Impact of Scope Changes on Software Growth

Dr. Jonathan Brown
Ms. Gail Flynn

Naval Surface Warfare Center, Dahlgren Division

Abstract: A recent Software Engineering Institute (SEI) technical report, the *Department of Defense Software Factbook,* summarizes historical Major Defense Acquisition Program/Major Automated Information System Software Requirements Data Report data for programs that have experienced software growth. The mean value reported for equivalent source lines of code growth is 106%. While accurate, this analysis captures total software growth, including the large impact of scope changes. Large, planned scope changes are outside the definition of what should be included in software growth. This paper introduces the notion of "Pure Software Growth," and differentiates large, planned scope changes from traditional software growth. Several large software programs are analyzed from this perspective to show the difference between pure and total growth and the unexpected impact this could have on estimates if not accounted for.

Keywords: software growth, pure growth, software size estimating, requirements creep, scope creep, software cost estimating

INTRODUCTION

Problem Statement/Purpose of Study

In software cost estimating, software size growth is expected and can have a dramatic impact on software cost estimation models, sometimes resulting in a 1-for-1 percentage increase in the final cost estimate. Historically, cost and software size estimators have attempted to predict software growth using subject-matter expert inputs and more recently, actual analysis of Department of Defense (DoD)-specific software trends (Jones & Hardin, 2007), (Lanham & Wallshein, 2015), (Clark, et al., July 2017). The software growth owing to scope that is added during the program is not insignificant. As seen in Figure 1, studies have shown that this scope, or requirements growth, can be upward of 160%.

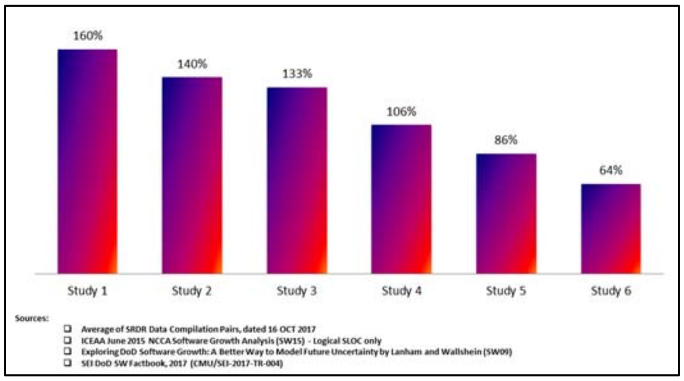

Figure 1

However, as Figure 2 shows, the historical data underlying these studies, e.g., Software Requirements Data Reports (SRDRs), capture all types of program growth, including growth because of scope changes (Clark, et al., July 2017) (Jones & Hardin, 2007) (Lanham & Wallshein, 2015) (OSD CADE, 2017). By comparing the initial software size to the final software size, the data are capturing all software growth of the original scope but also software growth owing to any scope that has been added to the program.

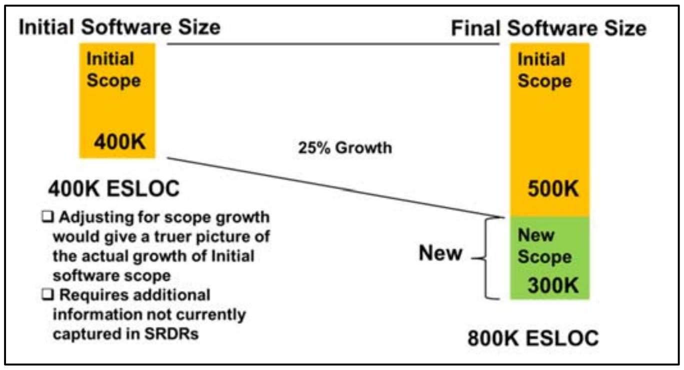
Initial Software Size
Final Software Size
Initial Scope
400K
25% Growth
Initial Scope
500K
400K ESLOC
Adjusting for scope growth would give a truer picture of the actual growth of Initial software scope
Requires additional information not currently captured in SRDRs
New
New Scope
300K
800K ESLOC

Figure 2

This paper defines and differentiates pure software growth from total software growth to differentiate between software growth of the original scope and growth added by new scope. This paper also analyzes several example systems in depth to determine the magnitude of pure growth vs. total growth, and demonstrates the potential impact on cost estimates and uncertainty analysis.

