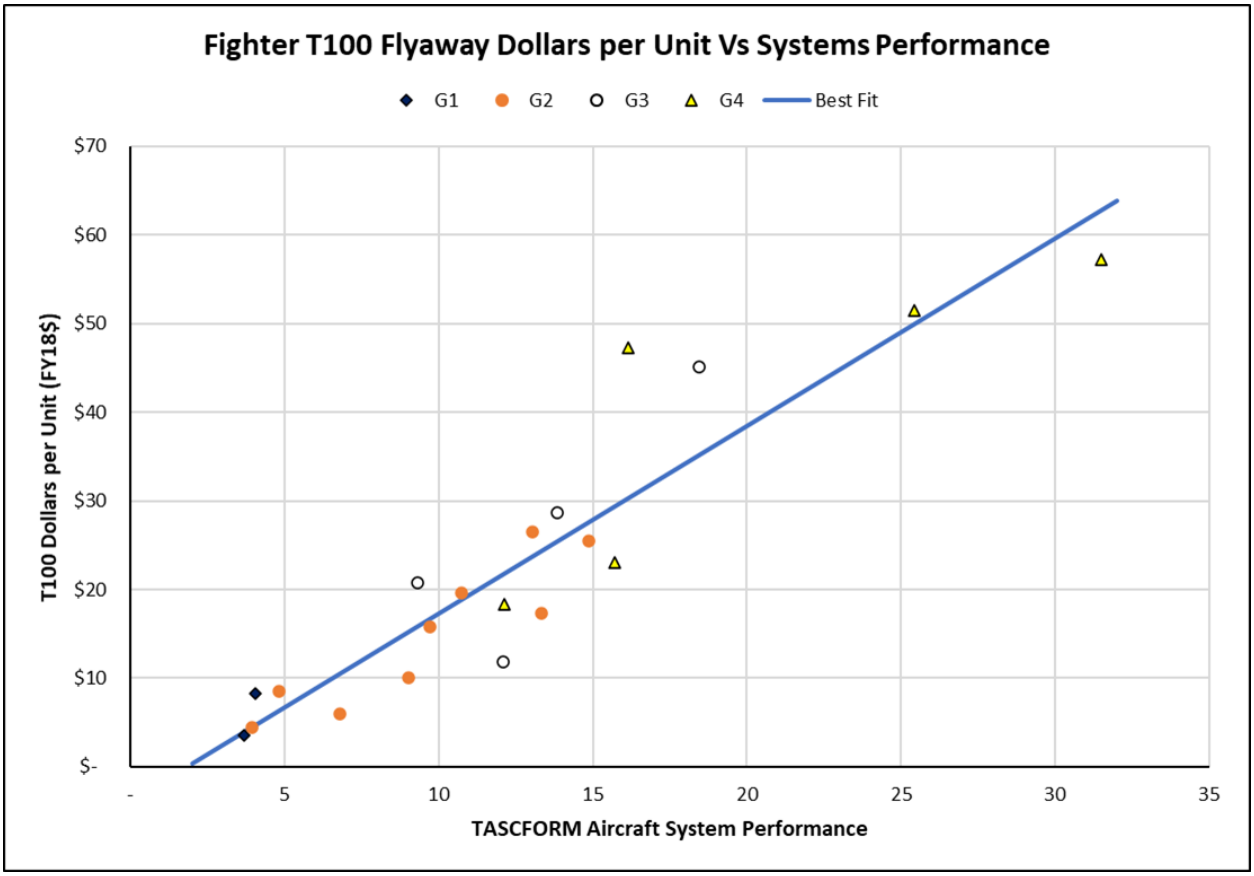


Figure 7. Fighter T100 Flyaway Dollars (FY18$) per Unit v. TASC Aircraft System Performance

engagement capability, navigational capability and survivability. The resulting numerical score can be interpreted against a baseline score for the F-4B (at 10). A tactical aircraft with an ASP score of 20 can be interpreted as a single aircraft is equivalent to two F-4Bs performing the same mission (Hildebrandt, 1986.)

Ideally, we would be able to update these metrics for the three aircraft added since the TASC study was completed: F-18E/F, F-22 and F-35. Unfortunately, the author has been unable to find ASP metrics developed for these aircraft. But we can still compare flyway costs to performance metrics through fourth generation aircraft.

A graph relating flyaway costs to TASC Aircraft System Performance is shown in Figure 7.

Examination of the linear best fit line demonstrates a R-square value of 85.2%, suggesting the increase in T100 flyaway dollars per unit is highly correlated to increases in aircraft system performance. It is also apparent from the chart that each generation has a higher level of systems performance:

	Avg. ASP
Generation 1	3.9
Generation 2	9.6
Generation 3	13.4
Generation 4	20.2

Table 4. Average Aircraft Systems Performance Score by Fighter Generation

It is also clear that this contradicts Augustine's assertion that "…the unit cost being most closely correlated with the passage of time rather than with changes in maneuverability, speed, weight, or other technical parameters." On the contrary, the graph suggests that the increase in unit cost is *intimately* correlated to the technical

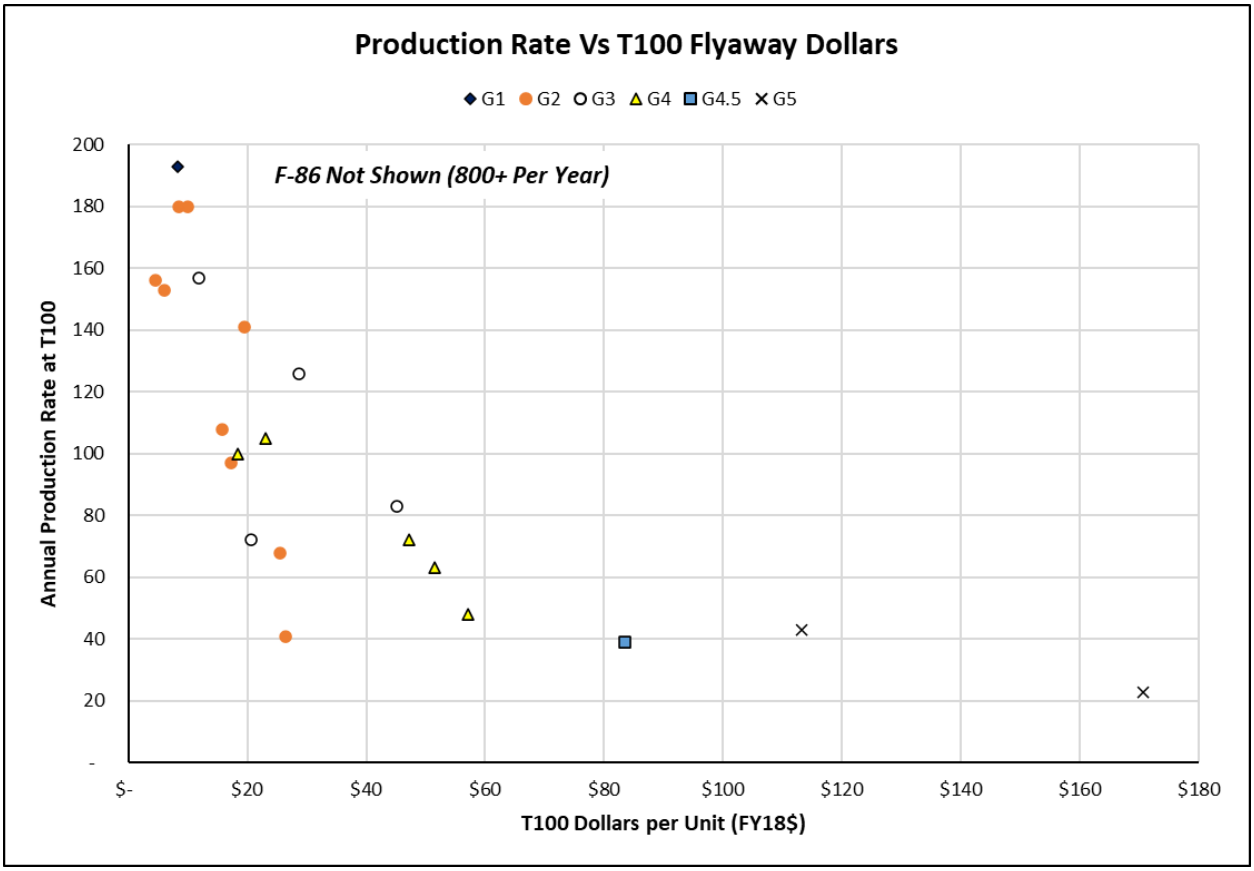


Figure 8. Production Rate v. T100 Flyaway Dollars (FY18$) per Unit

characteristics of the aircraft. While the ASP indices are only available through the fourth generation, it can be surmised that the equivalent systems performance values would be even higher for Generation 4.5 and 5 aircraft – as is, of course, the cost.

If we interpret the TASC systems performance as intended, this suggests a single fourth generation fighter jet is equivalent in performance to five (5) first generation aircraft. If this is correct, it helps explain another interesting fact that emerges from comparing U.S. fighter aircraft over time – the substantial decrease in production rates over time. This emerges vividly by plotting the annual production rate for the production lot in which the 100th unit delivers and correlating it to the flyaway cost, as shown in Figure 8.

Clearly, there has been a significant decrease in production rates with each generation of fighter aircraft at the 100th unit:

	Avg. Annual Rate (T-100)
Generation 1	510
Generation 2	125
Generation 3	110
Generation 4	78
Generation 4.5	39
Generation 5	33

Table 5. T-100 Annual Production Rates by Fighter Generation

It has been suggested (Eskew, 1990) that the reduced production rates are a factor in the increased costs over time. Many analysts suggest high production rates produce cost savings -- driven by material discounts associated with buying larger quantities, wider distribution of fixed support labor and overhead costs across the number of units produced, and steeper learning curve slopes (for a review of the literature, see Johnstone, 2017). Accordingly, the decrease in

production rates over time may potentially explain some of the increase in fighter aircraft costs.

The problem with this analysis is it asks the proverbial "chicken or the egg" question: Have lower production rates over time driven higher aircraft costs? Or have higher procurement costs forced the services to buy fewer and fewer fighters with each successive generation? Or (alternatively) have the improvements in capability of fighters allowed the services to simply buy less of them without sacrificing the ability to perform the mission? The answer to this question is beyond the scope of this paper; nonetheless, it is the author's intuition that the second and third answers are more plausible than the first.

Augustine's Law – True or False?

We return to Augustine's analysis of tactical aircraft and ask: are we really in danger of one day seeing the cost of a single aircraft equal the entire defense budget? The easiest way to answer this question is see how accurate Augustine's predictions – made almost forty years ago – have matched up against our most recent experience. Specifically, how have his extrapolations fared against reality not only for tactical aircraft cost, but for the defense budget and the gross national product as well?

2019 Values	Projected	Actual	Variance to Actual
Tactical Aircraft Unit Cost	$6.24B	$146M	4163%
U.S. Defense Budget	$351B	$689B	-49%
U.S. Gross Domestic Product (GDP)	$22.5T	$21.0T	7%

Table 6. Comparison of Projected v. Actual, CY2019

As we noted earlier, Augustine's book does not provide absolute numbers, but we can work them out from his charts. In the formulation of his law, Augustine assumed a nominal 15% increase per year in tactical aircraft costs in nominal dollars. Likewise, he assumed the defense budget would increase by 2.5% per year and the nominal GNP by 5.5% per year. By the year 2019 (Budget, 2019a; Budget, 2019b), we should have had the data in Table 6.

Graphically, if we fill in the actual experience since 1980 for these three categories, we get the results in Table 6. Graphically, if we fill in the actual experience since 1980 for these three categories, we see the results shown in Figure 9.

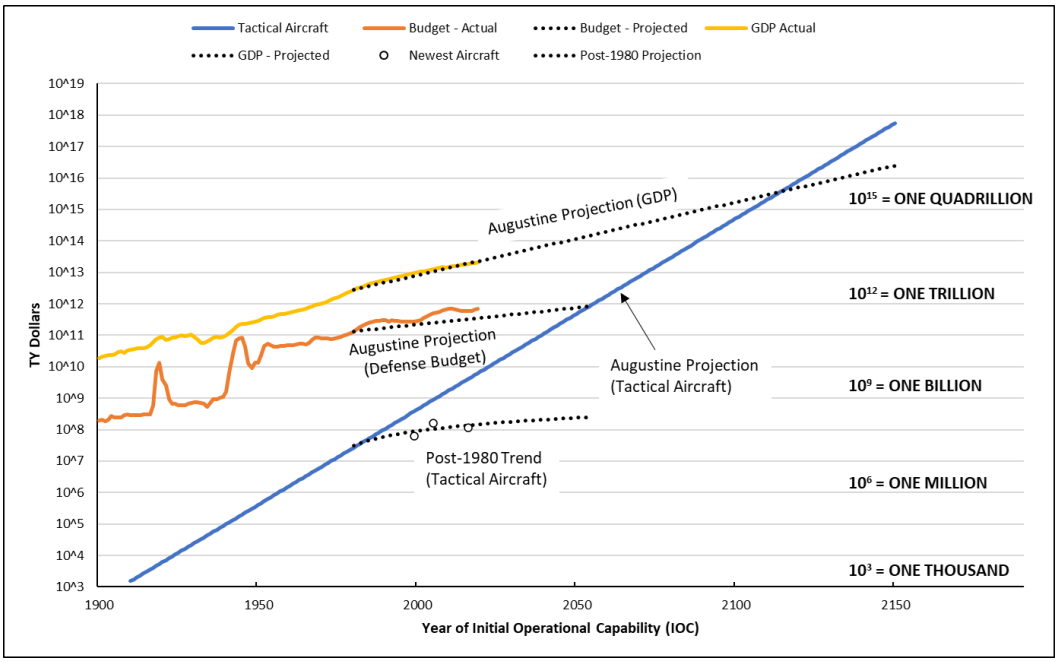


Figure 9. GDP, Defense Budget and Tactical Aircraft Trends, 1900-2050

Augustine's extrapolation for GDP was quite close to the eventual 2019 value. He overstated the actual amount by only 7%, an excellent mark for a projection forty years into the future. His projection for the defense budget, developed from the post-Vietnam War wind-down, did not anticipate either the Reagan era defense buildup or the Bush War on Terror and therefore was understated by 49%. But for tactical aircraft, Augustine's projection missed by four *thousand* percent. There is no six billion dollar fighter aircraft in 2019. Taking the F-18E/F, F-22 and F-35 unit cost into account, the Augustinian trend line seems to have deflected. Since 1980, the growth in tactical aircraft costs seem to have slowed to 4% nominal (then-year) growth per year. Based on this, the projected 2019 fighter cost is closer to $145 million per copy. Augustine's projections seem to have run into another empirical law – in this case, one coined by the late economist Herbert Stein (Stein, 1976): "If something cannot go on forever, it will stop."

Based on this analysis, it appears there is no longer any danger that our single aircraft cost will eventually overtake the total defense budget – in fact, based on this extrapolation, the lines seem to run roughly parallel at least through the prophetic year of 2154. Before we congratulate ourselves on our success, it seems worthwhile to mention that a 4% nominal growth in fighter cost year over year is still 1.5 times the inflation rate over the same time period. It means fighters are growing more expensive with each generation, and that this growth, if it persists, will continue to vex a Pentagon simultaneously trying to update its tactical and strategic aircraft fleet, its surface ship fleet and its nuclear weapons inventory.

Development and Program Cost

We now turn to Augustine's conclusion that there has been no change in aircraft design and development spans despite the increasing complexity of aircraft design and build over time. Using the same dataset of fighter aircraft, development spans were plotted first by calendar year the aircraft was fielded and second against T100 unit cost.

Based on a RAND database of acquisition milestones (Rothman, 1987), the length of development was defined as the span between a program's initiation and delivery of the first production unit. In this case, the Milestone B date was chosen as the beginning of the official development effort. This is slightly different from Augustine's analysis, which measured development contract award to first flight. First flight was not available from the RAND data, but the first production delivery was chosen as the next best alternative.

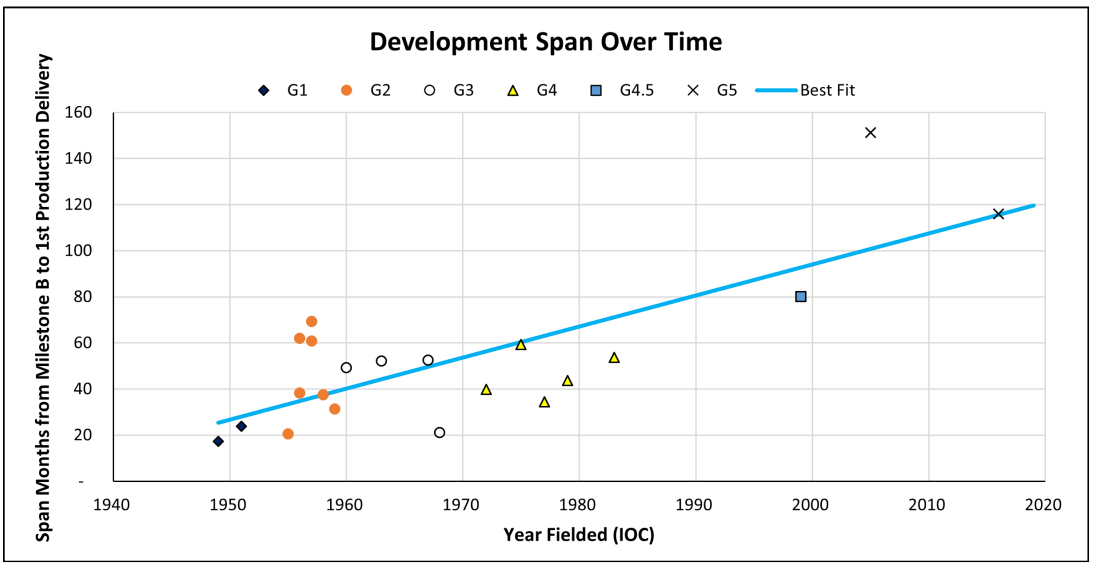


Figure 10. Development Spans Over Time by Fighter Generation

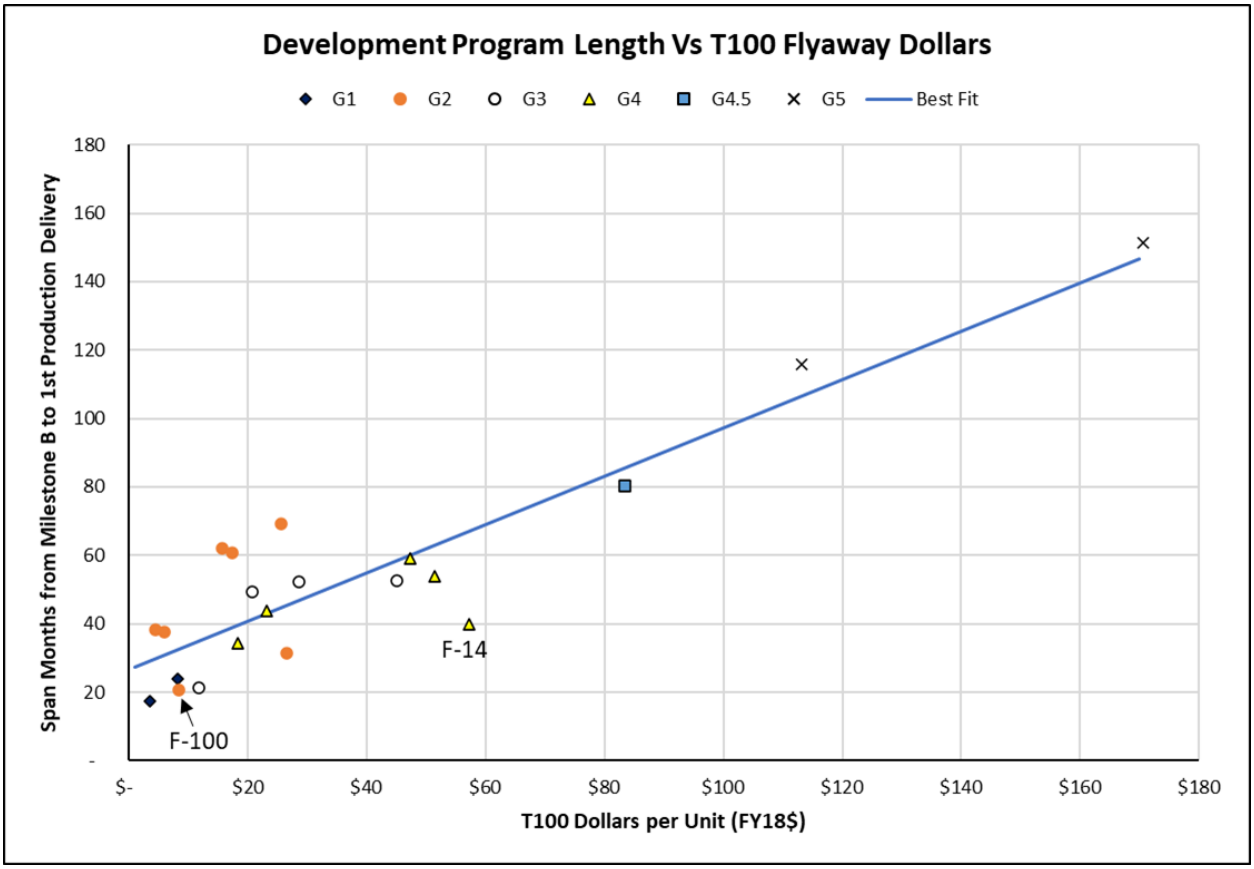


Figure 11. Development Program Length v. T100 Flyaway Dollars (FY18$) per Unit

Viewing the length of development spans across time, we see the results shown in Figure 10.

Based on the data through the early 1980s, it is easy to see how Augustine came to his conclusion – development spans do not show an especially strong trend across time. But once we introduce later data for F-18E/F, F-22 and F-35A, a different picture emerges. Fighter aircraft fielded after 1990 were in development substantially longer than prior generations.

If we correlate development spans to T100 unit cost, we see the results shown in Figure 11.

Once again, we see a correlation between development spans and flyaway unit costs. It is worth pointing out the two datapoints which are significantly below the trend line: the North American F-100 and the Grumman F-14. Performing a similar analysis, Eskew suggests that the F-100 and F-14 both benefited from inherited technology from other programs, thus shortening their development spans. The F-100 evolved from the earlier F-86 aircraft, while the F -14 inherited engines and avionics from the cancelled F-111B program (Eskew, 1990).

Examination of the linear best fit line demonstrates a R-square value of 82.2%, suggesting the increase in development span months is positively correlated to T100 flyaway dollars per unit. But once again this fit is highly influenced by the addition of the F-18E/F, F-22 and F-35A data points. Omitting those data points, the R-square drops to 23.9%.

It could be argued that that F-22 and F-35 represent peculiar circumstances. For example, the F-22 program, caught in the post-Cold War drawdown of defense spending, was rephased four times with additional time added to the

development program. In addition, it had design challenges surrounding stealth, integration of its avionics suites, and a new propulsion system. That does not explain, however, the longer span time associated with the F-18E/F, which had incremental improvements with minimal stealth, avionics systems mostly taken from its predecessor, and a derivative aircraft design (Younossi, 2005). Based on the experience of inherited technologies like the F-100 and F-14, one might expect it to be like fourth generation aircraft spans, and possibly even below them.

That development span times have increased since the 1980's is recognized by DoD leadership. In his 2013 implementation memo for the Better Buying Power initiative, Undersecretary of Defense (AT&L) Frank Kendall stated, "On average programs are taking about one year longer to complete development than they did 20 years ago, but the root causes of longer program cycle times are not obvious, and the data includes wide variations" (Kendall, 2013, Riposo, 2014). In short, then, Augustine's contention that "[t]he duration of the design and build phase of aircraft development programs has remained virtually unchanged for 40 years" must be considered challenged by the most recent program data.

Conclusions

For the moment, we will take Augustine's humorous "laws" and treat them as literal assertions of truth. We can summarize the laws into a series of propositions, and post conclusions against each, as follows:

Proposition 1: The price of tactical military aircraft is increasing at a factor of four every ten years, i.e., 15 percent per annum, and in time will overtake the United States defense budget and eventually its gross national product.

Conclusion: Taken literally, this proposition is clearly false. An analysis of military fighter unit cost incorporating the most recent vehicles shows a slower rate of increase than observed for previous generation fighters. Extrapolated into the future, those costs will not overtake either the defense budget or the gross national product. Having said that, the observed year-over-year increase of 4% per year excluding inflation drives increasingly expensive fighter aircraft (and lower quantities) over time – which was Augustine's point.

Proposition 2: This increase in unit cost cannot be explained by improvements in technical parameters (maneuverability, speed, weight, et al.) and are seemingly inherent characteristics of these systems.

Conclusion: This conclusion does not seem to be supported by analysis. The strong relationship between the TASC metrics of aircraft performance and unit cost for first through fourth generation fighters suggests that the increase in unit cost *is* correlated to the technical performance of the aircraft.

Proposition 3: Aircraft design and development spans have not increased over time despite advances in technology and design complexity.

Conclusion: This proposition was supported by the data through the early 1980's. However, military fighter aircraft fielded after 1990 *do* show an increase in aircraft design and development spans, a trend which has been recognized by DoD leadership. This change is an unwelcome development, since it represents a lengthening of the "time to market" for warfighters to make new capabilities available for the battlefield.

So why do Augustine's laws still have lasting appeal? Because most reader – recognizing that Augustine's tongue is firmly planted in cheek – recognize his laws should not be read literally but appreciated for the greater truth they represent. It is no question that military fighter unit costs are increasing at rates higher than inflation and that those rates have compounded significantly over time: fifth generation fighters are almost ten

times the cost of their first-generation predecessors. And this has profound implications for the sixth generation of fighter aircraft, which have already begun their initial DoD studies. For if history holds true, we can expect this next generation of aircraft will likely cross the $200 million per unit threshold.

Or they may not. The slowing of cost growth from the early 1980s when Augustine performed his original analysis gives some hope that the trend might be reduced further, or even reversed. It is possible that the Digital Revolution in design and manufacturing might offer a route to make this happen, as recently suggested by Dr. Will Roper, the former Undersecretary of the Air Force for Acquisition, Technology and Logistics. (Trimble, 2019.) At the same time, cost analysts must recognize such a happy event would be a strong break from the past seventy years of history. The feasibility of doing just that, however, is beyond the scope of this paper, and a subject for another time.

It seems appropriate to close with a word of warning: this analysis – and Augustine's own – hinge on the validity of extrapolating historical data into the future. One cannot do better than Mark Twain to outline the risks of doing so:

> In the space of one hundred and seventy-six years the Lower Mississippi has shortened itself two hundred and forty-two miles. That is an average of a trifle over one mile and a third per year. Therefore, any calm person, who is not blind or idiotic, can see that in the Old Oölitic Silurian Period, just a million years ago next November, the Lower Mississippi River was upward of one million three hundred thousand miles long, and stuck out over the Gulf of Mexico like a fishing rod. And by the same token any person can see that seven hundred and forty-two years from now the Lower Mississippi will be only a mile and three-quarters long, and Cairo and New Orleans will have joined their streets together and be plodding comfortably along under a single mayor and a mutual board of aldermen. There is something fascinating about science. One gets such a wholesale return of conjecture out of such a trifling investment of fact. (Quoted in Huff, 1954.)

Appendix – Commercial Jetliner Aircraft

Augustine's assertion that cost growth of highly complex systems is an inherent characteristic of those systems applies – as he indicated himself – to the commercial world as well as defense systems. In his book, Augustine graphs the cost of commercial airliners and produces a similar growth trend as tactical aircraft (Augustine, 1983).

