

Towards a Holistic, Total Engineering Cost Model

Gan Wang
BAE Systems
Reston, VA

gan.wang@baesystems.com

Jon Turnidge
BAE Systems
Reston, VA

jon.turnidge@baesystems.com

Alex Shernoff
BAE Systems
Reston, VA

alex.shernoff@baesystems.com

Ricardo Valerdi
MIT
Cambridge, MA

rvalerdi@mit.edu

Copyright © 2009 by Gan Wang, Alex Shernoff, Jon Turnidge, and Ricardo Valerdi. Published and used by INCOSE with permission.

Abstract. In this paper, we explore a new approach for a unified and interdisciplinary parametric model for estimating the total engineering effort in developing and delivering a software-intensive complex system. We begin by reviewing some of the limitations of using existing engineering discipline-focused tools for estimating total engineering cost and by articulating the benefits of such a holistic model. Applying a two step method combining heuristic analysis and data validation, we propose three hypotheses to expand the basic cost estimating relationship of COSYSMO, a systems engineering model, to the total engineering scope by including software size drivers. The implementation of the hypotheses and the validation approach are also discussed. We conclude the discussion by outlining the future work required to realize such a model and to apply it to supporting successful system development endeavours.

Introduction

What would you do if you were asked today to develop an engineering bid, say, for a cost proposal? How do you develop a total engineering estimate? An experienced estimator will lead you through the familiar steps: defining the project work scope, collecting and analyzing the historical data, selecting appropriate estimating techniques (i.e., analogy, parametric, bottom-up, or some combination of the three), and then developing the estimate using a cost model. When it comes to parametric models, there are many (commercial and academic) off-the-shelf tools or vendors to choose from, some are well known in the industry – SEER, PRICE, SLIM, COCOMO, just to name a few.

When you look into these tools, you will find that just about all of them are functionally or engineering discipline oriented. As shown in Figure 1, each model focuses on the tasks performed by a specific engineering function in support of a project.

There is one problem with this engineering discipline-focused approach: these tools provide only single function estimates, each with independent set of assumptions and mostly losing sight of the integrated engineering scope and the interdependency of functional tasks on the project. Using models that focus on individual functions requires someone to aggregate them into a complete, coherent estimate – the total engineering estimate – for a cost proposal. The process used to do this is often informal and the tool deployed is frequently ad hoc, most likely a

spreadsheet. And more importantly, the potential gaps and overlaps between the different estimates must be (but often not so adequately) resolved. Otherwise, you risk undermining your competitive position by double counting or, even worse, neglecting cost elements. Let's take a more careful look at this problem.

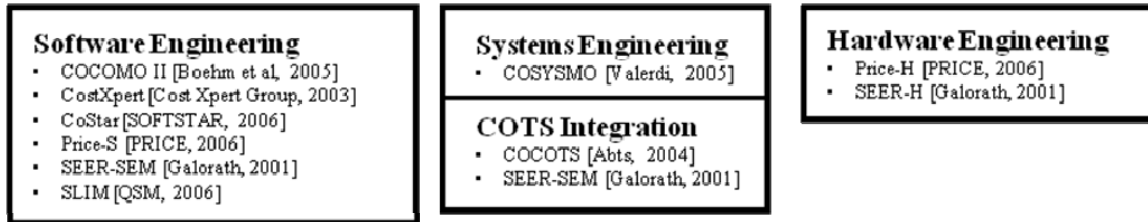


Figure 1. Functional Focus of Parametric Cost Models

Limitations of Functionally-Focused Tools

While these tools are useful for sub-elements of the project cost, they do not provide the complete answer. The shortfalls present several practical inconveniences.

First of all, when you assemble functional estimates into a total engineering estimate, there usually are gaps and overlaps in the estimate scope. Software estimating tools, such as SEER-SEM or COCOMO II, primarily estimate the development effort for software components or computer software configuration items (CSCIs), typically using the source lines of code (SLOC) based estimates for these items. There are factors in these models to account for the additional effort of integrating these items to the prime system or the prime mission product (PMP). Similarly, hardware models such as PRICE-H or SEER-H estimate the effort of developing hardware configuration items (CIs) using drivers like the component weight and size. They, once again, limit the estimate to these components under consideration. The overall system design, integration and test effort are ancillary considerations or after thoughts by these models.

For the system level development effort, the answer may be to use the Constructive Systems Engineering Cost Model (COSYSMO). It may be appropriate because its primary focus is on the system design and integration activities. This model has gained rapid popularity recently years in the industry (Valerdi 2008) and has been validated with data from several aerospace & defense companies. The issue is, however, where the hand-off point is to the component level development effort. Where does one draw the boundary between the system level model (e.g., COSYSMO) and the other models used for the component level estimates, so that there are no scope overlaps and, just as importantly, no effort gaps? There can be general application guidelines on how to reconcile the scopes (Wang, et al 2009), but it is difficult to institute hard boundaries that are consistently followed. The devils are in the details, as the exact scope and quality of an estimate is determined as much by the sizing inputs to the model as the historical data built into the calibrations. To adjust scope in the calibrations is a much more involved endeavor, requiring significantly greater effort and longer turnaround time. These guidelines are really stopgap measures that can prevent users from effectively benefiting from the potential value provided by the models.

For instance, the software estimate may include part of the integration effort related to the CSCIs under consideration. At the same time, the hardware estimate may also consider the

integration effort related to its CIs. The question is which estimate definitively covers the integration effort for the software CSCIs to go on to the hardware CIs. It is unclear since it depends on the assumptions made in both estimates. If we use COSYSMO and it covers the system integration effort, then you have to reconcile the integration efforts from all three estimates. Similar contentions may arise with design activities. How does one adjudicate the division of labor and ensure that each model provides a complementary estimate? On the other hand, which estimate covers specialty engineering activities such as security, information assurance, and environmental issues? How about efforts involved in operational tests, or site activation? Gaps and overlaps are an increasing larger issue when you start to consider other areas of the total engineering estimate.

