New Army Software Sustainment Cost Estimating Results

Authors

Cheryl Jones, US Army AFC, CCDC-Armaments Center
James Doswell, US Army FMC, DASA-CE
Dr. Bradford Clark, PhD, Software Metrics Inc.
Dr. Robert Charette, PhD, ITABHI Corporation
James Judy, US Army FMC, DASA-CE
Paul Janusz, US Army AFC CCDC-Armaments Center

Abstract

The U.S. Army is emphasizing the importance of making informed resource decisions regarding the sustainment of its portfolio of operational software systems. As a result, it has developed improved software sustainment cost estimation and performance analysis information methodologies for application across all system mission domains. This information is seen as a critical input to project and enterprise management decision making. This paper discusses a multi-year effort across the Army software community that establishes an objective software sustainment information infrastructure designed to provide objective software sustainment cost and technical information to Army decision makers at both the system and enterprise levels.

This is an update to the paper entitled “ESTIMATING SYSTEM SOFTWARE SUSTAINMENT COSTS: GENERATING CRITICAL DECISION INFORMATION TO INFORM THE ARMY SOFTWARE ENTERPRISE”, ICEAA, dated 5 June 2017

1 Executive Summary – Accomplishments

Through the support of DASA-CE leadership, the software sustainment initiative has succeed 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.

Through the initiative's effort, the Army pioneered a DoD software sustainment work breakdown structure (WBS) that is now being promulgated throughout the Army, and is being considered for use by other services. The WBS has created standard definitions of the different classes of software sustainment activities that Army programs are performing, and allows these activities to be quantitatively measured. It also permits software sustainment funding streams to be associated with work performed down to the software sustainment release level.

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 questionnaire served as a basis for the Office of the Secretary of Defense's (OSD) Software Resources Data Report for Maintenance (SRDR-M).

The initiative has also resulted in DoD's most comprehensive software sustainment database which has significantly enhanced the types and kinds of sustainment data available. The information in the database includes software release level data as well as management and process data on 192 Army systems in sustainment. The searchable database has over 411,000 data fields and contains actual execution information on over 700 capability releases, 300 Information Assurance Vulnerability Alert (IAVA) releases and 3,200 software licenses. 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 initiative's results have been provided to the Army software sustainment community, DASA-CE cost analysts, Life Cycle Management Command (LCMC) and Program Executive Office (PEO) cost analysts, as well as cost analysts from the other services. The results establish a robust foundation for software sustainment fact-based decisions, including:

- Allocations of Costs by WBS Elements
- Cost & Schedule Estimating Relationships
- Cost Benchmarks

The results are being used to improve not only Army software sustainment cost and schedule estimates, but management and software engineering practices, as well as to better understand the uncertainties and risks involved.

The initiative has firmly established a robust foundation for making software sustainment fact-based decisions and for understanding how software sustainment impacts Army readiness and combat effectiveness. More information is continually being added to the database, which will make it an even more unique and valuable Army software asset in the future.

2 Initiative Overview

It is commonly understood across the Department of Defense that software is a critical mission asset. [1] Software is where Army mission capability is continuously enhanced and sustained to meet new requirements, to adapt to new system interfaces, to integrate emerging technology, and to address known and projected security threats.

The Army's Chief Information Officer Lt. Gen. Bruce T. Crawford calls software "the next great frontier," because it's the system component where nearly all Army mission capability is now, and will be, instantiated. Software is where Army mission capability is continuously enhanced and sustained to meet new requirements, to adapt to new system interfaces, to integrate emerging technology, and to address known and projected security threats. [2]

Technical and management attention has traditionally focused on software development and acquisition, even though as much as 70% of the life-cycle cost of DOD system software is associated with its sustainment. [3] In fact, the majority of a system's mission capability is created and integrated during the sustainment portion of the system life-cycle. The 2018 Defense Science Board report titled, "Design and Acquisition of Software for Defense Systems, Department of Defense," notes that weapon systems like the Apache and Chinook helicopters continue to be effective platforms far "beyond their design life due largely to improvements via software upgrades."[4] The Chinook, for example, may be operational for nearly a century before it is retired, or some eighty years beyond its original designed life span.

As total software investment grows across all DOD systems, there has emerged a commensurate need to more effectively manage the resources allocated to the software systems that are in the field supporting the warfighting mission. This is especially the case with respect to software sustainment.

The U.S. Army has recognized the importance of making informed resource decisions regarding the sustainment of its operational software systems portfolio, and as a result, has emphasized objective software cost estimation and performance analysis as an integral part of its overall strategic software enterprise. Driving the need for accurate software sustainment technical and cost decision information is the proliferation of Army software systems resulting from 20 years of overseas conflict, and the recognition of projected future shortfalls in the resources that will be necessary to maintain these operational systems.

Army leadership, beginning with the Optimization of Software Acquisition, Development, and Sustainment initiative sponsored by the Secretary of the Army in 2013, and the concurrent software sustainment initiative under the Deputy Assistant Secretary of the Army for Cost and Economics (DASA-CE), has actively engaged the Army software community to examine how the service can best allocate and manage its software sustainment investments to achieve the Army's long and short term mission priorities and objectives. [5 - 13] To achieve these goals, however, requires software sustainment data that is visible, measurable and reliable.

3 Software Sustainment Decision Information

Objective software sustainment resource decision information is required at both the system and enterprise levels within the Army. At the system level, program managers and system software teams must decide what baseline change requirements to implement; how to prioritize the capability enhancements, maintenance corrections and adaptions, and security changes; and how

to structure incremental software product deliveries. Army enterprise level decisions determine resource priorities across the operational system portfolio, and trade-off funding and mission capability. The goal is for decision information to objectively tie investment costs to software product output and eventually to mission capability.

To address the information needs of the key Army software sustainment decision stakeholders, DASA-CE structured its initiative around a fundamental software sustainment information infrastructure. This includes:

1. Development of an Army software sustainment specific Work Breakdown Structure (WBS) adaptable to multiple mission domains. The WBS is derived from the processes, practices, and guidance followed by Army software sustainment technical teams and organizations, and provides a common and flexible structure for collecting cost and effort against validated software sustainment activities and products.
2. The creation of a baseline system software sustainment data repository. This is currently based on the collection, characterization, evaluation, and normalization of three years of historical software sustainment technical, cost, and context data from most of the 192 operational Army software systems. Additional data is currently being collected from FY18.
3. Analysis of the normalized software sustainment data to generate improved and validated WBS driven cost heuristics and statistically derived sustainment cost and schedule estimation relationships, benchmarks, and distributions. This effort specifically addresses the differentiations across top-level Army system software mission domains.
4. Development of a software sustainment cost estimation methodology linked to the data available at different system life cycle milestones. This includes the development of a risk focused estimation uncertainly model that takes into account data and information risk as well as program and technical risk in establishing estimation uncertainties.
5. Design and implementation of systemic Army software sustainment data collection methods, leveraging existing Army information systems and established OSD data collection and cost analysis constructs, specifically the Software Resource Data Report for Maintenance (SRDR-M). The data collection requirements are tied directly to stakeholder decision information requirements at all management levels.
6. Development of software sustainment cost analysis capabilities at all decision levels to support the enhancement of software sustainment cost models and to assess software sustainment cost performance across the Army on a continuous basis.
7. Development of specific recommendations to revise Army policy and guidance to better align the allocation and management of planned and applied resources to current software sustainment engineering processes across the service.

DASA-CE has completed its analysis of the initial Army software sustainment data set, and is actively supporting the implementation of the decision information infrastructure across the Army. Based on its efforts to date, an improved software sustainment cost estimation methods and models provides the basis for an effective decision information infrastructure.

4 Key Infrastructure Components

The software sustainment cost estimation initiative's first objective was to determine the availability and characteristics of the data collected. Based upon the complex sustainment environment in the Army, and the large number of variables that impacted the data, there exist clear shortfalls with respect to data availability, integrity, and usability.

Prior to this initiative there has been no requirement for Army systems to report actual software sustainment execution data, for contractor or Army organic (government) resources. This has impacted consistent enterprise insight into overall software sustainment portfolio performance.

Rather than being directly tied to output activities and products, life cycle software sustainment costs have been generally estimated as a percentage of the system's initial software acquisition cost. Some IT systems are estimated based on the projected size of the sustainment teams and other general "capacity" oriented cost elements. This results in an inaccurate cost estimation methodology that does not objectively project the actual cost of sustaining a software system.

Up until recently, DOD software sustainment has lived in the "shadows," meaning that controlling and justifying the cost of acquiring a system was of higher priority than controlling or justifying the cost of its sustainment. The Government Accountability Office's "GAO Cost Estimating and Assessment Guide: Best Practices for Developing and Managing Capital Program Costs," reflects this precedence. [14] However, as system software sustainment costs have grown and new system acquisitions have slowed, understanding and controlling the drivers of these costs has become more important to both DOD and Army leadership.

This initiative determined that there were three key components required to establish a sound foundation for generating the necessary information:

- Well defined and prioritized stakeholder information requirements
- An Army software sustainment Work Breakdown Structure (WBS)
- A repository of software sustainment execution data

4.1 Software Sustainment Information Requirements

An early objective of the initiative was to determine the information requirements of Army program and enterprise managers with respect to software sustainment. In other words, what information do decision makers at different levels of the Army enterprise need to know or have available to make effective software sustainment resource and/or technical decisions?

A number of workshops and meetings were held with a wide array of Army software sustainment stakeholders over the course of several years. Both common information requirements across all decision levels and unique information needs from each stakeholder were identified. A major finding of the decision information requirements analysis was that there is an overarching need for performance and cost information at a lower level of detail, especially with respect to executed software sustainment cost information mapped to specific software characteristics (i.e. software size) and products.

The three highest priority requirements that were identified are summarized in the rest of this section.

1. What are the software sustainment costs for individual systems?

This information need addresses the factors that may drive or influence software sustainment costs. Since no cost estimation decision is better than the data that supports it, the question is what kinds and types of data need to be collected to fully understand these factors? Importantly, can these factors in fact be measured and modeled?

Experience shows that to make a reliable software cost estimate of any kind, an appropriate workload measurement unit and its unit cost must be defined. For making a software sustainment cost estimate, this definitional exercise is not straightforward because of the lack of consistent measures and data available. This creates uncertainties in the eventual cost estimate, and even then, it may not be clear what factors create the highest levels of uncertainty in the estimate.

For instance, it is well known that there are various reasons as to why software sustainment costs increase such as system age, number of external interfaces and software code complexity. However, does the number of external interfaces drive sustainment costs more than software code complexity, and if so, by how much? Or does software sustainment practice drive sustainment costs more than either of those two factors?

Furthermore, new software systems development combined with sustainment practices such as Agile/DevOps/DevSecOps are being touted as a way to control software sustainment costs as well as fielding more secure systems to the warfighter quickly. How much will software sustainment costs decrease (if at all) using these approaches, given the lack of experience with these practices?

Additionally, there is a need for rough order of magnitude sustainment cost estimates early in the acquisition lifecycle where the amount of workload to sustain a new system is not known. It is estimated that 70% of a system's total lifecycle cost is in the sustainment phase. Are there validated rules-of-thumb or cost profiles that can inform such an estimate?

2. How much is the Army spending on software sustainment and what is it getting in return?

There are several different Army system commodities in use, including: Business, Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR), Chemical/Biological (Chem/Bio), Missiles, Fire, Aviation, Space, Test, and Vehicles. A critical question is which of these commodities are the least and most expensive systems to sustain in terms of annual costs? This is related to the information need from (1) above, which is, why would one type of commodity be more expensive than another?

Software changes can be classified as enhancements, maintenance and cyber security. Enhancement changes increase the software system's capability. There is a need to know how much sustainment funding is used to increase capability providing more effectiveness for the end-user. What percentage of funding is used to increase capability for each system?

In addition, cybersecurity is an increasing cost in sustaining Army systems. The question is how much is cyber security driving the cost of sustainment, and will this cost continue to rapidly increase as it has over the past decade? Furthermore, are COTS software products more or less expensive to sustain than non-COTS systems to ensure they are cyber-secure? Related to this is how much are COTS license costs increasing annually?

Software systems can have multiple releases over a year. Each release requires effort to design, implement, test, and validate proposed changes. What is the cost tradeoff between shorter versus longer release cycles and mission capability increases?

Furthermore, Army programs in software sustainment often have multiple funding streams associated with them. It is not always clear how an individual software release is related to which funding, especially when most of the funding is being sent to an Army contractor. Traceability of funding to actual work performed and paid for is a high priority information need.

3. How do changes in Army software sustainment funding affect Army, and by implication DoD, readiness?

It seems obvious that changes in funding would affect readiness, but it is extremely unclear how to quantify the impact. Even a modest cut in funding might have an outsized impact on a system's operational capability.

The reason is that there are *fixed sustainment costs* (e.g., software licenses, system facilities and operational management) and *variable costs* (e.g., future delivered capability, project management, certifications and accreditations, sustaining engineering, and field software engineering). In some systems, the fixed costs make up the majority of total sustainment costs incurred by a system.

Thus, even a small reduction in funding might have a major impact system readiness since the majority of the impact would affect variable costs, not the fixed costs. The temptation may be to cut "x" percent across all systems in sustainment, but for some systems, that would mean no capability improvements or defect repairs would ever occur.

4.2 Work Breakdown Structure

A viable software sustainment Work Breakdown Structure (WBS) is the cornerstone for credible cost estimates. Without a WBS, it is impossible to link performance to funding. In addition, not having a WBS makes it difficult to compare similar systems to one another for cost estimation purposes.

The current Army software maintenance WBS is shown in Figure 1. The elements of the WBS are based on actual practice across all Army systems. It defines the cost elements that comprise Army software sustainment. The WBS was designed to be tailored for specific system and organizational instantiations and can be adjusted to account for variations in domain driven technical characteristics and installed cost accounting systems.