To verify Augustine's claim, a similar cost database for commercial airliners was assembled. The data was limited to commercial passenger jets, both wide body

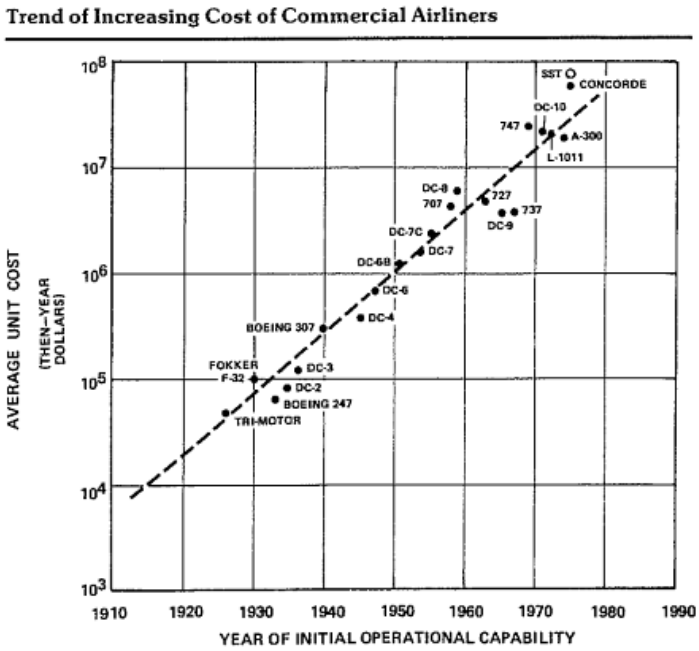


Figure 12. Reprinted with Permission of Norman Augustine

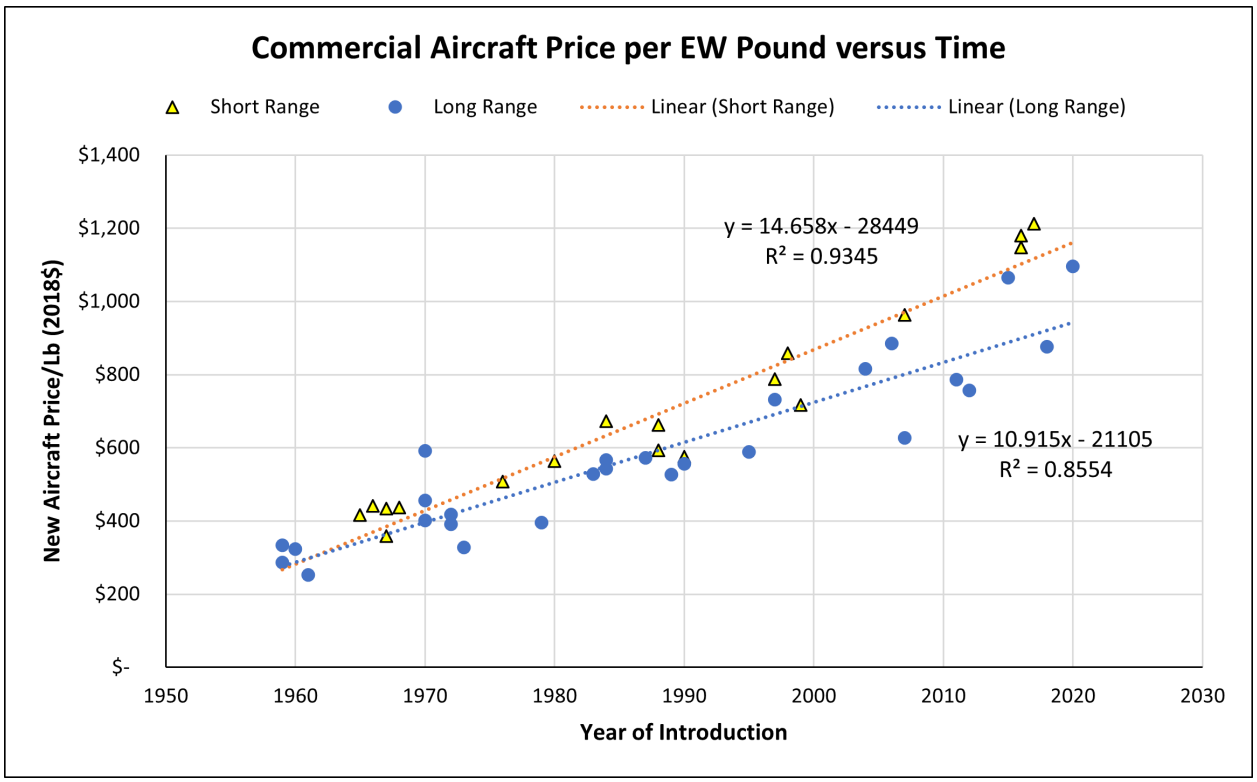


Figure 13. Commercial Aircraft Price per Empty Weight Pound Over Time

and narrow. Lee (1990) was the primary source for cost data for jetliners introduced before 1995. T100 unit cost is not available and even if it were, it would be of doubtful utility. Commercial aircraft prices are frequently delinked to actual costs. Manufacturers routinely offer early units at prices below cost, hoping to establish their product in the market and recover losses later in the product life cycle once the model is safely established in the marketplace. Instead, Lee reported the listed aircraft price at the time of introduction. Lee's values, reported in constant year 1995 dollars, were escalated to CY2018 dollars using the U.S. GDP implicit price deflator (Federal Reserve, 2019). For aircraft introduced after 1995, the cost data was supplemented by press releases of published prices and other sources (Airliner.net, 2013, Airbus, 2018a, Boeing 2019) normalized to CY2018 dollars. The final database was comprised of 46 aircraft models, the most recent being the Boeing 737 MAX.

It is useful to segregate the jetliner dataset between long range (>4000 nautical miles) and short range (<4000 nautical miles). Examples of short-range jetliners include the Airbus A220 and A320, the Boeing 717, 727 and 737, the Douglas DC-9 and the McDonnell Douglas MD-80. Example of long-range jetliners include the Airbus A300, A310, A330, A350 and A380; the Boeing 707, 720, 747, 767, 777 and 787; the McDonnell Douglas DC-10 and MD-11, and the Lockheed L-1011.

A view of the CY2018 dollars per pound against the year of introduction shows a sizeable increase over the 1960-2019 time period, as shown in Figure 13.

The annualized growth in constant year dollars is 2.4% per year for short-range aircraft and 2.0% per year for long-range aircraft. This is approximately half of the annualized growth in military fighter aircraft.

The taxonomy of commercial aircraft generations is much more poorly defined than its military equivalent, and there is no agreement among

sources. Table 7 summarizes changes in commercial jetliner design in safety features, engine performance, and materials usage (Airbus, undated, Airliners.net, 2007, Hiken, 2018). From this table, approximately four or five generations of aircraft designs can be identified, each more technologically advanced than the other. While this evolution is perhaps not as radical as its military equivalent – these aircraft examples operate at subsonic speeds, none carry weapons, and all are highly visible on radar – nonetheless it seems reasonable to suppose that the higher costs of commercial transport are also tied to higher levels of technical performance.

Year	Safety	Engines	Materials
	First commercial jetliner enters service in 1952.		
1950s	Dials and gauges in the cockpit with early auto-flight systems.	Turbojet engines	Metallic airframes
1960s	More elaborate auto-pilot and auto-throttle systems	Turbofan engines introduced.	
1970s		First generation of high bypass turbofan engines.	First introduction of composite materials in select applications
1980s	Electronic cockpit displays, improved navigation performance and Terrain Avoidance Systems		Composites usage approximately 1-5% by weight
1990s	Fly-by-wire technology, enabled flight envelope protection		Composites usage approximately 15% by weight on new models
2000s		Next generation of high bypass turbofan engines.	
2010s			Composites usage 30-50% by weight on new models

Table 7. Technology Advances in Commercial Aircraft Over Time

References

Airbus. (2018a). Airbus 2018 Price List Press Release, 15 January 2018. Retrieved from https://www.airbus.com/newsroom/press-releases/en/2018/01/airbus-2018-price-list-press-release.html.

Airbus. (2018b). Family Figures Farnborough 2018 Edition. Retrieved from https://www.google.com/search?ei=aDxcXemXNoqWsQWWrZmIDg&q=airbus+family+figures+2018+farnborough&oq=airbus+family+figures+2018+farnbor&gs_l=psy-ab.1.0.33i160.1315.8586..10933...0.2..5.799.9802.0j4j10j4j4j4j2.....0....1..gws-wiz.....0i71j0i67j0i131j0j0i22i30j33i22i29i30j33i299.TM1ds6K9qiQ.

Airbus. (Undated.) Technology's Contribution to Aviation Safety. Retrieved from https://accidentstats.airbus.com/statistics/generations-of-jet.

Airliners.net. (2007.) Airliner generations. Retrieved from https://www.airliners.net/forum/viewtopic.php?t=461777.

Airliners.net. (2013). Historical List Prices: Boeing And Airbus. Retrieved from https://www.airliners.net/forum/viewtopic.php?t=561643.

Aitoro, Jill. (October 25, 2016). "30 Years: A Norm Augustine Retrospective." Retrieved from https://www.defensenews.com/30th-annivesary/2016/10/25/30-years-a-norm-augustine-retrospective/.

Augustine, Norman R. (Spring 1979). "Augustine's Laws and Major System Development Programs." *Defense Systems Management Review, 2* (2), 50-76.

Augustine, Norman R. (1983). *Augustine's Laws and Major System Development Programs.* Reston, VA: American Institute of Aeronautics and Astronautics.

Boeing Company. (2019). About Boeing Commercial Airplanes. Retrieved from https://www.boeing.com/company/about-bca/.

Budget of the United States Government, FY2019. (2019a). Table 3.1, Outlays by Superfunction and Function, 1940-2023. Retrieved from https://www.govinfo.gov/content/pkg/BUDGET-2019-TAB/pdf/BUDGET-2019-TAB.pdf.

Budget of the United States Government, FY2019. (2019b). Table 10.1. Gross Domestic Product and Deflators used in the Historical Tables: 1940-2023. Retrieved from https://www.govinfo.gov/content/pkg/BUDGET-2019-TAB/pdf/BUDGET-2019-TAB.pdf.

Bureau of Labor Statistics (2019). BLS CPI Inflation Calculator. Retrieved from https://data.bls.gov/cgi-bin/cpicalc.pl?cost1=10000&year1=191301&year2=201906.

Department of Defense. (2012a). *Department of Defense Fiscal Year (FY) 2013 President's Budget Submission February 2012. Navy, Justification Book Volume 1, Aircraft Procurement, Navy, Budget Activity 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2013.aspx.

Department of Defense. (2012b). *Selected Acquisition Report: F/A-18E/F Super Hornet Aircraft (F/A-18E/F)*. Washington, D.C.: Defense Acquisition Management Information Retrieval. Retrieved from https://www.globalsecurity.org/military/library/budget/fy2012/sar/f-a-18e-f_december_2012_sar.pdf

Department of Defense. (2013). *Department of Defense Fiscal Year (FY) 2014 President's Budget Submission April 2013. Navy, Justification Book Volume 1 of 4, Aircraft Procurement, Navy, Budget Activity 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2014.aspx.

Department of Defense. (2014). *Department of Defense Fiscal Year (FY) 2015 President's Budget Submission March 2014. Navy, Justification Book Volume 1, Aircraft Procurement, Navy, Budget Activity 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2015.aspx.

DePuy, William E., Jr., Moyer, Rosemarie, Palmer, Paul R., Jr., McKinney, Bruce J., Kreisel, George R., Balut, Stephen J., McNichols, Gerald R. (1983). *U.S. Military Aircraft Cost Handbook*. Falls Church, VA: Management Consulting & Research, Inc.

Eskew, Henry L. (2000). "Aircraft Cost Growth and Development Program Length: Some Augustinian Propositions Revisited." *Acquisition Review Quarterly*, Summer 2000, 209-220.

Federal Reserve Bank of St. Louis. (2019). Federal Reserve Economic Data (FRED), Gross Domestic Product: Implicit Price Deflator. Retrieved from https://fred.stlouisfed.org/series/GDPDEF.

Franck, Raymond E., Jr. (1992). *Performance Choices in Post-Cold War Weapon Systems*. Maxwell Air Force Base, AL: Air University Press.

General Accounting Office. (2019). *F-35 Joint Strike Fighter: Action Needed to Improve Reliability and Prepare for Modernization Efforts*. GAO-19-341. Retrieved from https://www.gao.gov/products/GAO-19-341.

Hess, R.W., Romanoff, H. P. (1987). *Aircraft Airframe Cost Estimating Relationships: Study Approach and Conclusions*. R-3255-AF. Santa Monica, CA: RAND.

Hildebrandt, Gregory G., Sze, Man-bing. (1986). *Accounting for the Cost of Tactical Aircraft.* RAND N-2420-PA&E. Santa Monica, California: RAND Corporation.

Hiken, Alen. (2018). *The Evolution of the Composite Fuselage: A Manufacturing Perspective*. Retrieved from https://www.intechopen.com/online-first/the-evolution-of-the-composite-fuselage-a-manufacturing-perspective.

Huff, Darrell. (1954). *How to Lie with Statistics*. New York, NY: W. W. Norton and Company.

Johnstone, Brent M. (2017). Do Production Rates Really Matter? *Proceedings of the 2017 ICEAA Professional Development and Training Workshop*, Portland, Oregon. Retrieved from https://www.iceaaonline.com/pdx17papers/#DA03

Kendall, Frank. (2013, April 24). *Implementation Directive for Better Buying Power 2.0*. [Memorandum]. Retrieved from https://www.dau.edu/policy/PolicyDocuments/the564BBP%202.0%20Implementation%20Directive%2024%20April%202013.pdf.

Lee, Joosung Joseph. (2000). *Historical and Future Trends in Aircraft Performance, Cost and Emissions* (Unpublished Master's thesis). Massachusetts Institute of Technology: Cambridge, MA. Retrieved from https://dspace.mit.edu/handle/1721.1/8825.

Lockheed Martin. (2019, September 1). *F-35 Lightning II Program Status and Fast Facts*. Retrieved from https://www.f35.com/assets/uploads/documents/FG18-24036_009_F35FastFacts9_2019.pdf.

Resetar, Susan A., Rogers, J. Curt, Hess, Ronald W. (1991). *Advanced Airframe Structural Materials: A Primer and Cost Estimating Methodology*. R-4016-AF. Santa Monica, CA: RAND Corporation.

Riposo, Jessie, McKernan, Megan, Duran, Chelsea Kailioi. (2014). *Prolonged Cycle Times and Schedule Growth in Defense Acquisition: A Literature Review*. RR-455. Santa Monica, CA: RAND Corporation.

Rothman, M. B. (1987). *Aerospace Weapon System Acquisition Milestones: A Data Base*. N-2599-ACQ. Santa Monica, CA: RAND Corporation.

Stein, Herbert. (1976). "A Symposium on the 40th Anniversary of the Joint Economic Committee, Hearings Before the Joint Economic Committee, Congress of the United States, Ninety-Ninth Congress, First Session; Panel Discussion: The Macroeconomics of Growth, Full Employment, and Price Stability". Retrieved from https://babel.hathitrust.org/cgi/pt?id=umn.319510030778307&view=1up&seq=270.

The Analytic Sciences Corporation. (1980). *The TASCFORM-AIR Model: A Technique for Assessing Comparative Force Modernization in Tactical Aviation, Vol. I: Methodology Development*. TR-1334-3. Reading, MA: The Analytic Sciences Corporation.

Trimble, Stephen. (2019, September 23). NGAD Program Redirected to Promote Digital Industrial Revolution. *Aviation Week and Space Technology*. Retrieved from https://aviationweek.com/combat-aircraft/ngad-program-redirected-promote-digital-industrial-revolution.

United States Air Force. (2010). *United States Air Force FY2011 Budget Estimates, Aircraft Procurement, Air Force Volume I*. Retrieved from https://www.saffm.hq.af.mil/FM-Resources/Budget/Air-Force-Presidents-Budget-FY11/.

United States Air Force (2015, September 23). F-22 Raptor Fact Sheet. Retrieved from https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104506/f-22-raptor/.

United States Navy. (1999). *Department of the Navy, Fiscal Year (FY) 2000/2001 Biennial Budget Estimates, Justification of Estimates, February 1999. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4.* Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2000.aspx.

United States Navy. (2000). *Department of the Navy, Fiscal Year (FY) 2001 Budget Estimates, Justification of Estimates, February 2000. Aircraft Procurement, Navy, Volume I: Budget Activities 1-4.* Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2001.aspx.

United States Navy. (2001). *Department of the Navy, Fiscal Year (FY) 2002 Amended Budget Submission, Justification of Estimates, June 2001. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2002.aspx.

United States Navy. (2002). *Department of the Navy, Fiscal Year (FY) 2003 Budget Estimates, Justification of Estimates, February 2002. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2003.aspx.

United States Navy. (2003). *Department of the Navy, Fiscal Year (FY) 2004/2005 Biennial Budget Estimates, Justification of Estimates, February 2003. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2004.aspx.

United States Navy. (2004). *Department of the Navy, Fiscal Year (FY) 2005 Budget Estimates, Justification of Estimates, February 2004. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2005.aspx.

United States Navy. (2005). *Department of the Navy, Fiscal Year (FY) 2006/FY2007 Budget Estimates, Justification of Estimates, February 2005. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2006.aspx.

United States Navy. (2006). *Department of the Navy, Fiscal Year (FY) 2007 Budget Estimates Submission, Justification of Estimates, February 2006. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2007.aspx.

United States Navy. (2007). *Department of the Navy, Fiscal Year (FY) 2008/2009 Budget Estimates, Justification of Estimates, February 2007. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2008.aspx.

United States Navy. (2008). *Department of the Navy, Fiscal Year (FY) 2009 Budget Estimates, Justification of Estimates, February 2008. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2009.aspx.

United States Navy. (2009). *Department of the Navy, Fiscal Year (FY) 2010 Budget Estimates, Justification of Estimates, May 2009. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2010.aspx.

United States Navy. (2010). *Department of the Navy, Fiscal Year (FY) 2011 Budget Estimates, Justification of Estimates, February 2010. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2011.aspx.

United States Navy. (2011). *Department of the Navy, Fiscal Year (FY) 2012 Budget Estimates, Justification of Estimates, February 2011. Aircraft Procurement, Navy, Volume 1: Budget Activities 1-4*. Retrieved from https://www.secnav.navy.mil/fmc/fmb/Pages/Fiscal-Year-2012.aspx.

Yoon, Joe. (June 27, 2004). "Fighter Generations." Aerospaceweb. Retrieved from http://www.aerospaceweb.org/question/history/q0182.shtml.

Younossi, Obaid, Kennedy, Michael, Graser, John C. (2001). *Military Aircraft Costs: The Effects of Advanced Materials and Manufacturing Processes*. MR-1370. Santa Monica, CA: RAND Corporation.

Younossi, Obaid, Stem, David E., Lorell, Mark H., Lussier, Frances M. (2005). *Lessons Learned from the F/A-22 and F/A-18E/f Development Programs*. MG-276. Santa Monica, CA: RAND Corporation.

Brent Johnstone *is a Lockheed Martin Fellow and production air vehicle cost estimator at Lockheed Martin Aeronautics Company in Fort Worth, Texas. He has 34 years' experience in the military aircraft industry, including 31 years as a cost estimator. He has worked on the F-16 program and has been most recently the lead Production Operations cost estimator for the F-35 program. He has a Master of Science from Texas A&M University and a Bachelor of Arts from the University of Texas at Austin.*

Leveraging the Wisdom of Crowds with Modern Regression, Machine Learning, and Ensembles with Application to Army Software Sustainment

Christian B. Smart, Ph.D., CCEA®
Kimberly Roye, CCEA®
Cheryl Jones
James Doswell
Paul Janusz
Brad Clark

Abstract: Cost estimating often relies on the log-transformed ordinary least squares method for the development of cost estimating relationships. This method has weaknesses; the most significant of which is that it provides estimates that are biased low. These deficiencies can be corrected, and predictive accuracy can be improved, using modern regression methods and applying machine learning techniques. Statisticians have found that predictive accuracy can be even further improved through the combination of multiple models in an ensemble, or crowd approach.

The article discusses these methods in detail and applies them to an extensive dataset of 192 Army systems. Data analysis reveals several types of cost estimating relationships based on release type, release rhythm, and categories of data. This article discusses significance testing and goodness-of-fit metrics for all models developed.

Introduction

Cost estimating is defined as the process of collecting and analyzing historical data and applying quantitative models, techniques, tools, and databases to predict the future cost of an item, product, program or task. Commonly, Log-Transformed Ordinary Least Squares (LOLS) method is used in the development of cost estimating relationships (CERs). There are some disadvantages with using LOLS: estimates are usually biased low and, in some cases, the estimates are not optimal.

LOLS is biased in the sense that it is estimating the median, rather than the mean, of a lognormally distributed estimate. For a lognormal, the median is always less than the mean. This is an issue in cost estimating as estimates are rarely conducted in isolation, but rather as part of a larger WBS or a portfolio of systems. When these values are summed, the resulting total is less than the median of the total estimate. However, the means always add, so it is important to develop estimates at the mean value. (Smart 2017)

To correct for the bias, analysts can consider applying nonlinear regression methods and machine learning methods to develop models and predict future costs. In this paper, we will provide introductions to these less commonly used methods and discuss how the results can be combined to increase the predictive accuracy of cost estimates.

Using a dataset of Army software programs, the analysis will compare results of a log-transformed OLS model to the results using the application of cross-validation using a crowd approach to determine how predictive power is influenced by this method.

Understanding the Crowd Approach

Many people have encountered a situation in which they had to solve a math problem and used more than one way to solve it. As long as both methods yielded the same result, it is reasonable to think this would provide more confidence in the answer. In studying machine learning, research has shown that using multiple methods can improve predictive accuracy. For example, John Elder has been a pioneer in the use of ensemble models and has repeatedly found this to be the case (Elder 2003). Typically, when a model is too complex, it does not predict well in practice. However, the more models one uses, the more complex the overall prediction engine, but the predictions do better in practice than single models used by themselves. This was a revelation to the authors.

An early example of ensemble predictions is a competition to guess the weight of an ox at an English county fair in 1906. Eight hundred people entered the contest. The statistician Francis Galton (he coined the term regression) was interested in the results, and thought that the average, or mean, result, would be far from the actual weight. He was surprised to discover that the actual weight of 1,197 pounds was only one pound from the mean of the 800 submissions. (Surowiecki 2004)

More recently, the data science company Kaggle started hosting competitions to solve problems requiring prediction. The best known of these is the $1 million Netflix prize, where people submitted models to improve the company's recommendation system. The catch with Kaggle submissions is that you get a score on how you do, but you cannot see the actuals to see the error of each estimate. Ensemble techniques have consistently proven to be crucial to winning submissions to these competitions, including the Netflix prize. In that particular competition, two teams tied for first. The tie breaker went to the team who submitted first. The first-place finisher submitted twenty minutes before the second-place finisher and won the entire $1 million prize. (Siegel 2016)

Ensembles are prominent in weather forecasting, especially in storm prediction. Whenever a tropical storm or a hurricane is discussed in a weather forecast, there will be a predicted plot for several different models. Most model predictions will cluster near a common path, but occasionally one or two predictions may be very different. The true path will likely be much closer to the path most of the model predictions rather than the outliers. Using multiple models helps to avoid being influenced by such predictions.

The idea is counter-intuitive. In most applications, you should expect that the best is better than the average. You would not expect two mediocre athletes to perform better on average than a superstar. However, that is what happens with models. The average of several models, some of which may be good, and others that may be mediocre, will on average perform better than the best model. Studies have shown the improvement ranges between five and 30 percent. Even better, using ensembles has been shown to improve out-of-sample prediction, meaning that models will predict well when used in practice. (Siegel 2016)

In *The Wisdom of Crowds* by James Suroweicki (2004), the concept of the crowd approach is discussed. When you put together a large enough and diverse enough group of people and ask them to make decisions affecting matters of general interest, that group's decisions will be intellectually superior to the isolated individual. Applying this concept to cost estimating, we can determine that in the right circumstances, an average forecast is better than a single forecast.

The use of multiple techniques for prediction is called the ensemble approach. An ensemble is a group of items viewed as a whole rather than individually. Suppose we have multiple models from which we would like to choose the "best". The models could be constructed using different datasets, methods, variables or equation forms.

There are (at least) two ways to combine estimates:

- Using a simple average
- Using a more general method that considers correlation between the estimates

We will examine the benefits to be obtained by each of these methods.

Simple Averaging
Simple averaging combines the estimates by computing the means of the estimates.

$$(x_1 + x_2 + \cdots + x_n)/n$$

Let the residuals of a cost estimating relationship (CER) equation be defined by ε. The residuals could be:

- Absolute, as with a linear equation: $\varepsilon = y - f(x)$
- Percentage, as with a nonlinear equation: $\varepsilon = \frac{y-f(x)}{f(X)}$

Regardless of the residual form, the variance of an individual is defined as $\hat{\sigma}^2_{Individual} = E[\varepsilon^2]$. If there are multiple equations, the variance of the average of the CERs is $\hat{\sigma}^2_{Average} = E\left[\left(\frac{1}{N}\sum_{i=1}^{n}\varepsilon\right)^2\right]$. Assuming independence,

$$\hat{\sigma}^2_{Average} = E\left[\left(\frac{1}{N}\sum_{i=1}^{n}\varepsilon\right)^2\right] = \frac{1}{N^2}E\left[\sum_{i=1}^{n}\varepsilon^2\right] + \frac{1}{N^2}E\left[\sum_{i\neq j}\varepsilon_i\varepsilon_j\right]$$

$= \frac{1}{N^2}E[\sum_{i=1}^{n}\varepsilon^2] = \frac{1}{N}\left[\frac{\sum_{i=1}^{n}\varepsilon^2}{N}\right]$.

The quantity $\left[\frac{\sum_{i=1}^{n}\varepsilon^2}{N}\right]$ is the mean of the variances of the individual models. Thus, the SPE of the average of the models is the average of the variances divided by N.

For example, suppose for two nonlinear models, we have standard deviations equal to 30 percent and 50 percent, respectively. The first model has variance $=0.3^2=0.09$ and the second model has variance $=0.5^2=0.25$.

The variance of the simple average is $\frac{1}{2}\left(\frac{0.09+0.25}{2}\right) = 0.085$, which is lower than the better of the two models. By reducing the variance, we have also decreased the uncertainty in the estimate.

Weighted Average
With the simple average, we observe improvement over a single model. What happens when our estimates are correlated, meaning, they use the same data sources or we are comparing similar methods and model forms? When this occurs, the need to use the weighted average approach to incorporate correlation arises.

To calculate the weighted average among models, a correlation matrix, ***nxn***, between the estimates should be created.

The i,jth element of the correlation matrix is $\boldsymbol{E[\varepsilon_i\varepsilon_j]}$. Let $\boldsymbol{\alpha}$ denoted the ***nx1*** vector of weights for the estimates.

The SPE of the weighted average is $\boldsymbol{\alpha^T C\alpha}$**.** The weights should be constrained so that their sum is equal to 1, and the weights should be chosen to minimize the SPE.