Definition of Software Size Growth

Software size growth is the change in software size from the initial estimate to the final actual. This growth happens for many reasons, as listed in Table 1.

Table 1

Module 12 of the ICEAA CEBok®	2008 NCAA Software Development Cost Estimating Handbook	2007 Software Code Growth (Jones)
Underestimating required SLOC	Size projection errors	Underestimating the amount of new SLOC
Poor understanding of initial requirements	Requirements volatility	Underestimating the software complexity
Code reuse optimism	Product functionality changes	Overestimating the expected use of existing SLOC, i.e., modified and unmodified SLOC
New requirements added during development	Human errors	

There are similarities and constant themes for likely causes of software growth. For our purposes, total software growth consists of the software growth because of:

- Software size projection errors (source lines of code [SLOC] to code A)
- Underestimated software complexity (SLOC for "a" vs. SLOC for A)
- Optimistic amount of reuse (only need A, reuse B vs. A+B)
- Changes in the software's scope (only A+B, not A+B+C)

While it is definitely a part of total software growth, changes in its scope should be treated differently than software growth caused by other reasons. To put it another way, this is similar to the uncertainty in the number of end units procured for a hardware estimate. The final number of units procured will likely not be the initial number of units assumed; however, no adjustment is made to the estimate for this uncertainty. When the number of units procured is changed, the estimate—and hopefully the budget—are adjusted accordingly. Additionally, the actuals reported in the selected acquisition report (SAR) are adjusted for the changes to initial units.

Completely unrelated software scope additions are similar and should be estimated separately and adjusted for in the historical data. To help differentiate this scope, we will use the definitions of growth from Module 12 of the ICEAA Cost Estimating Body of Knowledge (CEBoK) to define pure software growth as total software growth removing all growth caused by changes in its scope, as illustrated in Figure 3.

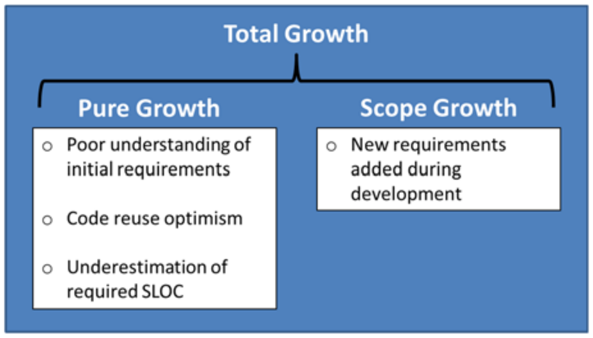
Total Growth
Pure Growth
Poor understanding of initial requirements
Code reuse optimism
Underestimation of required SLOC
Scope Growth
New requirements added during development

Figure 3

PURE SOFTWARE GROWTH ANALYSIS

Method

To illustrate this concept and to investigate the magnitude of pure vs total growth, we analyzed four separate programs. The programs were selected based on relevance and availability of data. To determine when and how much scope is added takes additional software data outside of that available in an SRDR. The specific data analyzed will be discussed under each example as it varies slightly from program to program.

Program 1

Program 1 is a large (~5000K DSLOC) complex DoD Combat Management System (CMS) software consisting of sensors, command and control, display, monitoring, engage, and training. The software development Program 1 analyzed was an upgrade to the existing CMS. The data analyzed were monthly reports of equivalent source lines of code (ESLOC), systems engineering technical review (SETR) packages, and program schedule. Figure 4 shows ESLOC change over time.