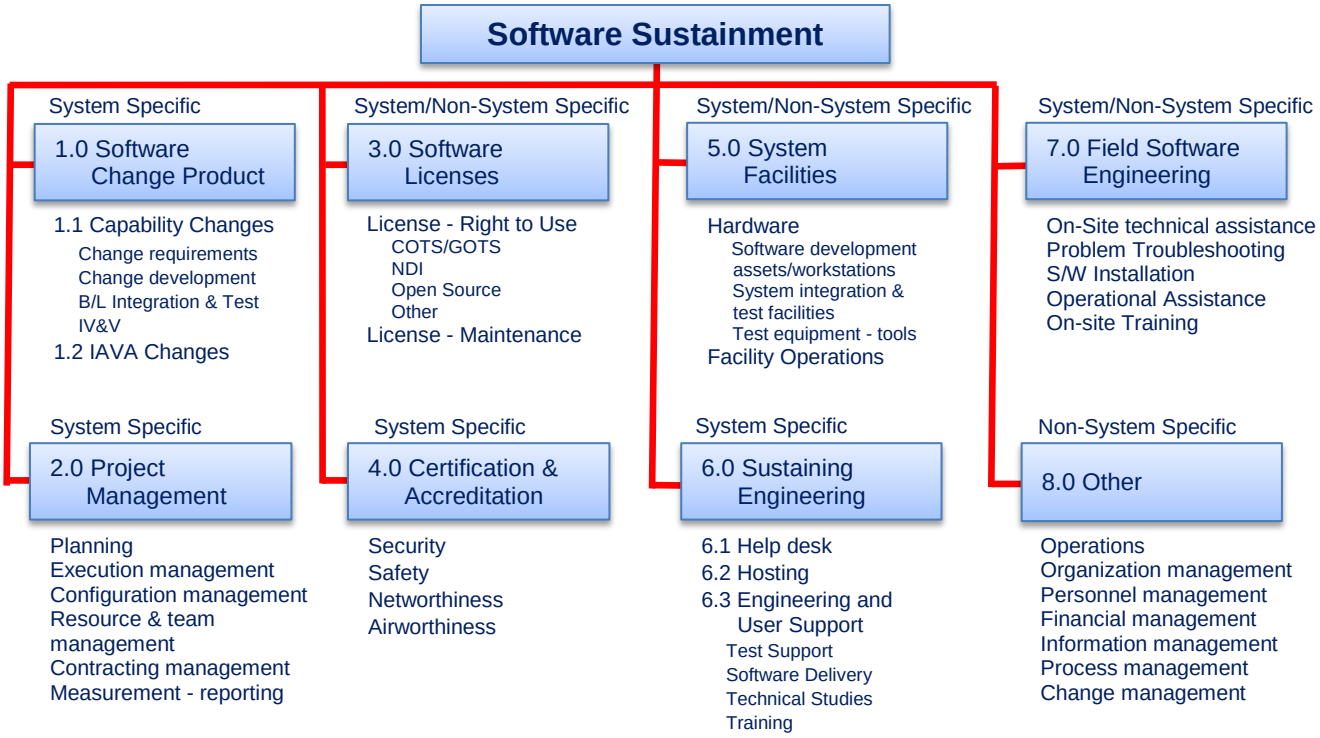


Figure 1. Army Software Sustainment Work Breakdown Structure

4.3 Software Sustainment Data Questionnaire and Repository

An Army Software Sustainment Data Questionnaire was developed to collect data and the data is stored in the data repository. Three general categories of data are collected using the questionnaire.

1. System characterization data that describes the technical and programmatic characteristics of the operational system and the system sustainment strategy. This data includes sustainment activity, release and change profiles, domain and mission characteristics, program technical and management risks, etc. This data provides information on how the software baselines are maintained, and supports the normalization of diverse program data sets. Note that this data is not directly mapped to the WBS.
2. System specific effort and cost data at the total system level, and for each of the WBS elements, including both government and contractor costs. Only costs that are attributable to a specific system are included. The software license WBS also has a detailed breakout that describes each license used by the system, including costs for each license (if purchased by the system), and other details.

3. Release level data for capability-releases and IAVA-only releases. This data includes system software sustainment cost and technical data mapped to specific output products and activities. This data includes release characterization data, release effort and cost, schedule information, output products (software size), incorporated changes, etc. Note that this characterization data is in addition to cost data for WBS 1.0, Software Change Product.

For all categories of data, the actual execution cost and effort data is obviously preferred, but Full-Time Equivalent (FTE) or planning data was collected if actual data was unavailable. The actual/estimated data was tracked in the repository for discrimination in data analysis.

Section 5.0 on Software Sustainment Data Characterization provides more information on the data collected.

4.4 The Importance of Understanding the Context

Every Army system is unique in some way. Therefore, the collection of system "context" data is required to enable the accurate interpretation, comparison and contrasting of the collected cost and technical software sustainment data.

Integral to the data analysis are the definitions of distinct software systems based on sustainment organization, commodities, application super domains, maintenance change types, sustainment phase, and number of software variants, platform variants, users and licenses.

Data analysis revealed that Army software sustainment activities are not "monolithic." That is to say that there is no single model or cost estimating relationship (CER) that can be defined to address the multitude of variables across Army software sustainment activities that will yield a valid cost estimate. All of the different products and activities that are being costed differently have to be taken into account, and their results integrated into a composite, context driven estimate. There exists too much variability across the program products and activities in question and the calibration of the CERs based on context, domain, etc., for a single model to be correct in all instances.

4.5 Army Software Sustainment Definition

Software sustainment (SWS) includes all software change activities and products associated with modifying a software system after a software release has been provided to an external party. The release, a composite of one or more changes, is the primary SWS change product. A release can be either a formal release or an engineering release. SWS may include software enhancements, software maintenance, and/or cybersecurity updates.

Software maintenance (SWM) includes defect repair, rehosting, adaptations, updates, and reconfiguration of the software. SWM is a type of change performed on the software.

SWS may be funded by multiple funding sources. Costs include both fixed and variable costs accrued at both the system and organizational levels for both organic (government) and contractor resources

5 Sustainment Data Characterization

A two-phase data collection activity was conducted for the initiative. During the first phase data was collected for five programs from each Army software sustainment activity, including PEOs and Life Cycle Management Centers (LCMCs), for a total of 56 systems. This first phase established an understanding of the software sustainment activities and data environment across the Army, which drove the refinement of the data collection questionnaire. This understanding was also reflected in the new DOD Software Resource Data Report for Maintenance (SRDR-M) that includes much of the same context, cost/effort, and technical data.

The second data collection phase collected software sustainment data across the remaining Army programs along with updates to some of the first phase systems. This included 136 additional systems and allowed analysis using a more complete data set. Both weapons and non-weapons systems comprise the dataset.

5.1 Data Overview

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 2.

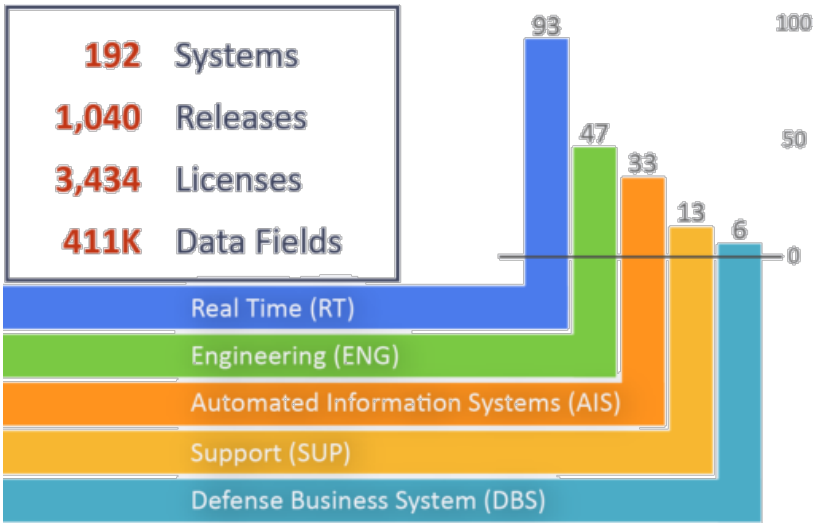


Figure 2. Data Demographics

When systems are 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).

5.2 System Age

The data contains systems that vary in years in sustainment up to 40 years and are in one of two phases (see Figure 3). Post-Deployment Software Support (PDSS) characterizes systems whose hardware component are still in production; however, the software components require sustainment activities to support fielded systems. PDSS systems are managed under the Program Executive Office (PEO), and are funded with RDT&E or Production funding. Post-Production Software Support (PPSS) systems are operationally sustained via a Life Cycle Management Center (LCMC) and are funded with O&M funds.

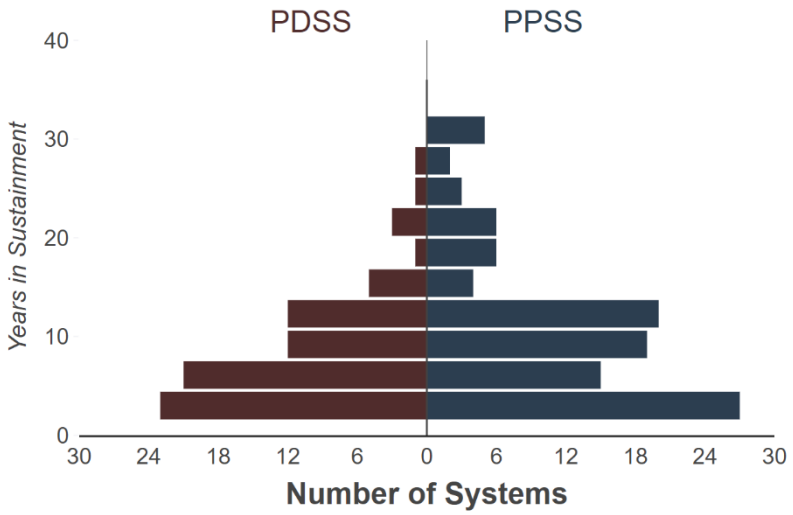


Figure 3. Age of Systems

5.3 Software Release Characterization

Figure 4 shows that releases are divided into capability releases (718) and IAVA-only releases (322). Capability releases modify software while IAVA releases scan the software for vulnerabilities. Of the capability releases, there were 318 primarily maintenance releases, 170 primarily enhancement releases, 195 hybrid releases that were a mix of maintenance and capability enhancements, and 16 releases classified as "other." IAVA-only releases were all classified as Cyber releases, and a few capability releases contained only IAVA updates, for a total of 341. Different types of releases were each analyzed for CERs.

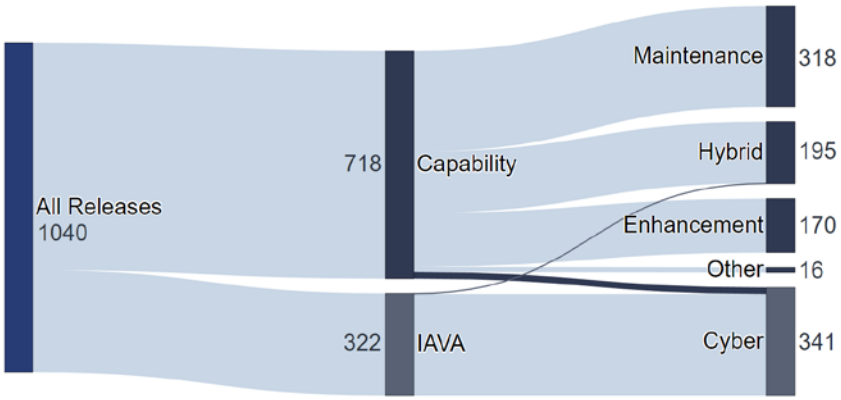


Figure 4. Releases by Change Type

5.4 Release Size Measures

Systems were asked to report the size measures that were used within their program. Figure 5 shows the size measures reported. Software Changes (SC) was the most common size measure with data provided for 571 releases. SCs are enhancements or maintenance changes to the software. There was variability in what constituted a software change and this is discussed later in the section on Lessons Learned. The second most common size measure reported was a count of the number of IAVAs, with data from 420 releases. Some systems reported the number of requirements implemented in the release, for 224 releases. Source lines of code (SLOC) counts were reported for 152 releases. A subset of those releases broke down the code counts for new, modified, reused, and autogenerated code. Other size measures included Function Points and RICE-FW objects reported in 39 releases. Story-Points were reported for 11 Agile releases.

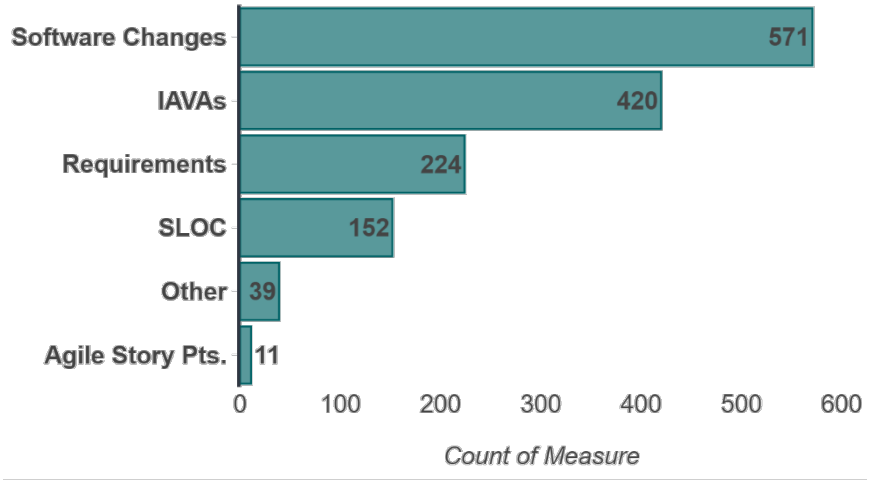


Figure 5. Releases by Size Measure

6 Cost and Schedule Estimation Relationships (CER/SER)

This analysis examined only the cost or effort to maintain software in the WBS Element 1.0, Software Change Product. Two types of analysis were employed: cost estimating relationships (CER) derivations and schedule estimating relationships (SER) derivations. The CER analysis segmented the data into categories and attempted to find an independent variable (e.g., equivalent source lines of code, number of software changes) to estimate the dependent variable, effort hours. The SER analysis segmented the data into categories and used the independent variable, effort hours, to estimate the dependent variable, duration.

The CER data were grouped into two release types: Capability releases and IAVA releases. A capability release changes the software to improve its capability or repair a problem. An IAVA release identifies cybersecurity issues that need to be evaluated within the system and in most cases can be resolved through a software patch distributed by the commercial software vendor. Note that if software requires a change as a result of an IAVA, it is categorized as a capability release due to repairing a problem. The IAVA analysis results are discussed in a separate section called IAVA Analysis.

6.1 CER Derivation for Capability Releases

This analysis investigated cost estimating relationships (CERs) for WBS 1.0, Software Change

Product, for capability releases. These CERs are utilized later in the acquisition lifecycle (post Milestone C) when expected release size (e.g., SLOC or number of software change counts) is known.

The data for WBS 1.0 is measured as a software release. Data was collected on capability release characteristics of size, cost, effort (hours), type of software changes, commodity, application super domain, and number of IAVAs performed as part of the change activity.