The second shortcoming is the cumbersomeness involved in conducting rapid, “round-trip” sensitivity analysis and cost trades. As we have indicated above, generating the total engineering estimate is a two-step process, first the functional estimates and then aggregating them to the total system estimate. Any changes to the estimate would have to go through the same two step process. It is inconvenient, if one wishes to conduct quick sensitivity analysis at the system level, and requires a little bit of juggle between multiple applications running at the same time and possibly among different people with different expertise (different functional models are typically owned and operated by different functional groups in an organization).

The third problem is the non-trivial cost and investment required in developing such an estimate. To generate a total engineering estimate, it would likely require multiple experienced personnel to run the functional models as these specialized models typically require specialized training and skill sets. Someone must then integrate the total estimate at the system level by creating a separate estimate. Depending upon the size of system and availability of special-skilled people, this task could incur significant cost and untimely cycle time. Practical experience corroborates the fact that it is a significant undertaking to estimate cost in proposals and business pursuits with a reasonable degree of accuracy. Organizations strive to cut cost and to be lean to stay competitive in today’s market, but this is difficult to do when engineering cost tools are fragmented.

A major contributor to these shortcomings is that each cost tool was developed to operate independently and, therefore, did not consider the scope and impact of other complementary tools in a deliberate manner. Independent assumptions were made about the functional scope in the development and calibration of each cost model, typically with an exclusive focus on its own center piece, a HW or a SW component. However, it is impractical for any individual project to limit to either hardware development or software development. Most systems today require considerations of multiple engineering disciplines and expertise in order to meet the customer requirements.

The question is whether an alternative approach is possible to integrate multiple perspectives in a more holistic manner for estimating total engineering cost.

A Holistic Approach to Estimating Total Engineering Cost

A holistic parametric model that can provide a single estimate for the total system cost including all the functional elements is desirable in more ways than one. It can be used to create

a rough-order-of-magnitude (ROM) estimate in the early phase of system life cycle to analyze business opportunities, assess competitive position in cost and risk, and support go/no-go or make/buy decisions, all in short turn-around time. It can also provide cross-check and independent validation for other estimating methods such as functional estimates or bottom-up bids. It is ideal to conduct quick-turn sensitivity analyses and architecture-level trade studies of alternative solutions in an interactive manner, for example, by turning the sizing “knobs” and directly observing the sensitivity of individual parameters on the cost impact of the complete system. Most importantly, it provides significant opportunities for cost savings and cycle time reduction in developing engineering estimates and system bids.

In this paper, we examine the feasibility of developing a holistic, total engineering cost model for estimating the end-to-end, total system development effort, which includes all the functional elements. In the following sections, we discuss the modeling approach used in this research, outline a number of hypotheses required to establish the basic cost estimating relationship (CER), and describe the methods to validate these hypotheses. We then conclude with suggestions for follow-on work.

Modeling Strategy

The basic modeling assumption is that we can identify a set of sizing parameters or “size drivers” and define the corresponding Cost Estimating Relationship (CER) that establishes sufficiently significant correlation between certain parameters (independent variables) and the total engineering effort (dependent variable) in developing a system. To manage the scope of the problem, we limit our initial investigation to the software-intensive systems or projects, which include only two major cost components – systems and software. The rationale is that if the relationship can be established in software-oriented systems, the same modeling approach can be extrapolated to consider other hardware or supportability drivers and scopes. This incremental approach will enable early harvest of low hanging fruits and effectively apply lessons learned to solving the bigger problem.

The total engineering cost and effort can be systematically captured using a two-dimensional construct. The first is engineering scope. In order to define a coherent total engineering scope involved in a program, we use a generic, contractor Work Breakdown Structure (WBS) defined based on the MIL-HDBK-881 (1998) and the EIA/ANSI 632 (1999) standard. The top level breakdown of this structure is listed in Figure 2 (a more detailed structure is listed in the Appendix). The handbook provides a consistent and visible framework for identifying defense materiel items and contracts within a program¹. It provides a range of examples to illustrate the applicability of the WBS across aircraft, electronic/automated, missile, ordnance, ship, space and surface vehicle systems. The hierarchical structure provides multiple levels that are focused on defining increasingly detailed information on the tasks needed to develop and deliver a Prime Mission Product (PMP). It also offers uniformity in definition and consistency in application for developing the basic construct and the top-level elements of the WBS and supersedes MIL-STD-881B (1993) per DoD Directive 5000.2. A standard WBS is a necessary step in a consistent and mature engineering process. More importantly, organizations must rely on a standard structure

¹ The Department of Defense is taking the actions necessary to reinstate MIL-HDBK-881 into a proper standard, but this has yet to be approved by the Office of the Secretary of Defense and the Defense Standardization Council.

to categorize costs in order to provide required reports to the government (Stem, Boito & Younossi 2006).

1.0 – System/Project
1.1 – Integrated Project Management (IPM)
1.2 – Systems Engineering
1.3 – Prime Mission Product (PMP)
1.4 – Platform Integration
1.5 – System Test & Evaluation (ST&E)
1.6 – Training
1.7 – Data Management
1.8 – Peculiar Support Equipment
1.9 – Common Support Equipment
1.10 – Operational / Site Activation
1.11 – Industrial Facilities

Figure 2. Generic, contract work breakdown structure (WBS)

The second dimension is the engineering disciplines or functional groups. For purposes of this paper, the total engineering work that is required for the development and delivery of a system is performed by the following six functional groups, each defined by the following standards or documents shown in Table 1.