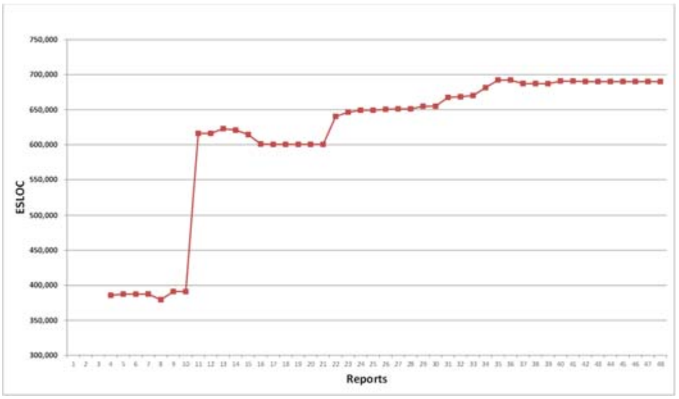

Figure 4

Notice the large spike at cost report # 10. Overlaying the SETR dates from the program schedule shows the large spike corresponds to critical design review (CDR) and the smaller spike corresponds to another minor review, as seen in Figure 5.

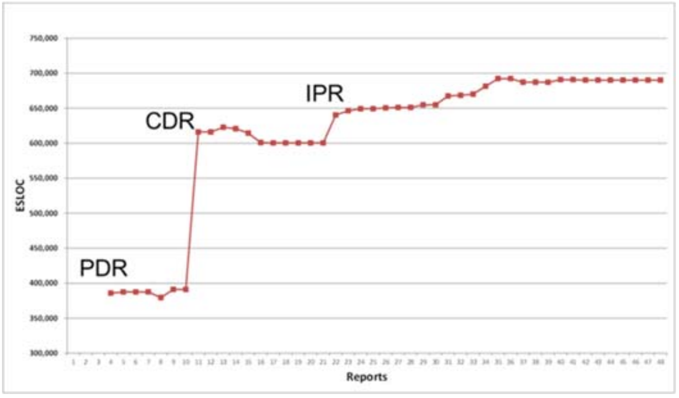

Figure 5

Reading the CDR documentation, we found scope was added for an open architecture redesign of the monitoring computer software configuration item (CSCI). Checking this against monthly reports verified there was a large increase, 140K, in the monitoring CSCI ESLOC. There was additional capability added to the Sensor CSCI. Checking the monthly reports we found a corresponding 56K ESLOC addition to the sensor CSCI in report #10. It was unclear what portion of this 56K from report #10 was tied to the capability increase and what was traditional growth. We assumed all growth from report #10 was tied to the capability increase and all growth not in report #10 was traditional growth. No additional increase was clearly tied to scope increases. Table 2 summarizes the results for Program 1.

Table 2

Pure Growth	28%
Scope Growth	51%
Total Growth	79%

Program 2

Program 2 is a large (~4000K DSLOC) complex DoD CMS software consisting of sensors, command and control, display, engage, and planning. The software development Program 2 analyzed was an upgrade to the existing CMS. The data analyzed were monthly reports of ESLOC, SETR packages, and program schedule. Figure 6 shows the change in ESLOC over time overlaid with the schedule.

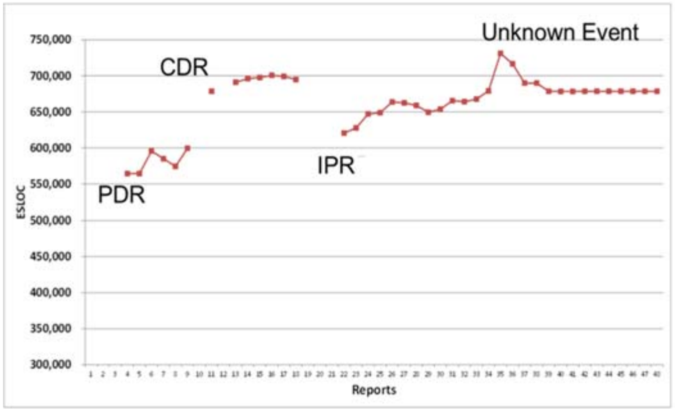

Figure 6

Unlike the previous program, there were gaps in the available data reports. In addition, there was a less-defined pattern. There were no specifics given in the SETR reports for why the ESLOC was rising and falling significantly. Judging from the graph, it is possible scope was added prior to preliminary design review (PDR) but removed prior to the next milestone, since the line would be more continuous if one shifted the CDR portion of the curve down ~75K ESLOC. There was also an anomalous event at report #35 but no mention was made in any report. Since no obvious scope was added, all growth was attributed as pure software growth. Table 3 summarizes the results for Program 2.