6.1.1 Ground Rules/Assumptions

The following ground rules or assumptions apply to each CER:

- The CER applies to WBS 1.0, Software Change Product, only for capability releases.
- Defense Business Systems were not included in this analysis.
- The term Total Cost includes both government and contractor costs.
- Data that was not within 50% of the reported annual labor hours per person-year and annual burden labor rate were labeled outliers and not used in this analysis.
- Due to the non-normal distribution of the raw data, both dependent and independent variables were transformed using $\log_{10}$. Zero values were represented as 0.1.
- All categorical variables (super domain, commodities, etc.) were represented as dummy variable (0,1).
- Ordinary Least Squares (OLS) regression was used to derive the CERs. The Minitab statistics tool and Python statistics libraries were used for OLS analysis.
- Software size can be expressed as Software Changes, Requirements, or Equivalent Source Lines of Code (ESLOC). ESLOC is a combination of new, modified, reused (code with no modification) and auto-generated code counts. The ESLOC formula to combine these different code types was as follows:

*ESLOC = New Code + (0.7 * Modified Code) + (0.15 * Reused Code) + (0.30 * Auto-Gen Code)*

- Three model fit statistics were used to evaluate the CER:
  - The R^2 is the Coefficient of Determination and represents the percentage of total variation explained by the regression model. It provides a measure of how well actual values of effort are estimated by the CER model factors. R^2 ranges between 0 and 100% and is the same in log-space as in unit-space.
  - Standard Error of the Estimate (SEE) is a measure of the accuracy of predictions made with a regression line.
  - PRED (30) is the percentage of CER estimates that fall within 30% of the actual values.
- All costs were normalized to Base Year FY18.

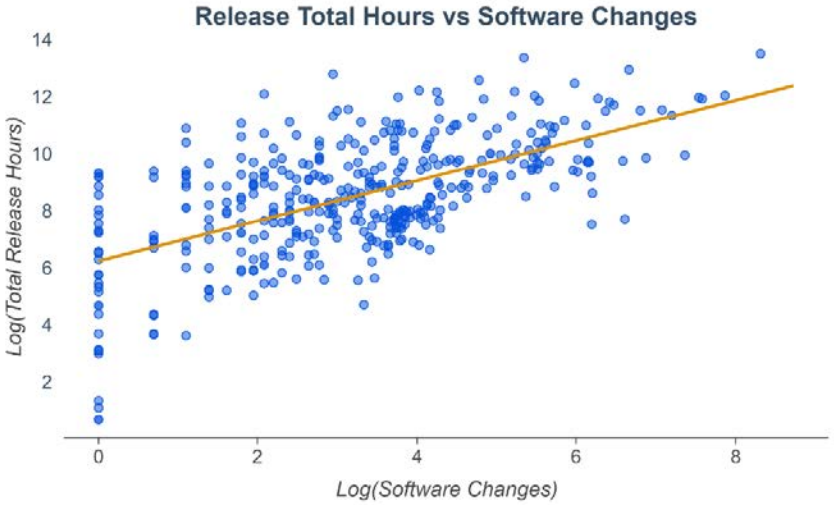
6.1.2 Methodology

After initially reviewing the raw data for CERs, the data was shown to have a high amount of variability. This led to an extensive effort to "*scrub*" the data to ensure it was as uniform as possible and to identify outliers. Scrubbing consisted of four parts:

1. All measures used for CER analysis were normalized to a common measurement unit, e.g., $110K was converted to $100,000 or 13K SLOC was converted to 13,000 SLOC. The format of the data in the repository was corrected to be consistently the same. The terminology for categorical data was also changed to make all labels consistently the same.

2. Data outliers were identified by comparing labor cost per hour to the reported annual burdened labor cost and annual hours per person-year. Each program was asked to report the annual labor rate for government and contractor as well as the labor hours per person-year for each. This annual labor rate was then compared to the labor rates reported for each WBS element, including WBS element 1.0, Software Change Product, (a software release). If the labor rate was more than 50% above or below the annual labor rate, that observation (data point) was considered an outlier.
3. Source lines of code is a release size measure that includes new, modified, reused, and autogenerated code. Equivalent SLOC was derived from raw SLOC counts by using formulas that combined the new, modified, reused, and autogenerated code counts as discussed in the assumptions above. Another anomaly with the SLOC count was the baseline code at the start of the release was often not included in the reuse count. This was corrected as well.
4. The release data was divided into two types of releases: Capability Releases and IAVA-only Releases. These release types are distinctly different as capability releases modify software and IAVA releases scan the software for vulnerabilities.

After the data was scrubbed, a regression analysis of the data was performed.



Variable	Coef.	Std Err	T-Value	P-Value
Constant	6.2438	0.172	36.231	0.000
Log(SC)	0.730	0.047	14.900	0.000

Figure 6. All CER Data Scatter Plot

Figure 6 shows a log-scale scatter plot of 397 observations for all capability release data that had the independent variable, software changes, and the dependent variable, total release hours. The plot shows a regression model of log(Hours) = 0.730*log(SC) + 6.2438 with a large amount of variation and an R^2 of 36.0%.

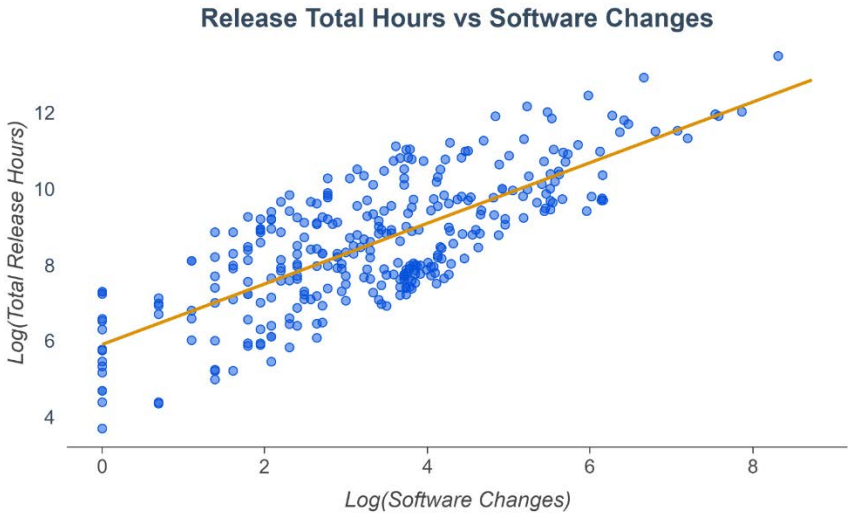
Due to the poor results, the data was then segmented into categories or groups to tighten variability using three strategies:

1. The upper and lower 10% of the data was trimmed from the dataset. 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.

2. Meta-data was used to derive multiple categories, each of which was analyzed for CERs using trimmed and untrimmed data
3. Unit cost was divided into quintiles resulting in strong CERs but presented the challenge of finding common characteristics for each quintile in which releases could be grouped.

The first strategy trimmed the upper and lower10% of the data based on unit cost (Hours/SC). Figure 7 shows the scatter plot and trendline on 317 observations. The regression model is log(Hours) = 0.7981log(SC) + 5.905 with an R^2 of 57.2%.



Variable	Coef.	Std Err	T-Value	P-Value
Constant	5.9052	0.145	40.618	0.000
Log(SC)	0.7981	0.039	20.532	0.000

Figure 7. Trimmed Data CER

The second strategy used categorical data to create similar data groups. Each category was used in CER regression analysis and in categorical trend analysis. Many of the groups did not reduce the variation in the data. The most significant groups are discussed further in this paper. Each group is shown below:

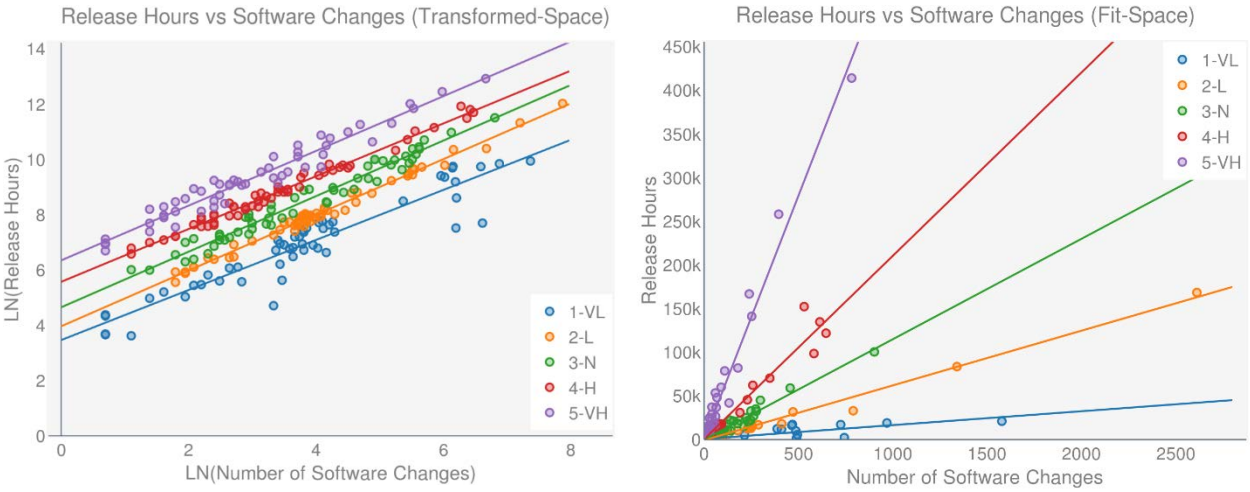
- Quintile Unit Cost (Hours per Software Change)
- Commodities (10)
- Change types (Enhanced, Maintenance, Cybersecurity)
- Number of Inter-Services Partners
- Acquisition Category (ACAT) Level
- Super Domains (RT, ENG, SUP, AIS)
- Sustainment Organization (17)
- Business models (Government, Contractor, Integrated)
- Location of Sustainment Organization (11)
- Sustainment Phase (MS-C LRP, MS-C FRP, O&S)/Time in Phase
- Number of Software variants
- Number of Platform variants

- Number of Users
- Number of Licenses

The third strategy divided the data into unit cost quintiles. The unit cost for each observation was ordered from smallest to largest and divided into five levels. As Figure 8 shows, this produced very tight variances. However, the challenge was to characterize each level so as to know which group an observation should belong.

The primary distinguishing characteristic was the *type* of software change beyond the high-level enhancement, maintenance and cybersecurity change types. As discussed in the Lessons Learned section in this paper, there was a variability in the definition of a software change. A detailed inquiry into each change type showed promising characteristics for describing each unit cost level. Unfortunately, the data was not collected consistently using the more detail change types.

Figure 8 shows five unit cost levels: 1-VL (Very Low), 2-L (Low), 3-N (Nominal), 4-H (High), and 5-VH (Very High). As the table in the figure shows, the central unit cost value, using either the mean or median, in each group increases non-linearly. The range of unit cost values in each group also increases non-linearly.



Release Hrs per Software Change	1-VL (Count: 49)	2-L (Count: 46)	3-N (Count: 41)	4-H (Count: 48)	5-VH (Count: 39)
Mean	26.8	60.8	138.3	413.4	2,725.2
Median	28.6	60.8	129.5	403.3	1,437.9
Min Value	2.0	46.9	78.2	212.1	718.0
Max Value	46.2	78.0	211.3	699.9	11,136.0

Figure 8. Unit Cost Grouping Levels

One of the investigations into finding a grouping strategy that reduced variation was to use a combination of the second and third strategies describe above. Categorical data was crossed-analyzed with the five-unit cost levels. The most promising categories were Commodities, Change Types, Inter-Service Number of Partners, and ACAT Level.

Counts of the number of observations in each category were cross-referenced with the

observation's unit cost level. The counts were normalized using the total number of observations in each category. Each category was examined for groupings of 50% or more. The groupings are highlighted in green in the tables below. The groupings were used to rank order the labels in the category.

The information in Table 1 shows commodities cross-referenced with unit cost levels. The table shows that Business and C4ISR have lower unit costs than Chem/Bio systems. Except that there is only one observation for Chem/Bio and its ranking is not certain. Chem/Bio has a lower unit cost than Missiles and Fire systems, Missiles and Fire systems have a lower unit cost than Test and Vehicle systems. Test systems having a high unit cost level was not expected.

Table 1. Release Unit Cost Level Count % by Commodity

Commodity	Cnt	1-VL	2-L	3-N	4-H	5-VH
Business	36	27.8%	11.1%	33.3%	19.4%	8.3%
C4ISR	131	22.9%	28.2%	14.5%	16.8%	17.6%
Chem/Bio	1			100.0%		
Missiles	7			57.1%	14.3%	28.6%
Fire	12	33.3%	16.7%	16.7%	33.3%	
Aviation	13		7.7%	15.4%	23.1%	53.8%
Space	12		8.3%	25.0%	25.0%	41.7%
Test	6	16.7%			50.0%	33.3%
Vehicles	6			33.3%	33.3%	33.3%

Table 2 shows the maintenance change types cross-referenced with unit cost levels. The three changes types were reported as percentages in the submitted data. An observation with a change type greater than 60% was labeled with that change type. In a few cases, the reported change type percentages were 50/50 and a mixed label was used.

Table 2. Release Unit Cost Level Count % by Change Type

Change Type	Cnt	1-VL	2-L	3-N	4-H	5-VH
Maintenance	120	26.7%	30.0%	11.7%	15.0%	16.7%
Cyber	12		25.0%	50.0%	25.0%	
Enh / Maint	30	6.7%	6.7%	40.0%	23.3%	23.3%
Enhancement	34	5.9%	5.9%	29.4%	20.6%	38.2%
Other	9	11.1%		22.2%	55.6%	11.1%
Enh / Other	3					100.0%

Table 2 shows that maintenance change types cost less than cyber-related changes. Enhancement change types are the most expensive. “Other” change types are changes that could not be categorized as maintenance, cyber or enhancement.

Table 3 shows the number of Inter-Service Partners cross-referenced with unit cost levels. Inter-Service partners are Marine Corp, Air Force, Navy, NATO, and others, e.g., National Guard, Foreign Military Sales. The table clearly shows that more partners on a system means higher unit cost.

Table 3. Release Unit Cost Level Count % by Inter-Service Partner

Inter-Service	Cnt	1-VL	2-L	3-N	4-H	5-VH
Army Only (1)	165	24.8%	23.0%	19.4%	18.2%	14.5%
2 partners	11	9.1%	36.4%	9.1%	27.3%	18.2%
3 partners	7			42.9%	14.3%	42.9%
4 partners	7	14.3%		14.3%	28.6%	42.9%
5 partners	33	6.1%	6.1%	24.2%	27.3%	36.4%

Table 4 show US DoD’s Acquisition Category (ACAT) levels cross-referenced with unit cost levels. There are three levels shown in the table 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. Table 4 shows that ACAT I & II programs have higher unit costs than ACAT III programs.

