ESTIMATING SYSTEM SOFTWARE SUSTAINMENT COSTS: GENERATING CRITICAL DECISION INFORMATION TO INFORM THE ARMY SOFTWARE ENTERPRISE

AUTHORS

Cheryl Jones, RDECOM ARDEC
James Doswell, ODASA-CE
Jenna Meyers, ODASA-CE
John McGarry, RDECOM ARDEC
Robert Charette, PhD, ITABHI Corporation
Bradford Clark, PhD, Software Metrics, Inc.
James Judy, ODASA-CE
Douglas Putnam, Quantitative Software Management, Inc.

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 enterprise management decision making. This paper discusses a multi-year effort across the Army software community to establish 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.

1.0 INTRODUCTION

It is commonly understood across the Department of Defense that software must be managed as a critical mission asset. [1] Software is the system component where nearly all mission capability is now instantiated, and where this 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. 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 may be associated with its sustainment. [2] In fact, the majority of a system's mission capability is created and integrated during the sustainment portion of the system life-cycle. 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 study sponsored by the Secretary of the Army in 2013, and the concurrent software sustainment study 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. [3 - 11]

2.0 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, adaptive, security, and corrective related changes, and how to structure incremental software product deliveries. Army enterprise level decisions center on determining resource priorities across the operational system portfolio, and trades between funding and associated mission capability. Decision information must 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 against validated software sustainment activities and products.

2. The creation of a baseline system software sustainment data repository. This is based on the collection, characterization, evaluation, and normalization of three years of historical software sustainment technical, cost, and context data from all operational Army software systems.

3. Analysis of the normalized software sustainment data to generate improved and validated WBS driven cost heuristics and statistically derived sustainment cost estimation relationships, cost benchmarks, and cost 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 driven 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 is currently completing 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, it is clear that improved software sustainment cost estimation methods and models will be the basis for an effective decision information infrastructure.

3.0 KEY INFRASTRUCTURE COMPONENTS

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

A key estimation issue centers on how, in actuality, required software system sustainment costs are derived. Rather than being directly tied to output activities and products, such as cybersecurity certifications and incremental releases, software sustainment resources have typically been associated with the projected size of the sustainment teams and other general "capacity" oriented cost elements. Life cycle software sustainment costs have been generally estimated as a percentage of the system's software acquisition cost. This resulted in an inaccurate cost estimation methodology that did not objectively project the actual cost of sustaining a software system.

In addition, there has been no requirement for Army software sustainment organizations to report executed software sustainment costs. This has limited the Army's ability to generate and refine cost estimating relationships for the primary software sustainment cost elements. This has also impacted consistent enterprise insight into overall software sustainment portfolio performance.

Up until recently, DOD software maintenance has lived in the “shadows,” meaning that controlling the cost of acquiring a system was of higher priority than controlling the cost of its maintenance. The Government Accountability Office’s “GAO Cost Estimating and Assessment Guide: Best Practices for Developing and Managing Capital Program Costs,” reflects this precedence. [12] However, as system software sustainment costs have grown as new system acquisitions have slowed, controlling these costs has become more important to both DOD and Army leadership.

The 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
- A repository of software sustainment execution data

3.1 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 did decision makers at different levels of the Army enterprise need to know to make effective resource and 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. Participants included senior Army leadership, multi-program resource and technical managers, and system level program and technical managers. The workshops reviewed the scope of software sustainment, the existing software sustainment stakeholder structure, required data, sustainment management and technical processes, and key decision information needs. Both common information requirements across all decision levels and unique information needs from each stakeholder were identified.

The existing definition of Army Software Sustainment, in terms of funding types, execution responsibility, and decision authority was reviewed with respect to alignment with current system software development and sustainment management practices. It was determined that the traditional scope of software sustainment activities and products was constrained by depot maintenance policy, which defined the scope of “Software Maintenance” to include only corrective and adaptive changes. Moreover, funding types and associated cost management constraints created a complex cost accounting environment that impacted the Army’s ability to generate, collect, and aggregate software sustainment cost and technical data. As a result, the initiative re-defined Army software maintenance to include all software change activities and products associated with modifying a software system after the Engineering & Manufacturing and Development (EMD) phase has completed and a software release has been provided to an external party. The primary software sustainment change product is considered to be a “maintenance release”, consisting of one or more baseline changes. These changes include both software enhancements and software corrections/adaptations and may be resourced by multiple funding sources (e.g., RTDE, Production, OMA, FMS, OCO, etc.).

In the initiative, the terms software maintenance and software sustainment are considered to be synonymous.

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 most significant information requirement at all management levels centers on the need to objectively project the cost to sustain a given operational software system or system portfolio.

3.2 Army Software Sustainment WBS

A viable software sustainment WBS is a defining characteristic of 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.

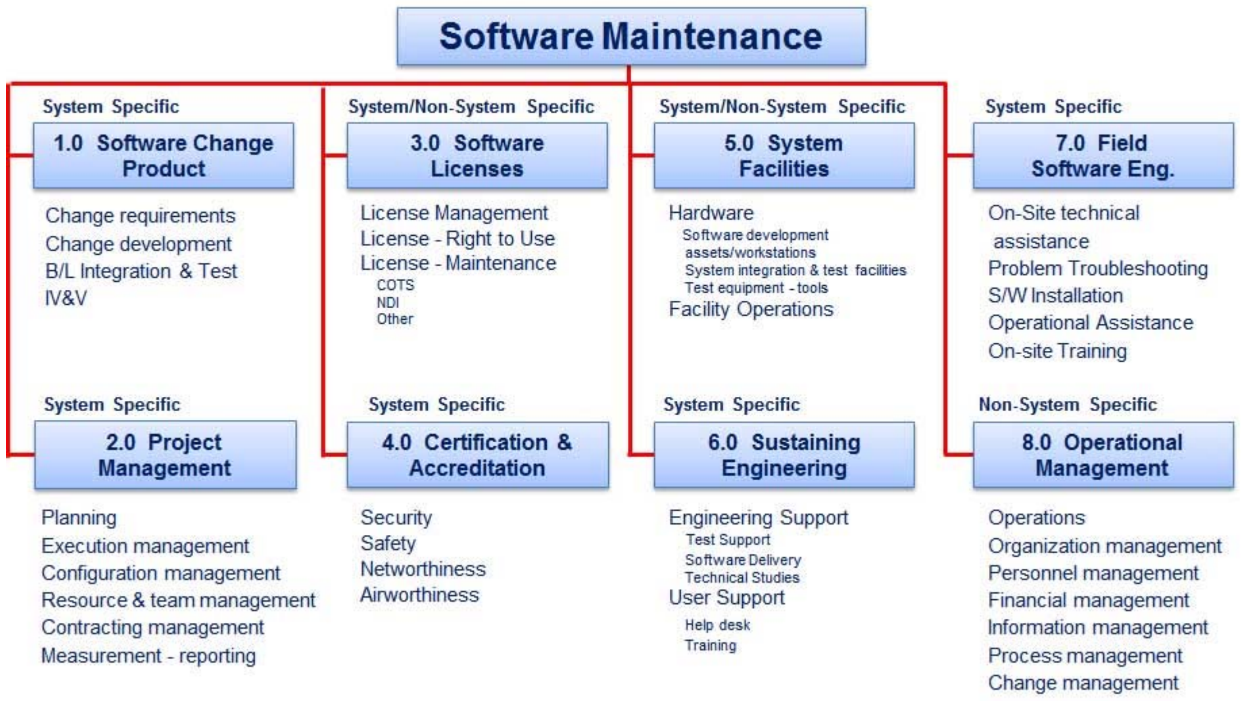


Figure 3.1 Army Software Maintenance Work Breakdown Structure (Version 4.4d)

The current Army software maintenance WBS is shown in Figure 3.1. 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.