Table 1. Engineering Functions Comprising Total Engineering Cost on Projects

Engineering Function	Process and Tasks As Defined By
Systems Engineering	ISO/IEC 15288, EIA/ANSI 632
Software Engineering	ISO/IEC 12207, MBASE/RUP
Electrical Engineering	MIL-HDBK-881, EIA/ANSI 632
Mechanical Engineering	MIL-HDBK-881, EIA/ANSI 632
Support Engineering	MIL-HDBK-881, EIA/ANSI 632
Project Engineering Management	MIL-HDBK-881, EIA/ANSI 632

This two-dimensional construct is employed to ensure a consistent scope of engineering and, subsequently, a singular set of cost elements corresponding to that scope. We use this structure to substantiate the notion of “total engineering effort” in the following context of this paper.

We use COSYSMO and COCOMO II as the respective systems and software reference models for this investigation. Both COSYSMO and COCOMO II are open models and, therefore, flexible to accommodate future extensions and improvements. Commercial models such as TruePlanning (formerly called PRICE) and SEER are proprietary, which give no insight into fundamental algorithms embedded in the CERs and would not permit investigations at this level without cooperation of their respective owners.

The COConstructive COSt MOdel II (COCOMO II) (Boehm 2000) was developed at the Center for Systems and Software Engineering at the University of Southern California (USC). It has been a widely used and accepted model among industry and government stakeholders since its initial creation in 1981. The basic assumption behind COCOMO II is that software engineering effort can be estimated as a function of the size of the software, in terms of lines of code or function points, and a series of effort multipliers. The COCOMO II approach has proven to be robust across a number of domains including aerospace & defense, IT, financial services and embedded systems. To date, a number of derivative models have been created to address specific needs of the estimation community including quality, product line investment, productivity and security (Boehm, Valerdi, Lane, et al 2005).

One of the most recent derivatives of COCOMO II, the Constructive Systems Engineering Cost Model (COSYSMO) (Valerdi 2008), is a parametric model for estimating the systems engineering and integration effort required for the conceptualization, design, test, and deployment of software and hardware systems and executing projects that develop such a system. The model has attracted quick attention from industry and government alike. Several organizations have developed internal implementation of the model including BAE Systems, Lockheed Martin and Raytheon (Valerdi, et al, 2007). At BAE Systems an internal estimating tool has been developed by calibrating the model to its own historical program data and used in various bid and proposal activities with satisfactory results (Wang, Valerdi, Boehm, et al, 2008). Increasing demands have risen from complementary engineering communities about the possibility of developing a similar model to estimate the total engineering scope.

COSYSMO defines a parametric relationship that provides a statistical relationship between systems engineering effort under nominal schedule, in person months, and four size drivers – system requirements (REQ), system interfaces (IF), system algorithms (ALG), and operational scenarios (SCN). COSYSMO defines a sizing quantity called “system size” from a weighted sum of the four size drivers. In addition, it considers the concept of engineering reuse by augmenting the system size with a five-category reuse model (Wang, Valerdi, et al, 2008). The parametric relationship is further adjusted by fourteen effort multipliers², which capture the understanding, complexity, operations, people and environmental factors that produce cost savings or penalties. The relationship is shown in Equation 1:

$$PM_{SE} = A \cdot \left[\sum_k \left(\sum_r w_r (w_{e,k} \Phi_{e,k} + w_{n,k} \Phi_{n,k} + w_{d,k} \Phi_{d,k}) \right) \right]^E \cdot \prod_{j=1}^{14} EM_j \quad (1)$$

Where:

PM_{SE} = *Systems Engineering* effort in Person Months (under nominal schedule)

A = calibration constant derived from historical project data

k = {REQ, IF, ALG, SCN}

² Requirements understanding, architecture understanding, level of service requirements, migration complexity, technology risk, documentation, # and diversity of installations/platforms, # of recursive levels in the design, stakeholder team cohesion, personnel/team capability, personnel experience/continuity, process capability, multi-site coordination and tool support.

$\mathbf{r} = \{\text{New, Modified, Adopted, Managed, Deleted}\}$

w_r = weight for defined levels of reuse

w_x = weight for “easy”, “nominal”, or “difficult” size driver

Φ_x = quantity of “k” size driver

\mathbf{E} = represents diseconomies of scale

\mathbf{EM} = effort multiplier for the j_{th} cost driver. The geometric product results in an overall effort adjustment factor to the nominal effort.

The approach we take is to examine the possibility of extending the COSYSMO estimating relationship to the total engineering cost by correlating the COSYSMO size drivers to the total engineering scope and by expanding the CER to include other functional drivers such as source lines of code (SLOC). The rationale is simple. COSYSMO covers the end-to-end system development life cycle, with emphasis on the front (system design) and back (system integration) ends. The estimate covers the top layer of the engineering cost but not those at the lower implementation levels. In other words, it provides the “width” but not the “depth”. It is conceivable, however, that its scope can be expanded vertically to include lower-level activities as described in the system Vee-model (Forsberg & Mooz 1995).

With this strategy in mind, we approach this problem in two steps:

1. Heuristic-based modeling. In this step, we propose a number of hypotheses based on experience and empirical observation of historical data that augments the basic relationship in equation (1) with the addition of new size drivers.
2. Validation of hypotheses through data collection and analysis. We collect historical program data based on the modified relationship and examine the correlations that substantiate the proposed hypotheses.

The next section provides a detailed, step-by-step derivation of each hypothesis.