With these constraints, we use the Lagrangian multipliers method and minimize

$$\boldsymbol{L = \alpha^T C\alpha - 2\lambda(\alpha^T\vec{1} - 1)}$$

where $\vec{\mathbf{1}}$ denotes the ***nx1*** vector of all ***1s.***

To minimize, we take the first derivative and set it equal to 0.

$$\frac{\partial L}{\partial \alpha} = 2C\alpha - 2\lambda\vec{1} = 0$$

Rewriting yields

$$2C\alpha = 2\lambda\vec{1}$$

Dividing both sides by 2 and multiplying both sides by C^{-1} results in

$$\alpha = \lambda C^{-1}\vec{1}$$

Multiplying both sides by $\vec{1}^T$ yields

$$\vec{1}^T\alpha = \lambda\vec{1}^T C^{-1}\vec{1}$$

The left side of the equation is the sum of the weights, which we constrained to be 1, thus

$$\lambda = \frac{1}{\vec{1}^T C^{-1}\vec{1}}$$

Since $\vec{1}^T C^{-1}\vec{1} = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}^{-1}$, if we plug the expression λ back into $\alpha = \lambda C^{-1}\vec{1}$ the result is

$$\alpha = \frac{C^{-1}\vec{1}}{\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}^{-1}}$$

The SPE of the weighted average method is equal to

$$MSE = \frac{(C^{-1}\vec{1})^T C(C^{-1}\vec{1})}{\left(\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}^{-1}\right)^2} = \frac{\vec{1}^T C^{-1}\vec{1}}{(\vec{1}^T C^{-1}\vec{1})^2} = \frac{1}{\vec{1}^T C^{-1}\vec{1}} = \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}^{-1}}$$

In the uncorrelated case,

$$\alpha_i = \frac{\frac{1}{\hat{\sigma}_i^2}}{\sum_{j=1}^{n}\frac{1}{\hat{\sigma}_j^2}} \quad MSE = \frac{1}{\sum_{i=1}^{n}\frac{1}{\hat{\sigma}_i^2}}$$

To compare the weighted average SPE with others (in the uncorrelated case), we need the following:

Lemma – If **a** and **b** are positive numbers, then $\frac{a}{b} + \frac{b}{a} \geq 2$ with equality when a =b

Proof:

$$(a - b)^2 \geq 0$$

if and only if

$$\mathbf{a}^2 - 2\mathbf{ab} + \mathbf{b}^2 \geq 0$$

if and only if

$$\mathbf{a}^2 + \mathbf{b}^2 \geq 2\mathbf{ab}$$

if and only if

$$\frac{a}{b}+\frac{b}{a}\geq 2$$

Comparing Simple Average to Weighted Average

The Variance of the weighted average is less than or equal to the SPE of the simple average with equality only when all the individual SPEs are the same. Note that:

$$Variance_{Simple\ Average}=\frac{1}{N^2}\sum_{i=1}^{N}\hat{\sigma}_i^2$$

$$\mathbf{Variance_{Weighted\ Average}}=\frac{1}{\sum_{i=1}^{n}\frac{1}{\hat{\sigma}_i^2}}$$

The weighted average variance is smaller than the simple average variance. To see this, consider

$$\frac{1}{N^2}\sum_{i=1}^{N}\hat{\sigma}_i^2\geq\frac{1}{\sum_{j=1}^{N}\frac{1}{\hat{\sigma}_j^2}}$$

This is true if and only if

$$\sum_{i=1}^{N}\hat{\sigma}_i^2\sum_{j=1}^{N}\frac{1}{\hat{\sigma}_j^2}\geq N^2$$

The left side of this inequality is a sum of ratios

$$\hat{\sigma}_i^2\frac{1}{\hat{\sigma}_j^2}$$

This expression is equal to 1 when $i=j$. When $i\neq j$, there is always a pair $\frac{\hat{\sigma}_i^2}{\hat{\sigma}_j^2}+\frac{\hat{\sigma}_j^2}{\hat{\sigma}_i^2}$. There are N values for $i=j$ and for other pairs there are $\binom{N}{2}=\frac{N(N-1)}{2}$ values. Therefore, the expression on the left, using the Lemma, is at least $\mathbf{N+2*\frac{N(N-1)}{2}=N+N^2-N=N^2}$ and equality only occurs if all the variances are equal. Therefore, the weighted average variance is smaller than the simple average variance and is strictly smaller when the variances are not all the same value.

We can also show that the weighted average has a variance smaller than the best single model in a crowd. Recall that $MSE_{Weighted\ Average}=\frac{1}{\sum_{i=1}^{n}\frac{1}{\hat{\sigma}_i^2}}$. To see that the variance of the weighted average is less than or equal to the minimum of the individual variances, note that

$\hat{\sigma}_{Min}^2\geq\frac{1}{\sum_{i=1}^{n}\frac{1}{\hat{\sigma}_i^2}}$ if and only if

$$\hat{\sigma}_{Min}^2\sum_{i=1}^{n}\frac{1}{\hat{\sigma}_i^2}\geq 1$$

Without loss of generality, assume $\hat{\sigma}_1^2=\hat{\sigma}_{Min}^2$. Then, $\hat{\sigma}_{Min}^2\left(\frac{1}{\hat{\sigma}_1^2}+\cdots+\frac{1}{\hat{\sigma}_N^2}\right)=1+\hat{\sigma}_{Min}^2\left(\frac{1}{\hat{\sigma}_2^2}+\cdots+\frac{1}{\hat{\sigma}_N^2}\right)\geq 1$.

In summary, the weighted average approach should be used when estimates or datasets are correlated. Though, in analyses with few data points, correlation estimates may not be as accurate as the simple average.

Practical Example Using Army Data

Army Data Description

Through the support of Office of the Deputy Assistant Secretary of the Army – Cost and Economics (DASA-CE) leadership, the software sustainment initiative has succeeded over the past five years of moving the U.S. Army from a position of making educated guesses on what was being spent on software sustainment and its utility, to being able to provide deep insights from an Army-wide perspective into how software sustainment is being performed, how much it costs, and what software is being delivered to the warfighter. The initiative created an Army Software Sustainment Data Questionnaire which is used to collect system context-information, annual cost and effort data, software release data, and data on software licenses.

The information in the database includes software release level data as well as management and process data on over 192 Army systems in sustainment. The information in the database supports the detailed analysis of software sustainment cost, schedule and risk drivers, and provides insight into the state of software sustainment management and processes practices.

The results establish a robust foundation for software sustainment fact-based decisions, including:

- Allocations of Costs by Work Breakdown Structure (WBS) Elements
- Cost & Schedule Estimating Relationships
- Cost Benchmarks

The amount of data collected resulted in over 411,000 repository data fields based on 192 Systems, 1,040 Releases and 3,434 software licenses, Figure 1.

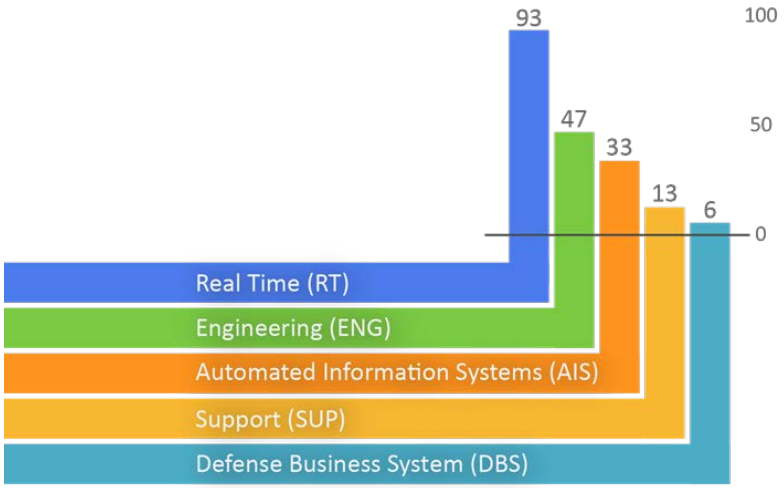


Figure 1. Data Demographics

The Army dataset used for this analysis includes the following variables:

- Total Release Hours (Dependent)
- Super Domain (Independent)
- Total Software Changes (Independent)
- Acquisition Category (ACAT) (Independent)

These variables were selected based on the causal analysis on the data (Jones, et al. 2020). Total Release Hours is defined as the effort (in hours) required to maintain software in the WBS Element 1.0, Software Change Product. This effort changes the software to improve its capability or repair a problem. When systems were divided into application super domains, there were 93 Real-Time Systems (RT), 47 Engineering Systems (ENG), 33 Automation Information Systems (AIS), 13 Support Systems (SUP), and 6 Defense Business Systems

(DBS). In the dataset used for this analysis, there are no DBS datapoints. These three independent variables were identified as being influential in past studies conducted by the army. See "Using Army Software Sustainment Cost Estimating Results (DASA-CE)" from September 2018 for more details.

Systems were asked to report the size measures that were used within their program. Software Changes (SC) was the most common size measure with data provided for 571 releases. SCs are enhancements or maintenance changes to the software.

US DoD's ACAT levels are also analyzed. There are three levels and an additional category for non-Program of Record (non-POR). The difference between each level depends on the location of a program in the acquisition process, funding amount for Research, Development, Test and Evaluation, total procurement cost, Milestone Decision Authority special interest and decision authority. ACAT I programs are major defense acquisition programs.

The upper and lower 10% of the data was trimmed from the dataset to subset extreme cases for separate analysis. Trimming was based on unit cost (total release hours / #software changes). While the data had been scrubbed for hours and cost outliers, some of the unit costs were extremely low and some were extremely high.

Using the trimmed dataset, we used two nonlinear methods and four machine learning methods to predict Total Hours given Super Domain, Total Software Changes, and ACAT level.

Nonlinear Regression Methods

We will introduce two nonlinear methods: Maximum Likelihood Estimation Regression for Lognormal Error (MRLN) and Zero-Percent Bias Minimum Percent Error (ZMPE). Dr. Christian Smart developed the MRLN method, which uses Maximum Likelihood Estimation (MLE) to directly estimate the mean lognormal without the use of transformations (Smart 2017). This method does not require lognormal transformations of either the dependent or independent variables.

The likelihood function, which represents the likelihood of obtaining the sample data, is:

$L(\theta) = \prod_{i=1}^{n} \Pr\,(X_i = A_i|\theta)$.

The vector, θ, maximizes the likelihood function in the MLE. Using this technique provides a major advantage: the likelihood function is almost always available. LOLS is an MLE of the median when the residuals are lognormally distributed.

Applying the MLE directly to the residuals yields an estimate of the lognormal mean. The goal for MRLN is to maximize the function:

$$l(\beta_0, \beta_1, \dots, \beta_p, \theta) = -\frac{n}{2}\ln(\theta) - \frac{1}{2\,\theta}\sum_{i=1}^{n}\left(ln(y_i) - \ln(\beta_0) - \sum_{j=1}^{p}\beta_j \ln(X_{ij}) + \frac{\theta}{2}\right)^2$$

Excel Solver can be used to perform this task. When Solver converges on a solution, Excel calculates the optimal values for $\boldsymbol{a}$ and $\boldsymbol{b}$ to form the power equation.

Dr. Steve Book developed the ZMPE method, which focuses on minimizing the sum of squared errors subject to the constraint that the sample bias is zero.

The goal of this method is to find a function, $y = f(x)(1+\varepsilon)$, with a multiplicative error, $\varepsilon = \frac{y-f(x)}{f(x)}$, that fits a data set so that the following are satisfied:

- Let $\boldsymbol{y}$ denote the actual and $\boldsymbol{f(x, a, b) = a\,X^b}$ denote the estimate. ZMPE minimizes

$$\sum_{i=1}^{n}\left(\frac{y_i - f(x_i, a, b)}{f(x_i, a, b)}\right)^2$$

subject to the constraint that the sample bias is zero,

i.e.,

$$\sum^{n}\left(\frac{y_i - f(x_i, a, b)}{f(x_i, a, b)}\right) = 0$$

The result is an optimal solution, $\boldsymbol{a}$ and $\boldsymbol{b}$, and can be calculated in Excel Solver.

Machine Learning Methods – A Refresher

In "Beyond Regression: Applying Machine Learning to Parametrics" (Roye, Smart 2019), multiple machine learning techniques were discussed in detail. Four supervised learning methods are used for this analysis and are briefly described in the following sections:

- Regression Trees
- Random Forests
- Support Vector Machines
- K-Nearest Neighbors

Supervised learning techniques in machine learning is the process of an algorithm learning from a subset of a given dataset, referred to as the training dataset. In supervised learning, the input variables and output variables are named. The algorithm learns from the mapping function from the input and output. With this method, the goal is to approximate the mapping function so that new outputs can be predicted using new input data.

Regression Trees

A decision tree is a decision support tool useful in classifying data. Tree-based methods are options for analysis, because the data are split into homogenous groups, and the graphs present these splits with the use of branches (called decision nodes) and leaves (terminal nodes). The goal of tree-based methods is to partition data into smaller regions where interactions are manageable. They are useful when there is a non-linear and complex relationship between dependent and independent variables. There are two types of trees: classification and regression trees.

Regression trees are used when the dependent variable of interest is continuous. Figure 2 presents the components of a regression tree.

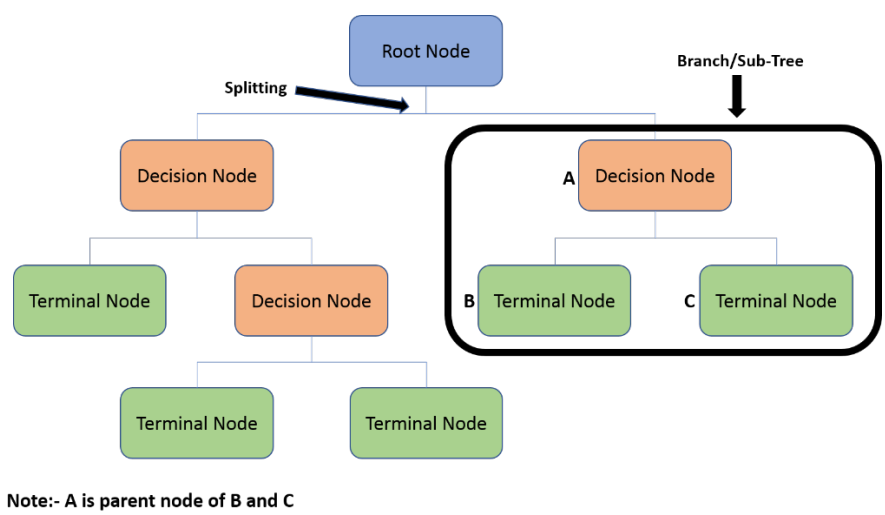


Figure 2: Regression Tree Layout

The root node represents the entire population, or most commonly, the sample dataset that is being explored. Decision trees recursively split a dataset into partitions based on a criterion. Starting with the root node of the tree, the method asks a sequence of yes and no questions to determine the decision nodes. The root node splits into two or more decision nodes. The decision nodes represent the first set of homogenous groups discovered within the dataset. When the algorithm determines which cut-off point minimizes the variance of y for a regression task, the branch ends in a leaf, or terminal node. Leaves represent a cell of partition and have a simple model for that cell; the model is the sample mean of the dependent variable.

Figure 3 provides the regression tree using the example data. At the first decision node, if a release has less than 61 software changes, the left side of the tree when followed to the next decision tree node which splits again on less than 13 software changes. This first decision node represents the first set of homogeneous software releases within the dataset. Based on each smaller group, the tree splits again on either the number of software changes, ACAT level I or II programs, or Real Time super domain releases. The tree ends at the terminal nodes and provides the average Total Release Hours (lognormally transformed) of the data points included in each node. In Figure 3, the numbers in the oblong circles above the nodes (root, decision and terminal nodes) are the average Total Release Hours and the percent of the sample. At the root node, the average Total Release Hours is 8.42 and since this is before any splits occur, there is 100% of the sample included.

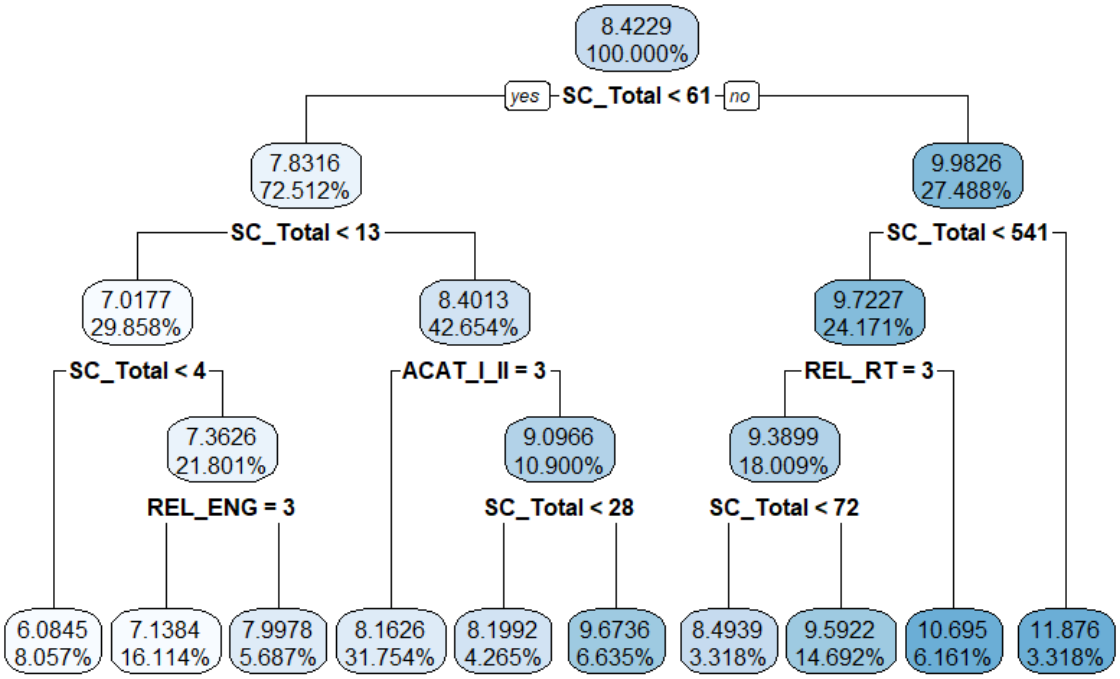


Figure 3: SW Sustainment Regression Tree

Random Forests

The Random Forest algorithm can be thought of as an ensemble approach using regression trees. This approach combines the estimates of multiple regression trees to produce an average. Random forests have been proven to provide better prediction ("wisdom of the crowd" effect). They are also more stable (robust to small amounts of noise). However, since the predictions are rather complex, there is no single equation or CER.

Random forest adds additional randomness to a model. The algorithm searches for the best feature among a random subset of features, which results in more diversity that usually results in better prediction.

Support Vector Machines

The application of Support Vector Machines (SVM) in the 1990s to optical character recognition was very successful. (Boser, et al., 1992) The basic idea for classification with this method is to maximize the margin between classes, which yields maximally robust classification. To apply to continuous output, the analogous idea is to find an equation that is:

- As "flat" as possible, i.e., the coefficients are as small as possible
- Emphasis on sparseness, parsimony
- Makes model less sensitive to errors in inputs
- Minimizes the residuals that are outside a specified range of the estimate (ε-insensitive), e.g., 15%

For a linear equation ***Y = a+bX***, with ***n*** data points the problem becomes

$$\boldsymbol{Minimize}\text{:}\ \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{a}^{\mathbf{2}}+\boldsymbol{b}^{\mathbf{2}})+\boldsymbol{C}*\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}}\boldsymbol{\delta}_{\boldsymbol{i}}$$

$$\boldsymbol{Subject\ to}\ |\boldsymbol{y}_{\boldsymbol{i}}-\boldsymbol{a}-\boldsymbol{b}\boldsymbol{x}_{\boldsymbol{i}}|\leq \boldsymbol{z}+\boldsymbol{\delta}_{\boldsymbol{i}}\ \boldsymbol{for\ all\ i}=\mathbf{1},\ldots\boldsymbol{n}$$

where the delta values are non-negative, and the loss function is insensitive to residuals less than z (user specified), and a weight equal to C is given to the errors (controls for degree of parsimony). For an example of insensitive losses, for a $10 million project, you may not care about the residual as long as it is no larger than $1 million.

Given a nonlinear equation ***Y = aX^b***, take log transforms of the data and apply the linear support vector set up. The insensitivity is now in log-space – the log of the differences between the actual and the estimate.

As an alternative to logarithmic transformation, you can apply the same notion to the absolute value of percentage difference between the actuals and the estimates, i.e.

$$Minimize\text{:}\ \tfrac{1}{2}(a^2+b^2)+C*\textstyle\sum_{i=1}^{n}\delta_i$$

$$Where\ \delta_i=\left\{\begin{matrix}\left|\frac{y_i-ax_i^b}{ax_i^b}\right|-0.15\ if\ \left|\frac{y_i-ax_i^b}{ax_i^b}\right|\geq 15\% \\ 0\ otherwise\end{matrix}\right\}\ for\ i=1,\ldots n$$

For solving this optimization problem, Excel's Solver capability can be used.

Results

First, we will present the CER developed using LOLS.

$$\begin{aligned}Y\ (Total\ Release\ Hours)\\ &= 6.42-0.55*ENG\ SD-0.055*RT\ SD-0.95*AIS\ SD+0.30*ACAT+0.72\\ &*Total\ Software\ Changes\end{aligned}$$

Super Domain and ACAT were defined as categorical variables (1 or 0). If ACAT is 1, then the datapoint was obtained from an ACAT I or II program. As a reminder for the Super Domain designations, Real-Time Systems (RT), Engineering Systems (ENG), and Automation Information Systems (AIS). Super Domain and ACAT variables are categorical variables, while Total Software Changes and Total Release hours are continuous variables.

The data used for the nonlinear and machine learning analysis was log-transformed to facilitate the comparison of the results to the LOLS model.

CERs were also developed using ZMPE and MRLN.

Next, we will discuss how to compute the simple and weighted averages using the estimates ($\hat{Y}$) for each of the methods.

For the simple average, the average of $\hat{Y}$ for MRLN, ZMPE, Regression Trees, Random Forest, SVM, and KNN was calculated. This average $\hat{Y}$ was used to calculate the goodness-of-fit statistics.

For example, consider the estimates for the first 10 datapoints for the six methods presented in Table 1.

Observation	MRLN	ZMPE	R-Tree	R-Forest	SVM	KNN
1	7204.21	9768.36	5448.91	8091.82	4074.67	5222.58
2	6885.73	9308.97	5448.91	8108.96	4017.75	4513.26
3	7166.47	4888.93	5448.91	4597.52	5310.05	5316.64
4	574.66	660.47	682.06	4990.52	1758.86	2467.58
5	9015.69	12405.18	5448.91	7944.49	4695.12	3620.33
6	2596.23	1657.38	5448.91	3695.92	3080.11	2635.91
7	8660.85	10453.97	24690.77	13522.31	16743.01	23840.75
8	6885.73	9308.97	5448.91	8108.96	4017.75	4513.26
9	31608.07	47208.50	22762.34	14850.36	38877.70	32809.06
10	2035.47	1278.89	1956.78	3248.21	2335.26	2843.57

Table 1: Estimates for Nonlinear Prediction Methods

We will then take the simple average of the estimate for the methods, such that:

$$Simple\ Average = \frac{\hat{Y}_{MRLN} + \hat{Y}_{ZMPE} + \hat{Y}_{RTree} + \hat{Y}_{RForest} + \hat{Y}_{SVM} + \hat{Y}_{KNN}}{6}$$

Table 2 presents the averages of the six explored methods for the first 10 data points. This new average estimate is then used to calculate the goodness-of-fit statistics.

Observation	MRLN	ZMPE	R-Tree	R-Forest	SVM	KNN	Average
1	7204.21	9768.36	5448.91	8091.82	4074.67	5222.58	6635.09
2	6885.73	9308.97	5448.91	8108.96	4017.75	4513.26	6380.60
3	7166.47	4888.93	5448.91	4597.52	5310.05	5316.64	5454.75
4	574.66	660.47	682.06	4990.52	1758.86	2467.58	1855.69
5	9015.69	12405.18	5448.91	7944.49	4695.12	3620.33	7188.29
6	2596.23	1657.38	5448.91	3695.92	3080.11	2635.91	3185.74
7	8660.85	10453.97	24690.77	13522.31	16743.01	23840.75	16318.61
8	6885.73	9308.97	5448.91	8108.96	4017.75	4513.26	6380.60
9	31608.07	47208.50	22762.34	14850.36	38877.70	32809.06	31352.67
10	2035.47	1278.89	1956.78	3248.21	2335.26	2843.57	2283.03

Table 2: Estimates and Averages for Nonlinear Prediction Methods.

For the weighted average, the calculation is a little more involved. First, all six models were included in the weighted average. The algorithm returned negative weights for two models – MRLN and KNN. Because of the ambiguity of negative weights, we decided to remove those models, and apply the weighted average over four models instead. This resulted in weights for ZMPE, Regression Trees, Random Forests, and SVM equal to 27.0%, 26.0%, 38.4%, and 8.6%, respectively.

MRLN	ZMPE	R-Tree	R-Forest	SVM	Weighted Average
7204.21	9768.36	5448.91	8091.82	4074.67	7511.86
6885.73	9308.97	5448.91	8108.96	4017.75	7389.51
7166.47	4888.93	5448.91	4597.52	5310.05	4958.84
574.66	660.47	682.06	4990.52	1758.86	2423.28
9015.69	12405.18	5448.91	7944.49	4695.12	8220.58
2596.23	1657.38	5448.91	3695.92	3080.11	3548.33
8660.85	10453.97	24690.77	13522.31	16743.01	15874.64
6885.73	9308.97	5448.91	8108.96	4017.75	7389.51
31608.07	47208.50	22762.34	14850.36	38877.70	27710.52
2035.47	1278.89	1956.78	3248.21	2335.26	2302.21

Table 3: Estimates and Weighted Averages for Nonlinear Prediction Methods.

Nonlinear models need different measures of goodness-of-fit than are used for linear ones. The goodness-of-fit measures we use are commonly used in nonlinear modeling and are the nonlinear analogues to traditional linear regression. These are *Pearson's R^2*, the *Standard Percent Error*, and *Sample Percent Bias*.

Goodness-of-fit metrics were calculated for each method and used to compare to the LOLS model.

- Pearson's R^2- The square of the correlation coefficient between the actual and estimated effort
- Standard Percent Error (SPE) – The standard deviation of the difference between the actual and estimated effort as a percentage of the estimated effort
- Sample Percent Bias – Average percentage error

Pearson's R^2, which we will refer to as R^2, is defined as:

$$\frac{\left\{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)\right\}^2}{\left\{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right\}\left\{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right\}}$$

Standard Percent Error is defined as:

The standard percent error is a nonlinear analog to the regression standard error, and is defined as

$$SPE = \sqrt{\frac{1}{n-k}} * \sum_{i=1}^{n}\left[\frac{y_i - f(x_i)}{f(x_i)}\right]^2 * 100\%$$

where $\boldsymbol{n}$ is the sample size, and $\boldsymbol{k}$ is the number of fitted coefficients. In this case, lower values are desired.

Sample Percent Bias, which we refer to as Bias, and is defined as:

$$\sum_{i=1}^{n}\left(\frac{y_i - f(x_i)}{f(x_i)}\right)/n$$

The goodness-of-fit statistics were calculated for the in-sample and out-of-sample datapoints. For machine learning techniques, a training and a test sample from the dataset are randomly sampled. For this analysis, an 80% training sample (211 data points) was taken from the dataset, with the remaining 20% (52 data points) being used for testing. The training sample is used to fit each machine learning model, while the test sample provides an unbiased evaluation of the final model fit on the training sample. The in-sample results are calculated from the training sample; the out-of-sample results are calculated from the test sample.

The machine learning methods were all biased low initially because of the log transformation. A sample bias correction adjustment was made, in line with the adjustment MRLN makes, to correct for this sample bias.

The in-sample results are shown in Table 4 and the out-of-sample results are displayed in Table 5.

Method (In-Sample)	R^2	SPE	Bias
LOLS	51.10%	178.74%	-58.17%
MRLN	51.10%	104.34%	2.20%
ZMPE	47.87%	97.93%	0.00%
Regression Trees	62.01%	124.40%	0.00%
Random Forest	47.08%	133.74%	0.00%
SVM	63.90%	135.72%	0.00%
KNN	57.50%	123.43%	0.00%
Simple Average	62.69%	97.05%	13.56%
Weighted Average*	61.45%	78.91%	37.19%

Table 4: Goodness-of-Fit Statistics for the In-Sample Data

Method (Out-of-Sample)	R^2	SPE	Bias
LOLS	92.52%	263.15%	-98.57%
MRLN	92.52%	151.20%	-22.78%
ZMPE	90.71%	143.79%	-17.99%
Regression Trees	84.84%	241.76%	-67.21%
Random Forest	45.59%	307.37%	-48.03%
SVM	45.18%	262.01%	-44.26%
KNN	65.55%	289.17%	-54.06%
Simple Average	91.88%	216.40%	-51.82%
Weighted Average*	90.10%	153.78%	-10.10%

Table 5. Goodness-of-Fit Statistics for the Out-of-Sample Data

For the in-sample data, the single best SPE belongs to ZMPE, followed by MRLN. Among single models, SVM has the highest R^2, followed by MRLN. The simple average has an R^2 comparable to the best of any single model, along with an SPE comparable to the best of any single model. The weighted average has a similar R^2 but a much smaller SPE – however it has a significantly positive bias.