3.3 Software Sustainment Data Repository

Data collection requirements were defined based on stakeholder decision information requirements and the activities and products defined by the WBS. An initial questionnaire was developed to collect the identified data.

Three general categories of data are collected using the questionnaire. The first is 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. This data is not directly mapped to the WBS.

The second category of data is 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.

The third category is release level data. 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.

For all categories of data, the actual execution cost and effort data is preferred, but Full-Time Equivalent (FTE) or planning data may be collected, if actual data does not exist at the lower WBS levels.

Section 4.0 on Software Sustainment Data Collection, Characterization, and Evaluation provides more information on the data collected.

3.4 The Importance of Understanding the Context

Every Army system is different. The collection of system "context" data is required to enable the accurate interpretation and comparison of the collected cost and technical software sustainment data.

Integral to the initiative been the definition of distinct software system "super domains" that include similar systems as characterized by their software sustainment products, activities, and costs. To date, four super domains have been defined:

- Engineering
- Real-Time
- Support
- AIS (automated information system)

With initial data analysis it became apparent 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 all of the variables across Army software sustainment activities that will yield a valid cost estimate. All of the different products and activities that are 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.

For example, if a system has been mandated to use commercial off-the-shelf software for the majority of its software functionality, then many of the software sustainment costs are directly tied to the vendor, and the costs of sustainment may be heavily skewed towards licensing costs. Similarly, software sustainment costs (and underlying CERs) are dependent on the type (or domain) of the software involved. The sustainment activities and costs involving real-time, embedded software systems are different from those of transaction-based enterprise resource management systems. Also, the sustainment of systems performed by government organizations versus contractor organizations using different business models need to be accounted for.

4.0 SOFTWARE SUSTAINMENT DATA COLLECTION, CHARACTERIZATION, AND EVALUATION

As part of this study, a multi-phase data collection activity was established. During the initial phase ("Phase I"), data was collected for five systems from each Army software sustainment activity, including PEOs and Software Engineering Centers and Directorates (SECs and SEDs), for a total of 56 systems. This initial 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 Phase I data provided an initial set of software sustainment context and quantitative data that is the basis for the analysis discussed in this paper.

Phase II data collection, which is now being completed, includes the collection of software sustainment data across the remaining Army programs. This includes approximately 200 additional systems and will allow additional analysis using a more complete data set. The systems that are include weapons systems C4ISR systems, as well as business systems.

One of the key findings of the Phase I data collection activities is the limited availability of objective data or associated policies that require objective data. A key recommendation from the initiative is for the development of systemic methods to collect and analyze the required data in the future.

4.1 Data collection

The process for data collection began with the PEO/SEC/SED identification of 5 systems per organization for Phase I data collection (56 in total). DASA-CE then met with the system representatives to explain the data collection questionnaire and clarify requirements. After that, the system representative completed and submitted initial draft of questionnaire. The questionnaire requested historical data for 3 years (2013, 2014, and 2015).

Once the questionnaire was received, the DASA-CE team reviewed the questionnaire, identified questions, and then met with each representative to discuss context questions and data issues. The system representative then updated the questionnaire based on DASA-CE findings. DASA-CE then reviewed the revised submission and continued to rework it with the system representative as necessary. The face-to-face discussions with the system representatives were key to producing good data that was useable for analysis.

Final data submission was then accepted and evaluated for availability, integrity, and usability using a structured methodology.

4.2 Data Evaluation

A rating criteria was used to evaluate both the annual effort/cost data for each program by WBS as well as the release effort, schedule, and associated technical data. The values for each rating (color) were used to derive a summary level rating. The rating criteria, documented in the Data Evaluation Guide, are shown below in Figure 4.1:

Table 1. Data Quality Levels

Color	Definition	Value
R	Red indicates there is no planning or actual data reported.	0
O	Orange indicates only planning data was reported.	1
Y	Yellow indicates FTE or partial, actual data was reported	2
G	Green indicates that actual data was reported.	3

Figure 4.1 Program Rating Criteria

The annual cost and effort data provided by each individual system was rated, using these criteria, as shown below in Figure 4.2:

					Initial System Overall				Detailed System Assessment							
Collect Phase	PEO	SEC	System	Definable Maint. Process	Total Program Effort/Cost	WBS 2-8		Project Mgmt (WBS-2)	License Management (WBS-3)	C&A Support (WBS-4)	System Facilities Management (WBS-5)	Sustaining Engineering (WBS-6)	Field S/W Engineering (WBS-7)	Operational Management (WBS-8)	License Costs
1	(PEO 4)	SEC 3	System 1	G	G	G		G	N/A	G	G	G	N/A	R	N/A
1	(PEO 4)	SEC 3	System 2	G	G	Y		Y	R	Y	Y	Y	N/A	R	G
1	(PEO 1)	SEC 2	System 3	R	G	Y		R	G	G	Y	Y	N/A	Y	R
1	PEO 1	(SEC 2)	System 4	G	R	O		R	R	R	R	R	N/A	R	R
1	PEO 1	(SEC 2)	System 5	G	G	Y		R	R	G	Y	G	G	G	R
1	PEO 1	(SEC 2)	System 6	G	G	O		R	R	Y	R	R	Y	R	G
1	PEO 1	(SEC 2)	System 7	G	G	G		G	G	G	G	G	G	G	G
1	PEO 1	(SEC 2)	System 8	G	R	R		R	R	R	R	R	R	R	R

Figure 4.2 Rating by Annual Cost and Effort Data

Data provided for each individual WBS element was rated, along with the overall system effort/cost data. In addition, an evaluation across the set of WBS elements was rated (in column WBS 2-8). As can be seen from this subset of data, some programs had detailed data by the requested WBS breakouts, while other programs did not have any effort/cost data. Many programs had data on some WBS elements (e.g., C&S support and license costs), but relatively few had data on every WBS element. This was not unexpected given the variability of the management and accounting processes in place across the sustainment community.

In a similar manner, the release data provided by each individual system was also rated. Each system had one to twelve releases, depending on the release rhythm and number of cyber-security releases. A sample of release evaluations is shown in Figure 4.3

				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 4.3 Release Evaluation Sample

In addition to rating individual technical, effort/cost, and schedule data, the evaluation also indicates the degree to which each release is useable for developing either cost estimating relationships (CERs) or schedule estimating relationships (SERs). This evaluation is based on whether one or more size measures are available, along with effort/cost or schedule information.