Formulation of Hypotheses

We propose three hypotheses about the basic cost estimating relationship of COSYSMO in order to expand its estimate scope to total engineering effort, as follows. Keep in mind that COSYSMO, as defined in equation (1), estimates only the systems engineering and integration effort as stipulated in EIA/ANSI 632 (1999).

Hypothesis #1: *There is sufficient correlation between the COSYSMO size drivers – system requirements, system interfaces, system-specific algorithms, and operational scenarios – and total engineering effort in a software-intensive project.*

This hypothesis implies that the COSYSMO parametric relationship can be expanded to estimate the total engineering scope of a system, with minimum change of its model form:

$$PM_{TE} = A \cdot SS_{SE}^E \cdot \prod_{j=1}^{14} EM_j \quad (2)$$

Where:

\mathbf{PM}_{TE} = Total Engineering effort in Person Months (under nominal schedule)

\mathbf{SS}_{SE} = System Size, and

$$SS_{SE} = \sum_k \left(\sum_r w_r (w_{e,k} \Phi_{e,k} + w_{n,k} \Phi_{n,k} + w_{d,k} \Phi_{d,k}) \right) \quad (3)$$

A keyword used in the hypothesis is the word “sufficient”. It intends to identify the degree to which the modified CER can provide an adequate level of estimating fidelity, which would be satisfactory for certain intended model applications. The significance of this hypothesis is that a simple model can be established in a top-down manner, without excessive levels of detail, to estimate the total, end-to-end engineering effort for developing a system. The immediate applications include early life cycle ROM estimates, trade studies for architecture and technical solutions, should cost, target cost, and cross-check or independent validation of estimates using alternative methods. All these with implicit simplicity of the model form.

To further improve correlation and, therefore, the estimating accuracy, we explore the possibility of expanding the system size definition to include other size drivers. The system size defined in equation (2) is in terms of equivalent requirements, a.k.a., eReqs. In other words, the size of the system estimated is in terms of number of requirements. This is easily understood and accepted notion as requirements are used to specify the complete system before it is built. In this context, however, this means that, in order to augment the system size with other size drivers, the quantities introduced need also to be requirements based to maintain the unit of measure. This leads to the second hypothesis.

***Hypothesis #2:** There is correlation between software requirements and source lines of code (SLOC) count in software-intensive projects.*

This hypothesis can be illustrated by a preliminary study result as shown by the scatter plot in Figure 3, in which multiple programs are plotted in terms of software requirements and SLOC count. In this context, software requirements are those requirements allocated to the software team from the systems team. They are typically derived from the system requirements by the requirement analysis and allocation activities at the system level. However, they do not include those requirements that are further derived at software level. The same classification rule applies as in counting system requirements, with three levels of difficulty and five categories of reuse. The SLOC is counted according to the code counting process defined by Boehm, et al (2000), in terms of equivalent source lines of code or eSLOC. These guidelines were developed to ensure a high level of consistency required in counting these requirements.

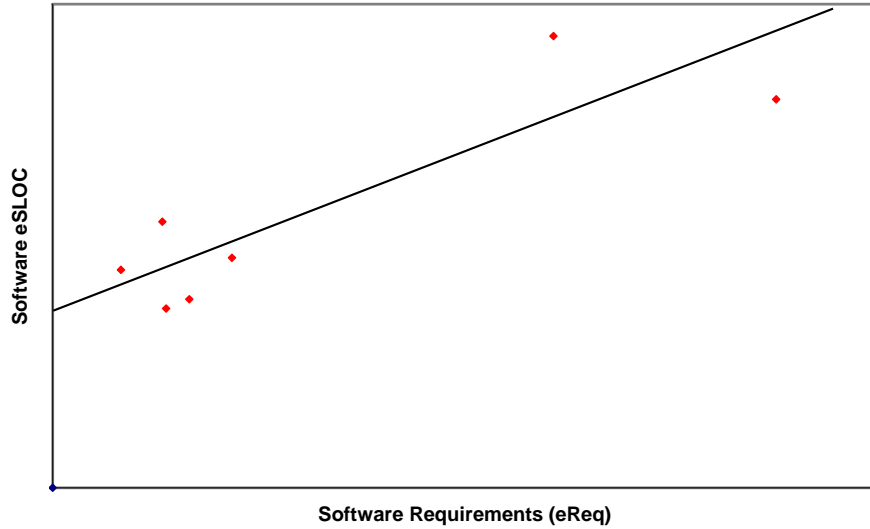


Figure 3. Relationship between Software Requirements and eSLOC (Preliminary Study Result)

Although not exceptionally strong, the plot does show reasonable positive correlation between SLOC and software requirements, and seemingly corroborates the assertion of correlation between software requirements and software code count. Better refined counting rules for software requirements may conceivably lead to enhanced correlation with SLOC count. These rules, and their consistent application, can further help strengthen the correlation between SLOC (software size metric) and system requirements (systems engineering size metric), which directly relates system definition to software implementation. This relationship between software and systems can be expressed as:

$$SS_{SW} = \sum_r w_r (w_e \Phi_e + w_n \Phi_n + w_d \Phi_d)_{SW} = f(SLOC) \quad (4)$$

Where,

- Φ_x = software requirement count
- $\mathbf{r} = \{\text{New, Modified, Deleted, Adopted, Managed}\}$
- w_r = weight for defined levels of reuse
- w_x = weight for “easy”, “nominal”, or “difficult” size driver

If the above relationship can be empirically validated, we can propose to further expand the COSYSMO-defined system size by including a software component. Therefore, Equation (1) can be modified as follows:

$$PM_{TE} = A \cdot (w_y \cdot SS_{SE} + w_s \cdot SS_{SW})^E \cdot \prod_{j=1}^{14} EM_j \quad (5)$$

Where,

SS_{SW} = Incremental System Size contributed by software engineering metrics
 SS_{SE} = Incremental System Size contributed by systems engineering metrics
 w_y = weight for system-level size drivers
 w_s = weight for software size driver

and

$$w_y + w_s = 1.0 \quad (6)$$

The incremental system size by software introduces an additional interchangeable and complementary size driver: software requirements or SLOC count. The correlation function $f()$ provides a bridge from a SLOC-based quantity to a requirement-based one. The selection of either is a matter of preference or may be determined by the availability of historical data. However, we recognize the fact that one of the de facto standard sizing parameters for software is SLOC³.