The out-of-sample data is a better indicator of how the models will perform when applied in practice, as the coefficients were not influenced by any of these data. For the out-of-sample data, ZMPE and MRLN are the two best single models – they both have R^2s in excess of 90%, and SPEs significantly better than the other methods. Out-of-sample R^2s can sometimes outperform in-sample R^2s if the training data contains more complicated relationships between the dependent and independent variables than the test dataset. The simple average has a -50% bias and a much higher SPE than MRLN or ZMPE. The weighted average, however, has an R^2 and an SPE, that is comparable to both MRLN and ZMPE, as well as a lower bias. The weighted average is better overall than any single model.

Conclusion

Whether the scenario involves guessing the weight of an animal at a county fair or trying to determine how much a new variant of a combat vehicle will cost, an ensemble approach can often produce a better estimate than a single model. This is a little counterintuitive. To obtain the cost estimate for a program, one might think it would be optimal to find the best cost model to give you an estimate. But a better result may be obtained by averaging estimates from two (or more) mediocre cost models instead. Ensembles seem to consistently produce more accurate estimates.

The authors applied the ensemble concept to estimating Army software sustainment costs. For these data, we have shown that weighted averages generalize better than most models and perform as well or better out of sample than the single best model. Though it is often most common to produce a single regression estimate, introducing the ensemble approach can increase the accuracy of prediction. The authors also applied machine learning methods that can compete with traditional regression methods.

In Appendix A, an additional concept of cross validation is discussed. Though this method was not implemented in this paper, it is an important concept to consider.

References

Boser, Bernhard E.; Guyon, Isabelle M.; Vapnik, Vladimir N. (1992). "A training algorithm for optimal margin classifiers". Proceedings of the fifth annual workshop on Computational learning theory – COLT '92. p. 144.

Dyson, F., "A Meeting with Enrico Fermi," Nature 427 (22 January 2004), page 297.

Elder IV, John F , Ph.D, (2003). The Generalization Paradox of Ensembles. Journal of Computational and Graphical Statistics, 12 (4), 853–864.

Harrell. F.E., *Regression Modeling Strategies*, 2010, Springer-Verlag, New York.

Jones, C., J. Doswell, B. Clark, R. Charette, J. Judy, and P. Janusz, "New Army Software Sustainment Cost Estimating Results," presented at the ICEAA annual workshop, Tampa, May 2019.

Jones, C., J. Doswell, P. Janusz, B. Clark, C. Smart, and K. Roye, "Improving Software Estimating Relationships for Army Software Sustainment Data," ICEAA annual workshop, Virtual, May 2020.

Mitchell, T., *Machine Learning*, 1997, McGraw-Hill, Boston.

Petty, C., C. B. Smart, and J. Lawlor, "Seven Degrees of Separation," presented at ICEAA Professional Development & Training Workshop, June 2015, San Diego, CA.

Roye, K., and C.B. Smart, "Beyond Regression: Applying Machine Learning to Parametrics," presented at the ICEAA Professional Development & Training Workshop, May 2019, Tampa, FL.

Siegel, E., *Predictive Analytics*, 2016, John Wiley & Sons, Hoboken, page 202.

Silver, N., *The Signal and the Noise: Why So Many Predictions Fail – But Some Don't*, 2012, Penguin Books, New York.

Smart, C.B., "The Signal and the Noise in Cost Estimating", presented at the ICEAA International Training Symposium, Bristol, October 2016.

Smart, C.B., "Maximum Likelihood Estimation for Regression of Log Normal Error," presented the ICEAA Professional Development and Training Workshop, Portland, June 2017.

Suroweicki, J., *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*, 2004, Doubleday, New York.

Dr. ***Christian B. Smart*** *is the Chief Scientist for Galorath Federal. He has a Ph.D. in Applied Mathematics, with more than 20 years of experience with the application of predictive analytics, machine learning, and risk management to defense and aerospace programs. In 2020, Dr. Smart published* Solving for Project Risk Management: Understanding the Critical Role of Uncertainty in Project Management *with McGraw-Hill. He is the 2021 recipient of the Frank Freiman Lifetime Achievement Award from ICEAA.*

Ms. ***Kimberly Roye*** *is a Senior Data Scientist with Bank of America. Starting her career as a Mathematical Statistician for the US Census Bureau, Kimberly transitioned to a career in Cost Analysis over 10 years ago. She has supported several Department of Defense hardware, software and vehicle programs, as well as NASA and the Department of Homeland Security (DHS). She is currently a lead developer of Machine Learning training for the Army and DHS. Kimberly earned a MS in Applied Statistics from Rochester Institute of Technology and a dual BS in Mathematics/Statistics from the University of Georgia.*

Ms. ***Cheryl Jones*** *is a senior systems engineer at the US Army Futures Command and the technical lead for the Army Software Sustainment Cost Estimation initiative. The objective of this project is to provide estimation approaches to accurately estimate and justify software resources. Ms. Jones is the project manager of Practical Software and Systems Measurement and the DoD representative to ISO SC7, System and Software Engineering. She is co-editor of ISO/IEC/IEEE 15288, Systems Life Cycle Processes.*

Dr. ***Brad Clark*** *is Vice-President of Software Metrics Inc. His area of expertise is software cost and schedule data collection, analysis and parametric modeling. He co-authored a book with Barry Boehm titled Software Cost Estimation with COCOMO II and another with Ray Madachy titled Software Cost Estimation Metrics Manual for Defense Systems. Dr. Clark received a Ph.D. in Computer Science in from the University of Southern California.*

Mr. ***Paul Janusz*** *is a senior software quality engineer at the US Army Futures Command at Picatinny Arsenal. He is responsible for implementing software measurement and independent verification and validation for a wide variety of armament software intensive systems. Currently, Mr. Janusz is supporting the Army Software Sustainment Cost Estimation initiative, assessing how software sustainment is being performed on Army programs, analyzing their estimated and actual measurement data, and evaluating the resulting impacts upon the software that is delivered to the warfighter. He is a member of the Practical Software and Systems Measurement (PSM) project, adapting the measurement framework to account for the issues faced by sustainment projects and the unique characteristics of continuous iterative development projects.*

Mr. ***James Doswell*** *is a senior software cost analysis at DASA-CE, responsible for independent government cost estimates for Army systems. He serves as the division's Software Team Lead, responsible for overseeing software analysis, developing predictive models, data collection, and organization wide software cost estimates.*

But Wait, There's More!
Using SiSE for your Cost, Schedule & Performance Needs

Katharine Mann
Ryan Hoang

Abstract: Software cost estimating is a challenging effort when there is little information known about a program early in the lifecycle. Historically, the Department of Homeland Security (DHS) has had difficulty in properly sizing software development efforts for Life Cycle Cost Estimates (LCCEs). In 2017, the DHS Cost Analysis Division piloted the Simple Function Point Analysis (SFPA) methodology, later renamed to Simple Software Estimation (SiSE), tying high level requirements in existing Acquisition documentation to a standard sizing metric. After demonstrating successes with several programs, additional value of SiSE results for program management beyond cost estimating was explored. This paper and presentation will cover the SiSE methodology, requirements definition and analysis, and how functional size can be used to effectively manage program cost, schedule, and performance. We will use real DHS programs, policies, and lessons learned to demonstrate the benefits of SiSE as a secret ingredient for program management success and describe how to engage with program managers to employ this tool in their programs. Finally, we will discuss our future research efforts and initiatives to implement SiSE across federal acquisitions. We believe this to be an innovative and exciting way to estimate and manage software development programs in any organization.

Keywords: Software cost estimating, Agile, Simple Function Points, Software sizing, Program Management, Stakeholder Engagement, Requirements, Functional Sizing

Introduction

1.1 Agile Requirements

Developing agile requirements is a different process than the traditional waterfall approach used in U.S. Government software acquisitions [1]. Gone are the days of defining and finalizing hundred-page requirements documents before typing a single line of code; gone are the months of development that may or may not deliver software that functions as originally intended. Instead, the Agile Manifesto prioritizes continuous delivery of working software, which often requires changing requirements late in development to suit the customer's needs. [2] This is a momentous paradigm shift in the acquisition of new IT software systems.

There have been difficulties in implementing Agile development approaches across the Government, as many Agile principles conflict with established processes. For example, Agile only plans work over weeks or months; however, the federal budget process requires funding requests to be prepared for submission to Congress nearly two years before those funds would be received. As a result, acquisition programs must be able to estimate future software development work based on vague or unknown requirements. As seen in Figure 1 the agile scrum development process is incremental and iterative, the key premise on delivering working software to the customer for continuous feedback and refinement. To address the need for flexible, user-centric requirements that still meet the federal acquisition regulations associated with taxpayer funding, cost estimators need a way to estimate software development from flexible high level agile requirements.

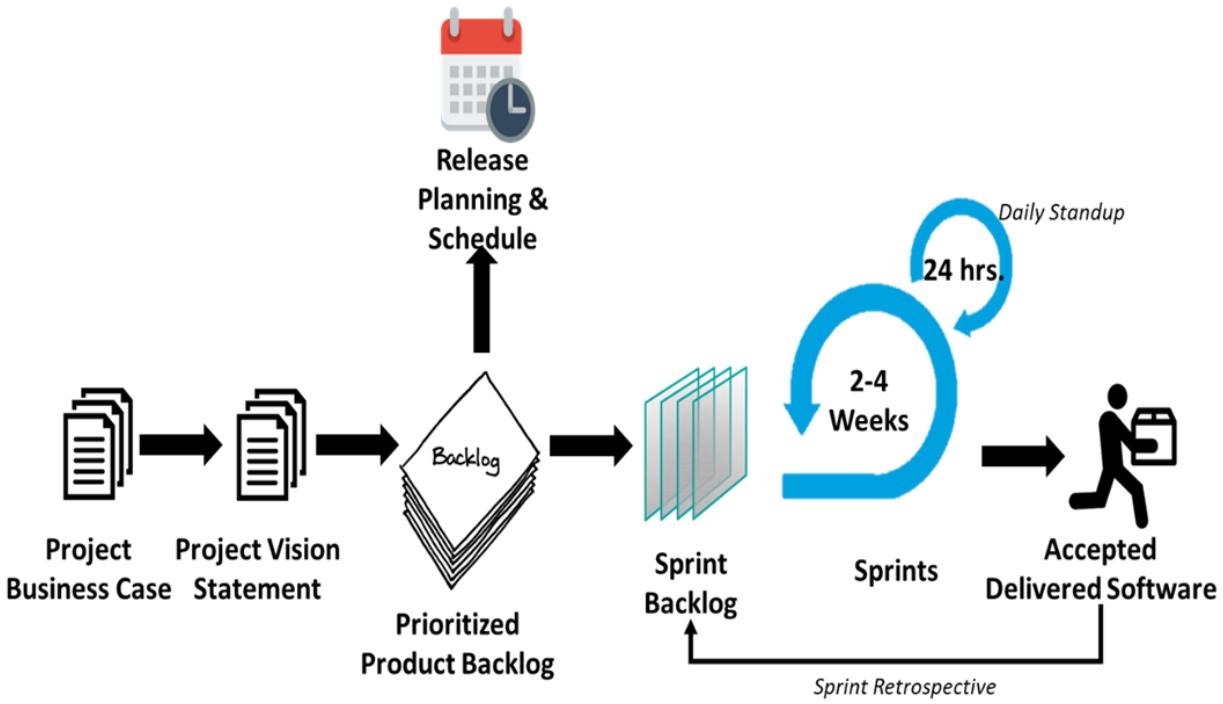


Figure 1: Agile SCRUM Process Chart, adapted from the SCRUM Body of Knowledge (SBoK, 2017)

1.2 Software Cost Estimating

The basics to estimating the cost of software development break down to the simple equation:

Effort = Size x Throughput

where Size is equal to the scope of working software produced to meet the customer's needs, expressed through a standard unit of measurement, and Throughput is the rate at which software developers can design, code, test, and deliver working software to the client. Effort can then be easily converted to cost as a build-up from labor rates and quantities.

Of the two components depicted in Figure 1, throughput is easier to quantify. Throughput is typically presented as a "per time" metric, such as Lines of Code per Hour, Function Points per Month, or Story Points per Sprint, and can be determined from historical development rates. Size is much more difficult to measure, as there is a myriad of ways to calculate the size of the software system being developed; below is a brief discussion of three common size measurements.

1.2.1 Story Points

In *Agile Estimating and Planning*, Cohn wrote: "Story points are a unit of measure for expressing the overall size of a user story, feature, or other piece of work. When we estimate with story points, we assign a point value to each item. The raw values we assign are unimportant. What matters are the relative values. A story that is assigned a two should be twice as much as a story that is assigned a one. It should also be two-thirds of a story that is estimated as three story points." [3] Story points, as stated by Cohn, are a subjective unit of measure. An individual Agile team assigns them to identify the relative difficulty of various tasks, and usually require a prioritized product backlog of user stories, which is not developed until much later in the acquisition process; therefore, it is difficult to aggregate and compare between development teams and different programs. Additional technical information is necessary to derive a normalized relationship between the effort performed by different teams and their assigned story points. While we believe that story points

are an important and indispensable part of the individual agile team planning process for sprints and backlog burn-down, story points do not lend themselves to long-term program management.

1.2.2 Software Lines of Code (SLOC)

Most commonly used within the Department of Defense, Software Lines of Code (SLOC) is the physical count of lines of text in the source code. As stated in the 2019 Defense Innovation Board Metrics for Software Development "The current state of practice within the DoD is that software complexity is often estimated base on the number of source lines of code (SLOC), and its rate of progress is measured in terms of programmer productivity. While both of these quantities are easily measured, they are not necessarily predictive of cost, schedule, or performance." [4] According to code.org, "Of course, every engineer knows that 'lines of code' is a silly measure…No software engineer measures the value of their work in lines of code. In fact, the best-designed programs often have the simplest designs and the fewest lines of code." [5] SLOC may provide some general idea of the scope of a development effort for an Rough Order of Magnitude estimate, but variations in code length due to type of programming language and the coding efficiency of individual developers means accuracy of SLOC estimates are likely inconsistent.

1.2.3 Function Points

The Function Point, developed by IBM's Allan Albrecht in 1979, is a standard unit of measurement based on how a system uses information. Capers Jones, leading authority in software estimating, stated: "Function Point metrics are the most accurate and effective metrics yet developed for software sizing and also for studying software productivity, quality, costs, risks, and economic value. Unlike the older 'lines of code' metric Function Points can be used to study requirements, design, and in fact all software activities from development through maintenance." [6] Function Points are agnostic of programing language or development methodology (e.g., waterfall, agile). Since its inception, the methodology governed by the International Function Point User's Group (IFPUG) established in 1986 is an International Organization for Standardization (ISO) standard. A Function Point is consistent regardless of who performs the count or what the system does. While the counting process involves some interpretation, experienced Function Point counters can produce counts for a system within 5% of each other. Figure 2 is a pictorial representation of system components that need to be understood when calculating Function Points. Drawbacks to Function Points can include the time required to learn the counting practice, time to conduct a full count, and the effort required to obtain certification.

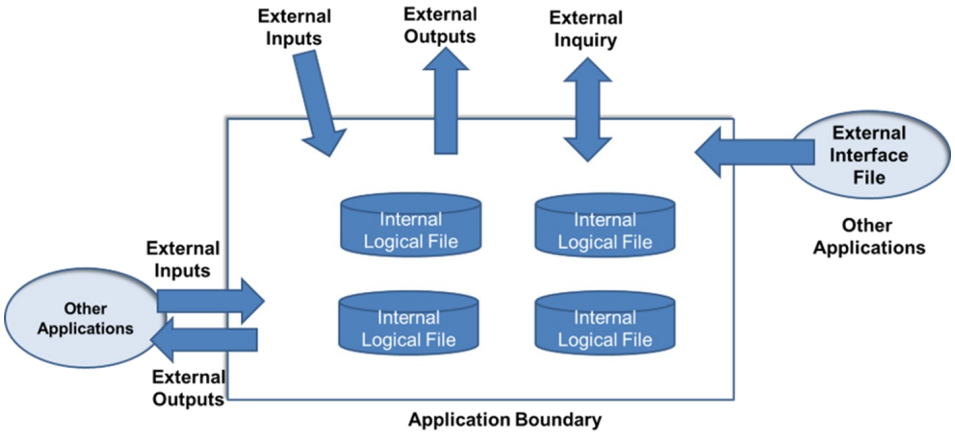


Figure 2: Pictorial representation of Function Point counting components of a system

1.3 Challenges to the Department of Homeland Security

Each year, DHS invests billions of taxpayer dollars into everything from helicopters for Customs and Border Protection (CBP), vessels for the U.S. Coast Guard, baggage screening equipment for the Transportation Security Administration (TSA), and complex software systems. These software systems are used for such purposes as administering Federal Emergency Management Agency (FEMA) grants, processing U.S. citizenship applications, and monitoring the enforcement of illegal immigration. In a recent report by the Government Accountability Office (GAO) in May of 2017 it was noted that "in fiscal year 2016, the department's IT budget of approximately $6.2 billion was the third largest in the federal government." [7] Like many other federal agencies in the U.S. Government, the Department of Homeland Security (DHS) has struggled with estimating the cost of and establishing realistic schedules for large IT programs.

One of the primary challenges experienced by DHS is accurately estimating the size of software development efforts. Many of these efforts result in public-facing systems and have many stakeholders with various needs, leading to complex sets of requirements. In the cost estimating field, developing an estimate is not conceptually difficult, as estimates are often just build-ups of labor; the justification of those inputs is what presents the major challenge. Agile development principles often conflict with established processes in the traditional acquisition lifecycle framework. Development teams will continually shift or add requirements as directed by the customer to deliver working software, but how can a program tell that it has completed what it originally set out to do? Understanding the true scope of programs is the missing piece to improving program management practices.

1.4 Charge by the Under Secretary for Management

In 2017, the DHS Under Secretary for Management (USM) charged the Cost Analysis Division (CAD) under the DHS Office of the Chief Financial Officer (OCFO) to find a way to improve cost estimates for Agile software development programs. There were two primary objectives:

1. Enhance the credibility and accuracy of a software development estimate and
2. Decrease the time required to develop the estimate.

At the time, DHS had designated five software development programs as pilots for implementing agile processes and best practices and providing lessons learned for other DHS endeavors. In addition to being highly visible major acquisitions, these programs were at various stages of the acquisition lifecycle and had experienced common challenges with cost, schedule, and performance. The charge by the USM provided a timely opportunity for the Cost Analysis Division to expand its technical knowledge of software development and attempt some novel estimating methods. From discussions with industry and Government partners, the Cost Analysis Division learned of the benefits of functional sizing techniques and identified functional sizing as a promising solution to the current dilemma.

2. SIMPLE SOFTWARE ESTIMATION (SiSE)

2.1 SiSE Process Overview

The Cost Analysis Division developed a custom software cost estimation process called Simple Software Estimation (SiSE), combining the open-source Simple Function Point (SiFP) method as published by Dr Roberto Meli (v1.01) and work pioneered and shared by analysts at the National Security Agency (NSA). [8] The SiSE Estimating

IFPUG Components	Low	Average	High
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

SFPA Components	Weighting Factor
Transactions (Create, Update, Delete, Report, Read)	4.6
Logical Data Groups (Saves)	7

Figure 3: Mapping between IFPUG and SiSE Components and Weightings

Process combines functional software sizing (i.e., quantifying business function/transaction types, system interfaces, and requirements counts from high-level acquisition documentation) together with software productivity rates (e.g., hours per function point) to determine Agile software development effort and costs. Note that we realize that there are Diseconomies of Scale (DoS) associated with software development estimating; however, we currently assume a simplified linear relationship between software functional size and productivity. The Simple Function Point method was acquired by the International Function Point User's Group (IFPUG) in September of 2019, which indicates growing interest in the underlying methodology. [9]

The SiSE functional sizing methodology leverages the process of IFPUG's ISO certified counting practices manual. The IFPUG counting practice estimates the size of software based on an understanding of the system's lowest-level business transactions (External Inputs, External Outputs, and External Inquires) and data storage interfaces (Internal Logical Files and External Interface Files) as seen in Figure 2. The IFPUG counting method requires the counter to quantify the complexity of each transaction or data storage component, based on a set of criteria. Then, depending on the component type and complexity, a Function Point value is assigned. The SiSE method was developed as an alternative to this lengthy and labor-intensive Function Point counting process.

The SiSE method maps the IFPUG components to two groups – Transactions (i.e., Create, Update, Delete, Report, and Read), which map to External Inputs (EI), External Outputs (EO), or External Queries (EQ), and Logical Data Groupings (i.e., Saves), which map to Internal Logical Files (ILF) and External Interface Files (EIF). Figure 3 illustrates this mapping between the IFPUG components and their Function Point counts, and the SiSE components and weightings.

The Cost Analysis Division's research illustrates that functional requirements are typically expressed as action verbs (e.g., "submit," "maintain," "receive"). Each requirement can be decomposed into one or more components (groupings of generic transactions and/or data groups) and corresponding weighing factors from the Simple Function Point method. Work done by functional sizing experts produced a lexicon of 140+ action verbs and their associated components. With the associated size for each action verb pre-defined, a functional size estimate for a set of requirements can be produced and totaled quickly.

To understand a software system's business transactions and estimate software requirements, the Cost Analysis Division uses a program's Concept of Operations (CONOPS), a high-level acquisition document developed early in our acquisition lifecycle that describes what functions

the completed system will do. The CONOPS is reviewed and validated by the DHS requirements and technical communities to ensure all required capabilities are captured before a program moves further through the acquisition lifecycle. The SiSE sizing step leverages the action verbs used in the CONOPS written functional requirements to quickly estimate a Simple Function Point size of the software. Once the initial size estimate is calculated, additional factors and risk may be applied to the estimated size to anticipate software growth, complexity, and program uniqueness.

The program office and the appropriate technical communities should then validate the final size estimate to ensure a consistent interpretation of the requirements used for the estimate. The Cost Analysis Division uses analogous historical and industry data to determine a throughput/ productivity rate used with the estimated Simple FP size to estimate the total software development effort for the program. This is then time-phased across the schedule to estimate the software development cost for a program's Life Cycle Cost Estimate (LCCE).

2.2 What is a Good Requirement?

It should be obvious that successfully conducting SiSE depends on a solid understanding of the functionality of the system being developed. It is impossible to assess the accuracy of any sizing estimate without understanding what a developed system does, and much of this understanding comes from written program documentation. With the shift to Agile methods also comes the mindset that documentation is secondary to developed software due to constantly changing requirements; therefore, it is crucial that the early, high-level program requirements are well written. There are many factors to consider when writing requirements [10]:

1. **User's perspective** – SiSE focuses on functional size, i.e. the actions that the system performs when it is operational. Good requirements should capture those actions. Non-functional requirements such as availability, maintainability and reliability, while important considerations during development, do not factor directly in SiSE.
2. **Unique / One action per requirement** – A good requirement should only describe one individual action. Including multiple actions in a requirement may cause confusion and lead to effort being underestimated.
3. **Clear and concise actions** – In the Agile spirit, requirements should be direct to keep documentation minimal. Keeping the written requirement concise also helps ensure that the functions are easily recognized.
4. **Consistent level of detail** – Good requirements for SiSE should be described at similar levels of detail. If a requirement is overly detailed or broken into multiple smaller actions, it may lead to that effort being overestimated relative to others.
5. **Testable / Verifiable** – Good requirements should have criteria to determine if the requirement has been developed properly and the capability met. This allows for development progress to be accurately tracked.

Various artifacts such as the CONOPS or a Functional Requirements Document (FRD) are produced as a program increases in maturity and describe requirements at differing levels of detail. The Cost Analysis Division is exploring using other DHS document sources for SiSE sizing to include the FRD, Requirements Traceability Matrix (RTM), and the Software Requirements Document (SRD). Analysts can also derive SiSE from user stories pulled from project management tools such as JIRA for sizing. Other federal agencies can utilize similar high-level requirements documentation if the organization does not employ CONOPS to the detailed software business function level. Performing SiSE with each of these documents will produce different

sizing estimates. Work is ongoing to investigate which requirements documents provide the most useful information to accurately estimate functional size.

3. BUT WAIT, THERE'S MORE!

As part of the Simple Software Estimating process, the Simple FP estimate quantifies the size of the functional requirements for a development effort. This number, when used by a cost estimator, provides a justifiable input for estimating cost. But after performing this sizing effort to produce just one number – the Simple FP estimated size, is that it? No! There are many ways that our size estimate, when utilized effectively by a program office, can provide maximum value by influencing many aspects of program management activities.

3.1 Developing Schedules – "When can this be delivered?"

One of the first items that a program needs to have agreement on is the realistic duration of the software development effort. There have been studies conducted that provide metrics on development rates for functional sizing (ex: FP/ team-month, etc.). Using a standard approach like Simple FP with an appropriate productivity rate to estimate our software development effort (in hours or person months) we can then estimate, using historical development rates, the schedule duration to complete the software development. If a schedule was already assigned to a program, this schedule estimation can assess the reasonableness of existing development timelines. The program can then justify to decision makers why pre-assigned milestones (or deadlines!) may be unrealistic and should be delayed or re-evaluated.

3.2 Estimating Resources – "What staff is needed?"

If timelines are already established, SiSE can assist program management with an easy way to quantify how many resources will be required to meet those deadlines. Using software development rate metrics together with the Simple FP size estimate the assigned milestone dates, an analyst can estimate the required team size and quantity to meet the desired schedule. If the available team size is insufficient, the analysis provides solid, objective justification to ask for additional program resources and funding.

3.3 Planning Agile Sprints – "What is everyone's workload?"

If based on good requirements (i.e., unambiguous, clearly stated, functional requirements), a cost analyst can use the Simple FP estimate, with a relevant productivity rate to estimate the effort required to develop each of those requirements. Because each requirement is objectively quantified, Agile teams can appropriately divide tasks when planning sprints and minimize potentially over-assigning work. Program managers can use also use this approach to assess team throughput and ensure that they are all producing similar amounts of functionality. This approach is far more objective and applicable across teams than the alternative velocity metric (expressed as Story Points per Sprint) typically used by Agile development teams.

3.4 Reviewing Vendor Proposals – "Is this bid realistic?"

This paper provides an overview on how programs can use SiSE to assess internal schedules and resourcing. This estimating process can also be applied to assessing vendor proposals for software development services, to validate that the scope of work is mutually

understood between the Government and contract offerors. The Simple FP sizing estimate can provide a quick cross-check to the overall amount of effort proposed, as well as gauge reasonableness of the delivery timeline and the staffing proposed to meet those dates. This will allow programs to better evaluate best-value proposals when awarding contracts.

3.5 Tracking Progress – "How is the program performing overall?"

The results of SiSE combines with other noted analyses to produce a baseline for accurate tracking of development progress. An initial cumulative Simple FPs "estimate to complete" chart can be plotted to project completion dates and effort, using assumed development rates and proposed staff. Plotting cumulative, delivered Simple FPs completed after each sprint against this initial projection can provide a program manager with valuable information in the form of a Burn Up chart. Program and project managers can track current development progress and see if the project progress is trending as planned. Deviations will provide an early indication of potential issues and allow the program to react pre-emptively. Establishing a visual representation of such progress also provides an instrument to initiate useful communication with leadership and focus the conversation on issues that require attention. An example of a visual representation can be seen in Figure 4.

3.6 DHS Examples of SiSE Use

The pilot of SiSE methodology on DHS programs has resulted in successes in various aspects of program acquisition processes. Three examples are highlighted in this section.