Once each individual system was rated, a summary of the Phase I Data was calculated and is shown below:

Initial System Overall				Detailed System Assessment							
	Definable Maint. Process	Total Program Effort/Cost	WBS 2-8	Project Mgmt (WBS-2)	License Mgmt (WBS-3)	C&A Support (WBS-4)	System Facilities Mgmt (WBS-5)	Sustaining Engineering (WBS-6)	Field S/W Engineering (WBS-7)	Operational Mgmt (WBS-8)	License Costs
R	14	11	11	28	30	11	21	27	17	32	12
O	0	2	15	2	2	8	6	3	2	1	3
Y	1	12	16	8	4	14	11	9	8	10	1
G	40	31	13	18	12	20	16	16	6	11	35
N/A	1	0	1	0	8	3	2	1	23	2	5
Total	56	56	56	56	56	56	56	56	56	56	56

Figure 4.4 Phase I Summary Data Chart

As can be seen from the summary in Figure 4.4, 43 systems were able to provide total effort or cost data (31 with actual data and 12 with FTE data), whereas 2 systems could only provide planning data, and 11 systems could not break out software effort or cost at all. At the lower level WBS elements, relatively few systems were able to provide effort or cost data. There was no expectation that the data would be complete, given that programs were asked for historical data from the past using a new WBS that was not in effect during those years. However, there was an expectation that most programs would be able to provide total effort or cost data, and a breakout of at least several WBS elements like C&A and license costs. Despite this specific lack of information, data was collected from enough systems to provide initial data analysis and demographics in terms of histograms of data distributions.

At the release level, data was received on 218 releases, including releases that provided enhanced capabilities, maintenance updates, and cyber-security updates as shown in Figure 4.5. CERs were developed for releases containing new capabilities and maintenance updates (plus cyber-security updates), and for releases that were cyber-security updates only.

Initial Release Overall			**Detailed Release Assessment**							
	CER Usability	**SER Usability**	**Size: Requirements**	**Size: External Interfaces**	**Size: SLOC**	**Size: Non-SLOC**	**Size: SW Changes**	**IAVAs**	**Effort (WBS-1)**	**Schedule (WBS-1)**
R	71	77	101	79	46	28	39	67	60	40
O	44	43	5	2	5	0	3	3	47	12
Y	23	22	3	2	4	2	5	0	23	27
G	79	75	70	55	69	12	116	106	87	138
N/A	1	1	39	80	94	176	55	42	1	1
Total	218	218	218	218	218	218	218	218	218	218

Figure 4.5 Release Level Data

For the Phase I analysis, releases that had actual effort/cost data (79), FTE data (23), and planning data (44) were used in the CER analysis, for a total of 146 releases. During Phase II and future analyses, the emphasis will be on actual data, when there are sufficient data points, or actual plus FTE data, when the data is more limited.

One of the interesting findings related to the data is the types of size measures that organizations use. Because there is much debate over the most appropriate sizing methods, the study gave submitters the choice of six size measures to use: requirements, external interfaces, lines of code, software changes (defects), IAVAs (cyber-security vulnerabilities addressed), or a user-defined count (such as RICE-FW, function points, story points). While 1/3 of the systems reported requirements measures, most of these turned out to be a count of changes made to the code (versus a SRS-type requirement). The use of source lines of code (SLOC) was still a primary size measure for many of the weapon system programs, and many of those programs had historical SLOC data available as well. Some 2/3 of the systems reported some type of software change measure (often defects, PTRs, or similar). Many of the COTS-intensive systems that had cyber-security concerns tracked Information Assurance Vulnerability Alerts (IAVAs), although not all systems did.

Business systems (ERP systems in particular) had the hardest time quantifying any sort of size measure. For instance, one system using agile development tracked story points, awhile another system tracked RICE-FW objects. No program reported using function points.

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.

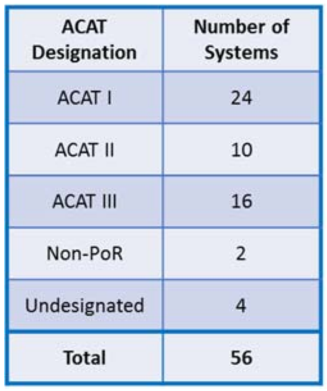
4.3 Data Normalization

Once the data was evaluated, it was entered into a repository. As part of this process, data was normalized for analysis purposes. Some of the normalization done included:

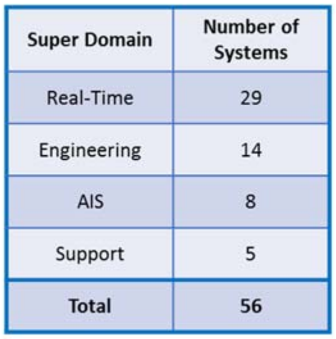
- Cost/Effort Normalization: If systems provided only effort data, then costs were estimated using the average labor rates provided. Likewise, if only cost data was provided, then effort was estimated.
- All cost data was converted to base year (BY) 2016 dollars, using the standard yearly conversion ratios for the types of dollars used.
- All lines of code types (e.g. physical, NCSS) were converted to logical lines of code.
- All IAVA only releases were combined into annual releases per system. Systems do IAVA releases anywhere from daily to weekly to monthly to quarterly, depending on the system. All data for one year were combined into one release for purposes of our analysis.
- Annual cost was adjusted to included release cost, if it was not originally included.

4.4 Data Demographics

The data collected in Phase I represents a diverse set of Army programs spanning different functional areas. The table below shows how the 56 programs in Phase I span the different acquisition categories (ACAT designation) and how they were allocated to super domains. Some of the systems have been in sustainment for many years while others represent systems in production that have just started maintaining a software baseline.



ACAT Designation	Number of Systems
ACAT I	24
ACAT II	10
ACAT III	16
Non-PoR	2
Undesignated	4
Total	**56**



Super Domain	Number of Systems
Real-Time	29
Engineering	14
AIS	8
Support	5
Total	**56**

Table 4.1 Phase I Data Collection by Different Program Type

4.4.1 Allocation of Costs Across the Software Sustainment WBS

One of the initial charts that was developed showed the allocation of costs across the WBS elements. This addressed an enterprise level information requirement concerning the allocation of executed costs for various sustainment activities and products. Changes to the software product accounted for 31% of the costs of software sustainment. The chart shown in Figure 4.6 is based on 113 data points from 43 systems that had sufficient WBS data.

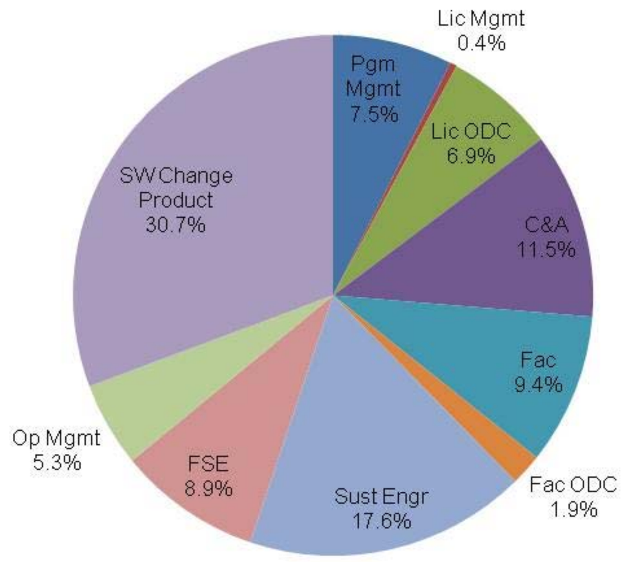


Figure 4.6 Allocation of Costs by WBS Element

This chart was developed using the following methodology:

- For each system, the percent allocation across the WBS was calculated. If a program reported multiple years of data, the average percent allocation across the years of data was used to represent the program.
- The average percent allocation for the super domain was then calculated using the average of the programs within each super domain.
- If a program reported all costs in only one WBS element, the program was not included in the analysis.
- Program level data was used in this analysis which includes government and contractor costs.
- Note: Some WBS elements like license costs (license ODC) and facility costs (facility ODC) were likely underreported in some cases.