The weights in Equation (5) measure the contributions to the overall system size by the four system drivers (i.e., requirements, interfaces, algorithms and scenarios) and the software driver (software requirements or SLOC). On the other hand, the values for the weights in Equation 6 can be determined by the expected work share between the systems engineering and software engineering functions. They can be calibration dependent, determined by the typical balance of the two functions in specific system implementations. A strategy for implementation can be through an iterative, two-step process. The first step applies the relationship in Equation (2) to obtain a ROM estimate and high-level breakdown of the effort according to the WBS as in Figure 2. The ratio derived from the effort associated with element 1.3 (PMP), to which any software CSCI-level efforts should belong, relative to those of other elements can be used as the basis for the weight calculation. Step two then applies the expanded relationship in Equation (5) that now includes the SLOC as a driver to obtain a higher-fidelity estimate.

However, up to this point, we still cannot meaningfully corroborate the expanded CER in equation (4) simply because the expanded system size may not result in a stronger correlation with the total engineering effort than the original COSYSMO defined system size as in Equation (2). To substantiate the relationship, we must consider the following hypothesis.

Hypothesis #3: There is a stronger correlation between software-expanded system size and the total engineering effort in software-intensive projects.

The key word is “stronger”. Otherwise, the expanded relationship offers no practical use.

³ The other widely-used software sizing approach involves function points (Jones 2008)

Intuitively, however, the relationship seems to make sense. As the system drivers capture the engineering activities and effort at the system design and integration level, SLOC count can more accurately capture the low-level implementation effort related to activities such as coding and unit testing. The incremental system size from SLOC could bridge the gaps where the four system-level drivers may have weak predictive power, particularly in WBS 1.3 or at the software subsystem implementation level.

If a stronger correlation can be established, then the expanded system size in Equation (5) is used in lieu of Equation (2), in later system design and implementation phases when the SLOC count is available or can be assessed. Whereas Equation (2) can be applied in earlier phases when detailed data such as SLOC is hard to obtain. Both relationships apply, but are best used in different system life cycle maximizing the benefit of each.

With hypotheses #2 and #3, the implication is that a total engineering model can be established, with higher level of confidence and fidelity, to estimate the end-to-end engineering effort for a system. If validated, this model can offer a practical alternative to existing functional-based models by providing a straightforward estimating approach and by mitigating the potential gaps and overlaps resulting from the integration of estimates from disparate models and tools to arrive at the total engineering cost at the system level.

One of the biggest advantages of the models as stated in equations (2) and (5) is the simplicity of the CERs, fundamentally changing the approach of how total engineering estimates are developed. It potentially offers significant cost savings and cycle time reduction opportunities in the estimation process and, more importantly, a systematic approach that yields more accurate estimates; both of which are welcomed steps towards improving market competitiveness.

Validation of Hypotheses

The validation of the proposed hypotheses above involves the collection and analysis of the historical program data. Between 2007 and 2008, we embarked on a data collection activity in conjunction with an internal development initiative led by the lead author at BAE Systems. Historical project data are collected from the major sites across the country between multiple lines of business (LoBs) in the Electronics, Intelligence and Support (EI&S) Operating Group (OG). The data identified are mostly engineering development and manufacturing projects, spanning the entire system development life cycle from concept to development, testing, and delivery. They are then organized by business areas and product lines, as well as a set of other contextual criteria. To support validation of hypotheses #2 and #3, we identified a subset of projects that are software development in nature.

Due to the variety of project and system types and legacy organizations involved, we have implemented a number of mechanisms that ensure the required level of consistency and fidelity in collected data.

At step one, we must determine what total engineering effort means or what constitutes the engineering scope as included in PM_{TE} on the left-hand side of Equation (2). This was necessary to ensure that calibration data was consistently collected and the estimated effort consistently

reported. The original COSYSMO defines its estimating scope by loosely aligning with EIA/ANSI 632 standard (Valerdi 2008) which we tried to emulate as closely as possible.

For purposes of modeling total engineering cost on a project, we institute the two-dimensional data collection structure as discussed in Section 2 – a reference contract WBS and a generic OBS. A deliverable-oriented structure centered on developing and delivering a PMP, the WBS, as in Figure 2, is designed to capture the complete engineering scope in a system development project. Also as stated, this structure is defined based on the MIL-HDBK-881 (1998) and the EIA/ANSI 632 (1999) standard to enforce consistency and gain acceptance. The work defined spans across multiple functions as defined by the OBS – systems, software, hardware, and support engineering – and applies to a nominal system development life cycle, from concept definition, to design and build, integration and verification, and to transition and validation of the system.