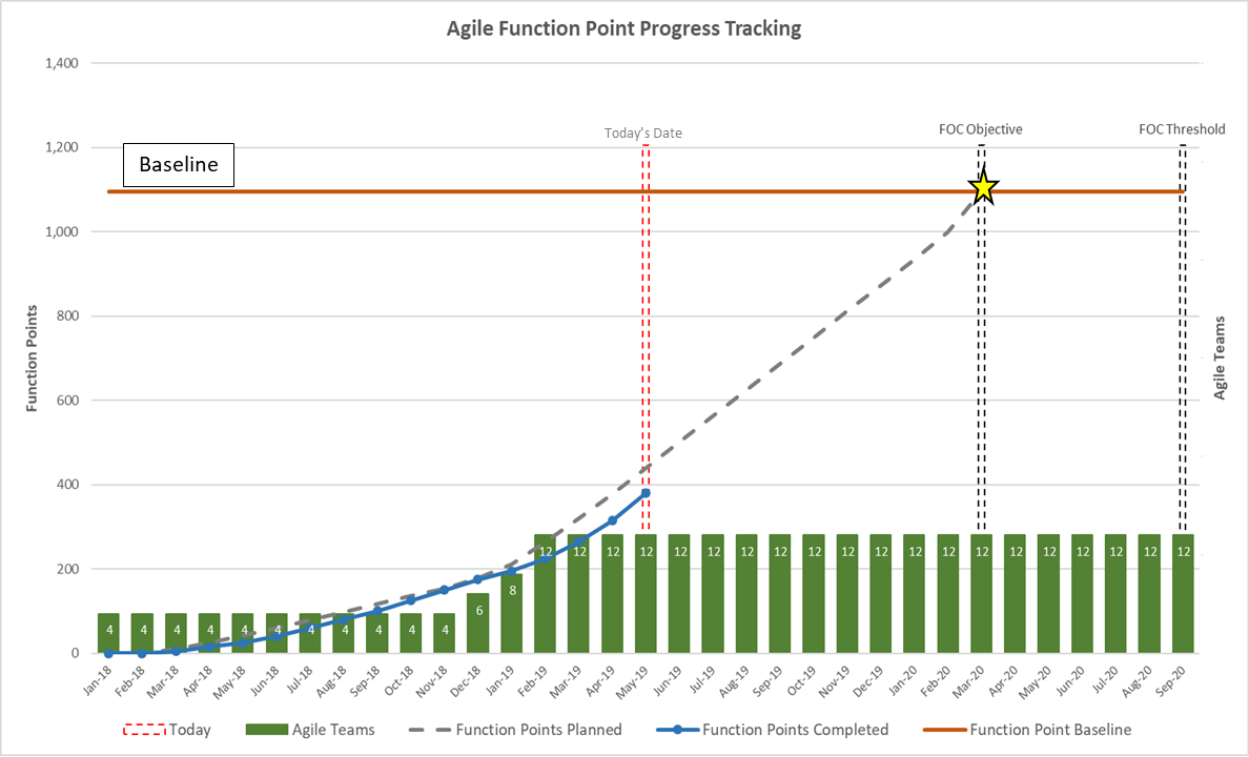


Figure 4. Example Function Point Tracking Chart

3.6.1 Program A

Program A was one of the first pilot programs for testing the Simple Function Point Analysis (SFPA) methodology, the predecessor to SiSE. The Program was a Level 2 ($300M-$1B total lifecycle cost) agile development program for a public facing web-based system. The program's CONOPS clearly detailed the user requirements via functional capabilities statements, which made it very easy to apply SiSE to estimate functional size. The estimated SiFPs were adjusted for risk and used with current throughput rates to update the program's LCCE for Department approval. Program A's LCCE helped to prove the methodology's viability for DHS programs; the program's requirements statements are some of the primary examples the Cost Analysis Division uses to educate other programs how to effectively write requirements for SiSE.

3.6.2 Program B

Program B is a Level 1 ($1B+ total lifecycle cost) program in the obtain phase of the acquisition lifecycle for a very complex, critical system with large computing/storage requirements and interfaces with systems both internal to DHS and external to stakeholders and partners. The program estimated Function Points for the system using the COSMIC sizing methodology [11].

As part of an Independent Cost Assessment (ICA) of the program's LCCE, the Cost Analysis Division used SiSE to size requirements described in one of the program's capability documents. The software development costs calculated through SiSE were within 8% of the program's estimate, a reasonable range for an independent cross-check. The Cost Analysis Division's ICA and the approval of Program B's LCCE demonstrated the value of SiSE for developing a software size estimate quickly and with similar accuracy to other standardized Function Point counting methods.

In addition to using Function Points in the development of the LCCE, Program B also implemented a progress tracking chart as described in Section 3.5. The chart is presented to stakeholders whenever the program meets with the DHS Acquisition Review Board for milestone decisions or program reviews. Trends projected in the chart have been consistent with progress observed as the program continues development activities.

3.6.3 Program C

Program C is a Level 2 program in the obtain phase of the acquisition lifecycle for a system that streamlines many unique process workflows into a single management platform. The program recently updated their LCCE to reflect a shift in acquisition approach to agile software development. The Cost Analysis Division was able to collaborate with the program to apply SiSE on business functions described in the program's CONOPS; the use of SiSE did not require Program C to create any new acquisition documents specifically for the LCCE update. Part of the program's updates also included re-baselining schedule milestones due to lower staffing levels than planned. The Cost Analysis Division used the Simple Function Point estimate along with throughput data and agile team quantities to project system development and identify a new date to reach Full Operational Capability. The recommended milestone dates were consistent with the schedule provided by the program's development contractor. The work done with Program C showed SiSE's ability to be performed on requirements regardless of development approach, as well as reduce program overdependence on contractors for program management activities.

4. SUCCESSES WITH SiSE IMPLEMENTATION

Over the last few years, there has been a large amount of progress made by the Cost Analysis Division in implementing SiSE in DHS. We highlight several accomplishments, as well as ongoing efforts to improve, refine, and expand the SiSE methodology to provide maximum value to the Department.

4.1 Leadership Support

DHS Leadership has supported the use of SiSE as a methodology to estimate software development sizing. They have recognized the objective nature of Simple Function Points and their standard calculation, as well as the link to functional requirements. Tracking SiFPs has begun to focus discussions of development progress on capabilities delivered rather than deadlines promised. In a May 2019 memo from the DHS Acting Chief Financial Officer, all new or re-baselining Major Acquisition programs are now required to use functional sizing for estimating software development effort.

4.2 Joint Agile Software Innovation Cost IPT (JASI CIPT)

The Joint Agile Software Innovation (JASI) Cost Integrated Product Team (IPT) was founded in 2018 by representatives from DHS, the National Security Administration (NSA), and the National Geospatial-Intelligence Agency (NGA) with three objectives:

1. Develop a pragmatic and defendable approach to estimate and measure software development through Functional Sizing
2. Improve data availability to enhance the credibility of estimates
3. Investigate new approaches to track, measure, and report progress of an agile program throughout its development lifecycle

Through JASI, the Cost Analysis Division provides SiSE training to over a dozen different audiences, with more sessions planned. JASI has also expanded to include membership from thirteen federal agencies in the Defense, Intelligence, and Civilian cost communities. These agencies all recognize the potential for SiSE to improve the development of cost estimates and are excited to implement SiSE in their own organizations. JASI CIPT won the 2019 Team Achievement award through the Washington Capital Area Chapter of the International Cost Estimating and Analysis Association (ICEAA) in recognition of the collaborative efforts of the team.

4.3 Adoption by New Acquisition Programs

The Cost Analysis Division's efforts to promote SiSE and provide training in the methodology to current acquisition programs have spread awareness across the department. Several early phase acquisitions have indicated that they are attempting to use SiSE as part of their program planning activities. To date, two DHS programs independently used SiSE to develop their Rough Order of Magnitude estimates.

4.4 Engagement with DHS Stakeholders

The Cost Analysis Division is working with acquisition stakeholders across the department to refine, improve and standardize SiSE. The Cost Analysis Division is collaborating with the Offices of the Chief Information Officer and Chief Technology Officer to improve and standardize written requirements and develop processes to validate Simple Function Point-based LCCEs from a technical perspective. The Cost Analysis Division is also engaging with the Office of the Chief Procurement Officer to facilitate collection of valuable performance metrics as part of future Agile development contracts.

4.5 Data Collection

The Cost Analysis Division is undergoing efforts with many DHS agile programs to collect data on completed software. Data being collected includes written requirements and respective Simple FP counts, agile team quantities and composition, effort to develop functional requirements, and actual costs, among others. The Cost Analysis Division intends to use the data to refine the number of Simple FPs assigned to various requirements statements, as well as develop DHS-specific throughput rates to improve size and schedule estimates for future programs. Simple FP estimates from different requirements documents will also be examined to determine which document type provides the most accurate and appropriate basis of estimate.

5. CONCLUSIONS

5.1 Benefits and Summary

The Cost Analysis Division believes SiSE offers many benefits to Agile acquisition programs. SiSE provides a faster, more reliable, and repeatable process for cost estimators to produ63ce credible estimates of functional size and development effort. The methodology leverages high-level documents created early in the acquisition lifecycle, allowing long-term analysis of system capabilities without being impacted by agile processes that can shift development priorities. Lastly, integrating functional sizing into other aspects of program management provides additional value to program managers by tying all activities to the same requirements and can be communicated consistently to leadership and decision makers.

5.2 Future Work

The SiSE methodology is still a "work in progress." We seek to improve this methodology based on data and lessons learned by programs as they progress through software development. All Cost Analysis Division efforts referenced in this paper are ongoing, with the hope that the SiSE process will soon become a standard not only within DHS, but across the U.S. Federal Government.

6. ACKNOWLEDGMENTS

The authors would like to thank Mr. David Seaver, Mr. Lyle Patashnick, Mr. Tyrese Johnson, and Mr. Collin Brooks for their mentorship and early work piloting SFPA/SiSE at DHS. Thank you to Ms. Carol Dekkers for her Function Point SME knowledge and enhancement of the methodology. Thank you to Dr. Wilson Rosa for his guidance and to the Cost Analysis Division and DHS OCFO leadership for their support and encouragement to write this paper.

References:

[1] Government Accountability Office, "Software Development Effective Practices and Federal Challenges in Applying Agile Methods," United States Government Accountability Office, Washington DC, 2012.

[2] Agile Manifesto, "Manifesto for Agile Software Development," 2001. [Online]. Available: http://agilemanifesto.org/. [Accessed 30 January 2020].

[3] M. Cohn, Agile Estimating and Planning, Upper Saddle River: Pearson Education, Inc., 2006.

[4] Defense Innovation Board, "Defense Innovation Board Metrics for Software Development," 9 July 2018. [Online]. Available: https://media.defense.gov/2018/Jul/10/2001940937/-1/-1/0/DIB_METRICS_FOR_SOFTWARE_DEVELOPMENT_V0.9_2018.07.10.PDF. [Accessed 9 February 2020].

[5] Code.org, "How many lines of code?," 2020. [Online]. Available: https://code.org/loc. [Accessed 30 Janaury 2020].

[6] C. . Jones, "Function points as a universal software metric," ACM Sigsoft Software Engineering Notes, vol. 38, no. 4, pp. 1-27, 2013.

[7] Government Accountability Office, "Progress Made to Implement IT Reform, but Additional Chief Information Officer Involvement Needed," 18 May 2017. [Online]. Available: https://www.gao.gov/products/gao-17-284. [Accessed 31 January 2020].

[8] Simple Function Point Association, "SiFPA," 2020. [Online]. Available: http://www.sifpa.org/en/metodo/. [Accessed 30 January 2020].

[9] International Function Point Users Group, "IFPUG Acquires the Simple Function Point Method," 12 September 2019. [Online]. Available: https://www.ifpug.org/ifpug-acquires-the-simple-function-points-method/. [Accessed 30 January 2020].

[10] D. French and C. Dekkers, "From Point A to Point Estimate: How Requirements Become Function Points," in ICEAA Professional Development & Training Workshop, Tampa, 2019.

[11] COSMIC, "COSMIC-Sizing," 2020. [Online]. Available: https://cosmic-sizing.org/ . [Accessed 18 February 2020].

Katharine Mann*, CCEA, is an Operations Research Analyst for the Department of Homeland Security, Cost Analysis Division (CAD) IT / Agile Software Development Team. Before joining DHS, Ms. Mann supported the NATO Communication and Information Agency (NCIA) in Brussels, Belgium, and various DoD programs. She is the Outreach Chair of the Washington Capital Area Chapter of ICEAA and Secretary of the Joint Agile Software Innovation (JASI) Government IPT. Ms. Mann has Undergraduate and Master's degrees in Industrial and Systems Engineering from Virginia Tech.*

Ryan Hoang*, CCEA, is an Operations Research Analyst for the Department of Homeland Security, Cost Analysis Division (CAD), IT / Agile Software Development Team. Since 2015, he has provided cost support to many major IT acquisition programs across DHS. He has B.S. and M.Eng. degrees in Chemical Engineering from Cornell University.*

Empirical Investigation of Engineering Change Order Percentages in Defense Contracts

Captain Kaiana M. Miller
Edward D. White, Ph.D.
Jonathan D. Ritschel, Ph.D.
Robert D. Fass, Ph.D.
Scott Drylie, Ph.D.

Abstract: Engineering Change Orders (ECO) are technical requirements changes to existing contracts. To account for the potential increase in contract costs stemming from ECOs, current acquisition practice is to estimate a dollar value to hold in management reserve (MR) in case of ECO occurrence. Estimators often rely on rules-of-thumb when developing these estimates. Specifically, many estimators use a 10% rule-of-thumb for estimating MR contract costs in the Development life cycle phase and a 5% rule-of-thumb for contracts in the Production life cycle phase. However, no empirical data appear to support or validate these 10% and 5% figures. Using a new data source, 1,216 contracts with ECOs were analyzed to determine the accuracy of the 10% and 5% rules-of-thumb as well as to determine if more accurate rules-of-thumb could be developed. Results suggest that if a contract is likely to have a positive ECO percentage, then 14% and 6% rules-of-thumb are more statistically appropriate for contracts in the Development and Production life cycle phases respectively. Lastly, Service, Contract Type, Commodity, Initial Program Size, and Schedule appear to impact ECO percentages.

Introduction

Since at least 1983, the Department of Defense (DoD) has instructed cost estimators to include in their contract estimates an additional percentage of the total costs to be held in reserve as a buffer against the possibility of an Engineering Change Order (ECO) (Gibson, 1983). An ECO is a tool used by management to direct a scope change to a contract (Engineering Change Proposals, 2021). This scope change is typically technical (e.g., correction of a design error that does not become evident until testing and modeling). Such scope changes amount to cost growth. Therefore, it would be beneficial to the government if accurate predictions could be made about the appropriate amount to hold in reserve. Reserving too much money limits the number of programs able to be funded. Reserving too little money puts a program at risk of being delayed or even cancelled.

The Government Accountability Office (GAO, 2008) determined that 63% of Major Defense Acquisition Programs (MDAPs) required contractual changes after system development. Such changes included administrative, engineering (also referred to as technical), and added non-technical work requirements changes. The same report showed that poorly defined requirements in acquisition programs can create significant cost growth. Major defense programs that had requirement changes after initial system development experienced mean cost growth of

72% from initial estimate, while those that did not have requirement changes experienced only 11%.

Per Gibson (1983) in a DoD ECO guidebook, a 10% estimate has provided reasonable coverage for the unanticipated requirements on many programs. The guidebook also provides suggestions for when to deviate from the 10%. However, no empirical data has been found that substantiates the validity of the 10% percentage. Some practitioners have continued to anchor estimates to that 10% rule-of-thumb (ROT) for Development contracts (DEV_{ROT}) in addition, to using a ROT for Production contracts ($PROD_{ROT}$). Specifically, the Air Force Life Cycle Management Center uses a 5% reserve for $PROD_{ROT}$ (S. Valentine, personal communications, 2021).

This article has three objectives. The first is to investigate whether DEV_{ROT} of 10% and $PROD_{ROT}$ of 5% provide a good estimate of the amount to be held in reserve for ECOs. If the first objective indicates that either ROT appears inaccurate, then the second objective is to develop a more accurate ROT to account for the percentage increase in cost due to ECOs. In conjunction with objective two, the third objective is to determine which factors, such as service, commodity type, contract type, or contract length, may drive differences in ECO percentages. These factors stem from previous research (Christensen & Templin, 2000; Arena et al., 2006; Bolten et al., 2008; Harmon & Arnold, 2013; Kozlak et al., 2017; Trudelle et al., 2017a, 2017b; D'Amico et al., 2018; Ellis et al., 2018) that indicate a possible association with program cost growth.

Methods

The data used for this article originated from the DoD contracting system known as Electronic Document Access (EDA). EDA is an online resource in which government contracting agencies upload scanned copies of actual contractual documents (EDA, 2017). The Defense Cost and Resource Center (DCaRC) commissioned a support contractor to establish a separate database from batches of contracts from EDA, which were identified of value by defense analysts. In October 2021, the support contractor provided the authors of this article the EDA data in the form of an Excel database.

To the best of our knowledge, the contracts in the current database were not chosen randomly. Each year, the DoD office of Cost Assessment Data Enterprise (CADE) sends out a data call to cost agencies DoD-wide requesting a list of contracts on which analysts would like information. Cost agencies then send their contract list to the support contractor, which in turn, then searches for them in EDA and transfers the data to the CADE database. The CADE database is updated on a quarterly basis.

Basic DoD contracts and their modifications comprise the database. The database includes a column of dollar amounts for each contract and modification, normalized for inflation to fiscal year (FY) 2020 using the 2020 OSD inflation table. Besides contract baseline cost, ECO cost, and ECO percentage (ECO cost / baseline cost), the database also contains contract number, service, commodity type, program, life cycle phase, contract type, contract start date, period of performance (PoP) end date, and schedule length in days (difference between PoP and contract start date).

Because only development and production contracts were germane to the analysis, we remove any Operating and Support (O&S) contracts in the database. To minimize the effect of error or unrealistic baselines, we omit any contract that exceeds 100% in absolute value for an ECO percentage. This exclusion is in-line with exclusion criteria from Ellis et al. (2018). Table 1 highlights the complete inclusion/exclusion criteria for the database we analyzed. The database contained 11,481 unique contracts with their respective modifications (if any) and reasons for modification. The Appendix lists the

INCLUSION CRITERIA	CONTRACTS ADDED	CONTRACTS REMOVED	CONTRACTS REMAINING
Original Dataset	11,481		11,481
Non-Technical Modifications		8,537	2,944
Blank Baseline or ECO Cost		12	2,932
Absolute Value of ECO % > 100%		498	2,434
O&S Contracts		1,218	1,216

Table 1: Inclusion/exclusion criteria describing the establishment of the final analyzed database.

programs which had contracts in the database. Due to the nature of the available data, the analysis in this research is solely at the contract level as opposed to the program level.

We analyze the finalized data using both descriptive measures and statistical inferential tests. The descriptive measures include means, standard deviations, coefficients of variation (CV), medians, interquartile ranges (IQR), and a quartile-based CV of IQR/medians. The inferential tests include *t*-tests, the nonparametric Kruskal-Wallis test (Wilcoxon Rank Sum when just comparing two populations) and Steel-Dwass multiple comparisons, Pearson's Chi-squared test for dependency between variables, associated odds ratios and confidence intervals for significant odds ratios. Given the large sample sizes for initially testing the current DEV_{ROT} and $PROD_{ROT}$ of 10% and 5%, assessment of normality is not needed. However, when conducting further tests comparing various services, commodities, and contract types (Table 2 lists those in our database), we chose the conservative nonparametric approach over the customary analysis of variance and subsequent Tukey analysis. For level of significance, we use an alpha of 0.05 for all the inferential tests. JMP Pro 15 was the software used to perform the statistical calculations.

SERVICES	COMMODITIES	CONTRACT TYPES
Air Force	AIS (Automated Information System)	Cost
Army	Decoys	Fixed
DoD (two or more services)	Electronics	Time and Materials (T&M)
Navy (includes Marines)	Engine	
	F-16	
	F/A-18	
	Ground Vehicle	
	Gun	
	Missiles	
	Non-lethal	
	Ordnance	
	Other Aircraft	
	Radar	
	Ship	
	Space	
	Targets/Drones	
	UAV (Unmanned Aerial Vehicle)	

Table 2: Breakdown of services, commodities, and contract types in the final analyzed database.

Results

From the 1,216 contracts in our final database, we first analyze the development ones. Table 3 reflects the summary statistics for these 448 contracts. With a *p*-value of less than 0.0001 for the accompanying *t*-test, empirical evidence suggests that DEV_{ROT} based on the mean value of 16.3% is statistically greater than 10%. But there is a high degree of variability present among the development contracts as reflected in both the standard deviation and IQR. This variability indicates that the median may also play a role in determining how much to estimate/reserve for ECO. A median value of 10.4% is very near the DEV_{ROT} value of 10% and is not statistically significant.

METRIC	DEVELOPMENT	PRODUCTION
Sample Size	448	768
Mean	0.163	0.088
Standard Deviation	0.279	0.286
Coefficient of Variation	1.711	3.253
Median	0.104	0.035
Interquartile Range (IQR)	0.265	0.141
IQR / Median	2.563	4.014

Table 3: Summary statistics for the 448 development and 768 production contracts. Numbers rounded to three decimal places.

Table 3 also reflects the summary statistics for the 768 production contracts. With a *p*-value of 0.0001 for the accompanying *t*-test, empirical evidence suggests that $PROD_{ROT}$ based on the mean value of 8.8% is statistically greater than 5%. Like the development contracts, production contracts also display a high degree of variability with respect to ECO percentage as reflected in the standard deviation and IQR.

DEV_{ROT} and $PROD_{ROT}$ are statistically different from one another with a *p*-value of less than 0.0001 for both the *t*-test and Wilcoxon Rank Sum test. This implies that development and production contracts statistically differ with respect to ECO percentages, with $DEV_{ROT} > PROD_{ROT}$. We present these results in separate subsections.

Development

Of the 448 development contracts, one contract had no ECO cost, 59 had negative ECO cost, while the remaining 388 (86.6%) had positive ECO cost. As stated previously in the Methods section, we conducted three Kruskal-Wallis (K-W) tests to determine any statistical difference among the services, commodity, and contract type with respect to ECO percentage. No statistical difference existed among services (*p*-value of 0.5387) or commodity type (p-value of .1022). For contract type, 65 contracts were identified as unknown (missing). Of the remaining 383, 251 (65.5%) were identified as cost, 85 as fixed, and 47 as T&M. A statistical difference appeared between cost and fixed (K-W *p*-value of 0.0385, with a subsequent *p*-value of 0.0277 for the Steel-Dwass (S-D) multiple comparisons). Table 4 highlights the metrics associated with cost, fixed, and T&M contracts, respectively. Note: both the mean ECO percentages for cost and T&M contracts were statistically (*p*-value < 0.0001) greater than DEV_{ROT} of 10%, while fixed type contracts were equivalent.

From studies noted in the Introduction section, we next address if dollar threshold and/or contract schedule length is associated with whether a contract is likely to exceed the DEV_{ROT} of 10%. The natural breaks are those set for DoD's classification. ACAT I is associated with Research, Development, Test and Evaluation (RDT&E) costs over $525M (in FY 2020 dollars); ACAT II with RDT&E programs between $200M and $525M (FY 2020); and ACAT III for all program less than $200M. The *p*-value for the Pearson's chi-squared test of independence was 0.4709. This finding suggests dollar thresholds associated with ACAT categories do not appear to affect the likelihood of exceeding the DEV_{ROT} of 10%.

Regarding contract length, only 346 (77.2%) of the contracts had complete schedule data (neither contract award date nor PoP end date

METRIC	COST	FIXED	T&M
Sample Size	251	85	47
Mean	0.172	0.102	0.211
Standard Deviation	0.264	0.271	0.399
Coefficient of Variation	1.531	2.654	1.89
Median	0.114	0.057	0.109
Interquartile Range (IQR)	0.269	0.215	0.421
IQR / Median	2.36	3.772	3.862

Table 4: Summary statistics for the 251 cost, 85 fixed, and 47 T&M development contracts. Numbers rounded to three decimal places.

were missing). We tested if contract length equal to or greater than five years had an increased chance of exceeding the DEV_{ROT} of 10%. This threshold is based on Trudelle et al., 2017a which showed that program length of five years or more appeared to be a statistically significant indicator of cost growth. The Pearson's chi-squared dependency test returned a *p*-value of 0.0007. The associated odds ratio of 3.99 (with an associated 95% confident interval of (1.71, 9.31)) suggests that development contracts that equal or exceed five years in length are four times more likely to exceed an ECO percentage of 10%.

Prior to preceding into a comparable analysis by narrowing to contracts with just net positive ECO cost (hereafter referred to as positive ECO cost), we investigate what factors (service, commodity, contract type) might indicate that a development contract is likely to experience positive ECO cost. The *p*-value for testing dependency between service and positive ECO cost was 0.9611, strongly suggesting no dependency whatsoever. Consistent to the previous finding regarding all ECO costs, development contracts, irrespective of commodity type, had a comparable chance of experiencing positive ECO with one exception, ground vehicles.

The ground vehicles in our database consisted of the Joint Light Tactical Vehicle (primarily Army/ Marine), the Joint Mine Resistant Ambush Protected (Army/Marine), the Logistic Vehicle System Replacement (Marine equivalent to the Army's Heavy Expanded Mobility Tactical Truck), and the Medium Tactical Vehicle Replacement (Marine equivalent of the US Army's Family of Medium Tactical Vehicles). These types of vehicles are approximately 3.81 times less likely to experience a positive ECO cost. The associated *p*-value for this Pearson chi-squared test was 0.0037 with a 95% confidence interval for the odds ratio of (1.45, 9.98). Table 5 contains the descriptive statistics for ground vehicles. As previously noted, high variability is present as shown by the standard deviation and IQR values.

METRIC	VALUE
Mean	0.125
Standard Deviation	0.389
Coefficient of Variation	3.112
Median	0.044
Interquartile Range (IQR)	0.559
IQR / Median	12.704

Table 5: Summary statistics for the 20 ground vehicle development contracts. Numbers rounded to three decimal places.

Regarding contract type, cost contracts appear to have a higher likelihood of experiencing a positive ECO cost compared to fixed and T&M. The *p*-value for the Pearson's chi-squared test was 0.0137 with an associated odds ratio of 2.05 and a 95% confidence interval of (1.15, 3.65). Overall, it appears that a cost development contract is more likely to experience positive ECO cost, while a ground vehicle like those in our database is less likely to experience positive ECO cost.

Next, we narrow to just those contracts with positive ECO cost, which represent 86.6% of all the development contracts in our database. Table 6 contains the descriptive measures of just the positive ECO contracts. As expected, both the mean and median are higher than those shown in Table 3. In addition, both the CV and IQR/Median are lower.

METRIC	VALUE
Mean	0.222
Standard Deviation	0.233
Coefficient of Variation	1.05
Median	0.14
Interquartile Range (IQR)	0.285
IQR / Median	2.045

Table 6: Summary statistics for the 388 development contracts with positive ECO cost. Numbers rounded to three decimal places.

Among just contracts that experienced positive ECO cost, neither service (K-W *p*-value of 0.2882) nor commodity (K-W *p*-value of 0.1371) appeared significant at the 0.05 level but contract type (K-W *p*-value of 0.0468) did. Given the *p*-value was very close to the 0.05 level, the subsequent S-D multiple comparisons lacked the statistical power to meet this significance threshold. S-D only indicated that fixed and T&M contracts might be different with a *p*-value of 0.0590. Table 7 highlights the metrics associated with cost, fixed, and T&M contracts for just those with positive ECO cost, respectively. Sixty contracts had missing information for contract type. Note: all contract types possessed a mean percentage statistically greater than DEV_{ROT} of 10% (*p*-value < 0.0001).

METRIC	COST	FIXED	T&M
Sample Size	223	68	37
Mean	0.224	0.182	0.337
Standard Deviation	0.218	0.208	0.326
Coefficient of Variation	0.973	1.138	0.967
Median	0.151	0.128	0.204
Interquartile Range (IQR)	0.286	0.227	0.617
IQR / Median	1.894	1.778	3.022

Table 7: Summary statistics for the ECO positive development contracts. Numbers rounded to three decimal places.

Next, we address if dollar threshold and/or contract schedule length might affect the chance a contract is likely to exceed the DEV_{ROT} of 10%. Using the same ACAT dollar thresholds as before, the *p*-value for the Pearson's chi-squared test of independence was 0.1382. This finding suggests dollar thresholds associated with ACAT categories may not affect the likelihood of exceeding the DEV_{ROT} of 10% when just examining contracts with positive ECO cost.