Of course, there are wide distributions of these costs across the different super-domains. The following chart in Figure 4.7 breaks out these percentages for each of the defined super-domain categories.

This allocation compares sustainment cost proportions across systems that are in different functional domains.

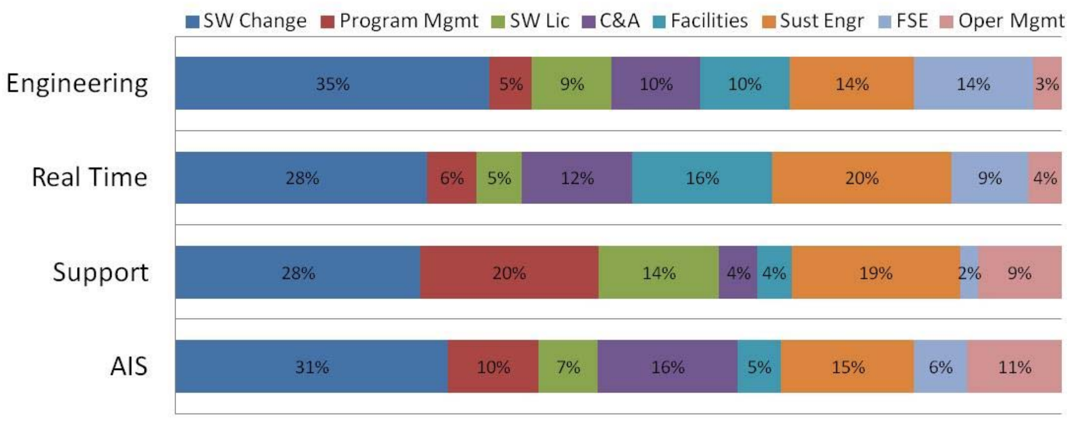


Figure 4.7 Allocation of Costs by Super Domain

The total number of systems and data points for each super-domain is shown below:

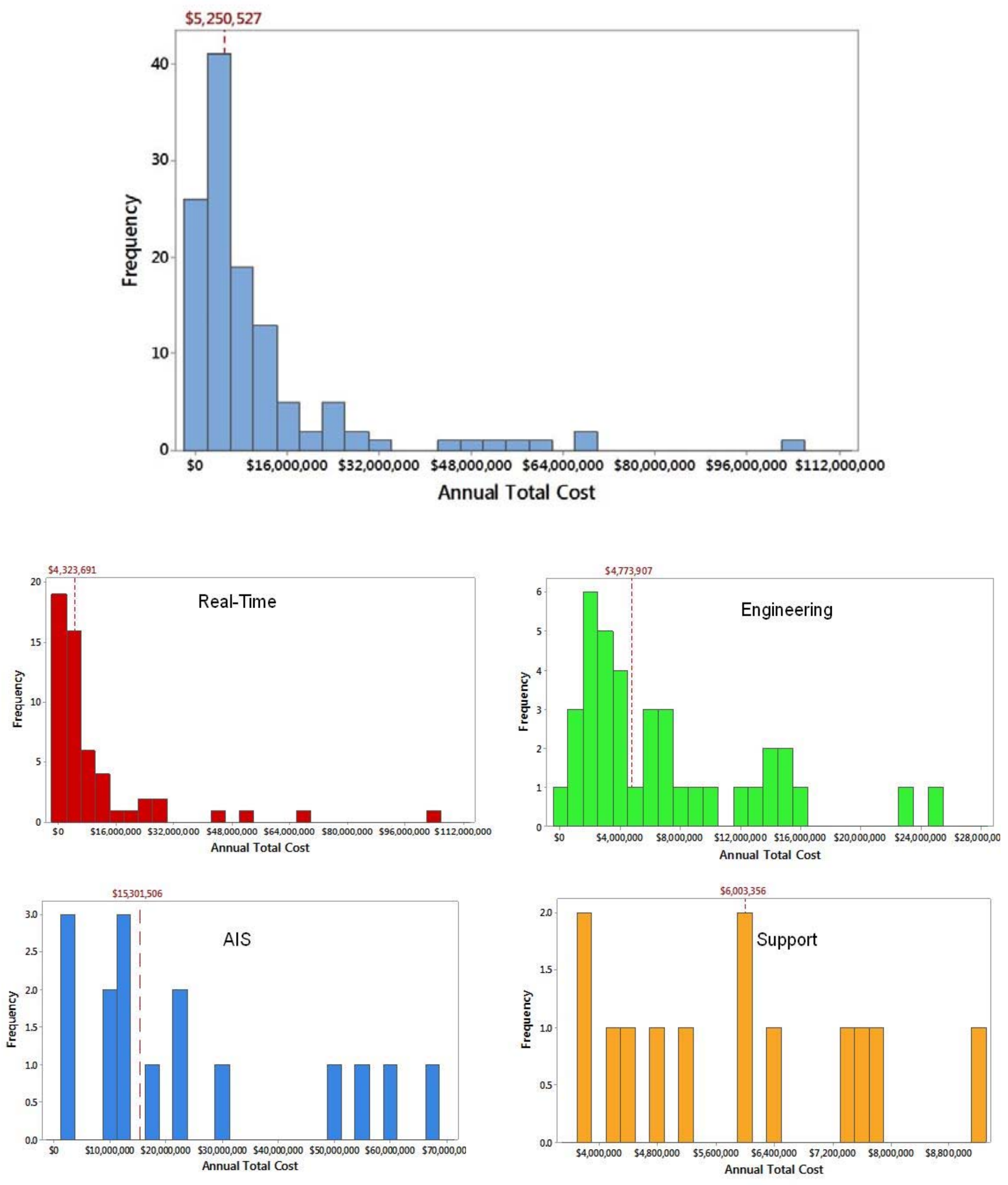
- Engineering: 12 systems, 35 data points
- Real-Time: 20 systems, 49 data points
- Support: 4 systems, 13 data points
- AIS: 7 systems, 16 data points

4.4.2 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 histograms shown in Figure 4.8 are shown for the entire dataset as well as stratified by Super Domain. The values for the first quartile (Q1), median, and third quartile (Q3) are shown which can be used to further inform an estimate using a program's risk profile.

The methodology for generating these histograms includes the following:

- Annual cost includes government and contractor costs.
- No outliers were removed for the Phase I analysis, due to the small sample size.
- Due to non-normal distribution of data, the median is used as central tendency value.
- Fields that were left blank or zero were not used.
- Each program may have reported multiple years of data. Each year of reported data represents an observation.



SD	N	Minimum	Q1	Median	Q3	Maximum
AIS	16	$2,783,335	$9,727,349	$15,301,506	$45,125,746	$66,448,489
ENG	38	$250,732	$2,320,639	$4,773,907	$10,734,471	$24,870,059
RT	55	$117,428	$1,363,244	$4,323,691	$10,355,772	$103,731,995
SUP	13	$3,729,674	$4,363,762	$6,003,356	$7,467,262	$9,120,451
All	122	$117,428	$2,350,442	$5,250,527	$11,948,232	$103,731,995

Figure 4.8 Distributions of Annual Costs

Additional histograms were developed for every WBS element per the collected data. There are obviously some outliers in this data. The Phase II analysis will take these outliers into consideration and attempt to clarify the context details that create unique circumstances in the data.