Table 4. Release Unit Cost Level Count % by ACAT Level

ACAT	Cnt	1-VL	2-L	3-N	4-H	5-VH
ACAT I	38	5.3%	15.8%	26.3%	18.4%	34.2%
ACAT II	41	31.7%	4.9%	9.8%	24.4%	29.3%
ACAT III	101	24.8%	31.7%	16.8%	13.9%	12.9%
Non PoR	2			100.0%		

6.1.3 CER Results

Ordinary Least Squares (OLS) regression was used to find CERs in the capability release data. The data was grouped by super domains, commodities, and maintenance change types (these categories had the best results). The dependent variable was total release hours (THrs). The most common independent variable was the total software changes (TSC). However, total system requirements (TReqts), total requirements implemented (TReqts_Imp), external interfaces modified (EI_Mod), software baseline (SWBase), and software change backlog (BL) were also used.

Table 5 shows the CER models, conditions for the OLS regression, the number of observations, the R^2, the standard error of the estimate (SEE), and the prediction level within 30% of the actuals (PRED(30)). Models with an R^2 above 0.7 are highlighted in red.

The strongest CERs in Table 5 have software changes (SC) as a common independent variable along with total system requirements, total requirements implemented, external interfaces modified, and software change backlog.

In a previous paper [15], CERs based on source lines of code (SLOC) as a size input did not perform very well. After collecting and analyzing more data, SLOC may not be a valid size measure for sustainment projects. This could be due to the large amounts of reused code reported from build to build.

Table 5. Cost Estimating Relationships

Model		Conditions	Obs	Adj R^2	SEE (Hrs)	PRED(30)
THrs = 463 * (TSC)$^{0.69}$		All data	329	0.36	48,385	17.3%
THrs = 341 * (TSC)$^{0.79}$		10% trimmed data	263	0.57	44,842	23.6%
AIS	THr = 242 * (TSC)$^{0.7341}$	10% trimmed & Super Domains (Categorical)	263	0.62	39,330	20.2%
ENG	THr = 386 * (TSC)$^{0.7341}$					
RT	THr = 736 * (TSC)$^{0.7341}$					
SUP	THr = 698 * (TSC)$^{0.7341}$					
Aviation	THrs = 1,452 * TSC$^{0.66}$	10% trimmed & Commodities (Categorical)	263	0.68*	40,886	23.2%
Business	THrs = 301 * TSC$^{0.66}$					
C4ISR	THrs = 364 * TSC$^{0.66}$					
Chem/Bio	THrs = 182 * TSC$^{0.66}$					
Fire	THrs = 1,531 * TSC$^{0.66}$					
Missiles	THrs = 1,114 * TSC$^{0.66}$					
Simulation	THrs = 577 * TSC$^{0.66}$					
Space	THrs = 1,005 * TSC$^{0.66}$					
Test	THrs = 1,742 * TSC$^{0.66}$					
Vehicles	THrs = 425 * TSC$^{0.66}$					
THrs = 608 * (TSC)$^{0.98}$ / (TReqts)$^{0.21}$		10% trimmed	32	0.84	32,228	25.0%
THrs = 330 * (TSC)$^{0.97}$ / (TReqts_Imp)$^{0.11}$		10% trimmed	65	0.74	63,904	23.1%
THrs = 296 * (TSC)$^{0.94}$ / (EI_Mod)$^{0.11}$		10% trimmed	41	0.74*	47,326	22.0%
THrs = 1,219 * (TSC)$^{0.75}$ / (SWBase)$^{0.04}$		10% trimmed	69	0.61*	36,567	26.1%
THrs = 757 * (TSC)$^{1.02}$ / (BL)$^{0.36}$		10% trimmed	45	0.74	81,719	15.6%
Cyber	THrs = 332 * TSC$^{0.79}$	10% trimmed & Change Type (Categorical)	263	0.59*	39,573	21.3%
Enhance	THrs = 531 * TSC$^{0.79}$					
Hybrid	THrs = 382 * TSC$^{0.79}$					
Maint	THrs = 281 * TSC$^{0.79}$					
Other	THrs = 284 * TSC$^{0.79}$					
THrs = 338 * TSC$^{0.77}$ * Enh%$^{0.10}$ * Maint%$^{0.02}$ * Cyber%$^{0.03}$ * Other%$^{0.01}$		10% trimmed & percentages of Change Types	263	0.60*	26,494	6.8%

***** High P-Values for one or more coefficients

Data models summarize, in a mathematical form, what the data means. The best-fit model in Table 5 with an R^2 of 0.84 indicates that the amount of work done, expressed as software changes, is influenced by the functional size of the system, expressed as total system requirements. Namely, the larger the system size (requirements) the less effort required per software change.

This can only occur if larger systems are more compartmentalized, or loosely coupled, than smaller systems. It has long been known that compartmentalized software is much easier to sustain than software that is a huge block of code. A software change in a compartmented system affects only the compartment thus reducing design and test effort. Larger software systems may be forced to compartmentalize functionality to decrease development complexity whereas smaller systems can relax the need for compartmentalization and still achieve the desired functionality. This best-fit CER implies that estimates should consider system size as a whole in relation to the changes to the system

6.2 SER Derivation for Capability Releases

This analysis investigated schedule estimating relationships (SERs) for WBS 1.0, Software Change Product, for capability releases. These SERs are utilized later in the acquisition lifecycle (post Milestone C) when software releases are periodically occurring.

Figure 9 shows the schedule data for WBS 1.0 based on 614 observations and measured in months. Some of the durations were less than three months indicating an emergency or patch release. Other release durations spanned years, indicating major or multiple rolled-up releases.

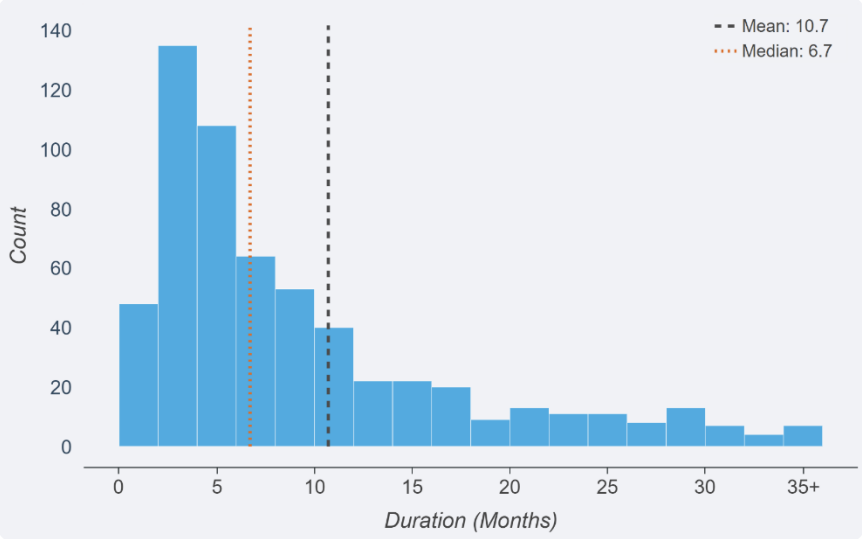


Figure 9. Release Duration Distribution

6.2.1 SER Ground Rules / Assumptions

The following ground rules or assumptions apply to each SER:

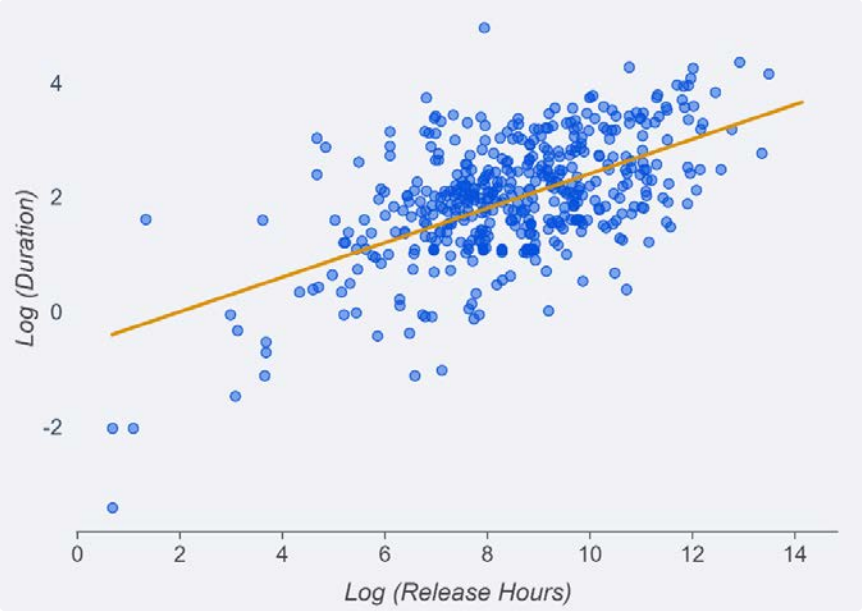
- The SER applies to WBS 1.0, Software Change Product, only for capability releases.
- The SER is based on effort (total release hours)
- Defense Business Systems were not included in this analysis.
- Observations were removed that did not report duration data

- When using trimmed data, the upper and lower 10% of data was removed based on Hours/Month
- Due to the non-normal distribution of the raw data, both dependent and independent variables were transformed using $\log_{10}$. Zero values were represented as 0.1.
- Ordinary Least Squares (OLS) regression was used to derive the SERs. The Minitab statistics tool and Python statistics libraries were used for OLS analysis.
- $R^2$was used to evaluate the SERs. This is the Coefficient of Determination and represents the percentage of total variation explained by the regression model. It provides a measure of how well actual values of effort are estimated by the CER model factors. R^2 ranges between 0 and 100% and is the same in log-space as in unit-space.

6.2.2 Methodology and Analysis Considered

The SER data has a high amount of variability. An OLS regression analysis was performed on capability release data.

Figure 10Figure 10 shows a log-scale scatter plot of 491 observations for all capability release data that had the independent variable, total release hours (THrs), and the dependent variable, months of duration. The plot shows a regression model of log(Duration) = 0.30*log((THrs) - 0.2578 with a large amount of variation and an R^2 of 31.4%.



Variable	Coef.	Std Err	T-Value	P-Value
Constant	-0.2578	0.076	-3.40	0.001
Log(THrs)	0.3001	0.020	15.01	0.000

Figure 10. All SER Data Scatter Plot

The above model is expressed as Duration = 0.55 * $THrs^{0.30}$ in standard number space. This model was compared to the COCOMO II Software Cost Estimation Model [16] equation for a nominal Time to Develop (TDEV): TDEV = 3.67 * $PM^{0.32}$. The exponents are very close. However, the constants differ because the sustainment SER uses total release hours as the independent variable and COCOMO II use Person-Months (PM).

In an effort to reduce variation and improve model fit, the upper and lower 10% of data was trimmed based on hours per month. Because software change counts were effective at predicting effort, they were included in multi-variable models. Table 6 shows the results of SER analysis.

Table 6. Initial Schedule Estimating Relationships

Model	Conditions	Obs	R^2
Duration(Months) = 0.55*(THrs)$^{0.3}$	All	491	31.4%
Duration(Months) = 0.15*(THrs)$^{0.45}$	10% Trimmed	393	44.6%
Duration(Months) = 0.45 * (THrs)$^{0.31}$ * (SC)$^{0.05}$	All	382	40.7%*
Duration(Months) = 0.15 * (THrs)$^{0.44}$ * (SC)$^{0.03}$	10% Trimmed	309	49.7%*

*** The coefficient for SC had a P-value > 0.1**

6.2.3 SER Results

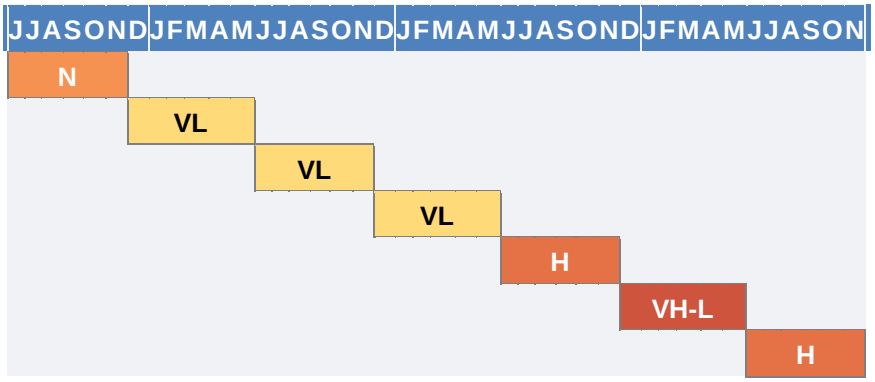
These poor results led to a search for an effective segmentation strategy that would reduce variation. The best segmentation strategy found to improve SERs was based on release duration categories. There are three categories of release duration: cyclic, sequential and concurrent.

1. Cyclic releases

Cyclic releases are releases with a fixed duration, e.g. three, four or five months, and are done sequentially with no overlap between releases. These releases have a regular pattern of start and end dates. When one release ends, the next release begins. Since each release is a fixed-duration, there is no need for an SER. Only the effort needs to be estimated based on the number software changes planned for the release. If the effort is a fixed level of effort (LOE), the number of software changes that can be implemented needs to be estimated based on the LOE.

Table 7 shows an example of a cyclic release for one system. At the top of the table are the first letter of the calendar month. The duration of the release is represented by a shaded box. The color of the box is a heat-map of the unit cost discussed in the CER section of this paper with the addition of splitting the highest unit cost, VH, into a lower, VH-L, and an upper, VH-U, half.

Table 7. Cyclic Release Example



For this system, the cycles are five months intervals. The different color bars shown that the unit cost varies. The SER for this system assumes fixed duration and LOE and the only estimate needed is the number of software changes that can be implemented under these constraints. The model is log(TSC) = 0.64*log(THrs) – 0.47 and is based on 11 observations (only 7 are shown in Table 7). The R^2 is 93.7%. The SER statistics are shown in Table 8.

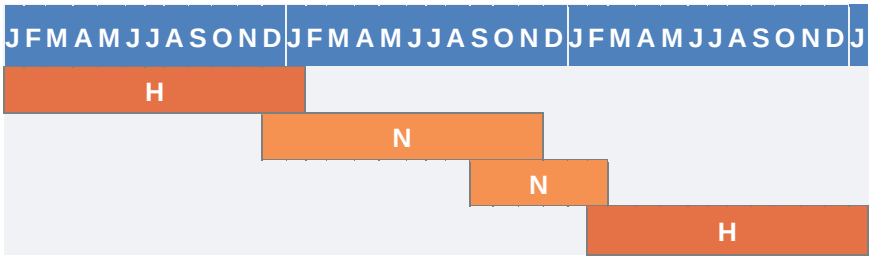
Table 8. Cyclic SER Statistics

Variable	Coef.	Std Err	T-Value	P-Value
Constant	-0.469	0.195	-2.41	0.04
Log(SC)	0.6402	0.055	11.56	0.000

2. Sequential releases

Table 9 shows an example of a sequential release for one system. These releases have variable durations and are done sequentially with some overlap between releases, i.e., the release start and end dates vary by release. Since each release duration varies, an SER is required to estimate a release's duration based on effort and the number of software changes to be implemented.