We did conduct *post hoc* analysis given this drop of *p*-value from 0.4709 (from the previous ACAT analysis) to 0.1382 with respect to investigating dollar threshold and exceeding DEV_{ROT} of 10%. We varied the dollar threshold incrementally between $10M to $100M and observed the spot whereby both the *p*-values and odd ratios change the most statistically. As shown in Table 8, that dollar amount appears to be around $30M. Table 9 highlights the metrics associated with contracts less than $30M and those equal to or greater than that value.

DOLLAR AMOUNT	ODDS RATIO	P-VALUE
<$10M	1.89	0.003
<$20M	1.94	0.0016
<$30M	2.17	0.0002
<$40M	1.75	0.0077
<$50M	1.62	0.0234
<$100M	1.57	0.0531

Table 8: Odds ratios and p-values for varying baseline contract amount from $10M to $100M in 2020 FY for ECO positive cost contracts.

METRIC	<$30M	$30M≥
Mean	0.268	0.17
Standard Deviation	0.255	0.194
Coefficient of Variation	0.949	1.144
Median	0.177	0.094
Interquartile Range (IQR)	0.318	0.216
IQR / Median	1.794	2.295

Table 9: Summary statistics for the ECO positive cost contracts less than $30M (in FY 2020 dollars) and those equal to or greater than $30M. Numbers rounded to three decimal places.

Regarding contract length, only 319 (82.2%) of the ECO positive contracts had complete schedule data (neither contract award date nor PoP end date were missing). We again tested if contract length equal to or greater than five years had an increased chance of exceeding the DEV_{ROT} of 10%. The Pearson's chi-squared test returned a *p*-value of 0.0009, suggesting dependency between them.

The odds ratio of 4.60 (with an associated 95% confident interval of (1.74, 12.17)) suggests that ECO positive development contracts that equal or exceed five years in length are more likely to exceed an ECO percentage of 10% than shorter contracts.

In summary, for development contracts it appears there is a high likelihood (86.6%) that these experience positive ECO cost. The median and mean ECO percentages are 14% and 22% respectively. Both are statistically greater than the DEV_{ROT} of 10%. Ground vehicles are less likely to experience positive ECO cost, while cost type contracts are more likely to experience positive ECO cost. Lastly, development contracts with a PoP equal to or exceeding five years and contracts less than $30M in FY 2020 dollars are likely to experience an ECO percentage greater than 10%. In the Discussion and Conclusion section, we quantify our ECO recommendations through a table utilizing the median values presented in this section due to the high variability reflected by the coefficient of variations and ratios of IQR/Median. We now turn to the results associated with the production contracts.

Production

For this subsection, we replicate the analysis and flow that we performed previously for development contracts but just for production contracts. Of the 768 production contracts, 148 had negative ECO cost, while the remaining 620 had positive ECO cost. These 620 contracts equate to approximately 80.7% of the production contracts. Such a relatively high percentage suggests that the analysis should also analyze the characteristics of just the positive ECO cost contracts to provide additional insight. Prior to this conditional analysis, we investigate inferential patterns among all the production contracts.

Testing ECO percentage differences among the services, the initial K-W produced a *p*-value of 0.0337, with Navy contracts being statistically lower than Air Force contracts (S-D *p*-value of 0.0276). However, it should be noted that a fair number of F/A-18 contracts (13 out of 71) had -100% ECO [Note: programs can have multiple contracts, thereby allowing this number to be plausible.], which drove the mean percentage of the F/A-18 to -7.2%. No other commodity had a negative ECO %. When excluding those 13 contracts, the K-W's *p*-value rose significantly to 0.1431, reflecting a truer picture among the services, suggesting that there appears to be no statistical difference among the services with respect to ECO %. Moving forward with any remaining inferential analysis in this part, we continue to exclude these 13 contracts.

For any K-W test, it is advisable to exclude any comparison group that has five or fewer observations because of low power issues (Kruskal & Wallis, 1952; Howell, 2002). Consequently, we removed the commodity groups Ship (3 contracts: DDG 51), Gun (2 contracts: CIWS and LW155), and AIS (2 contracts: GCSS-MC and DCGS-N). Therefore, we cannot make any inferential decisions regarding these commodities.

For the remaining commodities, we conducted a K-W test and found a statistical difference among the remaining commodities (*p*-value of 0.0160). The S-D multiple comparisons determined that Ground Vehicle and Other aircraft were statistically different from the other commodities (we separated the F-16 and F/A-18 from the Other Aircraft due to their relatively large number of contracts; Ellis et al. (2018) performed a similar measure for the F/A-18). The S-D multiple comparisons displayed a *p*-value of 0.0394 and noted that Ground Vehicle generally has lower ECO % compared to Other aircraft.

Investigating differences between contract types, we first needed to exclude production contracts labeled as unknown or missing. This number

totaled 95. For the remaining, we had 75 cost, 569 fixed, and 16 T&M contracts. Unlike development contracts that primarily consisted of cost type contracts, most of the production contracts were fixed contracts (86.2%). When inferentially comparing among cost, fixed, and T&M contracts, the only statistical difference appeared between cost and fixed (K-W *p*-value of 0.0003, with a subsequent *p*-value of 0.0002 for the S-D for multiple comparisons). We saw similar results for development contracts. Table 10 highlights the metrics associated with cost, fixed, and T&M production contracts, respectively.

METRIC	COST	FIXED	T&M
Sample Size	75	569	16
Mean	0.236	0.094	0.091
Standard Deviation	0.306	0.245	0.431
Coefficient of Variation	1.295	2.614	4.726
Median	0.102	0.031	0.104
Interquartile Range (IQR)	0.347	0.139	0.565
IQR / Median	3.402	4.484	5.461

Table 10: Summary statistics for the 75 cost, 569 fixed, and 16 T&M production contracts. Numbers rounded to three decimal places.

As with development contracts, we next analyze if dollar threshold and/or contract schedule length might affect if a contract is likely to exceed the $PROD_{ROT}$ of 5%. We use dollar thresholds again with respect to ACAT level but adjust accordingly for production. Specifically, ACAT I is associated with Procurement costs greater than $3.065B (in FY 2020 dollars); ACAT II with Procurement dollar amounts less than ACAT I but greater than $920M (FY 2020); and ACAT III for anything less. The *p*-value for the Pearson's chi-squared test of independence was 0.7181. This finding suggests dollar thresholds associated with ACAT categories do not appear to affect the likelihood of exceeding the $PROD_{ROT}$ of 5%.

Regarding contract length, 624 (82.6%) of the contracts had complete schedule data (neither contract award date nor PoP end date were missing). We tested if contract length equal to or greater than five years (due to Trudelle et al. (2017a) findings) had an increased chance of exceeding the $PROD_{ROT}$ of 5%. The Pearson's chi-squared test returned a *p*-value of 0.0510, which just barely misses our level of significance of 0.05, suggesting perhaps borderline dependency at best that production contracts that equal or exceed five years in length are slightly more likely to exceed an ECO percentage of 5%.

Prior to proceeding into a comparable analysis by narrowing to production contracts with just a positive ECO cost, we investigate what factors might indicate a production contract is likely to experience positive ECO cost. The *p*-value for testing dependency between service and positive ECO cost was 0.0026, strongly suggesting dependency. Further investigation reveals that Air Force production contracts are statistically more likely than Navy (which includes Marines) and Army to experience positive ECO cost. The associated *p*-value for this Pearson's chi-squared test is 0.0002 with an odds ratio of 2.12 with a 95% confidence interval of (1.42, 3.18).

Because a strong dependency appears between service and likelihood of incurring positive ECO cost, we need to perform conditional analysis on service before comparing among commodities. This conditional analysis is required because there is a natural dependency between service and commodity. Due to small sample size for some commodities, we excluded AIS, Gun, and Ship contracts. When comparing among Air Force commodities, UAV and F-16 production contracts are less likely to experience positive ECO cost compared to the other commodities of Decoys, Electronics, Engines, Missiles, Ordnance, Other Aircraft, Space, and Targets/Drones. The *p*-value for this Pearson's chi-squared test was less than 0.0001 with an odds ratio of 5.71 and a 95% confidence interval of (2.84, 11.50).

For just Navy, we compare only the commodities F/A-18, Missiles, Other Aircraft, and Ground Vehicle contracts because of small sample issues (having five or fewer contracts) for the other commodities of Decoys, Engine, Space, Target/ Drones, and UAV. The commodity that is the least likely to experience positive ECO cost are Ground Vehicle production contracts compared to the other three commodities tested (comparable to what we found with development contracts). The *p*-value is less than 0.0001 with an odds ratio of 3.30 and a 95% confidence interval of (1.99, 5.47). The next commodity, the F/A-18, is less likely to experience positive ECO cost compared to Missiles and Other Aircraft. Its *p*-value is 0.0078 with an odds ratio of 2.93 and a 95% confidence interval of (1.30, 6.63). For Army only commodities, 0.3329 was the resultant *p*-value suggesting no commodity is more likely to experience positive ECO cost than another.

Regarding contract type, the *p*-value associated with testing the assumption if contract type (cost, fixed, and T&M) affects the likelihood of experiencing a positive ECO cost was 0.0762, suggesting a weak association since it isn't less than 0.05. Tentatively, it appears T&M is the least likely to experience positive ECO cost with cost type contracts being the most likely.

Next, we narrow to just those production contracts with positive ECO contracts, which represent 80.7% of all the production contracts in our database. Table 11 contains the descriptive measures of just the positive ECO contracts. As expected, both the mean and median are higher than those shown in Table 2. In addition, both the CV and IQR/Median are lower.

METRIC	VALUE
Mean	0.157
Standard Deviation	0.227
Coefficient of Variation	1.446
Median	0.055
Interquartile Range (IQR)	0.18
IQR / Median	3.264

Table 11: Summary statistics for the 620 production contracts with positive ECO cost. Numbers rounded to three decimal places.

Among just production contracts that experienced positive ECO cost, neither service (K-W *p*-value of 0.7242) nor commodity (K-W *p*-value of 0.3347 after excluding commodities with five or less contracts) appeared significant at the 0.05 level but contract type (K-W *p*-value of less 0.0001) strongly did (after excluding 88 contracts listed as unknown or missing information). The subsequent S-D analyses suggested strong statistical differences between cost and fixed type contracts (*p*-value of 0.0002) and between T&M and fixed contracts (*p*-value of 0.0055). Table 12 highlights the metrics associated with cost, fixed, and T&M production contracts for just those with positive ECO cost, respectively.

METRIC	COST	FIXED	T&M
Sample Size	65	457	10
Mean	0.281	0.148	0.34
Standard Deviation	0.305	0.216	0.282
Coefficient of Variation	1.085	1.458	0.831
Median	0.144	0.054	0.283
Interquartile Range (IQR)	0.399	0.177	0.398
IQR / Median	2.771	3.278	1.407

Table 12: Summary statistics for the ECO positive 65 cost, 457 fixed, and 10 T&M production contracts. Numbers rounded to three decimal places.

Next, we investigate if dollar threshold and/or contract schedule length might affect the chance that a positive cost production contract is likely to exceed the $PROD_{ROT}$ of 5%. Using the same ACAT dollar thresholds as before, the *p*-value for the Pearson's chi-squared test of independence was 0.8821. This finding suggests dollar thresholds associated with ACAT categories do not appear to affect the likelihood of exceeding the $PROD_{ROT}$ of 5% for just contracts with positive ECO cost.

Regarding contract length, only 550 (88.7%) of the ECO positive contracts had complete schedule data (neither contract award date nor PoP end date were missing). We again tested if contract length equal to or greater than 5 years had an increased chance of exceeding the $PROD_{ROT}$ of 5%. The Pearson's chi-squared test returned a *p*-value of 0.2817, suggesting lack of dependency between them.

In summary for production contracts, it appears there is a high likelihood (80.7%) that these experience positive ECO cost. Given that occurs, the median and mean ECO percentages are 5.5% and 15.7% respectively. Although the median is relatively close to the $PROD_{ROT}$ of 5%, the mean is much greater than 5%. Ground Vehicle production contracts experience less ECO % compared to Other Aircraft but no single commodity experienced uniformly lower ECO %. Cost production contracts statistically exceed fixed contracts with respect to ECO %, while borderline significance suggests contracts with schedule lengths of five or more years might exceed $PROD_{ROT}$ of 5%.

Regarding the likelihood of experiencing positive ECO cost, Air Force contracts have a higher chance than Navy. Among Air Force contracts, UAV and F-16 production contracts have lower positive ECO % chance compared to other commodities. Among Navy contracts (to include Marine), the least positive ECO % are Ground Vehicle commodities, followed by contracts for the F/A-18. The remaining commodities were comparable. There appeared no statistical difference among commodity with respect to Army production contracts. Lastly, for just positive ECO % production contracts, fixed contracts were statistically lower than both cost and T&M contracts. In the Discussion and Conclusion section, we quantify our ECO recommendations through a table utilizing the median values presented in this section because of the high variability reflected by the coefficient of variations and ratios of IQR/Median.

Discussion and Conclusion

Three key conclusions can be drawn from our findings. One, if a program manager wishes to use a rule-of-thumb, life-cycle phase matters. No single rule should be applied, and the traditional one is likely inappropriate regardless. Two, it appears that the variables of Service, Contract Type, Commodity, Program Size, and Schedule all have some degree of influence on the appropriate percentage to hold in reserve in case of ECO occurrence. Three, there are factors which correspond to increased likelihood of a contract incurring a positive ECO percentage, and those percentages will differ depending on those factors.

Table 13 summarizes the overall descriptive results for both the development and production contracts in our database. The mean ECO percentages for these contracts are compared to either DEV_{ROT} or $PROD_{ROT}$. Statistically, mean values are higher than both ROTs at the level of significance of 0.05 with *p*-values < 0.0001. The

LIFE CYCLE PHASE	CURRENT ROT	MEAN ECO% - ALL	MEDIAN ECO% - ALL	MEAN ECO% - ONLY POSITIVE VALUES	MEDIAN ECO % - ONLY POSITIVE VALUES	PERCENT OF CONTRACTS WITH POSITIVE ECO COST
Development	10%	16.30%	10.40%	22.20%	14.00%	86.60%
Production	5%	8.80%	3.50%	15.70%	5.50%	80.70%

Table 13: Summary statistics for the development and production contracts with ECO cost. Numbers rounded to three decimal places.

differences between the means and medians for both the development and production contracts indicate the relatively high variability associated with ECOs. This is supported by the earlier results with respect to relatively large standard deviations, IQRs, CVs, and ratios of IQRs to medians.

Given our overall findings, we suggest that if a ROT is to be used for ECO, a four-tiered approach should be taken. First, the life cycle phase of the contract should be considered. Second, characteristics of the contract should be reviewed to determine whether there is an increased likelihood of incurring a positive ECO percentage. Third, a baseline ROT percentage should be chosen as a starting point; we advocate the median percentage of the positive ECO contracts as an initial value. Lastly, characteristics of the contract that our analyses considered strongly statistically significant should be reviewed to determine whether to adjust this baseline ROT estimate upward or downward. If pressed to provide one single ECO percentage for each life cycle phase, we recommend revising DEV_{ROT} from 10% to 14% and $PROD_{ROT}$ from 5% to 6%.

With respect to tailoring suggested ROT % based on known program factors, we make the following recommendations with these caveats. One, we modified our level of significance threshold to 0.01 to minimize the chance of perhaps a spurious statistical finding affecting our conclusions and recommendations. Two, we use medians to arrive at these percentages in lieu of means to minimize the effect of outliers. Lastly, if a contract contains two or more significant factors that cause the new baseline ROT to change, then we recommend taking the higher adjustment among the significant factors to arrive at a single percentage recommendation. Tables 14 and 15 provide our suggested recommendations with respect to development and production contracts, respectively. Note: for

FACTOR	ADJUSTMENT	FINAL ECO/ MR%
Commodity = Ground Vehicle	-9%	5%
Baseline contract <$30M (FY 2020)	4%	18%
Contract schedule >= 5 years	13%	27%

Table 14: Suggested ECO percentages based on factors and adjustments from the new DEV_{ROT} of 14%.

FACTOR	ADJUSTMENT	FINAL ECO/MR%
Army or Navy contract	-2%	4%
Cost type contract	9%	15%
T&M type contract	23%	29%

Table 15: Suggested ECO percentages based on factors and adjustments from the new $PROD_{ROT}$ of 6%

any fractional percentages, we do round up to the nearest percentage for ease of convenience plus allowing for the realization that mean ECO % were always larger than median ECO %.

We suggest that the ECO percentage estimates from Tables 14 and 15 should be used as an initial point estimate but should not be treated as an exact estimate. The means in all instances exceeded median estimates, and there is a great deal of variability associated with ECO costs. Cost estimators should use prior knowledge and other tools at their disposal to deviate from this point estimate when necessary. We again acknowledge that there is no one-size-fits-all ROT that should be used to estimate appropriate amounts to hold in MR in case of ECO. However, it does statistically appear that the original DEV_{ROT} and $PROD_{ROT}$ of 10% and 5% are generally lower than what we witnessed in our database.

References

Arena, M. V., Leonard, R. S., Murray, S. E., & Younossi, O. (2006). *Historical cost growth of completed weapon system programs* (Report No. TR-343). Santa Monica, CA: RAND.

Bolten, J. G., Leonard, R. S., Arena, M. V., Younossi, O., & Sollinger, J. M. (2008). *Sources of weapon system cost growth: Analysis of 35 major defense acquisition programs* (Report No. MG-670). Santa Monica, CA: RAND.

Christensen, D., & Templin, C. (2000). An analysis of management reserve budget on defense acquisition contracts. *Acquisition Review Quarterly*, *7*(3), 191-208.

D'Amico, C. N., White, E. D., Ritschel, J. D., & Kozlak, S. J. (2017). Unmasking cost growth behavior: A longitudinal study. *Defense Acquisition Research Journal*, *25*(1), 30–51. https://doi.org/10.22594/dau.17-783.25.01

Electronic Document Access. (2017), In AcqNotes [Online encyclopedia]. Retrieved from http://acqnotes.com/acqnote/careerfields/electronic-document-access-eda

Ellis, J.C., White, E., Ritschel, J.D., Valentine, S.M., Lucas, B., & Cordell, I.S. (2018). Likelihood and cost impact of engineering change requirements for DoD contracts. *Journal of Defense Analytics and Logistics*, *2*(1), 22-37. https://doi.org/10.1108/JDAL-02-2018-0002

Engineering Change Proposals. (2021). In Acquipedia [Online encyclopedia]. Retrieved from https://www.dau.edu/acquipedia/pages/articledetails.aspx#!329

Government Accountability Office. (2008). *Defense acquisitions: Assessments of selected weapon programs* (08-467SP). Retrieved from https://www.gao.gov/products/gao-08-467sp

Gibson, J. (1983). *The ASD ECO model users guide*. Wright-Patterson Air Force Base, OH: Aeronautical Systems Division.

Harmon, B. R., & Arnold, S. A. (2013). *Choice of contract type and other policy initiatives for reducing contract prices* (D-5002). Retrieved from https://apps.dtic.mil/sti/pdfs/ADA594093.pdf

Howell, D.C. (2002). *Statistical Methods for Psychology* (5th ed.). Pacific Grove, CA: Duxbury.

Kozlak, S. J., White, E. D., Ritschel, J. D. Lucas, B., & Seibel, M. J. (2017). Analyzing cost growth at program stages for DoD aircraft. *Defense Acquisition Research Journal*, *24*(3), 386–407. https://doi.org/10.22594/dau.16-763.24.03

Kruskal, W.H. & Wallis, W.A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, *47*, 583-621.

Trudelle, R., White, E. D., Koschnick, C., Ritschel, J. D., & Lucas, B. (2017). Estimating an acquisition program's likelihood of staying within cost and schedule bounds. *Defense Acquisition Research Journal*, *24*(4), 600-625. https://doi.org/10.22594/dau.17-775.24.04

Trudelle, R., White, E.D., Ritschel, D., Koschnick, C., & Lucas, B. (2017). Modeling median will-cost estimates for defense acquisition programs. *Journal of Defense Analytics and Logistics*, *1*(1), 19-33. https://doi.org/10.1108/JDAL-04-2017-0004

Appendix

3DELRR (Three-Dimensional Expeditionary Long -Range Radar)

ADM-141C (ITALD: Improved Tactical Air Launched Decoy)

ADM-160 (Miniature Air-Launched Decoy)

ADS (Active Denial System)

AEHF (Advanced Extremely High Frequency Satellite)

AGM-65 (Maverick)

AGM-84; RGM-84; UGM-84 (Harpoon SLAM-ER: Standoff Land Attack Missile-Expanded Response)

AGM-86A/B/C/D (ALCM: Air-Launched Cruise Missile)

AGM-88E (AARGM: Advanced Anti-Radiation Guided Missile)

AGM-142 (Have Nap)

AGM-154C (JSOW (Unitary): Joint Stand-Off Weapon Baseline Variant and Unitary Warhead Variant)

AGM-158 (JASSM/JASSM-ER: Joint Air-to-Surface Standoff Missile)

AGM-169 (JCM: Joint Common Missile)

AHLTA (Armed Forces Health Longitudinal Technology Application)

AIM-7; RIM-7 (Sparrow; Sea Sparrow)

AIM-9 (AIM-9X: Air-to-Air Missile Upgrade)

AIM-120 AMRAAM (Advanced Medium Range Air -to-Air Missile)

AQM-37 (Target Drone)

ASIP (Advanced Special Improvement Program models of Single Channel Ground and Airborne Radio System (SINCGARS))

AWACS (Airborne Warning and Control System)

AWS (AEGIS - MK 7 Advanced Shipboard Weapon System)

B-1 (Lancer)

B-2 (Spirit)

B-52 CONECT (B-52 Stratofortress Combat Network Communications Technology)

B-52H (Stratofortress)

B-61 Tail Kit (B61 Mod 12 Life Extension Program Tail Kit Assembly)

BGM-109 (Gryphon (Ground-Launched Cruise Missile))

BGM-178 (RATTLRS: Revolutionary Approach to Time-critical Long-Range Strike)

BQM-34 (Firebee)

BQM-74 (Chukar)

BQM-167 (Skeeter)

BTERM (Ballistic Trajectory Extended Range Munition)

C-5 (Galaxy)

C-17 (Globemaster III)

C-37A (Gulfstream V)

C-130 (Hercules)

CBU-97 (Sensor Fused Weapon (SFW))

CBU-105 (Sensor Fuzed Weapon)

CH-47 (Chinook)

CHAMP (Counter-electronics High Power Microwave Advanced Missile Project)

CIWS (Close in Weapons System)

CV-22 (Air Force variant Osprey)

DCAPES (Deliberate Crisis Action Planning and Execution Segments INC 2B)

DCGS Navy (Distributed Common Ground System Navy)

DDG 51 (Arleigh Burke Class Guided Missile Destroyer)

DEAMS (Defense Enterprise Accounting Management System)

E-2D (Advanced Hawkeye)

EA-18G (Growler)

EC-130H (Compass Call)

EELV (Evolved Expendable Launch Vehicle)

EPS (Enhanced Polar System)

EX-171 (ERM - Extended Range Munition)

F-15 (Eagle)

F-15 AN/ALQ-135 (Electronic Countermeasure)

F-15 ATP (Advanced Targeting Pod)

F-16 (Fighting Falcon)

F-22 (Raptor)

F-35 (Lightning II)

F-119 (F-22 Engine)

F-135 (F-35 Engine)

F-136 (F-35 Engine)

F/A-18 (Hornet)

FAB-T (Family of Beyond Line-of-Sight Terminals)

FMTV (Family of Medium Tactical Vehicles)

GBU-12 (Paveway II)

GBU-15 (Guided Bomb Unit 15)

GBU-24 (Paveway III)

GBU-39 (SDB I: Small Diameter Bomb Increment I)

GBU-53/B (SDB II: Small Diameter Bomb, Increment II)

GCSS-MC (Global Combat Support Systems - Marine Corps)

GPS III (Global Positioning System III)

GPS OCX (Global Positioning System Next Generation Operational Control System)

GQM-163 (Coyote)

GQM-173 (Multi-Stage Supersonic Target)

H-1 (Upgrade program)

HC/MC 130 (Recapitalization Aircraft)

HH-60 (Pave Hawk)

HMMWV (High Mobility Multi-Purpose Wheeled Vehicle)

IDECM (Integrated Defensive Electronic Countermeasures)

JAGM (Joint Air-to-Ground Missile)

JDAM (Joint Direct Attack Munition)

JLTV (Joint Lightweight Tactical Vehicle)

JPALS (JPALS - Joint Precision Approach and Landing System)

KC-46A (Pegasus)

LAIRCM (Department of the Navy Large Aircraft Infrared Countermeasure)

LRASM (Long Range Anti-Ship Missile)

LVSR (Logistics Vehicle System Replacement)

LW155 (Light Weight Howitzer 155 mm)

MC-130J (Commando II)

MGM-140 (ATACMS: Army Tactical Missile System)

MH-60R (Seahawk)

MH-139 (Grey Wolf)

MHS (Military Health System)

MIDS-LVT (Multi-Functional Information Distribution System - Low Volume Terminal (includes JTRS: Joint Tactical Radio System Terminals))

MIM-104A/B/C/D (Patriot)

MIM-104F (PAC-3: Patriot Advanced Capability 3)

MQ-1B (Predator)

MQ-4C (Triton)

MQ-9 (Reaper)

MRAP (Joint MRAP: Joint Mine Resistant Ambush Protected Vehicles)

MTVR (Medium Tactical Vehicle Replacement)

MUOS (Mobile User Objective System)

NAVSTAR GPS (Global Positioning System)

P-8A (Poseidon)

PIM (Paladin Integrated Management)

QF-4 (FSAT: Full Scale Aerial Target)

RIM-66 (Standard Missile 1 (SM-1MR))

RIM-116 (RAM BLK 2)

RIM-116A (RAM BLK 0)

RIM-116B (RAM BLK 1)

RIM-161 (SM-3: Standard Missile 3)

RIM-162 (ESSM: Evolved Sea Sparrow Missile)

RIM-174 (SM-6: Standard Missile-6)

RQ-4 (Global Hawk)

RUR-5 ASROC (Anti-Submarine Rocket (VLA: Vertical Launch))

SBIRS (Space-Based Infrared System)

SBSS B10 (Space-Based Space Surveillance Block 10)

SH-60/HH-60H/MH-60 (Sikorsky Seahawk)

SL-AMRAAM (Surface Launched - Advanced Medium-Range Air-to-Air Missile)

Space Fence (Space Fence Inc 1)

UH-60 (Black Hawk)

V-22 (Navy Osprey)

WCMD (Wind Corrected Munitions Dispenser)

WGS (Wideband Global SATCOM Program)

Weather Satellite Follow-on (WSF)

---

*Capt **Kaiana M. Miller** is a Cost Analyst in the Technology, Command and Control, and Special Programs Division of the Air Force Cost Analyst Agency at Joint Base Andrews. He received his BS in Management from The United States Air Force Academy and his MS in Cost Analysis from AFIT. Capt Miller's research interests include statistical modeling, predictive analytics, and social policy. (E-mail address: Kaiana.Miller.1@us.af.mil)*

*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)*

*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)*

*Dr. **Robert D. Fass** is an Assistant Professor in the Department of Systems Engineering and Management at AFIT. He holds a BA in Economics, an MBA from the University of New Mexico, and a Ph.D. in Business Administration and Management from New Mexico State University. Dr. Fass' research interests include cost analysis, decision analysis, risk analysis, behavioral economics, organizational behavior, organizational change, and government acquisition policy. (E-mail address: Robert.Fass@afit.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)*

Foundation of Structured Architecture, System & Cost Modeling

Danny Polidi
Mike Crist
Dr. V. Chandrasekar

This document does not contain technology or Technical Data controlled under either the U.S. international Traffic in Arms Regulations or the U.S. Export Administration Regulations

Abstract: Modern software packages exist to estimate system cost early in the system development and procurement process. This paper begins the development of a structured systems engineering approach to system design. This paper defines a standardized modular diagram for a RADAR system applied to military applications in the aerospace industry. This modular diagram with sub-system block elements will be used to create a system model. The standardized modular diagram will also be used to create a cost model, using the same modular sub-system block elements and industry standard historical cost data. The commercially available software packages which estimate system cost are limited in their ability to aid in system optimization towards multi-objective cost and performance goals, as many require a completed system design. Methods are needed to determine which components in a system would benefit from additional modeling such as using a multiphysics approach, and which design approach provides the best value (cost vs. performance) to the system. These methods are needed during concept development to aid in system scoping and cost estimation. To illustrate the benefits of cost optimization during early stages of design, this paper describes a sensitivity analysis approach applied to the design of an engineering system. This process seeks to use sensitivity analysis and a spiral design process to determine which cost drivers have the highest influence on overall system cost, and to realize high system performance while minimizing costs.