As the historical program data have also been captured in mostly different and disparate structures and, in many cases, using different financial systems, we went through a diligent and thorough process of mapping cost elements from historical structures into this WBS. The process was facilitated by the extensive training classes provided and cross-organization stakeholder workshops held to achieve agreement and consensus. With up to fourth level breakdown, the WBS specifies sufficient granularity and was able to account for all the engineering activities consistently. It was a time-consuming process but allowed for the data analysis to be carried out on a more accurate and an “apple-to-apple” comparison basis. A side lesson learned from this exercise was that a unified data structure, if implemented for the financial data collection (which may not be as a trivial endeavor as it sounds), may greatly benefit the metrics and measurement process and enable the engineering estimating capability.

The systems engineering size drivers (i.e., requirements, interfaces, algorithms, and scenarios), as well as the fourteen cost drivers, are based on the improved COSYSMO definition in the extended model (Wang, Valerdi, et al, 2008), which defines a driver classification construct with three levels of difficulty and five categories of reuse. The SLOC count is based on the standard COCOMO II definition in terms of eSLOC, which accounts for new, adapted and reused code with consideration of design modification, code modification, and code integration (Boehm, et al 2000). The COCOMO II cost drivers and scale factors are also collected as defined.

The following studies are conducted for validation of the stated hypotheses.

1. Total Engineering Effort vs. SS_{SE} . To substantiate Hypothesis #1, scatter plots of projects are developed based on similarity in project characteristics. The system size SS_{SE} is defined as in Equation (3) and the total engineering effort recorded according to the WBS defined in Figure 2. The degree of correlation is examined by applying regression analysis to the data and observing the goodness of fit. The derived mean function is the calibration for the sample population.
2. Number of Software Requirements vs. eSLOC. The software requirements are defined as the system derived requirements that are flown down to software through the requirement allocation process. The eSLOC count is based on the COCOMO II definition. The correlation is examined using distribution of software-oriented

projects. Regression of the data is also attempted to evaluate the goodness of fit. The objective is to substantiate Hypothesis #2.

3. Total Engineering Effort vs. Expanded System Size for Total Engineering or SS_{TE} . The correlation is analyzed with the same dataset as in the study 2 using scatter plotting and regression. The result is then compared with those in study 1. If improved correlation can be observed, then Hypothesis #3 is supported.

The data collection activities are still in progress as of the writing of this paper. Therefore, there is no conclusive evidence to support or reject the hypotheses. However, we can share some of the early results which may show an early trend. Figure 4 below displays a scatter plot of a small group of historical program data collected for validating Hypothesis #1, as described in the study 1 above. Figure 3 above displays the work-in-progress results for the correlation between software requirements and eSLOC as described in the study 2.

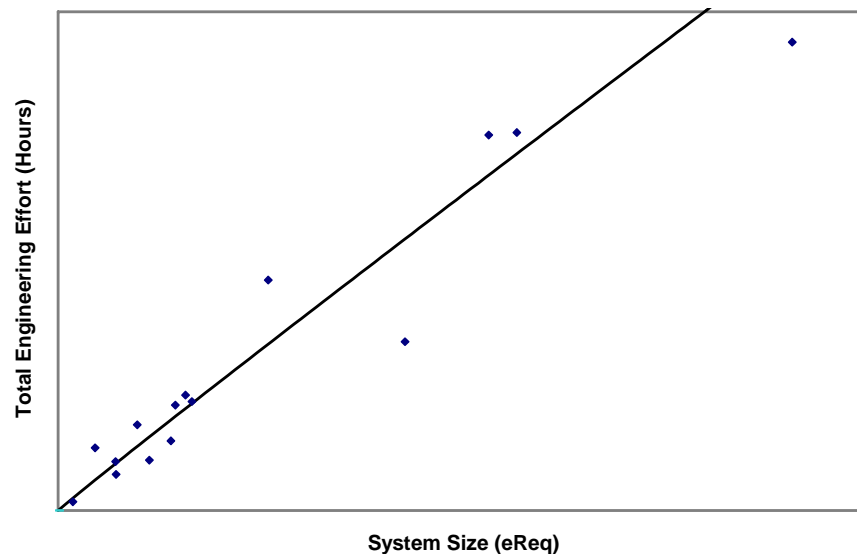


Figure 4. Scatter Plot of Historical Program Data Points

The data in Figure 4 is a subset of a much larger group of historical program data still in collection. (Other data points are still in progress.) Though still early in the validation process, the result is encouraging. The data set clearly exhibits the proper trend between the system size and the actual total engineering effort and a good correlation among the data points. A calibration can be derived from the set with a reasonable estimating fidelity. This early data collection result shows a promise in validating Hypothesis #1.

Conclusion and Future Work

In this paper, we examined the feasibility of a holistic, total engineering cost model. We proposed three hypotheses based on a modified COSYSMO parametric relationship. The goal is

to develop a total engineering model based on a set of system and software size drivers to estimate the total, end-to-end engineering effort in developing a system. If validated, this model can be applied to develop estimates of software-intensive systems and projects in a top-down manner that otherwise depend on successful aggregation of independent software and systems engineering estimates.

A logical corollary hypothesis is the extension of system size to including software function points, which is a candidate for follow-on study. The same methodology can be extended to explore the model expansion to hardware.

A holistic model of this type can significantly improve the efficiency and effectiveness of developing ROM estimates, conducting technical solution trade studies, providing validation to bottom-up and functionally oriented estimating approaches. Most significantly, it can be applied with a potential of great cost savings and cycle time reduction and, hence, greatly improve market competitiveness for organizations.

The validation work is on going through data collection and analysis activities at BAE Systems. Early result shows encouraging promises for Hypothesis #1 and, to some degree, Hypothesis #2. The validation work is in progress for analysis and validation of all the hypotheses. In addition, an extended model prototype has been developed based on these hypotheses to develop total engineering ROM estimates. The goal is to operational pilot this model and to evaluate its effectiveness in the proposed applications. We are prepared to present the ongoing results as progress is made.