5.0 INITIAL COST ESTIMATION RELATIONSHIPS (CER) AND BENCHMARKS

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 Benchmarks. The CER analysis attempted to find one or two product attributes (equivalent source lines of code or number of software changes) to estimate the value of another attribute (cost, hours, or duration). Benchmarks attempt to show the central tendency (using either the mean or median) of the ratio of two attributes, e.g. cost per software change.

The CERs and benchmarks were grouped by either Super Domain or IAVA/non-IAVA releases. An IAVA is an Information Assurance Vulnerability Assessment - a count of the number of identified cybersecurity issues that needed to be evaluated/addressed within the system. Note that actual software changed generated by an IAVA were costed in WBS 1.0.

5.1 CER Derivation

This analysis investigated cost estimating relationships (CERs) for WBS 1.0 - Software Change Product. These CERs are utilized later in the acquisition lifecycle when expected release size (i.e. SLOC or number of software change counts) or duration is known.

The data for WBS 1.0 is measured as a software release. Data was collected on release characteristics, size, cost, effort (hours), duration and number of IAVAs performed.

5.1.1 Ground Rules/Assumptions

The following ground rules or assumptions apply to each CER:

- The CER applies to WBS 1.0 Software Change Product only. CERs are grouped using one of two methods, either by Super Domain or whether it was an IAVA only release or a non-IAVA release.
- The fiscal year is independent of the start and end date for each release. Releases can span multiple fiscal years.
- The term Total Cost includes both government and contractor costs.
- Due to the lack of data, outliers were not removed with the exception of two CERs.
- Only CERs with an $R^2 > 60\%$ are shown.
- All costs were normalized to Base Year 2016.

5.1.2 Methodology

Ordinary Least Squares (OLS) regression was used to derive the CERs. The Minitab statistics tool and the Cost Analysis Statistics Package (CO$TAT) 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 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)*

The CERs are linear and nonlinear models. The nonlinear models were created using a log-log transformation of the dependent and independent variables.

Three model fit statistics were used to evaluate the CER:

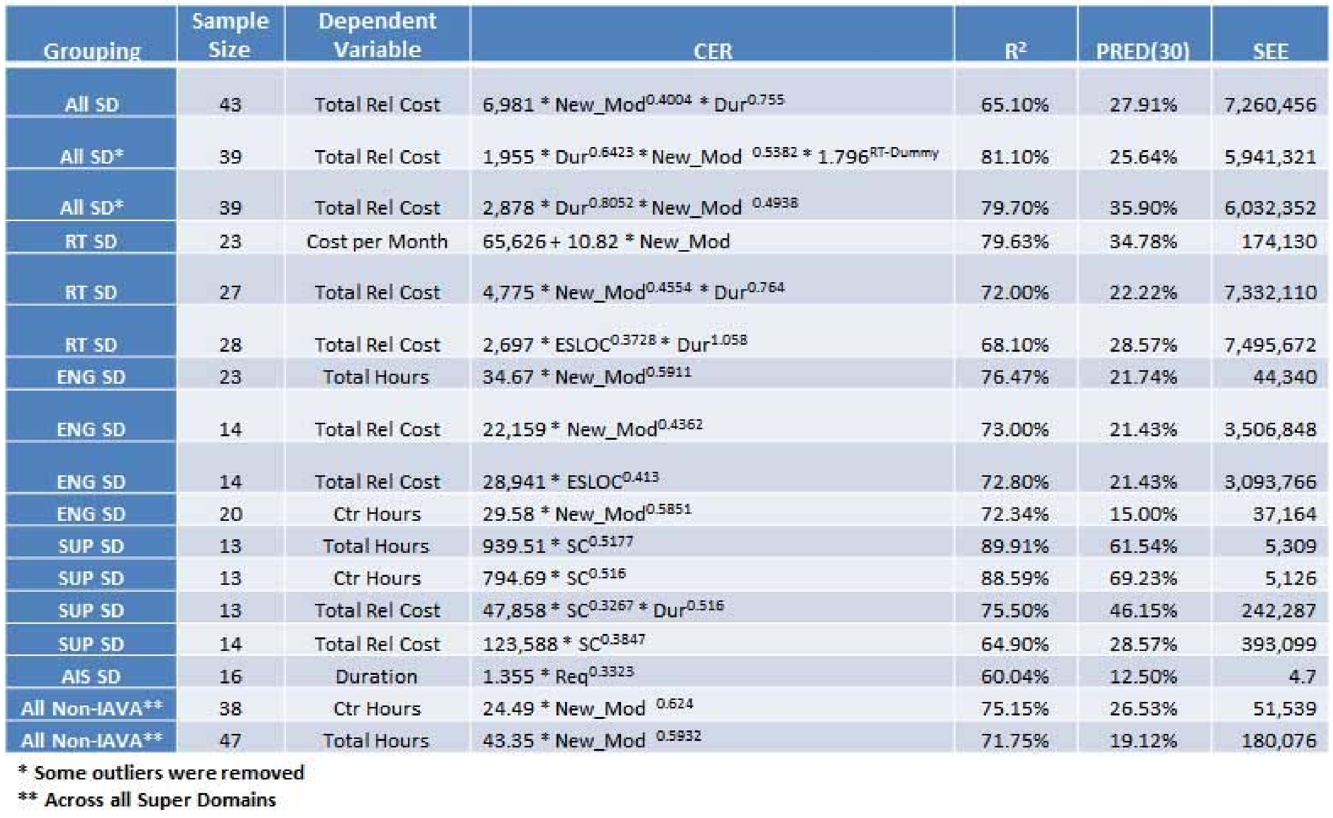
- The R^2 is the Coefficient of Determination and represents the percentage of total variation explained by the regression model. It provides a measure of how well actual values of effort are estimated by the CER model factors. R^2 ranges between 0 and 100% and is the same in log-space as in unit-space.
- 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.

5.1.3 CER Results

The CERs that had a R^2 of greater than 60% are displayed in Table 5.1. The results are grouped by Super Domain (SD) and IAVA/non-IAVA release. The sample size for each CER is provided. The dependent variable is the attribute that is being predicted. The CER shows the dependent variables and their coefficients derived from OLS. The fit statistics are provided in the last three columns.