This work demonstrates that a system can be defined as a standard set of block diagrams for an airborne RADAR for military applications created by integrating a wide sample of the available examples. And where each of the example block diagrams could be considered a subset of the more generalized form. This work describes using the generalized block diagrams to create a WBS structure as the foundation for both a system model and a cost model. This work applies a sensitivity analysis to a cost model in order to direct a system designer towards a trade study for the purposes of system optimization. And finally, this work introduces a method using component cost sensitivity to determine the range of possible cost improvements to bound project return on investment.

I. Introduction

There are several very good commercially available cost estimation packages. To use these packages, first a system must be defined. The system must be defined in terms of hardware blocks. The hardware blocks can be arranged with a hierarchy such as a Work Breakdown Structure (WBS). Once the system is defined, the system can be entered into the cost estimation package. The package essentially converts each hardware component into a corresponding cost. In this way the cost of a system can be estimated.

In order to analyze a system, it is necessary to have a system upon which to perform the analysis. Typically, for a new effort a system is defined from the perspective of the designer

where certain features were a priority to the respective designer. Those priorities are reflected in the various block diagrams which are produced and can be seen as areas of increased fidelity while other areas of the block diagram are simplified or even combined with other functions into one sub-block. For the current example of an airborne RADAR for military applications there are many example block diagrams within the existing literature. However, although the end application is the same, the various block diagram examples vary widely. It can be considered that the block diagrams were tailored for each application and demonstrate the priority of the respective designer. This paper demonstrates the development of a generalized set of block diagrams for an airborne RADAR for military applications. The block diagrams were created by integrating a wide sample of the available examples where each of the examples could be considered a subset of the more generalized form.

The focus of this paper will be divided into four main topics: block diagram development, systems engineering model development, systems cost model development, and sensitivity analysis concepts applied to a system cost model.

The first section, Block Diagrams, will discuss the available literature on the topic. Specifically, research into the existence of industry standard block diagrams for an airborne based RADAR for military applications and subsequently the development of one where a standard did not exist. The level one system RADAR block diagram is defined along with the level two sub-blocks: antenna, transmitter, synchronizer, receiver, etc. A solid block diagram is frequently the best way to begin a new design. It becomes a pivot point upon which everything else is developed. All major radio frequency (RF) interfaces and divisions of functions can be seen (digital control will not be addressed).

In the second section, Model The System, a structured approach to system engineering is started and the elements of the block diagram are described. The elements are ready to be loaded into a system engineering tool and form the basis of future work which would be expanded to include operational view diagrams, logical view diagrams and other system engineering artifacts.

In the third section, Model The Cost, a structured approach to system cost modelling is discussed and the format of the model is described. The cost model utilizes the same functional blocks as defined in the block diagram section. This forms the basis for future work which will eventually lead to a robust modular cost model to describe a range of RADARs and their associated estimated costs.

In the fourth section, Sensitivity Analysis Applied To A Cost Model, a concept is introduced whereby a sensitivity analysis could be applied to a cost model to direct a system designer towards a trade study for the purposes of system optimization. In addition, it is shown that a sensitivity analysis provides an upper bound of potential cost improvements which then forms the basis for a Return On Investment (ROI).

This work is novel in that it demonstrates that a system can be defined as a standard set of block diagrams for an airborne RADAR for military applications created by integrating a wide sample of the available examples. And where each of the example block diagrams could be considered a subset of the more generalized form. This work is novel in that it describes using the generalized block diagrams to create a WBS structure as the foundation for both a system model and a cost model. This work is novel in that it introduces a sensitivity analysis applied to a cost model in order to direct a system designer towards a trade study for the purposes of system optimization. And finally, this work is novel in that it introduces a method using component cost sensitivity to determine the range of possible cost improvements to bound project return on investment.

II. Related Work

A. Literature Assumptions and Search Terms.

The available literature was consulted, primarily through the use of Google searching. The assumption of the author was that a standardized block diagram for an airborne RADAR for military applications already exists. The assumption was that there was a standard upon which all designs were based. Research was done using keyword search terms such as "standard block diagram", "RADAR block diagram", "airborne RADAR block diagram", etc.

B. Literature Results.

For each effort, many search results were obtained. There were countless block diagrams for all types of RADARs. And, for the specific platform of airborne RADAR, again, there were many different results. However, although the results varied each version of a block diagram had some similarities. The differences appeared to be due to the focus of the respective system designers. In other words, if the designer's focus was upon a specific sub-function, that area of the block diagram had significantly more fidelity. Conversely, other areas of the block diagram would be abbreviated or even combined with other sub-functions. In this way the designer could highlight an area or sub-block for increased emphasis.

III. Block Diagrams

A robust block diagram is frequently the best way to begin a new design. And it is a recommended first step. It is not uncommon for engineers to jump right into a design and begin designing. Each engineer responsible for a portion of the system has ideas on how to best proceed. And frequently those best ideas are competing rather than complimenting one another. Therefore, a robust discussion early on regarding system goals is critical. Without a clear set of system goals, it is unlikely a system will be designed correctly on the first attempt. And a first step towards defining those goals is to create a block diagram. It becomes a pivot point upon which everything else is developed. All major interfaces and divisions of functions can be seen.

Not only do block diagrams align system and sub-system designers but also block diagrams are key tools for cost analysts. The most obvious area of contribution is defining interfaces. With a good visual representation, interfaces are considered, and meaningful requirements can be created. Those requirements affect many variables including cost, performance, life span, operation, etc. A cost analyst does not need to be a system designer. But having some familiarity with the basic building blocks of the system is critical. A good cost analyst should actively participate in the early generation of system block diagrams to help influence the direction of the system design.

A. Interfaces and Functions.

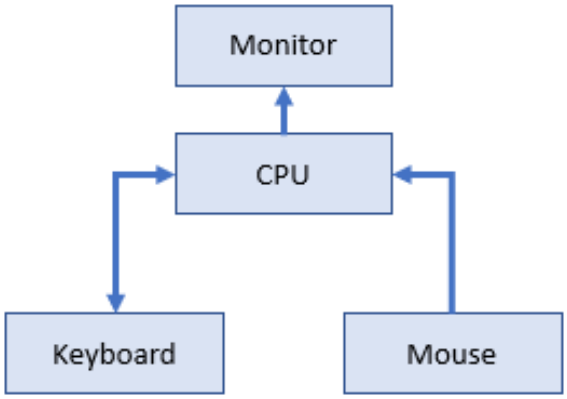


Figure 1. Computer Block Diagram.

Figure 1 shows a simple block diagram of a standard computer. In this case, the computer is made up of four sub-blocks. For purposes of this paper, the entire computer will be considered Level 1 while the sub-blocks indicated in the figure will be considered Level 2.

From this figure, the central processing unit (CPU) is the main sub-block in that it interfaces with all the other blocks. And none of the other sub-blocks talk directly to one another. By observing the arrows, some of the interconnects are 2-way communication, such as between the keyboard and CPU, while other interconnects are 1-way communication, such as from the CPU to the monitor. In addition, there clearly are four blocks. Each block is labeled by function. Each block has distinct responsibilities for the system performance. And it could be clearly defined what those interfaces should be for those blocks to communicate with one another. For a team designing a computer system, this very simple block diagram already contains very valuable information which will help guide the designers towards a successful system design. This is the value of a block diagram. It can be done early, simply, and it can contain an enormous amount of critical system information.

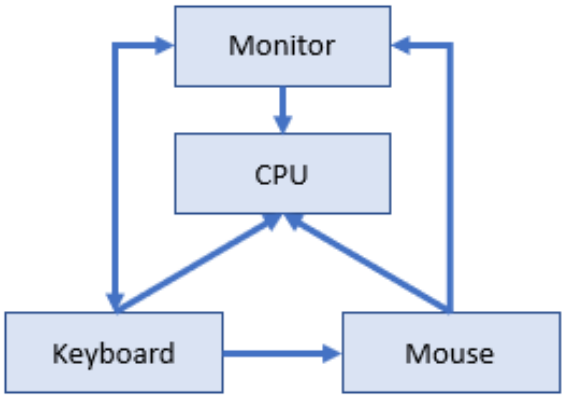


Figure 2. Computer Block Diagram.

As a comparison, imagine a block diagram as the one indicated in Figure 2. Although the blocks are the same as Figure 1, the interfaces are clearly more complex. And the interactions between the blocks more closely resemble a network rather than a command-and-control structure such as that indicated in Figure 1. Clearly, block diagrams offer a shorthand to an enormous amount of information in a simple easy to read format.

B. Standardized RADAR Block Diagram.

With no clear standardized block diagram, the author used the available information to piece together one comprehensive solution. The goal here was to generalize the sub-blocks in such a way as to incorporate a wide sample of the available examples. Any sample block diagram could be considered a simplified, or tailored version of the more generalized form.

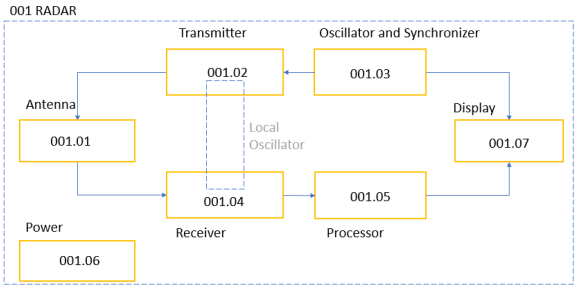


Figure 3. Generalized RADAR System Block Diagram.

Figure 3 is a generalized RADAR system block diagram for an airborne based military application. The outer dashed line can be considered Level 1, the complete RADAR System. The sub-blocks indicated in the figure comprise the Level 2 blocks and consist of antenna, transmitter, etc. This block diagram as well as the Level 3 block diagrams appear in the appendices.

What can be concluded here is that every airborne RADAR system for a military application will have an antenna. Antenna designs can vary widely. It could have any number of radiator elements: 1, 10, 100, 1000, etc. It could have any type of radiator: notch, patch, whip, etc. However, it most certainly will have some form of antenna. And so that sub-block appears in the block diagram. In this case it has been labeled 001.01 signifying the first (01) of the Level 2 sub-blocks. All of the sub-blocks have been correspondingly numbered. This will come up again within this paper when the WBS structure and models are discussed.

The block diagram also contains arrows which demonstrate the direction and flow of information between the sub-blocks. It can be seen that the directional flow is exclusively 1-way. It should be noted that this is limited to the signal flow including radio frequency (RF) interfaces. There could potentially be cases of multi-directional flow for purposes of digital control which are not addressed. For example, the processor might turn off the transmitter and then receive some feedback that the transmitter has indeed been disabled. However, that is a control signal and is not captured by this signal flow diagram.

The local oscillator appears as a dashed line and crosses into the transmitter and receiver as well as the white space in between. This is done because while those two sub-blocks require a local oscillator (LO) for operation, in some hardware configurations they have a resident dedicated LO, while in some configurations there is a separate sub-block dedicated to the LO function. And this representation is deliberately created to accommodate either physical hardware solution or implementation.

The descriptions for each sub-block (Level 1, 2, or 3) were generated in the same manner as the block diagrams. The available literature was widely explored, and the various descriptions were collected. Then, the various descriptions were combined into a higher order, more general version for which all descriptions could be considered a simplified, or tailored version of the more general versions presented here.

C. Sub-Block: 001.01 Antenna.

A numbering convention was selected, and each element is assigned a unique identifier. The numbering convention identifies hardware "Levels" (1, 2, 3, etc.). The antenna appears in Figure 4 as element 001.01 which was previously noted to designate the first of the Level 2 elements. The numbering convention was consistently applied throughout the development for the block diagrams, system model, and cost model.

The antenna is the coupling element between free space and the other RADAR elements. The antenna transfers the RADAR energy from the transmitter into free space. And the antenna collects the echo energy from free space and delivers it to the receiver for down conversion and processing.

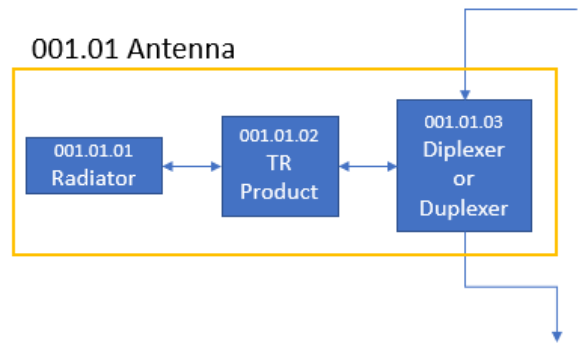


Figure 4. Antenna Block Diagram.

As illustrated in Figure 4, the antenna sub-block can be further decomposed into Level 3 blocks: radiator, transmit-receive (T/R) product, and duplexer. The element numbers have been assigned as indicated in the figure.

An anticipated criticism regarding this block diagram would come from the perspective of the hardware configurations. Typically, for an airborne system, there are two main hardware configurations. The first typical configuration is where each element is independent. Each element has a radiator and a T/R product. And then a bank of those channels is combined to make an array, a vertical configuration. The second configuration is where all the radiators are assembled in a bank of radiators, almost like a plate of radiators. And then those radiators are mated against a bank or plate of T/R products, a lateral configuration. A designer might argue that the image captured in Figure 4 is only one of the

two possible configurations. However, the representation in Figure 4 is a functional view irrespective of the hardware configuration. Therefore, both configurations are applicable. The block diagrams contained within this document are created to showcase the functional sub-blocks and the associations between them. The diagrams were intended to satisfy all physical instantiations.

The radiator is an exchanger between the propagating waves and the electric currents. The T/R product is a device where common circuitry for both transmit and receive functions are combined into a single module or element. The duplexer in a high-power RADAR system is the element that switches the antenna path between the transmitter and the receiver paths for a system where the two paths share an antenna. It is also used to protect the receiver from high power transmissions entering directly from the transmitter.

D. Sub-Block: 001.02 Transmitter.

The transmitter appears in Figure 5 as element 001.02. The transmitter modulates, or up converts, the wave to a transmission frequency. Then, if required to increase the signal power before transmission, the transmitter amplifies the wave for the antenna to send into operating space.

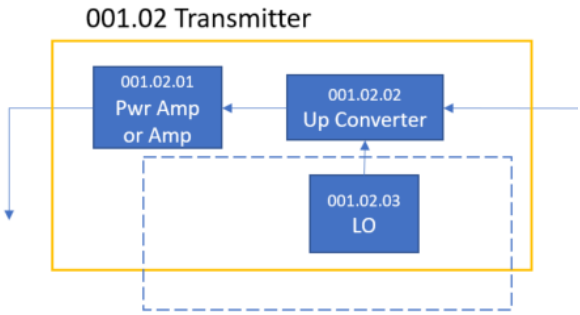


Figure 5. Transmitter Block Diagram.

As illustrated in Figure 5, the transmitter sub-block can be further decomposed into Level 3 blocks: power amplifier, up converter, and local oscillator (LO). The element numbers have been assigned as indicated in the figure.

The power amplifier is a device that converts a low power signal into a higher power signal. In the past, the high-power amplifier was more likely to be some sort of traveling wave tube. But certainly, the more contemporary approach would be a solid-state high-power amplifier. Regardless of the hardware configuration, the block diagram is a functional view and represents either approach. The LO is an oscillator which is used to change the frequency of the signal.

Local oscillators often employ some means of a phased locked loop (PLL). Typically, it is easier to have a stable oscillation when the frequency generated is low. And it is easier to generate a high frequency oscillation when the oscillation is less stable. By means of a PLL, it is possible to take the best of both and create a device which is stable at high frequencies. In the paper Digital Control Of Frequency Locked Oscillator, Microwave Journal March 2020, stable high frequency oscillations were achieved by locking a single oscillator to itself. It was accomplished by passing the signal through a long semi-rigid cable and then frequency locking to the time delayed reference.

The up converter is a nonlinear electrical circuit that creates new frequencies from two signals applied to it. Most commonly, this is accomplished with a mixer. In the case of a transmitter, the mixing product is a multiple of the sum of the input signal and the LO signal. The purpose of the up converter is to modulate the signal from the synchronizer for the antenna. But it is widely known that by means of a mixer, multiple harmonics are generated. And, by use of filtering, any higher order harmonic can be isolated, amplified, and used as the mixing product.

E. Sub-Block: 001.03 Synchronizer.

The synchronizer appears in Figure 6 as element 001.03. The synchronizer coordinates the timing of the RADAR. It generates timing pulses that are used to control the RADAR pulse repetition frequency (PRF). Signals are sent simultaneously to both the transmitter and the display to align the sweep echo pulses.

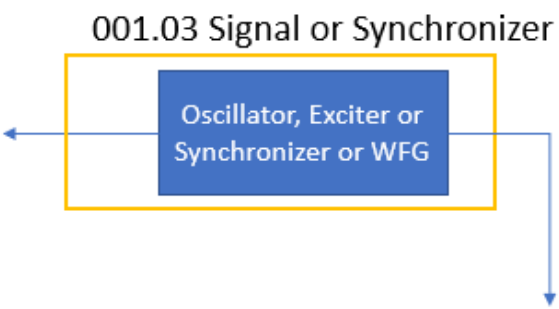


Figure 6. Transmitter Block Diagram.

The Synchronizer sub-block can be further decomposed into Level 3 blocks. However, for purposes of this analysis, the synchronizer will be considered as elemental, and cannot be further subdivided. This is done because there does exist a variety of synchronizer architectures and further analysis will need to be performed to determine a standardized Level 3 architecture. Thus, no element numbers have been assigned as indicated in Figure 6.

F. Sub-Block: 001.04 Receiver.

The receiver appears in Figure 7 as element 001.04. The receiver detects an incoming echo signal bounced off of a target, receives, amplifies, demodulates, and converts the analog signal to digital format for further analysis in the digital processor.

As illustrated in Figure 7, the receiver sub-block can be further decomposed into Level 3 blocks: low noise amplifier, down converter, intermediate frequency (IF) amplifier, filters, 2nd down converter, detector, and analog to digital converter. The element numbers have been assigned as indicated in the figure.

The local oscillator here is the same as that discussed in the RADAR and transmitter sections. As mentioned earlier, the LO is an oscillator which is used to change the frequency of the signal. In this application the LO is used to down convert a signal while in the transmitter it is used to up convert a signal. The low noise amplifier (LNA) boosts the signal while adding as little additional noise as possible. The goal is to maximize the signal to noise ratio (SNR) of the echo signal.

In a receive chain, the signal to noise ratio is determined primarily by the first element. The first element of the chain dominates the entire chain's performance. Low noise amplifiers, as the name suggests, are a special sub-class of amplifiers designed for this purpose. Therefore, an architect should assume an LNA as a first element.

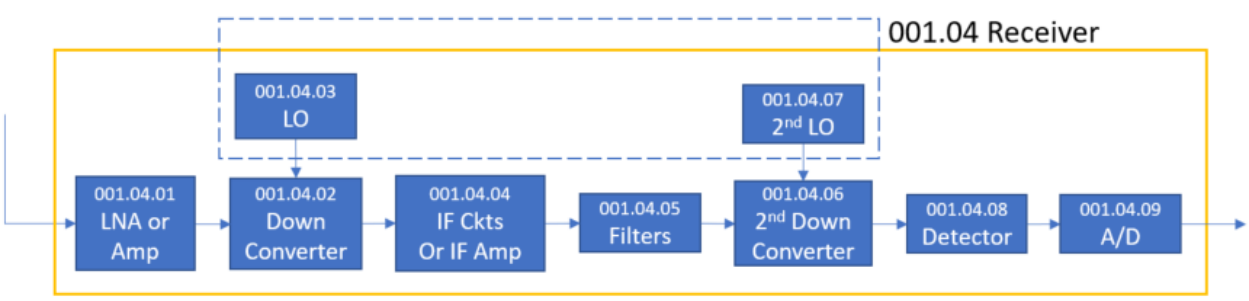


Figure 7. Receiver Block Diagram.

The down converter is a nonlinear electrical circuit that creates new frequencies from two signals applied to it. Most commonly, this is accomplished with a mixer. In the case of a receiver, the mixing product is a multiple of the difference of the inputs signal and LO signal. The purpose of the down converter is to demodulate the signal and the RF frequency of the LNA to a lower or intermediate frequency (IF) where amplification and filtering can be done more easily. Generally, multiple mixers would be used.

Later in the signal chain, the signal will be converted from analog to digital. The signal must be digitized for meaningful data processing of a modern system. At some future time, it may be possible to directly convert an X-band signal to digital, but in today's practical terms that is not yet possible. Therefore, the conversion is required. The detector and analog to digital (A/D) converter, as their names imply convert an analog signal into a digital signal. To convert the signal from analog to digital, a digital clock must be used. If the transmitted signal is "high," using the Nyquist criteria, the sampling frequency must be "higher." This is the fundamental limitation of the A/D converter. As mentioned, as time passes the technology is improving, but in today's practical solutions, the architect's options are still somewhat limited.

G. Sub-Block: 001.05 Processor.

The processor appears in Figure 8 as element 001.05. The processor decides if an echo is a target and determines if and how to present a depiction to the display. Typically, this may include number, location, and movement of targets.

The processor sub-block can be further decomposed into Level 3 blocks. However, for purposes of this analysis, the processor will be considered as elemental, and cannot be further subdivided. This is done because the processor is primarily a digital device, and the focus of this analysis is upon the analog path for the signals. Thus, no element numbers have been assigned as indicated in Figure 8.

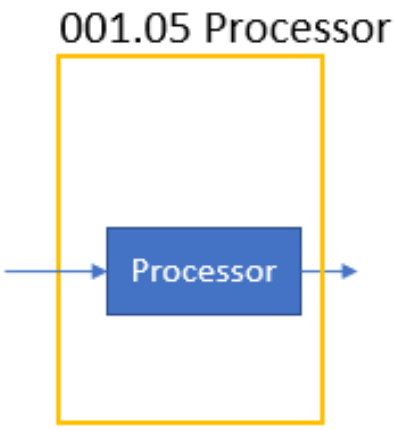


Figure 8. Processor Block Diagram.

H. Sub-Block: 001.06 Power.

The power block appears in Figure 9 as element 001.06. The power block converts the primary power from the platform to the required forms needed for each sub-block.

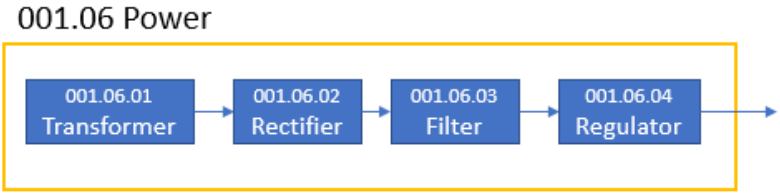


Figure 9. Power Block Diagram.

As illustrated in Figure 9, the power sub-block can be further decomposed into Level 3 blocks: transformer, rectifier, filter, and regulator. The element numbers have been assigned as indicated in the figure.

The transformer is a device which transfers electrical energy through electromagnetic induction. The transformer is comprised of coils. As current passes through a coil it generates an electro-magnetic field. If a second coil is placed within that field, a current is generated in the second coil. By adjusting the ratio of turns for each coil, a voltage can be stepped up or down. In

a familiar power supply, such as for a laptop, the power supply converts 120 volts AC from the wall outlet down to 12 volts DC for use by the computer. The transformation from 120 volts to 12 volts, called a step down, is done by means of a transformer. Because the transformer is comprised of coils, the transformer is always the heaviest component in the power supply.

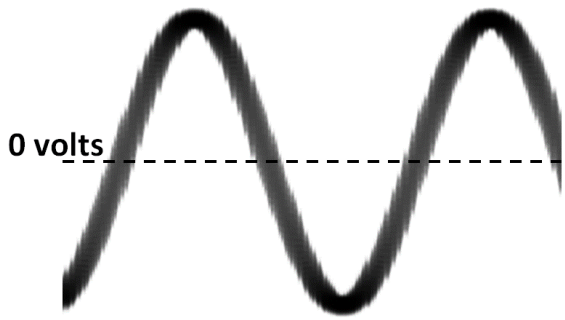


Figure 10. Sinusoidal Signal.

The rectifier is a device which converts electricity from AC to DC. The most common rectifier is a bridge rectifier and can be created with four diodes. For a sinusoidal signal (Figure 10) half of the time the voltage is positive, and half of the time the voltage is negative. Through the rectifier, the negative half cycle of the sinusoidal signal is flipped up to be positive (Figure 11). As a result, the wave form is now a series of positive going sinusoidal voltage "bumps," like humps of a camel's back.

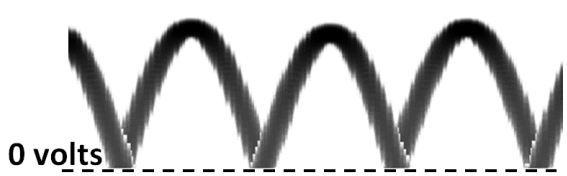


Figure 11. All Positive Voltages.

The filter is a device used to remove unwanted frequency components. After the voltage has been transformed into a series of positive going sinusoidal voltage "bumps," it is necessary to smooth it out. When the voltage value approaches zero the slope of the curve is negative. Once the voltage has reached zero volts, the voltage begins to rise and has a positive slope. The transition between a negative voltage slope and a positive voltage slope is instantaneous. The instantaneous nature of the voltage in the time domain corresponds to a high frequency effect in the frequency domain. By removing that high frequency component of the signal, by means of a filter, the corresponding waveform will be smoother.

The regulator is a device which stabilizes a DC voltage independent of the load current. The signal, post filtering, approximates a fixed DC value. However, the value is not stable. The voltage continues to have some residual effects from its sinusoidal origin. For use in a system, voltages must be stabilized. A regulator removes frequency components of a "dirty" DC signal and clamps it to a predetermined value. Post regulation, the signal is a "clean" DC value.

I. Sub-Block: 001.07 Display.

The display appears in Figure 12 as element 001.07. The display presents a depiction, in a usable form, of received targets. Typically, this may include number, location, and movement of targets.

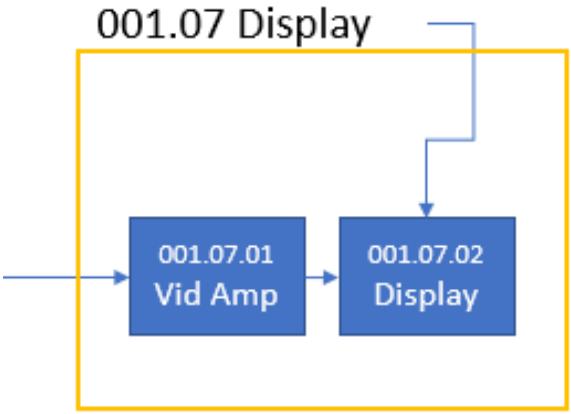


Figure 12. Display Block Diagram.

As illustrated in Figure 12, the display sub-block can be further decomposed into Level 3 blocks: video amplifier and display. The element numbers have been assigned as indicated in the figure.

The video amplifier is a device which is designed to process video signals. The display is a device which is used for presenting images and video. Typically, this would be a cathode ray tube (CRT) or more recently some type of liquid crystal display (LCD) such as a laptop screen or monitor.

IV. Model the System

Once a rigorous block diagram for a system has been created the next step is to begin modeling the system, or otherwise referred to as architecting a system.

Not only do system models align system and sub-system designers but also system models are key tools for cost analysts. System models include many artifacts such as documented requirements, documented use cases, logical view diagrams, operational view diagrams, etc. Just as with block diagrams, a cost analyst need not be an expert in all these areas but certainly a firm understanding would be most helpful. An awareness of system modeling and the related artifacts enables a cost analyst to better understand the system and then to estimate a cost with a higher fidelity. Just as with block diagrams, a good cost analyst should actively participate in the development of a system model to help define the system design.