Table 3

Pure Growth	20%
Scope Growth	0%
Total Growth	20%

Program 3

Program 3 is a large (~4000K DSLOC) complex DoD CMS software consisting of sensors, command and control, display, engage, and planning. The software development Program 3 analyzed was a major upgrade to the existing CMS. The data analyzed were quarterly reports of ESLOC, SETR packages, and program schedule. Figure 7 shows the change in ESLOC over time overlaid with the schedule.

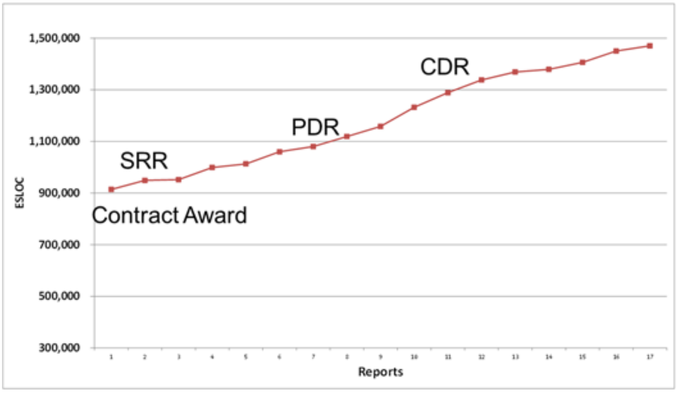

Figure 7

This program exhibited a relatively smooth software growth profile and did not trigger any immediate red flags. However, reading the SETR packages and notes of the quarterly reports we found there was a large change in the capabilities of the system; an updated target was not part of original scope. In the documentation, the developer summarized this change added 340K ESLOC from report #8 to 13. Table 4 summarizes the results for Program 3.

Table 4

Pure Growth	24%
Scope Growth	37%
Total Growth	61%

Program 4

Program 4 is a medium-sized (~2000K DSLOC) complex DoD CMS software consisting of sensors, command and control, display, and training. The software development Program 4 analyzed was an upgrade to the existing CMS. The data analyzed was 15 software metrics reports spanning 4 years of development. Figure 8 shows the change in ESLOC over time.

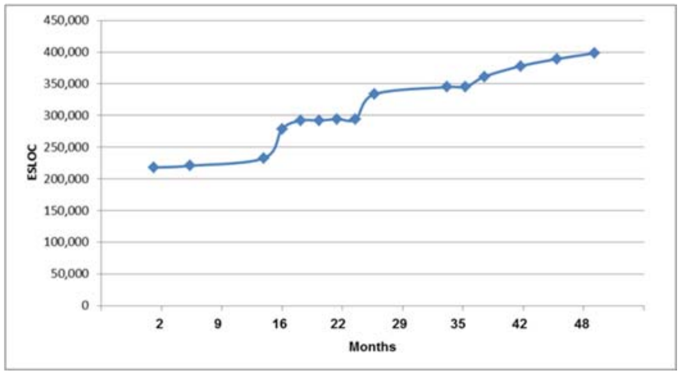

Figure 8

This data point was more challenging to analyze since we did not have access to the SETR information for this software development. However, the software metrics reports did describe software growth. While most reasons listed would be categorized as total growth (e.g., specific CSCIs required more updates than anticipated, one CSCI's actuals were higher than estimated), there were items identified that were direct scope increases. Scope was added for additional capability for sensor identification and unplanned changes driven by external interoperability changes. These corresponded to the "bump" just before May 2013. No identified scope increase drove the second bump in Feb 2014. Table 5 summarizes the results for project 4.