The independent variables in the CER column are abbreviated. The abbreviations are explained below:

- ESLOC = Equivalent Source Lines of Code explained earlier
- New_Mod = Sum of New and Modified Lines of Code
- SC = Software Change Count (Problem Reports, Defects, Issues, Change Requests, etc.)
- Dur = Release Duration in months
- Req = Software Requirements (SRS equivalent requirements)



Grouping	Sample Size	Dependent Variable	CER	R²	PRED(30)	SEE
All SD	43	Total Rel Cost	$6{,}981 * New\_Mod^{0.4004} * Dur^{0.755}$	65.10%	27.91%	7,260,456
All SD*	39	Total Rel Cost	$1{,}955 * Dur^{0.6423} * New\_Mod^{0.5382} * 1.796^{RT\text{-}Dummy}$	81.10%	25.64%	5,941,321
All SD*	39	Total Rel Cost	$2{,}878 * Dur^{0.8052} * New\_Mod^{0.4938}$	79.70%	35.90%	6,032,352
RT SD	23	Cost per Month	$65{,}626 + 10.82 * New\_Mod$	79.63%	34.78%	174,130
RT SD	27	Total Rel Cost	$4{,}775 * New\_Mod^{0.4554} * Dur^{0.764}$	72.00%	22.22%	7,332,110
RT SD	28	Total Rel Cost	$2{,}697 * ESLOC^{0.3728} * Dur^{1.058}$	68.10%	28.57%	7,495,672
ENG SD	23	Total Hours	$34.67 * New\_Mod^{0.5911}$	76.47%	21.74%	44,340
ENG SD	14	Total Rel Cost	$22{,}159 * New\_Mod^{0.4362}$	73.00%	21.43%	3,506,848
ENG SD	14	Total Rel Cost	$28{,}941 * ESLOC^{0.413}$	72.80%	21.43%	3,093,766
ENG SD	20	Ctr Hours	$29.58 * New\_Mod^{0.5851}$	72.34%	15.00%	37,164
SUP SD	13	Total Hours	$939.51 * SC^{0.5177}$	89.91%	61.54%	5,309
SUP SD	13	Ctr Hours	$794.69 * SC^{0.516}$	88.59%	69.23%	5,126
SUP SD	13	Total Rel Cost	$47{,}858 * SC^{0.3267} * Dur^{0.516}$	75.50%	46.15%	242,287
SUP SD	14	Total Rel Cost	$123{,}588 * SC^{0.3847}$	64.90%	28.57%	393,099
AIS SD	16	Duration	$1.355 * Req^{0.3323}$	60.04%	12.50%	4.7
All Non-IAVA**	38	Ctr Hours	$24.49 * New\_Mod^{0.624}$	75.15%	26.53%	51,539
All Non-IAVA**	47	Total Hours	$43.35 * New\_Mod^{0.5932}$	71.75%	19.12%	180,076

* Some outliers were removed
** Across all Super Domains

Table 5.1. Cost Estimating Relationships

The strongest and most useful CER in Table 5.1 is the estimation of total release hours based on software changes (SC) for the Support Super Domain.

$$\textit{Total Hours} = 939.51 * SC^{0.5177}$$

This CER had a high PRED (30) of 89.91% based on 13 data points. Unfortunately this is for only one Super Domain and there are not many data points.

Other CERs based on source lines of code (SLOC) as a size input do not perform very well. Perhaps SLOC is not a valid measure of size for sustainment projects. More data is required to support this conclusion.

Other derived CERs with statistically significant predictive results are presented in Table 5.1. It is expected that subsequent analysis using the Phase II data set will provide additional insight into the predictive variables useful for software change product costing.

5.1.4 Future CER Research

There is additional research pertinent to future CER analysis. Only two groupings were presented in this paper. However with additional data expected from Phase II, CERs can be created for application domains (these are contained within the Super Domains), Maintenance Organizations, Operating Environments, and type of release (enhancement, defects, cybersecurity).

The analysis as currently presented addresses cost estimating relationships. Equally important are schedule estimating relationships (SER). This is a future area for analysis.

Analysis to date has indicated that there is a repeatable pattern of software sustainment releases described in terms of size, content, and schedule across the life cycle phases of each system. A primary area for future research is the analysis of these patterns, which we define as software release rhythm. A release can be large or small. Large releases usually contain major enhancements. Small releases are usually bug fixes. The frequency of major to minor changes needs to be understood and quantified as it has a large influence on creating a full lifecycle cost estimate.

5.2 Benchmarks

The purpose of this analysis is to provide benchmarks for the WBS 1.0 Software Change Product. These benchmarks are useful for providing mean or median statistics and can be useful for performing a sanity check on software sustainment cost estimates.

5.2.1 Ground Rules/Assumptions

The following ground rules or assumptions accompany each benchmark:

- The benchmarks apply to WBS 1.0 Software Change Product only.
- 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 releases.
- A year is a fiscal year from October 1 to September 30.
- The term Cost, Hours and FTE includes both government and contractor data.
- Due to the lack of data, outliers were not removed.
- All cost data were normalized to Base Year 2016.

5.2.2 Methodology

Basic descriptive statistics using the CO$TAT were used to express benchmark values. The data is not normally distributed. The descriptive statistics show results using both a unit space and lognormal distribution (which approximates a normal distribution). The lognormal results are transformed back into unit space.

The following benchmark statistics were used to express the central tendency and dispersion of the data:

- Count (N): the number of data points in the sample size.
- Mean: Estimated sample value representing the population central value; equal to the sum of the values divided by the number of values, i.e., arithmetic mean.
- Standard Deviation (Std Dev): Standard Deviation measures the amount of variation or dispersion from the mean in a sample. Plus or minus one standard deviation from the mean is a range that includes about 68% of the data.
- Standard Error (SE): A standard error is the standard deviation of the sampling distribution of a statistic. Standard error is a statistical term that measures the accuracy with which a sample represents a population. The standard deviation measures the amount of variability or dispersion for a subject set of data from the mean, while the standard error of the mean measures how far the sample mean of the data is likely to be from the true population mean.
- Coefficient of Variation (CV): The coefficient of variation is a measure of spread that describes the amount of variability relative to the mean. Because the coefficient of variation is unit-less, it can use instead of the standard deviation to compare the spread of data sets that have different units or different means. It can be found by dividing the standard deviation by the mean. It is generally expressed as a percentage.
- Minimum (Min) Value: Min value is the small value in the sample.
- Q1: Numerical value for the lower 25% of ranked data (1st Quartile), i.e., the value half way between the lowest value and the median in a set of ranked values.
- Median: Numerical value separating the higher half of a sample from the lower half, i.e., the middle value in a set of ranked values.
- Mode: The mode is the value that appears most often in a set of data.
- Q3: Numerical value for the lower 75% of ranked data (3rd Quartile), i.e. the value half way between the median and the highest value in a set of ranked values.
- Maximum (Max) Value: Max value is the largest value in the sample.

The dispersion statistics are useful in understanding risk about a mean or median value, i.e., how well the value represents the true population mean or median.

5.2.3 Cost per Software Change by Super Domain

This analysis, shown in Figure 5.1 attempts to show what it costs to implement one software change during sustainment for each super domain. Both a table of the relevant data, and histograms showing the distribution of the data are provided. The cost per software change is segregated by super domain. The software change count only includes reported changes in a program's software release. Software Changes are also commonly referred to as problem reports, change requests, defects, etc.

The figure shows the total cost for Government and Contractor per change. The data includes IAVA and non-IAVA releases.