There is a very important book on the topic entitled "Architecting Information-Intensive Aerospace Systems" by Dr. John M. Borky. In it, the author writes that architecting is done "to create systems and enterprises that are well organized, expandable and evolvable, robust under the stresses of real-world use, and affordable to own and operate. In short, the essence of the art and science of architecture is manifested in results that are beautiful in the eyes of their users while satisfying those users' practical needs."

Robust models created through Model-Based Systems Engineering (MBSE) are the foundation of the entire System Engineering (SE) process and provides a clear and unambiguous definition of the system.

While there are several tools which may do similar functions, for architecting the system in this paper, the COTS software tool used was chosen because it contained all the tools required to document requirements, document use cases, create logical view diagrams, operational view diagrams, and other system engineering artifacts.

A. System Modelling Approach.

When creating a structured architecture utilizing a COTS system engineering tool, one very useful structure is indicated in Table 1.

System Engineering Model
Components
Internal Block Diagrams
Packages
a_Requirements
b_UseCases
c_Structure
d_Behaviour
e_Data
f_Services
g_Context
PredefinedTypes (REF)
z_Default

Table 1 System Model Structure.

These items are referred to as packages. This is a very solid structure and provides an architect with designated locations for creating system artifacts. With a structure such as this, virtually any artifact required, or created, can be sorted, and stored into one of these packages.

B. RADAR System Modelling Structure.

An indentured set of numbers was created for this system and appears in Table 2. In Table 2, the hierarchy of the system design with Level 1, 2, & 3 sub-block names and numbers can be seen. These numbers form the basis for a Work Breakdown Structure (WBS).

Level 1	Level 2	Level 3	Block Name
001			Radar
	001.01		Antenna
		001.01.01	Radiator
		001.01.02	TR Product
		001.01.03	Duplexer
	001.02		Transmitter
		001.02.01	Power Amplifier
		001.02.02	Up Converter
		001.02.03	Local Oscillator
	001.03		Synchronizer
		001.03.01	Synchronizer
	001.04		Receiver
		001.04.01	Low Noise Amplifier
		001.04.02	Down Converter
		001.04.03	Local Oscillator
		001.04.04	IF Amplifier
		001.04.05	Filters
		001.04.06	2nd Down Converter
		001.04.07	2nd Local Oscillator
		001.04.08	Detector
		001.04.09	Analog to Digital Converter
	001.05		Processor
		001.05.01	Processor
	001.06		Power
		001.06.01	Transformer
		001.06.02	Rectifier
		001.06.03	Filter
		001.06.04	Regulator
	001.07		Display
		001.07.01	Video Amplifier
		001.07.02	Display

Table 2 Indentured System Numbering Structure.

C. Work Breakdown Structure (WBS).

A Work Breakdown Structure (WBS) (MIL-STD-881D) is a tool used to define a project in discrete work elements in a hierarchical format. It displays and defines the product, or products, to be developed and/or produced. It relates the elements of work to be accomplished to each other and to the end product. As described in the *smallbusiness* website, "The main purpose of a WBS is to reduce complicated activities to a collection of tasks." It is a very useful management tool. By arranging the architecture in such a manner, not only will it describe the breakdown of the hardware from Level 1 to Level 2 and so on, but it can also form the foundation of the set of tasks required to design the hardware. For example, a radiator is designated as block 001.01.01. The design cycle for any block, such as a radiator, will likely follow a standard design cycle: requirements, preliminary design, detailed design, and integration, verification & validation (IV&V). Those are phases which are made up of tasks and could be designated with the further indentured designators:

001.01.01.01 Requirements Phase
001.01.01.02 Preliminary Design Phase
001.01.01.03 Detailed Design Phase
001.01.01.04 IV&V Phase

These phases could additionally be designated with even lower-level numbers and even more specific tasks. The point is, creating a system structure with an eye towards a WBS is a best practice and frequently is a contract requirement.

D. RADAR COTS System Model.

The RADAR system demonstrated earlier as a set of block diagrams can be loaded into a COTS system engineering modeling tool (Table 1). The information from the block

Figure 12

diagrams could be loaded into the requirements package. The various block diagrams can be loaded into the structure package. And, in general, all system artifacts could be documented within the COTS tool.

The numbering and indenture of the entries should remain consistent with that presented in the earlier sections of this paper (Table 2). The specifications within the requirements package should also be consistent with the information contained within the block diagrams.

A robust system model helps to align system and sub-system designers and are key tools for cost analysts. A good cost analyst need not be an expert in system engineering tools. But some familiarity with system engineering tools would be very advisable. And early participation in a product life cycle will help a cost analyst to not only influence the direction and the development of the system design but also to then be in a far better position to generate system cost estimates.

V. Model the Cost

Once a rigorous block diagram and system model for a system has been created the next step is to begin modeling the system cost. As mentioned, with early participation in a product life cycle a cost analyst will be in a far better position to generate system cost estimates.

A new emphasis introduced here is a structured approach which includes a modular approach to modeling the system cost. At a later phase in the system design, it will be necessary to perform trade studies. The most common trade will be between two performance profiles. For example, "better" performance using more power vs. "worse" performance using less power. This is a very common trade in industry.

To achieve the two profiles, a modular approach will be used to swap out blocks for either "better" or "worse" performance. This is the elegance of a modular approach to system architecture. If a parallel effort could be taken to create a corresponding cost model for each block, then as blocks are swapped in and out for performance trades, a cost trade could simultaneously be performed.

As with system modeling, there are several COTS cost tools which may do similar functions. For costing the system in this research, the software package utilized was selected because it contains all the elements required to enable a user to create a modular cost model. Blocks can be created and turned on and off to simulate substituting one block for another. For each cost model block, there are parameters which can be adjusted to influence cost. Those parameters correspond to various ranges of hardware design details ranging from a very high level of detail to a very low level of detail depending on the user's familiarity with the hardware being modeled. All of the cost data within the COTS tool is pulled from industry standards, so no cost data *needs* to be loaded. Of course, for any user the tool data can be modified for specific applications and past performance actuals. Some cost tools include information regarding the sensitivity of a particular parameter being adjusted. The presentation of the sensitivity factors is currently a bit crude, but it should be possible to pull the data for further analysis outside of the tool.

A. Cost Modelling Approach.

Unlike the system tool structure which focused on artifacts and a manner by which to organize them (See Table 1), a cost tool focuses on the hardware, or more precisely, the indentured organization of the hardware. This allows a system architect to utilize the WBS numbering system directly in the tool. The tool directly estimates cost based on what hardware will be included. So, to create a cost model a user needs to first consider the indenture of the hardware.

Level 1 – Deliverable hardware

Level 2 – 1st Sub-block of hardware

Level 2 – 2nd Sub-block of hardware

Level 2 – 3rd Sub-block of hardware

Etc.

B. RADAR System Cost Modelling Structure.

As with the system model, an indentured set of numbers was created for this system cost model and appears in Table 3. The entries in Table 3 are very similar to those from Table 2. However, Table 3 has additional rows for "Roll Up." A cost tool could call out Level 2 hardware, for example an antenna. However, an antenna is also a collection of Level 3 hardware blocks. In this case, both options are included in the cost model. And when the model is run to produce an estimate either, but not both would be selected.

C. RADAR COTS Cost Model.

The RADAR system demonstrated earlier as a robust set of block diagrams and system model can be loaded into a COTS cost estimation tool. With the structured approach, the numbering and indenture of the entries should remain consistent with that presented in the earlier sections of this paper (Table 3).

Level 1	Level 2	Level 3	Block Name
001			Radar
	001.01		Antenna
		001.01.01	Radiator
		001.01.02	TR Product
		001.01.03	Duplexer
	001.02		Transmitter
		001.02.01	Power Amplifier
		001.02.02	Up Converter
		001.02.03	Local Oscillator
	001.03		Synchronizer
		001.03.01	Synchronizer
	001.04		Receiver
		001.04.01	Low Noise Amplifier
		001.04.02	Down Converter
		001.04.03	Local Oscillator
		001.04.04	IF Amplifier
		001.04.05	Filters
		001.04.06	2nd Down Converter
		001.04.07	2nd Local Oscillator
		001.04.08	Detector
		001.04.09	Analog to Digital Converter
	001.05		Processor
		001.05.01	Processor
	001.06		Power
		001.06.01	Transformer
		001.06.02	Rectifier
		001.06.03	Filter
		001.06.04	Regulator
	001.07		Display
		001.07.01	Video Amplifier
		001.07.02	Display

Table 3 Indentured Cost Numbering Structure.

As part of the COTS tool, each sub-block contains functional parameters which can be tuned for the specific application. For example, the weight of the specific hardware sub-block could be modified. What is initially loaded is an industry standard value to be used as a starting point.

This is the domain of the cost analyst. A good cost analyst having participated early in the design life cycle of the product will have familiarity with the WBS and product requirements. With a robust cost model which mirrors the system model a cost analyst is well positioned to generate a robust cost estimate and can rapidly participate in trade study alternatives. Because of the modular nature of the model structure, it is

possible to turn blocks "on" or "off" to select what is to be included for an estimate. In this way, blocks can be swapped in a modular fashion allowing a cost analyst to work with the system architect the ability to perform cost trades.

VI. Multivariable Analysis & Trade Studies

A robust structured system modelling approach utilizes the concept of modularity. If the system is comprised of modules, then the possibility exists where modules could be swapped to modify the system for various performance characteristics. At the same time, if the cost model mirrors the system model, then as the system is being defined, a rough cost estimation could be determined simultaneously.

Even with a modular approach, when designing a system more than one variable must be considered. Choices are made regarding those variables. In most cases, variable choices have competing impacts. For example, one design architecture may have "better" performance using more power vs. "worse" performance using less power. Decisions for a sub-system need to be evaluated at a system level. A system designer needs to consider the design as a system and realize that any change potentially has an impact beyond the sub-system. It is not usually possible to make an architecture or hardware change irrespective of the larger view of the system. This is really the heart of system engineering, consideration of an entire system, not just a collection of sub-system parts.

This is particularly important when considering cost because it is not possible to swap out cost as modular blocks and estimate new costs without understanding that there are affects to the system. There are multilevel impacts when modular blocks are substituted. Simply swapping out a block and estimating cost gives a first order indication of the cost impact. But until the design is finalized it is only a rough estimate. There is a spiral approach to design. As choices are made, impacts are assessed, costs can be estimated, new choices are made, and eventually the design spirals into a solution.

To decide between competing variables a trade study can be employed. A trade study is a useful tool which allows a designer to compare and contrast the various possible choices to determine which solution would be "best" for the given application.

To perform a trade study, first the various options are clearly defined. Criteria must be selected. Criteria are the items which are impacted by the options. Typical criteria are cost, schedule, performance, supportability, etc. A matrix is made with the options vs. the criteria, Table 4. A grade is given in the matrix for each criterion and option. Then the criteria are assigned a weight. The grades are scaled by the weighting factors. And then a score for the options can be calculated by adding up the weighted grades for each option. The option with the highest score "wins" the trade study and represents the "best" solution.

		Option #1		Option #2		Option #3	
	Weight	Grade	Weighted Grade	Grade	Weighted Grade	Grade	Weighted Grade
Criteria #1	5%	9	0.45	1	0.05	9	0.45
Criteria #2	5%	9	0.45	5	0.25	9	0.45
Criteria #3	15%	9	1.35	5	0.75	5	0.75
Criteria #4	10%	5	0.5	9	0.9	1	0.1
Criteria #5	30%	5	1.5	5	1.5	9	2.7
Criteria #6	25%	5	1.25	5	1.25	9	2.25
Criteria #7	10%	1	0.1	5	0.5	5	0.5
Total	100%		5.6		5.2		7.2

Table 4 Sample Trade Study Matrix.

VII. Sensitivity Analysis Applied to a Cost Model

The limitation of the commercially available cost estimation packages is that it is essentially a unidirectional process. A user defines a system and uses the cost estimation package to estimate cost. The user can then experiment with alternatives or modifications to the system and estimate the corresponding associated system cost. What is missing is a bidirectional interaction with the software package. There is very little guidance from the cost estimation package which suggests to the user system modifications for consideration. It lacks suggestions to the designer which modifications would have the greatest impact to the overall cost of the system.

It is desirable to have a feature within a cost estimation package which can analyze the components of the system to determine which components have the greatest impact. In other words, which components have the highest sensitivity for modification as it pertains to the overall cost of the system.

Although sensitivity analysis is well understood the application of sensitivity analysis upon a cost model for the purposes of maximizing the impact to the overall system cost is novel. It should be possible, and is explored in a follow-on paper, an effort to generate a cost sensitivity algorithm of the various components in a system to analyze a system and determine which subsystem components in a chosen design solution have the highest sensitivity to cost for the overall system. The analysis should highlight the areas to which a system designer could apply focus to reduce the overall system cost early in the life cycle of a program.

A. Sensitivity Analysis Potential.

Lack of adequate cost analysis tools early in the design life cycle of a system contributes to non-optimal system design choices both in performance and cost. A goal is to develop algorithms for an automated tool/approach utilizing cost element sensitivity to enable a system designer the ability to understand the relative cost impacts of various decision/choices which affect system design early in the design cycle for an airborne based RADAR system for military aerospace applications.

Most cost estimations are a unidirectional process. First a design is selected then the design cost is estimated. If the cost is not good the only feedback is typically to "reduce" cost. Then a new design is chosen, and the design cost is again estimated. But typically, the process lacks meaningful feedback which demonstrates how or where to make design changes to impact cost most significantly. Instead, the designer typically modifies an area of particular interest to the designer.

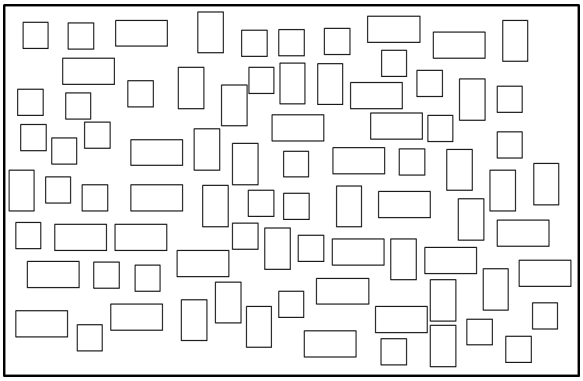


Figure 13. Complex System of Sub-System Blocks.

Consider a complex system made up of sub-system blocks, Figure 13. To estimate the cost of the entire system, the cost of the sub-system blocks is estimated and then rolled up into the top-level system cost. Typically, the cost is too high and there needs to be some effort to reduce the overall cost. So, trade studies are performed which focus on specific sub-system blocks. Of course, if any given block is modified, there will be an effect on other blocks known as secondary effects. For example, "better" performance using hardware which requires more power vs. "worse" performance using hardware which requires less

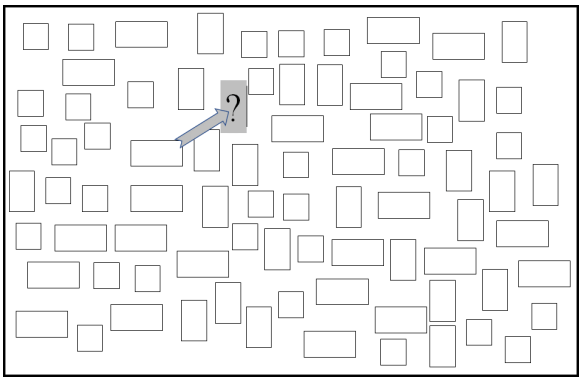


Figure 14. Arbitrary Sub-Block Selection.

power. A change such as that may have a secondary effect of increased copper thickness on other printed wiring boards which would then increase costs in other areas of the design. The secondary effects must be dealt with and considered but that would occur later in the process when a trade study is performed. Initially, there is the challenge of trying to determine which sub-block to apply focus to impact cost most significantly for the entire system, Figure 14. This is where the tools are significantly lacking. In the absence of sophisticated tools, the selection becomes somewhat arbitrary. It is desirable for the cost analyst to actively participate with the system designer to identify the areas of focus where the greatest impact to overall cost could be achieved.

Used in conjunction with a COTS cost estimation tool, it should be possible to develop an algorithm to understand the system sub-blocks in terms of cost sensitivity to overall system cost. With such an algorithm a cost analyst could analyze a system and determine the relative cost sensitivity for each sub-block.

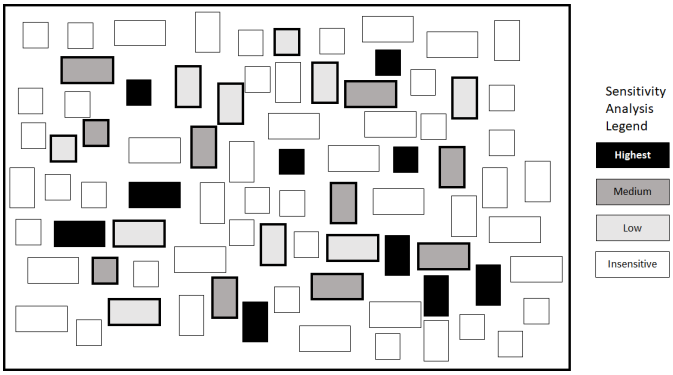


Figure 15. Sensitivity Analysis Results.

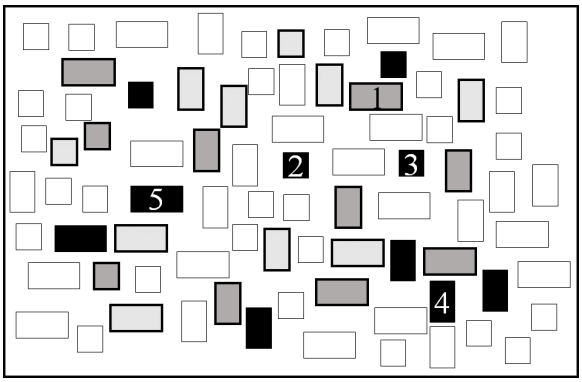


Figure 16. Selection of Five Sub-Blocks.

Once the sub-system blocks have a relative sensitivity value, the sub-blocks could be ranked from most sensitive to least sensitive, Figure 15. Then using knowledge of the system, a cost analyst could suggest a few sub-blocks to focus attention for reasonable improvement goals, Figure 16.

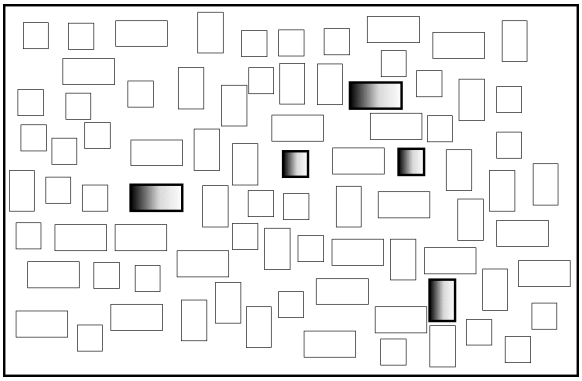


Figure 17. Simultaneous Modification of Five Sub-Blocks.

Once a few sub-blocks have been selected for reasonable improvement goals, and using the COTS cost estimation tool, an estimate could be made to determine the impact of cost to the entire system from simultaneous improvements to these few selected sub-blocks, Figure 17.

The overall impact estimation would provide a bound, or maximum, for potential cost improvements. To realize any potential component cost improvements there would need to be some amount of investment of resources. Any investment up to the estimated maximum potential value would yield a profit. This then forms the basis for a Return On Investment (ROI).

Again, a full trade study would need to be performed to evaluate the potential impacts to the rest of the system. But those trade studies would be paid for using the ROI estimations.

The limitation of the commercially available cost estimation tools is that it is essentially a unidirectional process. A user defines a system and uses the cost estimation package to estimate cost, Figure 18. What is missing is a bidirectional interaction with the software tool. By performing a sensitivity analysis upon the cost model, a cost analyst can offer suggestions to the system designer where to focus attention to most significantly impact overall system cost.

VIII. Summary

This paper presents an approach to generating a set of block diagrams for describing a standardized modular RADAR system applied to military applications in the aerospace industry. The resulting block diagrams were created using a compilation from a wide sample of available industry data and references integrated into a higher level, more generalized version. In addition to block diagrams, generalized block descriptions were also created along with a generalized numbering structure.

This paper demonstrates an approach to implementing the generalized block diagrams and numbering structure to create both a system model as well as a cost model for the RADAR under consideration. By means of the numbering structure, the system and cost models could be generated in such a way as to be modular to facilitate eventual trade studies for performance and cost improvements.

This paper discusses the potential advantages of a cost sensitivity algorithm applied upon the system cost model to analyze and determine which subsystem components in a chosen design solution have the highest sensitivity to overall cost. This paper illustrates that such an analysis could direct system designers to the areas of focus to most significantly impact the overall system cost early in the life cycle of a program.

Finally, the paper discusses using the sensitivity analysis results to select a few sub-blocks for reasonable improvement goals for simultaneous improvements. Simultaneous improvements to these few selected sub-blocks would provide a bound, or maximum, for potential system cost improvements. Any investment up to the estimated maximum potential value would yield a profit. This then forms the basis for a Return On Investment (ROI). The ROI estimate would in turn fund future trade studies.

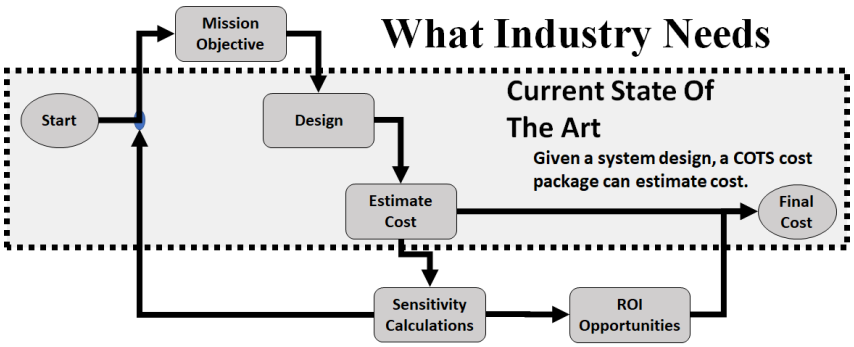


Figure 18. Ideal Cost Estimation Process.

References

Unknown. (2018). Math Science Notes. Retrieved 28 July, 2018, from http://mathscinotes.com/wp-content/uploads/2013/08/2_Radar_systems.pdf

Unknown. (2018). CopRadar. Retrieved 28 July, 2018, from https://copradar.com/rdrparts/

Mas3565560. (2018). Scribd. Retrieved 28 July, 2018, from https://www.scribd.com/doc/39306065/Basic-Components-of-Radar

Unknown. (2003). H2g2. Retrieved 28 July, 2018, from https://h2g2.com/edited_entry/A743852

Unknown. (2018). Fas. Retrieved 28 July, 2018, from https://fas.org/man/dod-101/navy/docs/es310/radarsys/radarsys.htm

http://mathscinotes.com/wp-content/uploads/2013/08/2_Radar_systems.pdf

https://copradar.com/rdrparts/

https://www.scribd.com/doc/39306065/Basic-Components-of-Radar

https://h2g2.com/edited_entry/A743852

https://fas.org/man/dod-101/navy/docs/es310/radarsys/radarsys.htm

http://www.uninettunouniversity.net/allegati/1/CommonFiles/Eventi/it/27/442/Radar%20System%20Design(final%20version).pdf

https://www.youtube.com/watch?v=943Ci3Xv9Ig

https://www.slideshare.net/vagheshp/radar-system-components-and-system-design

https://msi.nga.mil/MSISiteContent/StaticFiles/NAV_PUBS/RNM/310ch1.pdf

http://nptel.ac.in/courses/101108056/module1/lecture3.pdf

https://prezi.com/hj6kvajz5tew/components-of-radar-system/

http://faculty.nps.edu/jenn/Seminars/RadarFundamentals.pdf

https://www.nsstc.uah.edu/data/outgoing/outgoing/rwade/animations/ats671/GBRS_Radar_Lec2/Lecture03.ppt

https://m.eet.com/media/1111904/817_radar1.pdf

https://www.ll.mit.edu/publications/journal/pdf/vol12_no2/12_2devphasedarray.pdf

http://coreel.com/components-of-radio-detecting-and-ranging-radar/

https://en.wikipedia.org/wiki/Radar

https://en.wikipedia.org/wiki/Antenna_(radio)

https://en.wikipedia.org/wiki/Radiator

https://en.wikipedia.org/wiki/Transceiver

https://en.wikipedia.org/wiki/Frequency_mixer

https://en.wikipedia.org/wiki/Local_oscillator

https://en.wikipedia.org/wiki/Analog-to-digital_converter

https://www.google.com/search?q=power+supply+block+diagram&rlz=1C1MKDC_enUS772US772&tbm=isch&source=iu&ictx=1&fir=BEfxb62e0o6LOM%253A%252C9Ci9lxlphfzUKM%252C_&usg=__RU1G0uhkN2XZzelhpCHZ6JA2bTc%3D&sa=X&ved=0ahUKEwishdvI7I3cAhUqr1QKHag1CcAQ9QEIPjAE#imgrc=BEfxb62e0o6LOM

https://en.wikipedia.org/wiki/Transformer

https://en.wikipedia.org/wiki/Rectifier

https://en.wikipedia.org/wiki/Voltage_regulator

https://en.wikipedia.org/wiki/Electronic_filter

https://en.wikipedia.org/wiki/Amplifier#Video_amplifiers

https://en.wikipedia.org/wiki/RF_power_amplifier

https://smallbusiness.chron.com/importance-work-breakdown-structure-54294.html

Borky, John. *Architecting Information-Intensive Aerospace Systems*. Publisher, 2017.

Mr. ***Danny Polidi*** *received a B.S. and M.S. degree in Electrical Engineering from the California Polytechnic State University, San Luis Obispo, CA in 1990 and 1991. Current Ph.D. candidate for Systems Engineering at CSU. Upon graduation, started working at Space Systems/Loral on high frequency, microwave designs for space applications. Later, at Radian Technology, he became the Product Manager of the Digitally Tuned Oscillator Product Line where he was responsible for designing new circuits, writing code, and production. At NANOmetrics, Danny managed all Electronic Engineering activities. From 2004 – present he has worked at Raytheon as a Section Manger, Team Lead, Cost Account Manager and has been certified as a Program Manager.*

Mike Crist *received his B.S. in Computer Science from Embry-Riddle Aeronautical University in 2003 and his M.S. in Electrical Engineering from The University of Texas at Dallas in 2005. He is currently enrolled at Colorado State University working on a Ph.D in Systems Engineer. Mike has had a variety of roles over his 20+ year career, including embedded software developer, FPGA developer, circuit card designer, cost account manager, integrated product team lead and engineering tool strategist. He currently works on staff for an Electrical / Mechanical Design Center on various model based engineering projects.*

V. Chandrasekar *(S'83–M'87–F'03) received the bachelor's degree from IIT Kharagpur, Kharagpur, India, and the Ph.D. degree from Colorado State University (CSU), Fort Collins, CO, USA. He has been a Visiting Professor with the National Research Council of Italy, Rome, Italy, the University of Helsinki, Helsinki, Finland, the Finnish Meteorological Institute, Helsinki, IIT Kharagpur, and Indian Institute of Science,, an Affiliate Scientist with the National Aeronautics and Space Administration (NASA)'s Jet Propulsion Laboratory, Pasadena, CA, USA, a Distinguished Visiting Scientist with the NASA Goddard Space Flight Center, Greenbelt, MD, USA, and a Distinguished Professor of Finland (FiDiPro). He has also been a Director of the Research Experiences for Undergraduate Program for over 25 years, where he is involved in promoting research in the undergraduate curriculum. He is currently a University Distinguished Professor with CSU. He is also the Research Director of the National Science Foundation Engineering Research Center for Collaborative Adaptive Sensing of the Atmosphere. He has been actively involved in the research and development of weather radar systems for over 30 years. He has played a key role in developing the CSU-CHILL National Radar Facility as one of the most advanced meteorological radar systems available for research and continues to work actively with the CSU-CHILL radar, supporting its research and education mission. He is an avid experimentalist conducting special experiments to collect in situ observations to verify the new techniques and technologies. He has served as an Academic Advisor for over 70 graduate students. He has authored two textbooks and five general books and over 250 peer-reviewed journal articles.*

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