Table 9. Sequential Release Example



For this system, releases vary between 6 and 13 months. The different color bars show that unit cost varies. The SER for this system is log(Months) = 2.036*log(SC) – 0.157*log(Hrs) – 1.30 based on 4 observations. The R2 is 99%. The SER statistics are shown in Table 10.

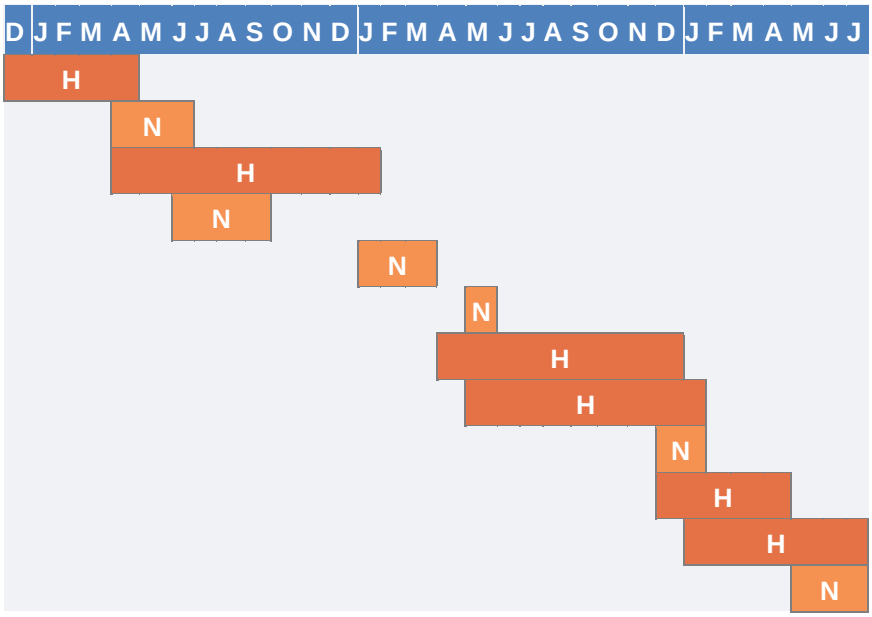
Table 10. Sequential SER Statistics

Variable	Coef.	Std Err	T-Value	P-Value
Constant	-1.300	0.492	-2.64	0.231
Log(Hrs)	-0.157	0.256	-0.61	0.650
Log(SC)	2.036	0.411	4.96	0.127

3. Concurrent releases

Table 11 shows an example of a concurrent release for one system. These releases have multiple concurrent developments with variable durations and overlapping start and end dates. Because each release varies in duration, an SER is required to estimate duration based on effort and the number of software changes.

Table 11. Concurrent Release Example



For this system, the duration of releases vary between 1 and 9 months. Again, the different colors show that unit cost varies among releases. The SER for this system is log(Months) = 0.996*log(Hrs) - 0.033*log(SC) - 3.437 based on 14 observations (only 12 are shown in Table 11). The R^2 is 98%. The SER statistics are shown in Table 12. In this SER, the coefficient for SC is statistically insignificant.

Table 12. Concurrent SER Statistics

Variable	Coef.	Std Err	T-Value	P-Value
Constant	-3.437	0.137	-25.06	0.000
Log(Hrs)	0.9962	0.0395	25.22	0.000
Log(SC)	0.0332	0.0201	1.65	0.119

6.3 Future CER/SER Research

The large amount of data in the repository has provided an exceptional opportunity to learn about Army software sustainment costs and DoD software sustainment costs in general. There are still many analyses that need to be done on this dataset to fully understand all its implications for cost estimation and the drivers of software sustainment costs.

An area that still needs investigating is the cost of cyber security in sustaining software systems. What are the factors that drive the cost to ensure the software is secure? Has there been a trend of increasing costs over time and, if so, what drives that increase? The Army is changing the cyber security framework from DoD's Information Assurance Certification and Accreditation Process (DIACAP) to a risk management framework. How will this change impact short-term and long-term sustainment costs?

The release data used in the release CER analysis presented here is based on start to finish cost and technical data. Releases can span one or more fiscal years. Yet actual release data is collected annually. Reexamining CERs using annual release data has shown the promise of better results. This will be pursued further.

The SER analysis focused on the type of release (Cyclic, Sequential, Concurrent) and was system-specific. Future SER analysis will examine which type of release is most cost effective, cyclic, sequential or concurrent. Another focus will be to expand the number of systems in each type of release to derive a more generalized model. Other strategies of grouping SER data also need to be explored.

The data repository contains a large amount of software license data. Additional analysis will investigate whether higher license costs are correlated to higher sustainment costs. The question of using COTS software to reduce cost also needs additional analysis, i.e., by using COTS, can it be demonstrated there is a cost savings.

There was mention earlier in this paper about fixed versus variable sustainment costs. The data provides the opportunity to study the impact of reducing investment (budget cuts) on the amount of delivered capability.

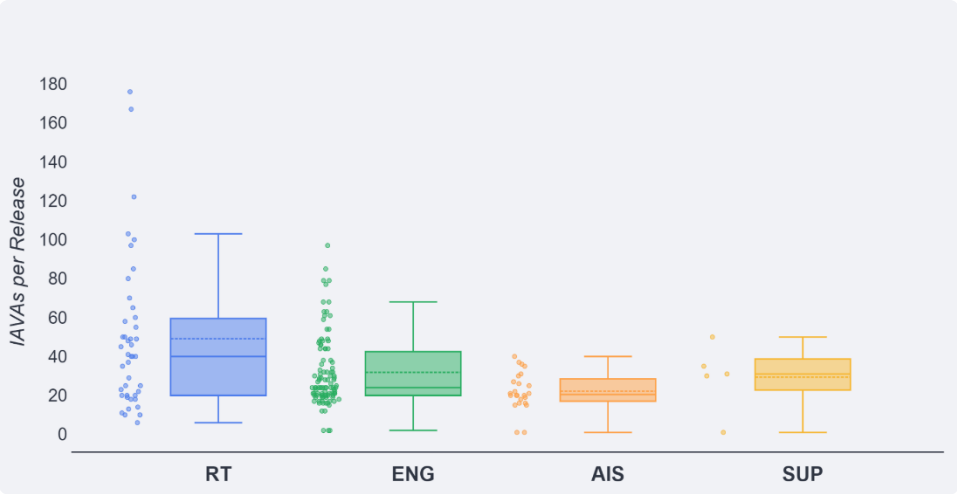
More data will be collected in FY18. This provides an opportunity to analyze smaller grouped observations for CERs as well as review the CERs presented in this paper. As more Agile data is collected, there is an opportunity to explore the differences between Agile versus traditional development approaches for efficiencies in sustainment.

7 IAVA Release Analysis

IAVA releases scan software for known vulnerabilities as different from capability releases that change software. This section discusses the analysis done to data on IAVA releases.

7.1 IAVAs per Release

The boxplot and statistics table in Figure 11 show the mean and median number of IAVAs for a release. This benchmark uses IAVA-only release data. The results are stratified by super domains to show differences in domains. The real-time (RT) domain has the highest number of IAVAs per release followed by engineering (ENG) and both Automated Information Systems (AIS) and Support (SUP) domains.



Super Domain	Count	Mean	Median
RT	48	45.875	40
ENG	112	30.875	24
AIS	39	21.744	20
SUP	7	21.286	30

Figure 11. IAVAs per Release

Caution should be used with this data, however, as the number of IAVAs per release may be influence by the number and frequency of releases.

7.2 IAVA CERs

This analysis investigates the cost estimating relationships (CER) for WBS 1.0, Software Change Product, for IAVA-only releases. The data is the number of IAVAs, the independent variable, with either release hours or release cost being the dependent variable. A CER could be used with the mean or median number of IAVAs per release to derive a total release cost.

Figure 12 shows two scatter plots for 115 observations of the number of IAVAs versus release hours on the left and release cost on the right. The plots are in log-scale. The CER represented by the trend line through each plot has an R^2 of 11.7%. Unfortunately, the data has too much variance to support a CER.

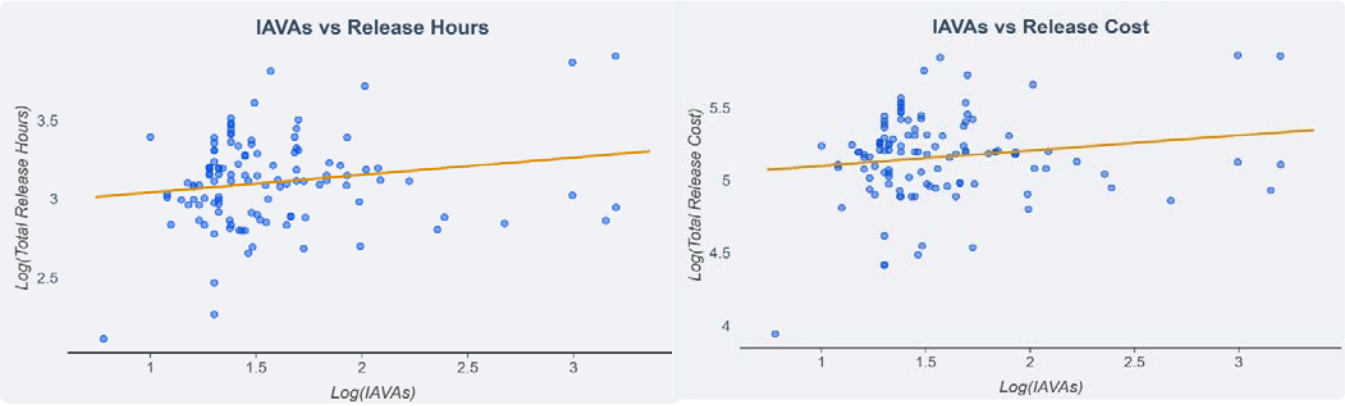


Figure 12. All IAVA CER Data

A strategy used in the capability release CER analysis to categorize data based on unit cost was attempted with IAVA data. The unit cost is based on hours per IAVA. The data was ordered from smallest to largest unit cost and divided into quintiles, five levels of unit cost: VL, L, N, H, and VH.

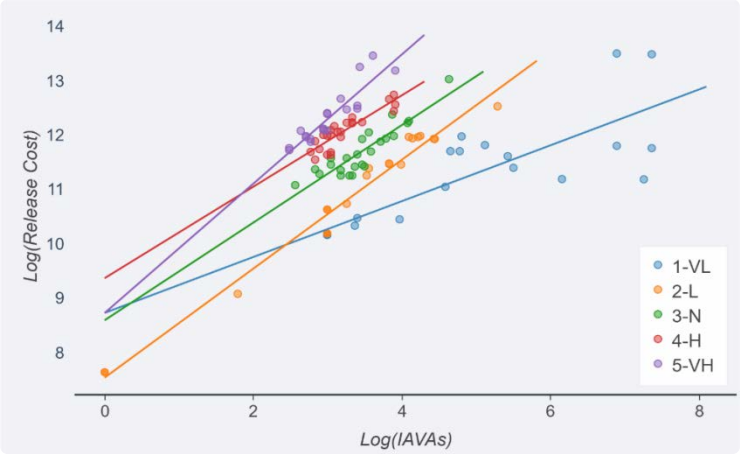
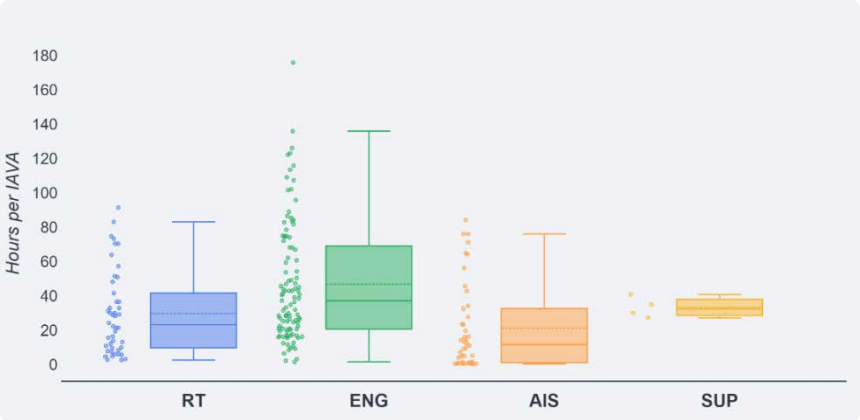


Figure 13. Scatter Plot of IAVA Unit Cost Levels

Figure 13 shows the results of a scatter plot of the different unit cost groups. Compared to the same plot for capability releases (see Figure 8), these results show that the number of IAVAs is not related to the cost of an IAVA release.

In the absence of a valid CER, hours per IAVA was analyzed. Figure 14 shows the mean and median hours per IAVA for each super domain. The boxplot compares the medians side by side and shows the variation in each super domain group.

Cost per IAVA can be used to bound the number of IAVAs a program can expect to do given a fixed budget.



Super Domain	Count	Mean	Median
RT	50	64.72	40.5
ENG	113	42.48	25
AIS	56	497.84	30
SUP	6	19.67	15.5

Figure 14. Hours per IAVA

8 Benchmarks

The purpose of this analysis is to provide benchmarks for all WBS elements. These benchmarks are useful for providing mean or median statistics and can be useful for performing a sanity check on software sustainment cost estimates.

8.1 Ground Rules/Assumptions

The following ground rules or assumptions accompany each benchmark:

- Some benchmarks are grouped by super domains: Real-Time (RT), Engineering (ENG), Support (SUP), and Automated Information Systems (AIS). Super domains are used to group similar software complexity and capability.
- A year is a fiscal year from October 1 to September 30.
- The terms Cost, Hours and FTE includes both government and contractor data.
- All cost data were normalized to Base Year FY18.

8.2 Methodology

This section addresses two types of benchmarks, annual cost execution according to the WBS and cost/effort per software change.

8.2.1 Annual Cost Benchmarks

The data collected spans across multiple years for a given system with not all WBS elements being required in every year. In order to arrive at a representative depiction of the spending allocation for a program, it was necessary to average the years reported by each individual system. This is more effective that just an average across the WBS for all systems because it more accurately captures the intra-system relationships in WBS execution.