	RT SD		Eng SD		Sup SD		AIS SD	
Statistic	Sample	Log Normal	Sample	Log Normal	Sample	Log Normal	Sample	Log Normal
N	30		16		13		10	
Mean	$50,877	$52,711	$203,835	$240,950	$36,095	$39,484	$52,752	$59,728
Std Dev	$76,033	$119,536	$329,675	$795,859	$35,829	$39,077	$59,344	$83,427
SE		$17,433		$41,199		$11,687		$10,672
CV	1.49	2.27	1.62	3.3	0.99	0.99	1.12	1.4
Min	$920		$3,508		$2,916		$3,316	
Mode		$3,462		$5,862		$14,177		$11,782
Max	$403,743		$1,336,963		$109,267		$189,180	

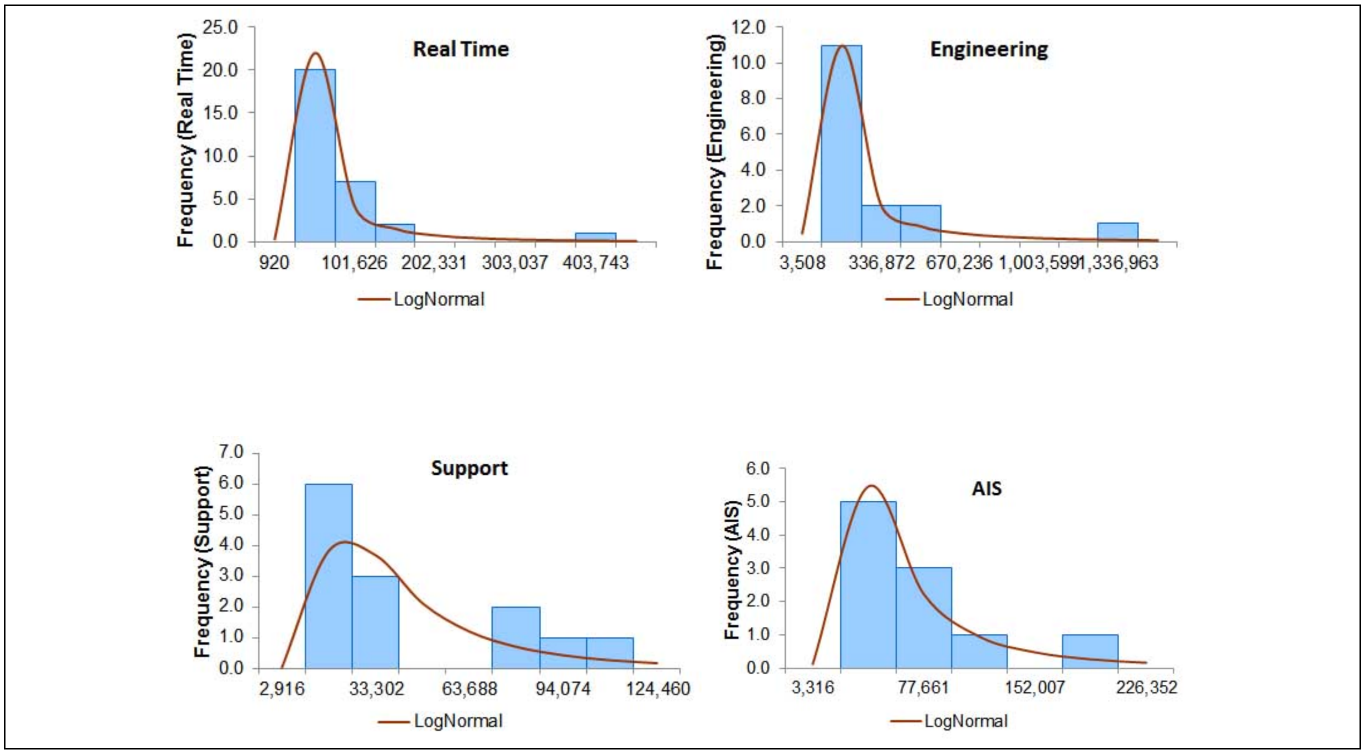
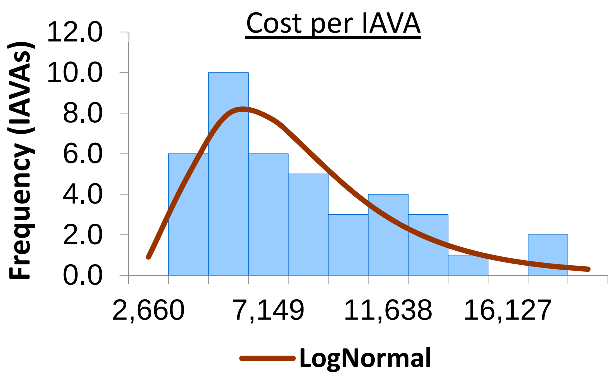


Figure 5.1 Cost per Software Change by Super Domain

Some of the histograms in Figure 5.1 show extreme values which influence the mean value. The mode for each histogram may be of more interest. See the definition of mode above.

5.2.4 Cost per IAVA

The chart and analysis in Figure 5.2 shows the cost of implementing an IAVA generated change. This benchmark uses IAVA only release data. The cost includes both contractor and government costs. The results are not stratified by super domain due to the lack of sufficient data.



Statistic	Sample	Log Normal
N	40	
Mean	$7,547	$7,608
Std Dev	$3,768	$3,783
SE		$537
CV	0.49	0.49
Min	$2,660	
Mode		$5,462
Max	$17,624	

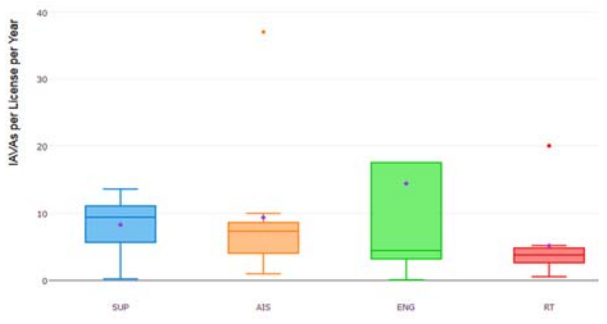
Figure 5.2 Cost per IAVA

5.2.5 IAVAs per License per Year

This analysis attempts to show how many IAVAs are addressed per software license per year. The proposition is that the more licensed software a program uses, the higher the cost in conducting IAVAs. The benchmark on cost per IAVA is used in conjunction with this benchmark.

The information in Figure 5.3 combined with the number of software COTS products in the software application provides a rule-of-thumb of how much IAVAs cost the program per IAVA cycle (quarterly or annually). More data is needed to determine if IAVA costs are different for different groups of data, e.g. super domains.

Figure 5.3 shows this benchmark for each super domain. The IAVA rhythm for each release is different for each program. The data is normalized to a yearly amount.



Statistic	RT	ENG	SUP	AIS
N	9	9	8	9
Mean	5.18	14.45	8.29	9.39
Min	0.56	0.08	0.23	1
Q1	2.71	3.82	6.34	4.65
Median	3.8	4.46	9.43	7.33
Q3	4.7	7.75	11.06	8.18
Max	20	55.89	13.64	37

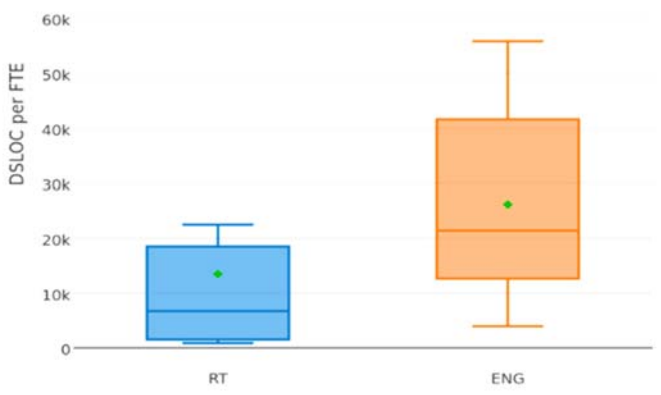
Figure 5.3 IAVAs per License Per Year

The more COTS intensive systems in the SUP and AIS super domains have more IAVAs per license per year. Combined with the cost in Figure 5.2, there are indications these domains spend more on IAVAs.