Some possible limitations of this approach may include the ability to generalize the results to project domains outside of those considered in the sample population (i.e., engineering development and manufacturing projects). Another is that the approach is prone to inconsistency in counting rules applied to different data collection approaches, particularly when the data is collected at the different levels of system abstraction. The third is the potential mismatch across organizations in defining the total engineering tasks and scope and how much these definitions deviate from the MIL-HDBK-881 handbook/standard.

Despite these limitations we feel that the expanded CERs defined in this paper can provide a useful step in better understanding the relationships between the cost and cost drivers of various engineering disciplines and their eventual impact on the cost of the total system. We hope to continue evolving these ideas and validating the concept, and to refine our approaches for the benefit of more accurate and more effective cost estimating techniques to support successful development of systems.

Appendix

Reference Contract Work Breakdown Structure

(Based on MIL-HDBK-881)

- 1.0 System / Project**
- 1.1 *Integrated Project Management (IPM)*
- 1.1.1 Technical Management
- 1.1.2 Technical Reviews
- 1.1.3 Change Management
- 1.1.4 Technical Process and Quality Management
- 1.1.5 Acquisition & Supply Management (Subcontract & Technical Oversight)
- 1.1.6 Information Technology & Infrastructure
- 1.1.7 Dismantle and Disposal
- 1.2 *Systems Engineering*
- 1.2.1 Systems Engineering Management
- 1.2.2 ConOps & Stakeholder Analysis
- 1.2.3 Requirement Analysis & Management
- 1.2.4 Prime Mission Product (PMP) Design
- 1.2.5 Modeling & Simulation
- 1.2.6 Logistics Engineering
- 1.2.7 Reliability, Maintainability, Safety (RMS) Engineering
- 1.2.8 Specialty Engineering
- 1.3 *Prime Mission Product (PMP)*
- 1.3.1 Subsystem / Configuration Item (CI) 1...n (Specify Names)
- 1.3.1.1 IPT Engineering Management
- 1.3.1.2 Design
- 1.3.1.3 Design Analysis and Verification
- 1.3.1.4 Construction/Acquisition
- 1.3.1.5 Integration, Assembly, Test & Checkout (IATC)
- 1.3.1.6 Discrepancy Report (DR) Maintenance
- 1.3.2 PMP Application Software
- 1.3.2.1 IPT Engineering Management
- 1.3.2.2 Design
- 1.3.2.3 Construction/Acquisition
- 1.3.2.4 Integration, Assembly, Test & Checkout (IATC)
- 1.3.2.5 Discrepancy Report (DR) Maintenance
- 1.3.3 PMP System Software
- 1.3.3.1 IPT Engineering Management
- 1.3.3.2 Design

- 1.3.3.3 Construction/Acquisition
- 1.3.3.4 Integration, Assembly, Test & Checkout (IATC)
- 1.3.3.5 Discrepancy Report (DR) Maintenance
- 1.3.4 PMP Integration, Assembly, Test & Checkout (IATC)
- 1.3.5 Operations/Production Support
- 1.4 *Platform Integration*
- 1.4.1 External Interface & Technical Liaison Coordination
- 1.4.2 Transition to Use
- 1.4.3 Initial Spares & Repair Parts
- 1.5 *System Test & Evaluation (ST&E)*
- 1.5.1 ST&E Management
- 1.5.2 Design Solution Verification
- 1.5.3 Development Test & Evaluation (DT&E)
- 1.5.4 Operational Test & Evaluation (OT&E)
- 1.5.5 ST&E Mock-ups / Prototypes / Simulations & Test Equipment
- 1.5.6 ST&E Test & Evaluation Support
- 1.5.7 ST&E Test Facilities
- 1.6 *Training*
- 1.6.1 Equipment
- 1.6.2 Services
- 1.6.3 Facilities
- 1.7 *Data Management*
- 1.7.1 Technical Publications
- 1.7.2 Engineering Data
- 1.7.3 Management Data
- 1.7.4 Support Data
- 1.7.5 Data Repository
- 1.8 *Peculiar Support Equipment*
- 1.8.1 Peculiar Test & Measurement Equipment
- 1.8.1.1 IPT Engineering Management
- 1.8.1.2 Design
- 1.8.1.3 Design Analysis and Verification
- 1.8.1.4 Construction/Acquisition
- 1.8.1.5 Integration, Assembly, Test & Checkout (IATC)
- 1.8.1.6 Discrepancy Report (DR) Maintenance
- 1.8.2 Support & Handling Equipment
- 1.8.2.1 IPT Engineering Management
- 1.8.2.2 Design
- 1.8.2.3 Design Analysis and Verification

- 1.8.2.4 Construction/Acquisition
- 1.8.2.5 Integration, Assembly, Test & Checkout (IATC)
- 1.8.2.6 Discrepancy Report (DR) Maintenance
- 1.9 *Common Support Equipment*
- 1.9.1 Common Test & Measurement Equipment
- 1.9.1.1 IPT Engineering Management
- 1.9.1.2 Design
- 1.9.1.3 Design Analysis and Verification
- 1.9.1.4 Construction/Acquisition
- 1.9.1.5 Integration, Assembly, Test & Checkout (IATC)
- 1.9.1.6 Discrepancy Report (DR) Maintenance
- 1.9.2 Support & Handling Equipment
- 1.9.2.1 IPT Engineering Management
- 1.9.2.2 Design
- 1.9.2.3 Design Analysis and Verification
- 1.9.2.4 Construction/Acquisition
- 1.9.2.5 Integration, Assembly, Test & Checkout (IATC)
- 1.9.2.6 Discrepancy Report (DR) Maintenance
- 1.10 *Operational / Site Activation*
- 1.10.1 System Assembly, Installation & Checkout (On-Site)
- 1.10.2 Contractor Technical Support
- 1.10.3 Site Construction
- 1.10.4 Site Conversion / Upgrade
- 1.11 *Industrial Facilities*
- 1.11.1 Construction
- 1.11.2 Acquisition / Modernization
- 1.11.3 Maintenance