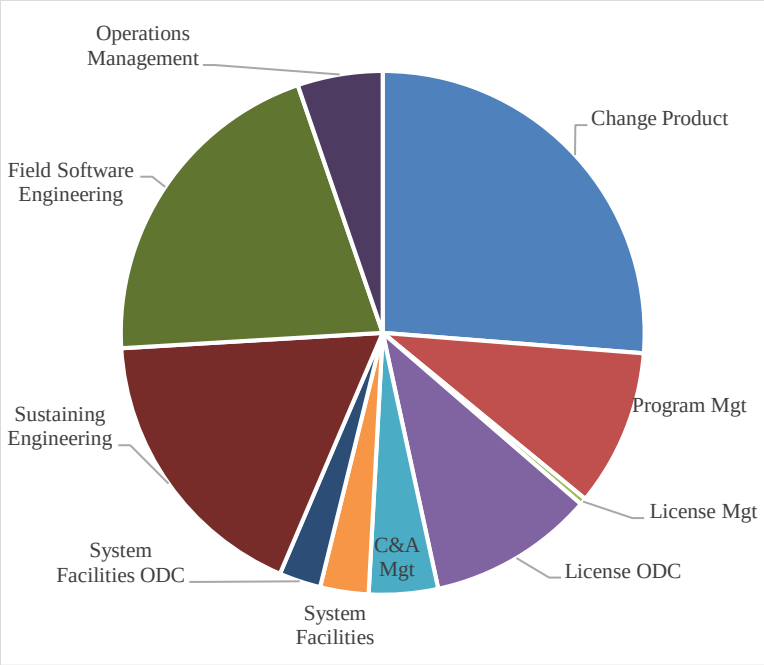
8.2.2 Productivity (Hours per Software Change) Benchmarks

Software changes were the most commonly reported size measure and is the measure chosen to produce productivity benchmarks. Hours per software change, is a measure that can be quickly used to develop rough order of magnitude estimates for sustainment releases. Due to the varying definitions of a Software Change, the distribution of hours per software change took the shape of a right skewed lognormal distribution. To address the nature of the skewed distribution the dataset was trimmed using plus or minus 1.5 *x* the Inner Quartile Range (IQR) for each Super Domain.

8.3 Distributions

8.3.1 Allocation of Costs Across the Software Sustainment WBS

One of the initial charts that was developed showed the allocation of costs across all WBS elements and two Other than Direct Costs (ODC) for licenses and system facilities. This addressed an enterprise level information requirement concerning the allocation of executed costs for various sustainment activities and products.



Change Product	**$369,869,728**
Program Mgt	$136,508,861
License Mgt	$5,780,421
License ODC	$144,535,446
C&A Mgt	$60,051,407
System Facilities	$42,064,243
System Facilities ODC	$36,258,937
Sustaining Engineering	$248,703,767
Field SW Engineering	$291,449,365
Operations Management	$74,056,525
Total	$1,409,278,700

Figure 15. Allocation of Costs by WBS Element

Figure 15 is based on 190 averaged annual WBS cost observations. It shows the cost sensitivity between all of the Sustainment WBS elements. The largest cost is WBS 1.0. Software Change Product. The smallest cost was WBS 3.0, Software Licenses, although License ODC ranked 4th in cost. The second and third highest costs were WBS 7.0, Field Software Engineering, and WBS 6.0, Sustaining Engineering. Figure 1 describes each WBS element.

Each system reported up to three years of annual FY costs. The costs for each system was averaged. Figure 15 shows the sum of the averaged costs for each WBS element. This represents a portfolio view of average annual execution.

8.3.2 Distribution of Annual Cost

In addition to examining the cost percentages for each WBS, the analysis also looked at the distribution of costs across the set of systems. This can be used for early-on cost estimating when little detail is known about the program, or as a cross-check for software sustainment estimates developed by other methods. For each WBS element, the stacked bar in Figure 16 shows the allocation of all WBS costs across a super domain.

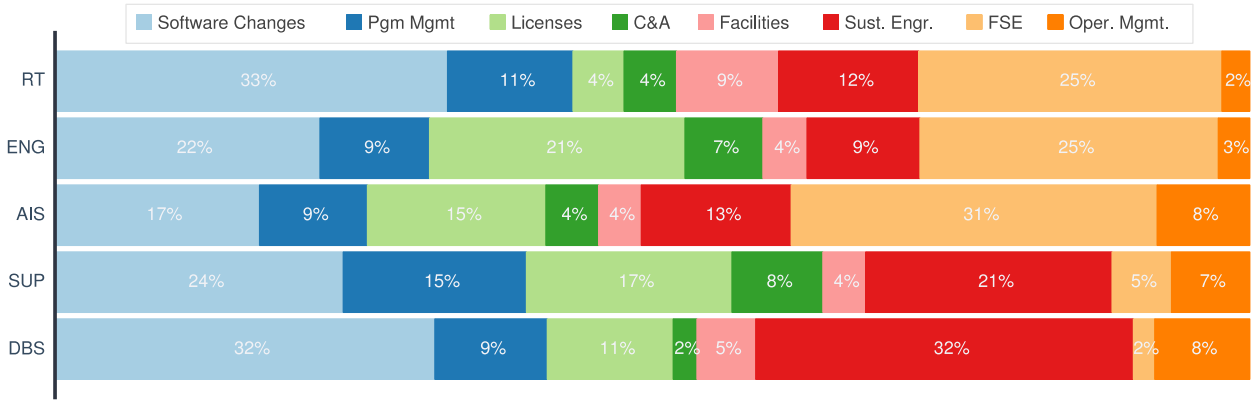


Figure 16. Allocation of WBS Costs by Super Domain

The methodology for generating the stacked bar chart above includes the following:

- Annual cost includes government and contractor costs.
- No outliers were removed.
- Fields that were left blank or zero were not used.
- Each program may have reported multiple years of data. Each system was averaged to represent a single observation.

8.4 PDSS vs PPSS Sustainment Phases

The total annual cost across WBS elements was divided between PDSS and PPSS sustainment phases. Recall from the section on System Age, that PDSS is between initial deployment and the end of production. PPSS are systems being sustained after the production line has been terminated.

Figure 17 shows that while the cost allocations do not change dramatically from PDSS to PPSS, there is a discernible shift in Software Change Product (SW Change Product) and Sustaining Engineering (SE). PDSS has a higher SW Change Product cost allocation across super domains due to enhancement change types. This is shown in Figure 18.

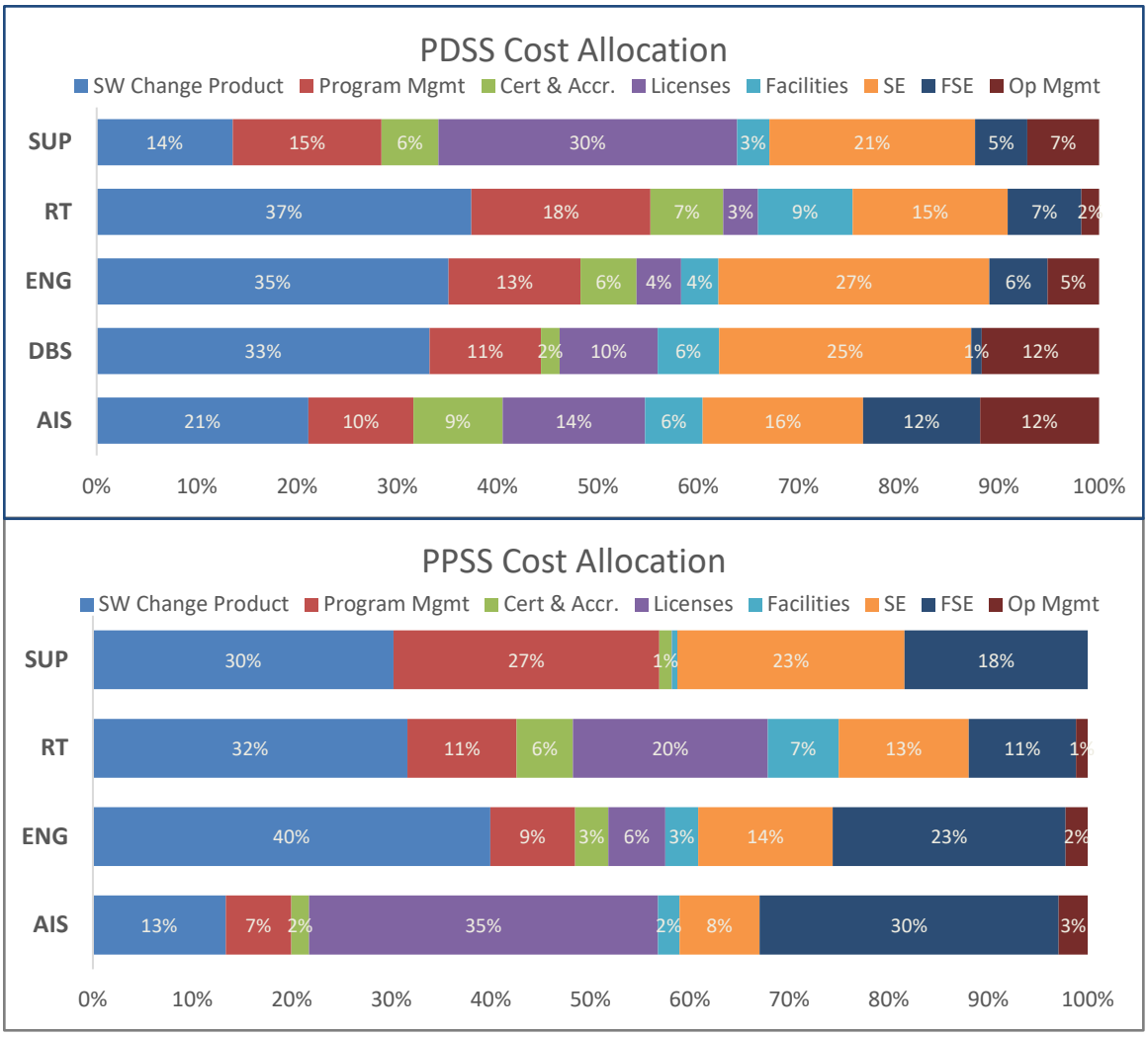


Figure 17. PDSS vs PPSS Cost Allocation

Figure 18 shows the cost allocation for WBS 1.0, Software Change Product, by software change type and sustainment phase. The most noticeable shift in cost is with enhancements change types going from PDSS to PPSS.

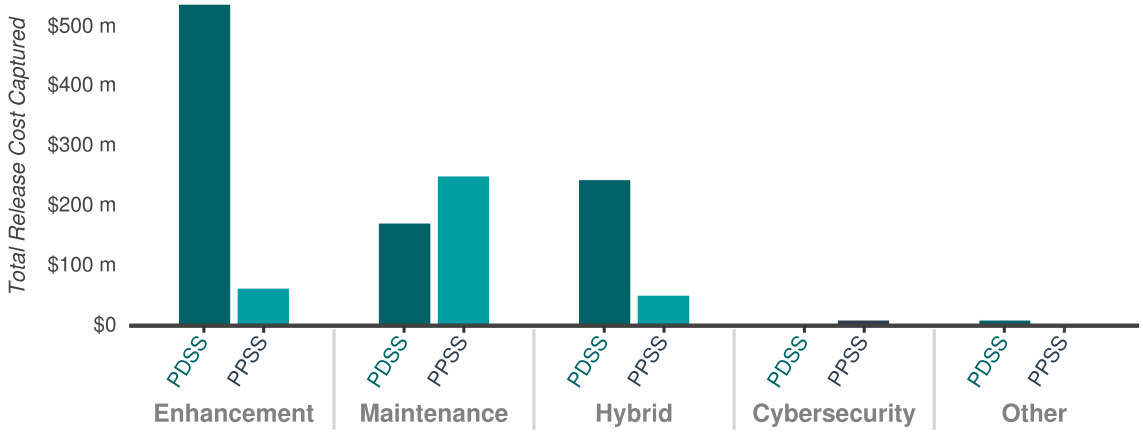
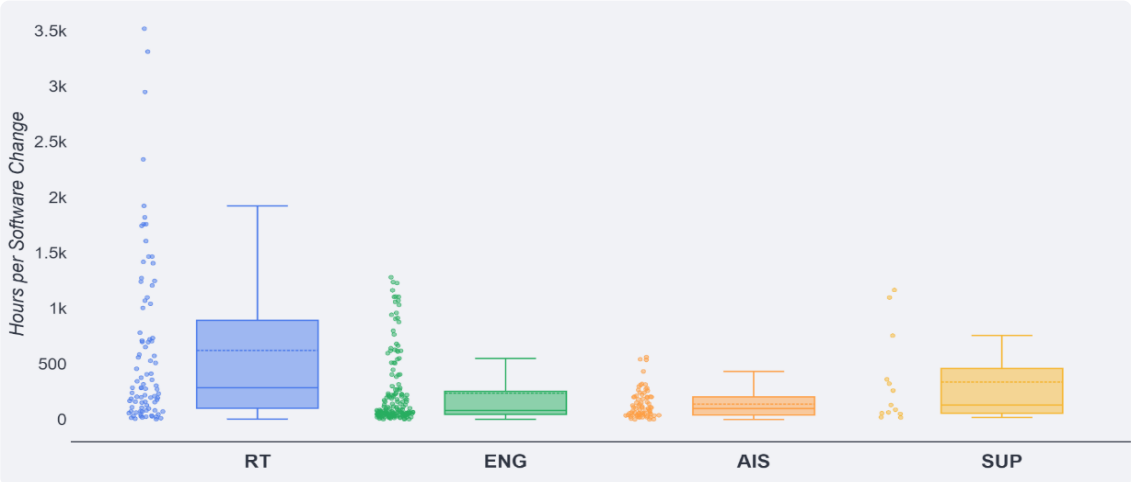


Figure 18. Release Costs by Change Type and Sustainment Phase

8.5 Hours per Software Change and Software Changes per Release

Figure 19 shows how many hours it takes to implement one software change during sustainment for each super domain. Both a histogram and a table of statistics show the distribution of the data. Some of the distributions in Figure 19 show extreme values which influence the mean value. The median value for each histogram may be a more valid indicator of central tendency. The total hours shown includes both Government and Contractor hours. The data is from capability releases.

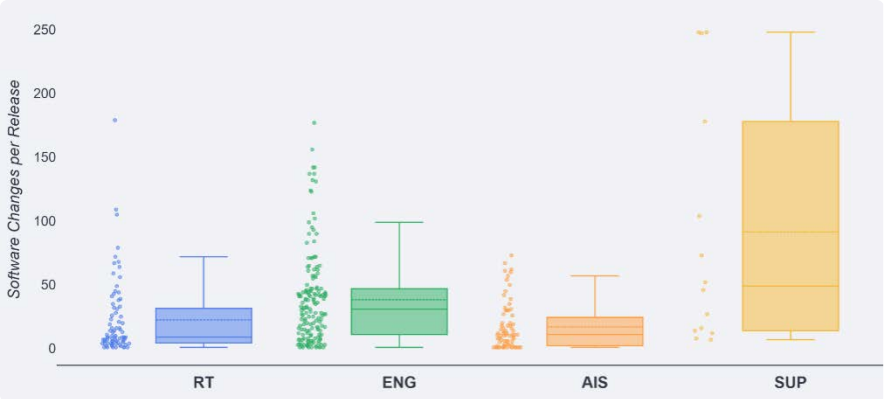


Super Domain	Count	Mean	Median
RT	101	1481.6	407.4
ENG	186	629.8	102.5
AIS	79	244.9	111.5
SUP	14	490.7	197.0

Figure 19. Hours per Software Change by Super Domain

Figure 20 shows the number of software changes in a release by super domain. The data shows a lot of variability within each domain. This may be caused by the different definitions used to describe what constitutes a change. This issue is discussed in the section on Lessons Learned.