5.2.6 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. Figure 5.4 shows this data for the phase 1 data set.

DSLOC counts all code types equally. ESLOC counts 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), Program 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 the labor rates for Government and Contractor reported for each program. Only the Real Time and Engineering super domains had sufficient data to derive DSLOC per FTE.



Statistic	RT	ENG
N	16	6
Mean	13,501	26,172
Min	878	3,983
Q1	1,564	14,682
Median	6,736	21,436
Q3	18,188	36,771
Max	80,734	55,863

Figure 5.4 Delivered Source Lines of Code (DSLOC) per Full Time Equivalent (FTE)

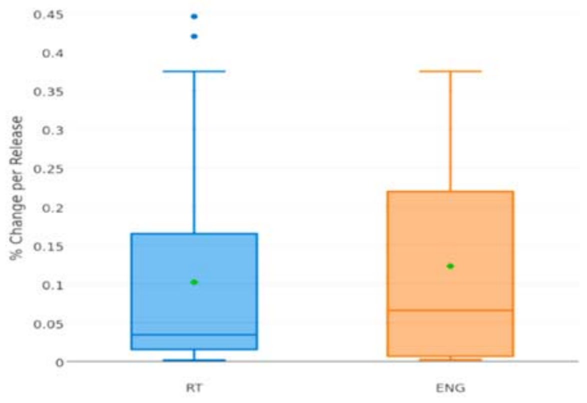
It appears initially that the number of FTEs to support a code base varies. However, there are only six date points for the ENG super domain. More data is needed to investigate the difference between all four super domains.

5.2.7 Baseline Percent Change

This analysis attempts to show how much (expressed as a percentage) the software changes in a release, as shown in Figure 5.5. Baseline percent change was calculated for each release as follows:

(New Code+ Modified Code) / Delivered Code (DSLOC)

The earliest baseline size reported was used to represent DSLOC. Only the Real Time and Engineering super Domains had sufficient data to derive Baseline Percent Change. The percentage change is a common input to software cost estimation models.



Statistic	RT	ENG
N	38	18
Mean	10.3%	12.3%
Min	0.2%	0.2%
Q1	1.6%	0.7%
Median	3.4%	6.6%
Q3	15.7%	21.7%
Max	44.6%	37.5%

Figure 5.5 Baseline Percent Change

5.3 CER and Benchmark Data Issues

There were a number of issues with the collected data. These issues contributed to the variation in the data making inferences and conclusions difficult. One issue is the lack of standardized software sustainment processes across the system portfolio. There was not an original standard set of activities (WBS elements) and milestones against which data could be collected.

A second issue was the inability to map executed cost/effort data to software sustainment output activities and software change products. Generally a program receives sustainment funding not linked to a specific set of changes. It is up to the discretion of the program on how the money is spent. Consequently it was impossible to know the costs of different types of changes, e.g., enhancements, defect fixes and cybersecurity. Compounding this issue is that multiple funding streams are often used to support sustainment activities and these streams are separately managed. This sometimes resulted in incomplete cost data.

A third issue specifically related to cost is that for many systems, the Government is heavily leveraged on contractors to do the work which limits insight into cost data. This resulted in incomplete reporting of contractor cost data.

Another issue is that release size is often not known. Either source lines of code for different code types (new, modified, reused, auto-generated, and deleted) or the number of software changes by type or priority is not tracked. This severely limits the ability to derive cost estimating relationships and benchmarks. Because of this, it is difficult to relate the amount of software sustainment work to funding.

5.4 Uncertainty: The Cost Risk - Uncertainty Estimation Determination (CRED) Model

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." [12] The Guide also states that each WBS element should be assessed for its level of cost, technical and schedule risk.

It is often the case in cost estimates that risk or uncertainty is not addressed. One reason given is that performing risk or uncertainty analysis is seen by programs as being too costly, too complex to perform, or the critical risks/uncertainties are perceived outside the control of the program, and therefore, cannot be managed. Nevertheless, programs that fail to adequately address risk or uncertainty in their cost estimates end up being programs that typically overrun their budgets and schedules, and fail to meet their technical objectives.

As part of the DASA-CE cost estimation initiative, a prototype Cost-Risk Uncertainty Determination (CRED) model was developed that 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. 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. 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. The third attribute category is used to assess the external factors that may impact a systems sustainment effort, such as policy mandates such as security requirements or budget reductions. The final attribute category involves the software sustainment environmental factors related to the “Red-Yellow-Green” program ratings discussed in Section 4.2. 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 allows for the generation of 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 risk and understand contributing factors.

The CRED model is currently being refined under Phase II. The plan is to validate the CRED model as more actual versus prediction software sustainment costs estimate information is gathered and analyzed.

6.0 INITIATIVE FINDINGS

The Army software sustainment cost estimation initiative has provided valuable insight into a portion of the software lifecycle that was previously misunderstood and underappreciated in the management of the software life cycle. Key top-level findings from the initiative are integral to improving the Army’s ability to more efficiently manage how sustainment resources are applied to multiple system and mission capability requirements. These insights include:

1. The need for objective software sustainment cost and performance information is universal across all Army decision levels. Especially critical are accurate near term and life cycle cost estimates that can objectively project software sustainment costs. All decision makers need to know, with some accuracy, what it will cost to sustain the software in a given system.

2. The availability, integrity, and usability of system software sustainment data is tied directly to the capability of the management and technical processes and discipline that govern the software sustainment environment. This is true at both the system and enterprise management levels.

3. An effective software sustainment decision information environment requires that software engineering change practices be directly aligned with funding/resource accounting processes and Army sustainment policy and statute. The misalignment of these factors drives inconsistencies in processes and associated data that limit objective software sustainment cost and performance analysis.

4. The current software sustainment cost environment is characterized by resources and funding being applied to establish and maintain multiple system change capabilities across many government and contractor organizations for each system. Each system is largely autonomous in its sustainment management, and this impacts the ability to manage software sustainment from an enterprise perspective. There is clearly a need to align software sustainment cost estimates, resource execution, and performance assessment directly to software sustainment activities and products across the Army's system base in a consistent manner.

5. Effective management of system software sustainment must recognize the changing nature of the system software life cycle. Development and integration of software derived mission capability occurs throughout the life-cycle, and the associated products released to the warfighter are continuous. Software development during acquisition and sustainment are becoming more similar than different, and the dividing line between software development and maintenance is no longer definitive and has become somewhat artificial. How we fund and manage the software life cycle in terms of capability integration, cyber-security constructs, and corrective changes must adapt.

6. The ultimate objective is to relate software sustainment investments across the system life-cycle to mission capability across either a single system or system portfolio. To do this first requires a software sustainment cost management process founded on accurate estimates of and effective performance tracking to the software change products that provide that mission capability.

7.0 FUTURE EFFORTS

The initiative is now in its second phase. DASA-CE and senior Army management is initiating efforts for collaboration across the services, DOD, and industry to further refine software sustainment data collection and analysis across multiple environments and technical domains. A key part of this collaboration will be the sharing of software sustainment data and joint development and validation of advanced cost estimation methods.

8.0 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] 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.

[3] 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.

[4] 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.

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

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

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

[8] 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.

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

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

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

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