References

- Abts, C. *Extending the COCOMO II Software Cost Model to Estimate Effort and Schedule for Software Systems Using Commercial-Off-The-Shelf (COTS) Software Components: the COCOTS model*, PhD Dissertation, University of Southern California, 2004.
- Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D. J. and Steece, B., *Software Cost Estimation With COCOMO II*, Prentice Hall, 2000.
- Boehm, B., Valerdi, R., Lane, J., Brown, A. W., COCOMO Suite Methodology and Evolution, *CrossTalk - The Journal of Defense Software Engineering*, Vol, 18, No. 4, pp. 20-25, 2005.
- Chen, Y., Boehm, B., Madachy, R., Valerdi, R., An Empirical Study of eServices Product Sizing Metrics, *3rd ACM-IEEE International Symposium on Empirical Software Engineering*, Redondo Beach, CA, August 2004.
- Cost Xpert Group, Inc. *Cost Xpert 3.3 Users Manual*. San Diego, CA: Cost Xpert Group, Inc., 2003.
- EIA/ANSI 632. *ANSI/EIA-632-1988 Processes for Engineering a System*, 1999.
- Forsberg, K., and H. Mooz, Application of the “Vee” to Incremental and Evolutionary Development, *Proceedings of the National Council for System Engineering (NCOSE) Conference*, St. Louis, MO, July 1995.
- Galorath, Inc. *SEER-SEM Users Manual*. El Segundo, CA: Galorath Inc., 2001.
- Jones, C., *Applied Software Measurement: Global Analysis of Productivity and Quality*, 3rd Ed. McGraw-Hill Professional, 2008.
- MIL-HDBK-881, *Work Breakdown Structure*, Washington, D.C., 2 January 1998.
- PRICE, Program affordability management, <http://www.pricesystems.com>, accessed December 30, 2006.
- QSM, SLIM-Estimate, http://www.qsm.com/slim_estimate.html accessed on 12/30/2006.
- SOFTSTAR, <http://softstarsystems.com> accessed on 12/30/2006.
- Stem, D. E., Boito, M. and Younossi, O., *Systems Engineering and Program Management: Trends and Costs for Aircraft and Guided Weapons Programs*, Project Air Force (U.S.), Rand Corporation, 2006.
- Valerdi, R. *The Constructive Systems Engineering Cost Model (COSYSMO)*, PhD Dissertation, University of Southern California, May 2005.
- Valerdi, R., *The Constructive Systems Engineering Cost Model (COSYSMO): Quantifying the Costs of Systems Engineering Effort in Complex Systems*, VDM Verlag, 2008.
- Valerdi, R., Rieff, J., Roedler, G., Wheaton, M., Wang, G., Lessons Learned from Industrial Validation of COSYSMO, *17th INCOSE International Symposium*, June 2007, San Diego, CA.

Wang, G., Valerdi, R., Ankrum, A., Millar, C., Roedler, G., COSYSMO Reuse Extension, *18th INCOSE International Symposium*, June 2008.

Wang, G., Valerdi, R., Roedler, G., Ankrum, A., Gaffney, J. E., Harmonizing Systems and Software Estimation, *19th INCOSE International Symposium*, 2009.

Biographies

Gan Wang, Ph.D., is an EI&S Engineering Fellow at BAE Systems. He has been engaged in the research and development of systems engineering processes, engineering cost modeling, life cycle cost estimation, decision support methods, and system-of-systems engineering and management methodologies. Prior to joining BAE Systems, Dr. Wang has spent many years developing real-time geospatial data visualization applications for mission planning and rehearsal, battlefield management, command and control (C2), flight simulation, and aircrew physiological training systems. He also developed control systems and aircraft simulation models for various man-in-the-loop flight training systems. He has over twenty years of experience in systems engineering, software development, research and development, and engineering management involving complex, software-intensive systems.

Alex Shernoff is a Systems Engineer at BAE Systems. He has been working on the deployment and implementation of COSYSMO at BAE Systems to localize it to organizational platforms and processes and to support systems engineering productivity and efficiency analysis. Prior to this effort, Alex assisted in the planning and logistics of BAE Systems continuing support for the annual INCOSE Conference. Prior to joining BAE Systems, Alex graduated with a B.S. in Electrical Engineering from The Ohio State University. He specialized in high voltage power and electromagnetism.

Jon Turnidge is a Software Engineer at BAE Systems where he has been working on the development of the Total Engineering Estimation Model and Process. Prior to joining BAE Systems, he has worked in the Information Technology field for several firms in the Washington, D.C. area providing database modeling and software engineering services to the federal government. Jon holds a B.S. in Computer Science and Mathematics from the University of Puget Sound.

Ricardo Valerdi, Ph.D., is a Research Associate with the Systems Engineering Advancement Research Initiative and the Lean Advancement Initiative at MIT. He received his doctoral degree in systems engineering from USC in 2005, where he developed the COSYSMO model for systems engineering cost estimation. He is also a Senior Member of the Technical Staff at the Aerospace Corporation. Prior to his doctoral studies, Dr. Valerdi was a systems engineer at Motorola Radio Network Solutions Group. He serves on the Board of Directors of INCOSE as Treasurer and is a member of the Measurement Working Group.