Number of software changes per release can be used to size future releases when program specific data is unknown. The resulting size can be used with the cost benchmark in Figure 20 or with a CER to create a rough order of magnitude estimates.

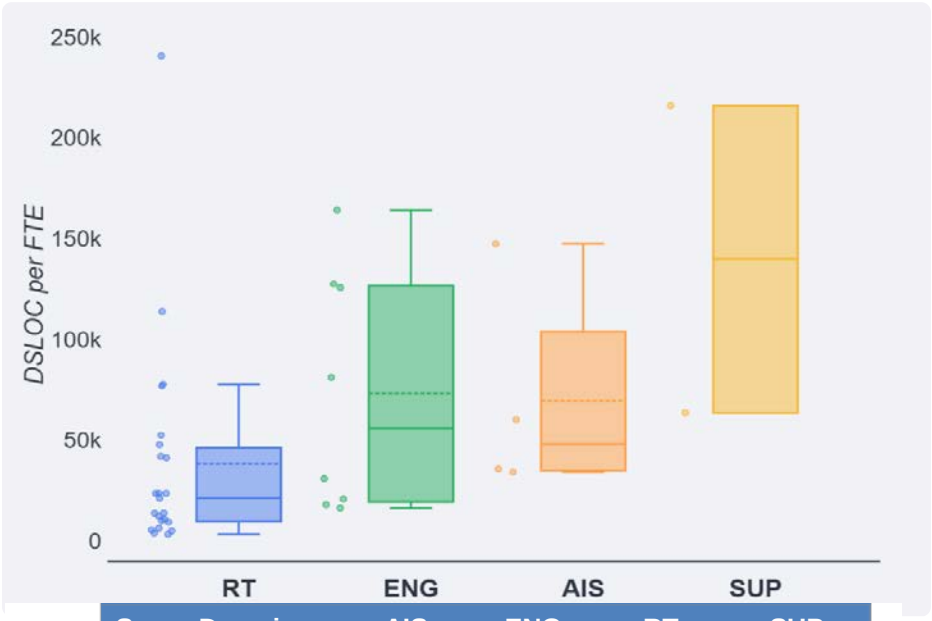


Super Domain	Count	Mean	Median
RT	80	22.5	9.0
ENG	161	38.3	31.0
AIS	75	17.0	11.0
SUP	14	91.4	49.0

Figure 20.Number of Software Changes per Release

8.6 Delivered Source Lines of Code (DSLOC) per Full Time Equivalent (FTE)

Some sustainment budgets are estimated based on the size of the software code base and the number of FTEs required to maintain that size. In this estimate, the larger the code base, the more FTEs are required.



Super Domain	AIS	ENG	RT	SUP
Count	4	8	23	2
Mean	69,492	73,166	38,306	139,953
Median	48,094	56,181	21,221	139,953

Figure 21. DSLOC per FTE

DSLOC counts all code types (new, modified, reused, and autogenerated) equally. Whereas ESLOC counts treat different code types proportionally. The earliest baseline size reported was used to represent DSLOC. FTE counts were derived by including the following WBS Elements: SW Change Product (1.0), Project Management (2.0), Sustaining Engineering (5.0), and Certification and Accreditation (4.0). FTE counts include Government and Contractor effort. FTEs were derived by using labor hours per man-year and labor rate reported for both Government and Contractor for each program.

Figure 21 show DSLOC per FTE for each super domain. The data has a non-normal distribution making the median value the most representative. The Real-Time (RT) domain has the lowest DSLOC per FTE ratio of about 21,000 to 1. This means that more FTEs are required per 1,000 DSLOC than the other domains. Intuitively this makes sense as programming languages typically used in Real-Time systems tend to be at a lower level and more verbose. The Automated Information Systems (AIS) and Engineering (ENG) have a DSLOC per FTE ratio of about 50,000 to 1. There is not enough data for the Support (SUP) domain to draw a conclusion.

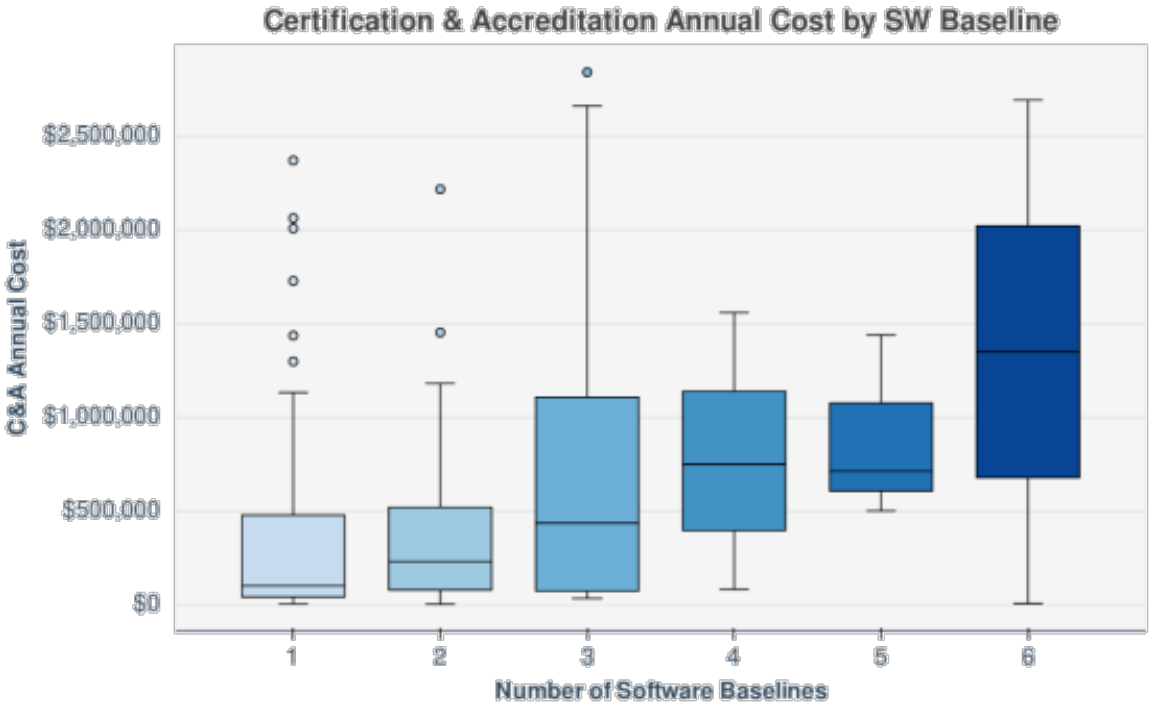
9 Cost Impact of Software Baselines

Outside of standard cost estimating CERs and benchmarks, the consolidated dataset can be used to provide insight into management related questions. The Army LCMCs wanted to know the cost impact of sustaining multiple concurrent software baselines of fielded systems.

Intuitively multiple baselines would impact cost in three main areas of the WBS. WBS 1.0, Software Changes, WBS 2.0, Project Management, and WBS 4.0, Certification and Accreditation.

For WBS 1.0, corrected defects or software changes have to be replicated on each baseline of the software and tested to ensure changes work as intended. For WBS 2.0, sustaining multiple baselines increases the number of configurations, deployments, and changes tied to each baseline. WBS 4.0 is affected through the requirement of certifying each baseline and release through the Risk Management Framework.

Two analytical approaches were taken to evaluate the impact of sustaining multiple baselines. The first approach shown in Figure 22 groups the systems by the number of baselines sustained and compares the distribution of annual Certification and Accreditation cost across the systems. While there is variation within the dataset, there is an increasing trend in costs as the number of fielded baselines increases.



# of Baselines	1	2	3	4	5	6
Count	50	34	16	10	5	4
Median Cost	$104,942	$234,411	$438,457	$752,890	$716,197	$1,353,313

Figure 22. Cost Impact on Software Baselines

The second approach uses Partial Dependence applied to a trained Random Forest Regression model. Partial Dependence is used to provide insight and understanding into machine learning algorithms by providing the marginal effect of a given variable on the predicted outcome or, in this case, the annual cost. Partial Dependence can be interpreted similarly to linear regression coefficients. It seeks to answer the question of what is the impact of a given dependent variable on the independent variable if all other variables are held constant. In this implementation, this can be translated to “What is the impact of having multiple software baselines if all other variables in a given system are exactly the same”.

Random Forest Regression was used as the base estimator for the Partial Dependence analysis. The model was fit using average annual cost of Software Changes, Project Management, and Certification and Accreditation as the independent variables. Super domain, number of releases, number of software changes, number of IAVAs, ACAT level, system age, and software baselines served as the dependent variables. Once trained, the Random Forest model performed at 97.6% accuracy on unseen data.

Figure 23 show a partial dependence plot using the random forest learning method.

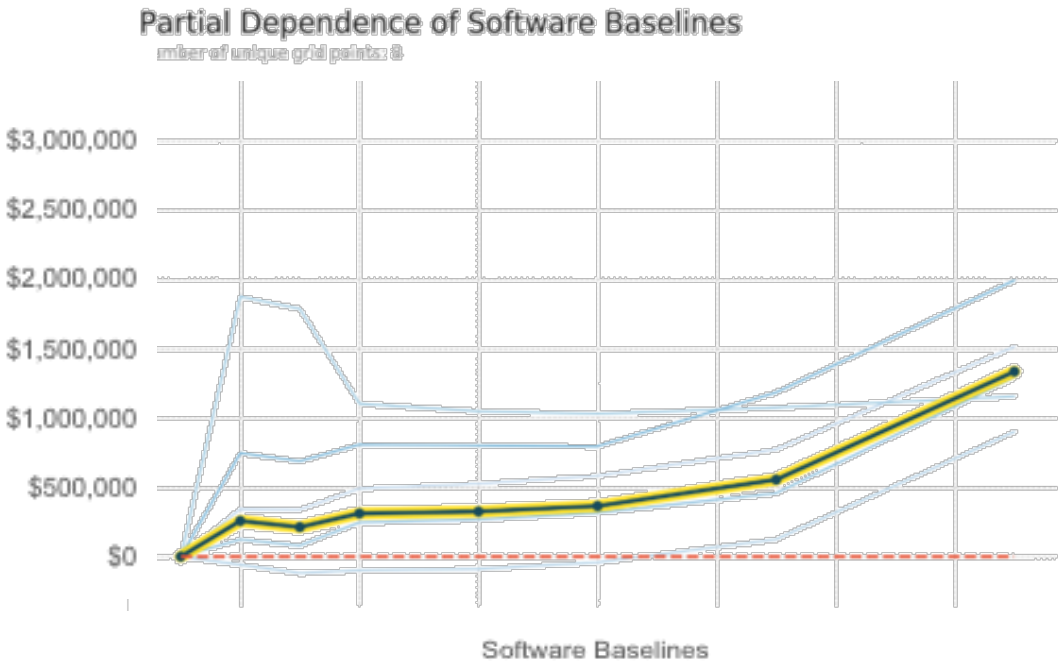


Figure 23. Partial Dependence Plot

The yellow line in Figure 23 represents the average of the predicted cost for each record when changing the number of baselines. The blue lines represent 5 quantiles of the predictions, meaning there are some data points that are more affected than others by having multiple baselines. Below 0 does not imply a negative cost, it means that the record's predicted cost was less than the average cost at that number of baselines

On average it is ~$250K to go from supporting 1 baseline to supporting 2 baselines. Once a program is managing multiple baselines, the cost impact grows steadily until the number of baselines is greater than 9.

10 Cost Risk/Uncertainty

10.1 Data Uncertainty

The data submitted by Army systems during data collection was evaluated for completeness and reasonableness. If the data was not available or was 50% outside the cost boundaries as set by the labor hours per person-year and the annual burdened cost rate, it was labeled as being red, as shown in Figure 24. If the data was estimated and within cost boundaries, it was labeled as being "yellow." If the data was actual execution data and within cost boundaries, it was given a "green" label.

Both annual cost data for each WBS element and release-level data were evaluated. A representative sample of the 1,040 releases is shown in Figure 24. Each system had one to twelve releases, depending on the release rhythm and number of cyber-security releases.

				Initial Release Overall		Detailed Release Assessment							
PEO	SEC	System	Release	CER Usability	SER Usability	Size: Requirements	Size: External Interfaces	Size: SLOC	Size: Non-SLOC	Size: SW Changes	IAVAs	Effort (WBS-1)	Schedule (WBS-1&2)
(PEO 4)	SEC 3	System 1	Release 1	Y	Y	G	G	G	N/A	G	G	Y	G
(PEO 4)	SEC 3	System 1	Release 2	Y	Y	G	G	G	N/A	G	G	Y	G
(PEO 4)	SEC 3	System 2	Release 1	G	G	R	R	Y	N/A	G	G	G	G
(PEO 4)	SEC 3	System 3	Release 1	G	G	G	N/A	G	N/A	G	N/A	G	R
(PEO 4)	SEC 3	System 4	Release 1	G	R	R	N/A	G	N/A	G	N/A	G	R
(PEO 4)	SEC 3	System 4	Release 2	G	R	R	N/A	G	N/A	G	N/A	G	R
(PEO 1)	SEC 2	System 5	Release 1	Y	R	R	R	G	R	G	N/A	Y	R
PEO 1	(SEC 2)	System 5	Release 2	G	G	R	R	G	N/A	R	R	G	G
PEO 1	(SEC 2)	System 5	Release 3	G	G	R	R	G	N/A	R	R	G	G
PEO 1	(SEC 2)	System 5	Release 4	G	G	R	R	G	N/A	R	R	G	G
PEO 1	(SEC 2)	System 5	Release 5	G	G	R	R	G	N/A	R	R	G	G
PEO 1	(SEC 2)	System 5	Release 6	G	G	R	R	G	N/A	R	R	G	G
PEO 1	(SEC 2)	System 6	Release 1	R	R	G	G	Y	N/A	G	R	R	G
PEO 1	(SEC 2)	System 6	Release 2	R	R	G	G	Y	N/A	G	R	R	G
PEO 1	(SEC 2)	System 6	Release 3	R	R	G	G	Y	N/A	Y	R	R	G
PEO 1	(SEC 2)	System 7	Release 1	R	R	G	Y	G	N/A	G	R	O	R
PEO 1	(SEC 2)	System 7	Release 2	R	R	G	Y	G	N/A	G	R	O	R
PEO 1	(SEC 2)	System 8	Release 1	G	G	G	G	G	N/A	G	G	G	G
PEO 1	(SEC 2)	System 9	Release 1	R	R	Y	G	G	N/A	Y	N/A	R	R
PEO 1	(SEC 2)	System 9	Release 2	R	R	Y	G	R	N/A	N/A	N/A	R	G
PEO 1	(SEC 2)	System 9	Release 3	R	R	Y	G	G	N/A	R	N/A	R	G

Figure 24. Release Evaluation Sample

In addition to rating individual technical, effort/cost, and schedule data, the release-level evaluation also indicates the degree to which each release is useable for developing either CERs or SERs. This evaluation is based on whether one or more size measures are available, along with the availability of effort/cost or schedule information. In order to have an acceptable rating in CER or SER usability, a system needed at least one size measure, and a measure of effort/cost and schedule.