Table 5

Pure Growth	46%
Scope Growth	18%
Total Growth	64%

Examples Summary and Comparison to Recent Examples of Software Size Growth Analysis

Not all programs analyzed were determined to have identifiable scope increases. However, there were large differences between the measured pure software growth and total software growth for the programs that did experience measurable scope increases. The total software growth measured was also similar to the average ratios found for several recent studies, as seen in Figure 9.

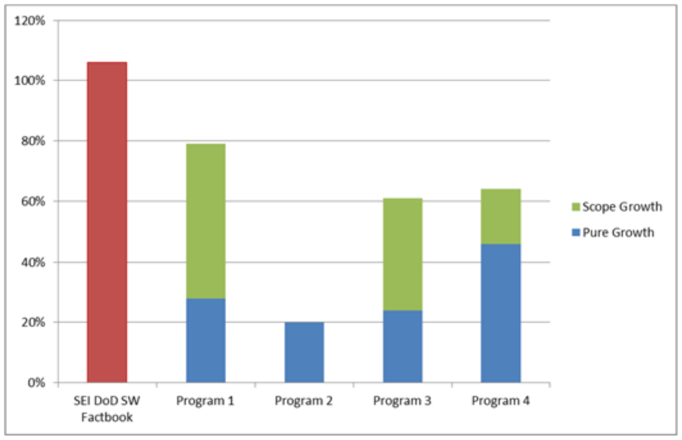

Figure 9

IMPACT ON SOFTWARE COST ESTIMATES

Methodology

A hypothetical software cost model was developed using generic and typical inputs. One year of development was calculated in base year 2018 (BY18 $) using the model illustrated in Figure 10:

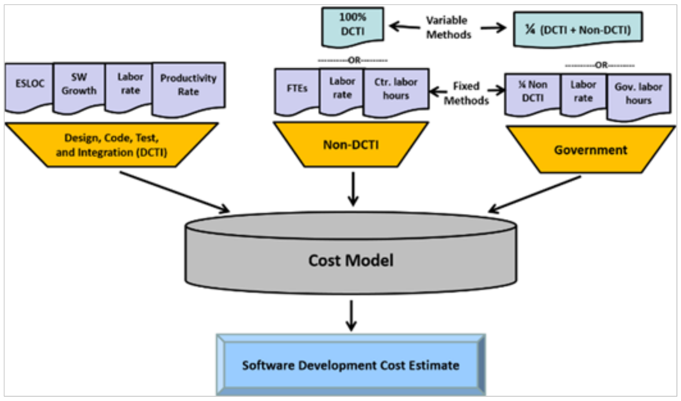

Figure 10

Based on the research of Major Defense Acquisition Programs (MDAPs) mentioned earlier, total growth was 0.79 and pure growth was 0.28. Also, if non-DCTI [design, code, test, and integration] was varied, so was government; likewise, if non-DCTI was fixed, so was government. ESLOC was then changed to analyze the impacts of the choice between total and pure growth on different program sizes.

The goal in the following subsection is to demonstrate the choice is an important one, regardless of which growth one chooses to use.

Point Estimate Impacts

One way to view the effects seen from this model is displayed in Figure 11, where cost estimates obviously increase as the effort increases, although at different rates. Notice the estimates are largest when using total growth with non-DCTI varied. The focus is not on the hour or dollar value outcome but on the percentage impact to a cost model based on software growth choice, total or pure.