The implication of Figure 24 is significant. If programs do not collect and use historical data to make predictions of their future costs, their estimates are assumed to possess some level uncertainty. Programs that fail to adequately address risk or uncertainty in their cost estimates end up being programs that typically overrun their budgets and schedules, as well as fail to meet their technical objectives.

10.2 Estimation Uncertainty

An important characteristic of a credible cost estimate is a provision for program uncertainties, according to the GAO Cost Estimating and Assessment Guide. The Guide states that," Uncertainties should be identified and allowance developed to cover the cost effect… Known costs should be included and unknown costs should be allowed for." [14] The Guide also states that each WBS element should be assessed for its level of cost, technical and schedule risk.

As part of the DASA-CE cost estimation initiative, a simple prototype Cost-Risk Uncertainty Determination (CRED) model was developed based on the information or knowledge gap between *what should be known* at the point in time of the estimate and *what actually is known*. Figure 24 is an example of what should be known at the time of data collection and what is actually known. The number of red labels is troubling.

The CRED model identifies major factors that materially impact a software sustainment cost estimate at any point in a system's life cycle. Its objective is to improve the credibility of a software sustainment cost estimate by:

1. Identifying, characterizing and accounting for different cost performance factors (e.g., software product attributes, management factors, external program activities) and human estimation biases (e.g., anchoring, optimism bias, etc.) that may be sources of risk/ uncertainty that can result in creating material impacts on a software sustainment cost estimate.

2. Making visible the "knowledge gap" (if any) between what should be known (as defined by Army or DOD policy, regulation, or accepted best practice) and what is known about the system being maintained (or is to eventually enter sustainment) - that can be used to calculate a range of uncertainty associated with the estimate.
3. Completely documenting the key program issues and related performance factors that may influence the cost estimate and why.

Four general attribute categories of risk/uncertainty factors have been defined that are known to materially influence the system's complexity or efficiency/effectiveness of the system's sustainment process, and therefore, the eventual cost of maintaining the software system.

1. The first involves technical characteristics of the system itself like the number of external interfaces the system has, the execution timing constraints the system must meet, COTS product incorporation, critical technology usage, data rights, and so forth.
2. The second involves the program/project management factors that can influence how the efficient/effective a software sustainment effort is likely to be, e.g., its technical process capability, technical personnel capability, facilities and infrastructure support available, etc.
3. The third involves assess the external factors that may impact a systems sustainment effort, such as policy mandates such as security requirements or budget reductions.
4. The final attribute category involves the software sustainment environmental cost factors related to how well software sustainment tasks and activities are performed.

The total number of attribute categories and risk/uncertainty factors has been kept deliberately to a small number in order to encourage the CRED model's use by cost analysts and program managers alike.

The CRED model is meant to enable risk assessments of programs that allow cost estimators to generate risk profiles of Army programs. It can be used in conjunction with risk parameters associated with CERs shown earlier, and is designed to be used by programs throughout life cycle to proactively minimize software sustainment risk and understand factors that drive software sustainment costs.

The CRED model is currently being refined. Effort is being currently expended to validate the CRED model as more actual versus estimated software sustainment cost information is gathered and analyzed.

11 Lessons Learned

The DASA-SE software sustainment initiative has highlighted many important lessons to be learned in regard to Army software sustainment practice and how to estimate software sustainment costs. The first lesson is that there is great variability in what is called a software change. Within WBS 1.0, Software Change Product, the effort associated with software releases are captured. A software release is sized using the count of the number of software changes. A software change describes a change where source code or scripts are altered whether it be added, deleted or modified. Since there is significant variability across the programs in the definition of a software change, a more in-depth analysis was required to understand the costs of different types of software changes. The three different change types identified are broken down into sub-categories as follows:

- Enhancements
  - New capability: ECPs, new requirements
  - Redesign / rewrite: 100% new code, new architecture
- Maintenance
  - Defect repair: bug fixes, PTR fixes
  - Reconfiguration: threat loads, EW parameters
  - Rehost: migration from Windows to Linux
  - Testing: interoperability testing
  - Update: weapon tables, switch configurations, Operating System
  - Update, Defect repair (see above)
  - Upgrade: upgrading the application version to the next higher version
- Cyber
  - Vulnerabilities: enhance security posture not resolved
  - through IAVA process

More specifically identifying the type of software change has proved useful for reducing the cost variability.

There was also variability in what was considered the *start* and *end* of a release. Different development methodologies create different strategies for when work on software changes occurs for a release. Some methodologies may have frequent and continuous approval of changes and others may convene a Configuration Control Board at longer intervals to approval multiple changes. Examples of release start-end definitions are:

> *"Start of the analysis and design phase for the CCB approved ECPs. End date is shown as the conclusion of (acceptance test)."*
>
> *"Start date is start of contractor development. End date is date of release of software*

from (the LCMC)."

"Release took 8 months from requirements to V&V."

"Kick-off meeting to Software Release CCB."

"The start date is the beginning of the FY13. During this period the program was operating using a Rapid Development Cycle where solutions would be engineered and implemented in response to urgent needs from theater. Government customer would give approval during weekly CCB meetings. There is no single date to represent the approval of all of the requirements."

More focus should be given to defining what constitutes the start and end of a release. Generally, work cannot start with approval and it ends when the work is validated and ready for delivery to the users.

An additional lesson learned concerns the challenges of using the CERs, SERs and benchmarks to make estimates for each of the WBS elements.

- WBS 1.0, Software Change Product, uses IAVAs and software changes as size counts. The challenge is the inconsistent definition of the size measure and effort is generally not tracked by release. Effort is tracked annually for a release.
- WBS 2.0, Project Management, includes activities for project execution. However these roles and responsibilities may be spread across the WBS. Some of the execution activities are paid by overhead.
- WBS 3.0, Software Licenses, involves managing licenses for government and contractor on the project and enterprise-wide licenses. These licenses are paid from multiple sources and are often not tracked or associated with a particular program that may be using them.
- WBS 4.0, Certification and Accreditation, deals with activities involved with DIACAP, RMF and STIGs. There are different types of C&As. It is difficult to track preparation versus the certification versus post-certification repairs.
- WBS 5.0, System Facilities, includes laboratory infrastructure and management. The challenge is facilities are paid by various sources and some resources are inherited.
- WBS 6.0, Sustaining Engineering, focuses on help desk, hosting and delivery/test support. The cost category is generally misunderstood and the activities are inconsistently reported.
- WBS 7.0, Field Software Engineering, includes field maintenance, installation and troubleshooting. The challenge is the difficulty in estimating these activities because they are often shared among multiple programs.
- WBS 8.0, Operational Management, addresses enterprise and business management. The estimation challenge is this activity is generally treated as overhead and spread across multiple programs.

A further lesson learned is that it often takes multiple iterations with the data provider to clean up the data provided. This may be caused by a misunderstanding of what data is being requested or a lack of complete data

- It is worth the effort to clean up the submitted data

Another lesson learned is that data for some of the WBS elements was reported "unavailable" because the work was funded by different organizations, because costs were applicable to multiple systems, or because data was not tracked at lower WBS levels.

Release data was collected for a full release – yet it is tracked annually.

- Future analysis will evaluate annual release data and aggregate release data that spans multiple fiscal years

12 Conclusions and Next Steps

As documented throughout this paper, the Army DASA-CE software sustainment initiative has succeed 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 what it was being used for, 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.

There now exists an Army software sustainment work breakdown structure (WBS) that is being promulgated throughout the Army, The WBS has created standard definitions of the different classes of software sustainment activities that Army programs are performing, and allows these activities for to be quantitatively measured. It also permits software sustainment funding streams to be associated with work performed down to the software sustainment release level.

There is also an Army Software Sustainment Data Questionnaire which is being used to collect system context-information, annual cost and effort data, software release data, and data on software licenses. The questionnaire serves as a basis for OSD's Software Resources Data Report for Maintenance (SRDR-M).

In addition, there now exists a comprehensive DoD software sustainment database which has significantly enhanced the types and kinds of sustainment data available. 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 next steps are to implement periodic data collection. Army G4 (Logistics) has undertaken annual data collection starting with FY18 PPSS actual execution data. This data will be utilized to inform future sustainment funding decisions. The Army OSMIS has been identified as the data repository for data collection and storage.

OSD has also started an initiative to collect software sustainment data. This initiative participated in the formulation of their data collection mechanism, the Software Resources Data Reporting for Maintenance (SRDR-M). The SRDR-M closely aligns to the DASA-CE SWS WBS and data requirements. Moving forward, the SRDR-M will be utilized to collect SWS data from Army programs and perform analysis.

Acronyms

ACAT	Acquisition Category
AIS	Automated Information System super domain
BL	Software Change Backlog
BY	Base Year
C&A	Certification and Accreditation
C5ISR	Command, Control, Communications, Computers, Cyber,hh Intelligence, Surveillance, and Reconnaissance
CADE	Defense Cost and Resource Center
CER	Cost Estimating Relationship
Chem/Bio	Chemical/Biological
COTS	Commercial Off The Shelf
CRED	Uncertainty Estimation Determination
CSCI	Computer Software Configuration Item
Cyber%	Percent of the release that is Cybersecurity updates
DASA-CE	Deputy Assistant to the Secretary of the Army for Cost and Economics
DBS	Defense Business System commodity
DIACAP	DoD Information Assurance Certification and Accreditation Process
DISA	Defense Information Systems Agency
DoD	US Department of Defense
DSLOC	Delivered Source Lines of Code
ECP	Engineering Change Proposal
El_Mod	External Interfaces Modified
ENG	Engineering super domain
Enh%	Percent of the release that is Enhancements to the system
EW	Electronic Warfare
FSE	Field Software Engineering
FTE	Full Time Equivalent
IAVA	Information Assurance Vulnerability Alert
IAVM	Information Assurance Vulnerability Management
ICEAA	International Cost Estimating and Analysis Association
LCMC	Life Cycle Management Centers
LOE	Level of Effort
Maint%	Percent of the release that is Maintenance changes
O&S	Operations and Sustainment
ODC	Other than Direct Costs
OLS	Ordinary Least Squares statistical regression
OMA	Operations and Maintenance Army funding
OPA	Other Program Army funding
OSD	Office of the Secretary of Defense
OSMIS	Operation/Sustainment Management Information System
PDSS	Post-Deployment Software Support
PEO	Program Executive Office
PM	Person-Months of effort
POM	Program Objective Memorandum
POR	Program of Record

PPSS	Post-Production Software Support
PTR	Problem Trouble Report
RDT&E	Research, Development, Testing, and Evaluation
RMF	Risk Management Framework
RT	Real-Time super domain
SC	Software Changes
SEC	Software Engineering Center
SER	Schedule Estimating Relationship
SLOC	Source Lines of Code
SRDR	Software Resources Data Report
SRDR-M	Software Resources Data Report for Maintenance
STIG	Security Technical Implementation Guides
SUP	Mission Support super domain
SW	Software
SWBase	Software Baseline SLOC
SWS	Software Sustainment
TDEV	Time to Develop
THrs	Total release hours
TReqts	Total Requirements in a system
TReqts_Imp	Total Requirements Implemented in a release
TSC	Total Software Changes for a release
WBS	Work Breakdown Structure

References

[1] Ferguson, J. “Crouching Dragon, Hidden Software: Software in DoD weapon systems,” *IEEE Software*, July/August 2001, Vol. 18 No.4:105-107.

[2] Crawford, Bruce T., “Military Software Is 'Next Great Frontier’,” National Defense, 21 April 2017.

[3] Jones et al., “Investigation into the Ratio of Operating and Support Costs to Life-Cycle Costs for DoD Weapon Systems,” *Defense ARJ*, January 2014, Vol. 21 No. 1: 442–464.

[4] Defense Science Board. Design and Acquisition of Software for Defense Systems, Department of Defense, February 2018.

[5] McGarry, John et al. “Software Maintenance, Sustaining Engineering, Software Maintenance, Sustaining Engineering, and Operational Support: Key Factors That Impact Program and System Performance,” Systems and Software Technology Conference, 25 April 2012.

[6] Judy, James et al. “Software Maintenance, Sustaining Engineering, Software Maintenance, Sustaining Engineering, and Operational Support: Estimating Software Maintenance Costs for U S Army Weapons Estimating Software Maintenance Costs for U.S. Army Weapons Systems,” PSM User’ Group Meetings and Workshops, 28 July 2012.

[7] Judy, James. “Software Maintenance and Sustaining Engineering Cost Estimation,” PSM User’ Group Meetings and Workshops, 28 July 2012.

[8] McGarry, John and Robert Charette, “Workshop: Life Cycle Maintenance Cost Estimation Model,” PSM User’ Group Meetings and Workshops, 31 July 2012.

[9] Judy, James et al., “Estimating Software Maintenance Costs for U.S. Army Systems,” COCOMO Forum 2013, 16 October 2013.

[10] Judy, James et al., “Software Maintenance Cost Estimating Relationships – One Size Does Not Fit All,” 28th International Forum on COCOMO and Systems/Software Cost Modeling, 23 October 2013.

[11] Judy, James et al., “How Much Does Software Maintenance Cost?,” ICEAA Workshop 2015, 9 June 2015.

[12] McGarry, John and Robert Charette, “Army Software Maintenance - Addressing the Critical Issues,” 17th Annual PSM Users’ Group Workshop, 22 February 2016.

[13] Jones, Cheryl et al., “Workshop: Transitioning Defense Software Maintenance/ Sustainment to DevOps,” 17th Annual PSM Users’ Group Workshop, 23 February 2016.

[14] Government Accountability Office. *GAO Cost Estimating and Assessment Guide, Best Practices for Developing and Managing Capital Program Costs*, GAO-09-3SP, March 2009.

[15] Judy, James et al. “Estimating System Software Sustainment Costs: Generating Critical Decision Information to Inform the Army Software Enterprise”, ICEAA, 5 June 2017

[16] Boehm et. Al., Software Cost Estimation with COCOMO II, Prentice Hall PTR, Upper Saddle River, NJ, 2000.