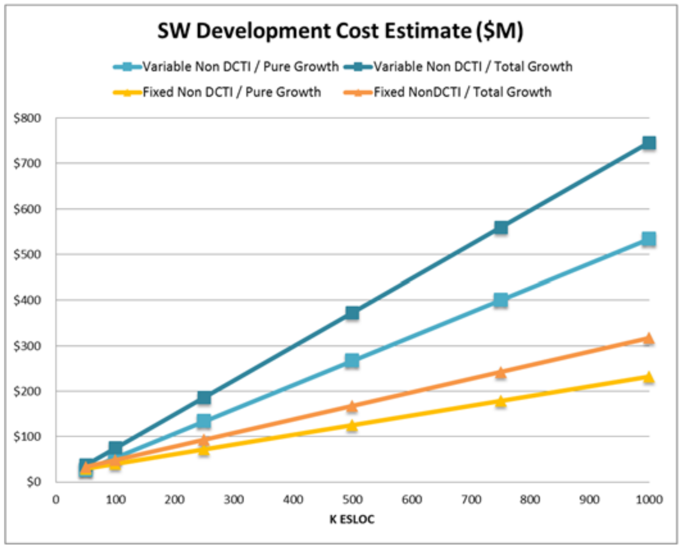

Figure 11

Figure 12 reflects results when non-DCTI fixed and variable methods are crosschecked at 50K ESLOC only. We can clearly see the impact the choice between pure and total estimate can have on one's software development effort, dependent on the size of the effort.

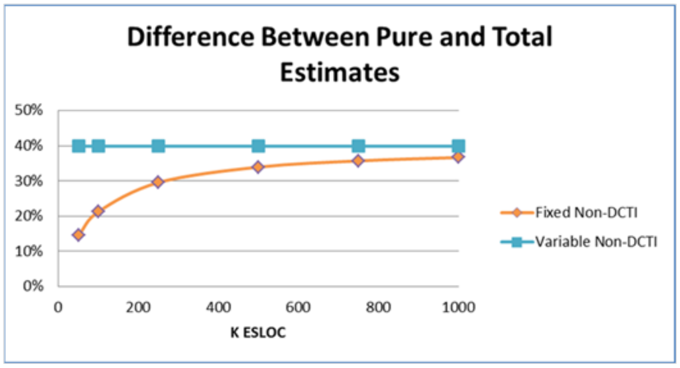

Figure 12

According to these results, as development efforts increase in size and the only variation in a model is the choice between pure growth (28%) and total growth (79%), using variable non-DCTI could result in a 40% difference in the cost estimate no matter the program size. If using a fixed non-DCTI, however, the difference in the cost estimate increases, approaching about 40%, as the size of the effort increases.

Figure 13 reflects results when non-DCTI fixed and variable methods crosschecked at 250K ESLOC only.

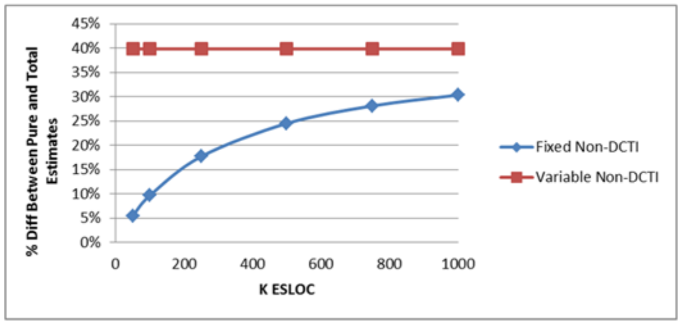

Figure 13

Figure 14 reflects the results when non-DCTI fixed and variable methods crosschecked at all ESLOC tested.

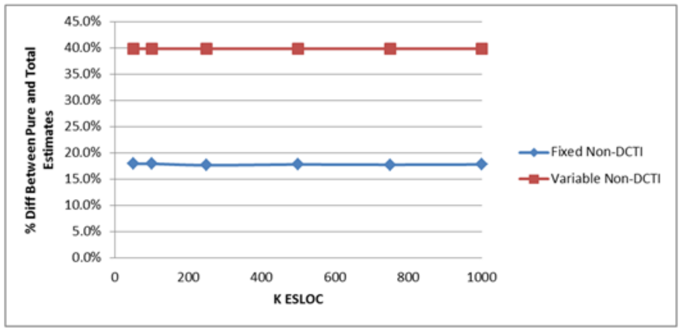

Figure 14

The results shown, based on our model, indicate that one would expect between a 15 and 40% difference for an MDAP software development cost estimate using total growth instead of pure growth; the larger the software development effort, the greater the difference between the cost estimates.

Cost Uncertainty Analysis Impacts

The subsequent subsection demonstrated the large impact using pure vs. total software growth can have on a cost model point estimate. It can also have an impact on risk bounds. Assuming an analyst has collected data on relevant historical examples of software growth and completed the analysis to calculate the pure and total software growth, there are five possible combinations that we considered for risk analysis, as seen in Figure 15.

	Pure Growth	Total Growth
Low	A (min)	D
Most Likely	B	E
High	C	F (max)

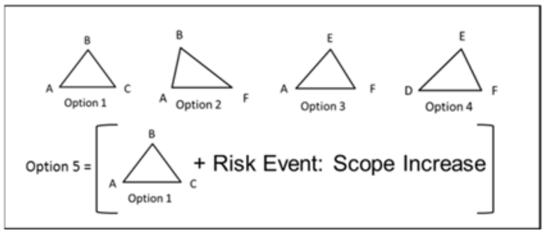

Figure 15

These options represent possible software growth outcomes, but next we'll determine if any of first four represent possible outcomes that should be captured in an uncertainty analysis. To frame this discussion, we use the three categories of uncertainty described in the *Joint Agency Cost Schedule Risk and Uncertainty Handbook* (JA CSRUH):

- Section 1.3.1, "Uncertainty to be Captured"
- Section 1.3.3, "Uncertainty That Could be Captured"
- Section 1.3.4, "Uncertainty That Should Not Be Captured"

The easiest two options to discuss are L-Pure, PE-Pure, H-Pure (Option 1) and L-Total, H-Total, PE-Total (Option 4) cases. In each case, the endpoints (or statistics) of the set are either pure or total, and the analyst has determined either pure or total software growth is the correct one to be captured. Option 1 reflects no scope creep, only pure growth; Option 4 reflects a situation in which the analyst believes the program (and/or all programs) will have scope creep. These are both valid and represent the minimum uncertainty for software growth since this range capture the uncertainty in the software growth equation. Capturing the uncertainty in the estimating equation falls into Section 1.3.1.

The other two options represent a mix of pure and total software growth concepts. We first examine L-Pure, PE-Total, H-Total (Option 3). Using total software growth as the point estimate, the analyst has assumed software growth owing to scope creep is the most likely outcome. In that case, the correct bounds would be L-Pure, H-Total since it is possible for a program to be completed without scope creep. This puts this option into Section 1.3.1. As we mentioned earlier, this scenario is similar to the L-Total, PE-Total, H-Total, since L-Pure and L-Total are similar.

The most interesting case is L-Pure, PE-Pure, and H-Total (Option 2). By definition the difference between pure and total software growth is scope increases or scope creep. This is explicitly called out in Section 1.3.3. Since scope creep is uncertain but possible, it should be included in the upper risk bounds or risk statistics for software size growth.

Figure 16 displays the results of our hypothetical dataset for the different options mentioned above. These estimates were calculated using Tecelote Research Inc.'s ACEIT RI$K software.

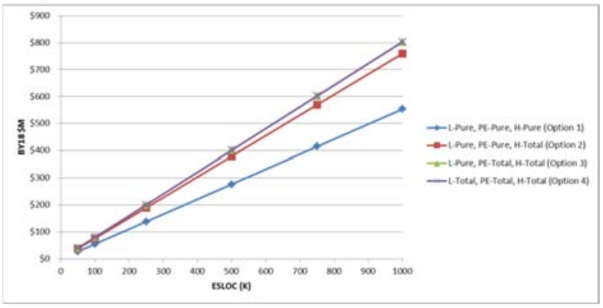

Figure 16

Figure 16 reflects probability allocated at 50%, and Figure 17 reflects probability at the mean. Again, the dollar value is not as important in this study as is the magnitude of the impact the risk bound choice can have on the cost estimate.

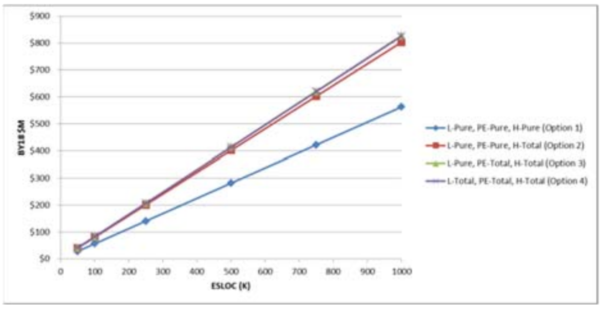

Figure 17

Figure 18 clearly points out the largest differences between the options and Option 4.

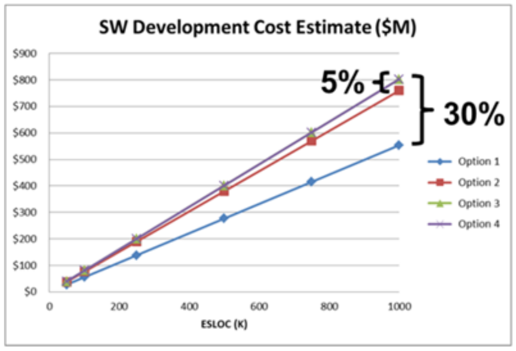

Figure 18

Option 4 was considered as the base option because total growth is most commonly used. Regardless of ESLOC, one can see from Figure 18 that the difference of software development cost estimates between base option and Option 1 is obviously much larger than between base option and Option 2 or Option 3.

These results all point toward a conclusion that Option 1 (using pure growth only) is not advisable in any case for MDAPs since it can lead to underestimation, but Option 2 (L-Pure, PE-Pure, and H-Total) has risk bounds that analysts need to consider since it can lead to significant differences in the estimate, irrespective of a program's size.

Another possibility that was not assessed with our model was Option 5. This option involves using pure growth only in the uncertainty for software growth but adding a risk event to the model where scope creep occurs. This would require an additional assumption to quantify how likely scope creep is to occur on a given program.

CONCLUSIONS

The goal of this research was to define the concept of pure software growth and to demonstrate the importance of choosing between using pure or total growth in an estimate. Three of four examples in this paper exhibited scope growth that accounts for a large percentage of total software growth measured for each program. One program exhibited no measurable scope growth. While it is only a small sample, it is likely similar to what one would encounter using a much larger dataset like the SRDR database.

Given the potential impacts to the cost estimate, as demonstrated, it is important to document whether total growth or pure growth is used in a cost estimate. Even if total growth is used, it is important to document that a portion on the software growth is avoidable if scope creep can be avoided in a program. If there is separated pure vs total software growth, analysts can quantify how much costs can be avoided. It is also essential to document risk boundaries and the assumptions used to support them.

FUTURE WORK

The programs analyzed in this research were only in the real-time, command and control domain. It would be judicious to also analyze programs in other real-time domains and in the super-domains of Engineering, Support, and Automated Information Systems.

REFERENCES

Clark, B., Miller, C., McCurley, J., Zubrow, D., Brown, R., & Zuccher, M. (July 2017). *Department of Defense Software Factbook.* Software Engineering Institute, Software Solutions Division. Hanscom AFB, MA: Carnegie Mellon University. Retrieved from http://www.sei.cmu.edu

Jones, R. P., & Hardin, P. (2007). Software Code Growth: New Approach Based on Historical Analysis of Actuals. *ISPA/SCEA Joint Annual International*

Conference and Workshop (p. 15). New Orleans, LA: Technomics.

Lanham, N., & Wallshein, D. (2015). Exploring DoD Software Growth: A Better Way to Model Future Software Uncertainty. *ICEAA Professional Development and Training Workshop* (p. 18). San Diego, CA: Naval Center for Cost Analysis.

OSD CADE. (2017, 10). SRDR Data Compilation. *SRDR Data Compilation as of 20171016*. Retrieved from https://www.osd.cade.mil