Reuse in Systems Engineering

Gan Wang, *Member, IEEE,* Ricardo Valerdi, *Member, IEEE* and Jared Fortune, *Member, IEEE*

Abstract—Reuse in systems engineering is a frequent but poorly understood phenomenon. Nevertheless, it has a significant impact on estimating the appropriate amount of systems engineering effort with models like the Constructive Systems Engineering Cost Model (COSYSMO). Practical experience showed that the initial version of COSYSMO, a model based on a "build from the scratch" philosophy, needed to be refined in order to incorporate reuse considerations that fit today's industry environment. The notion of reuse recognizes the effect of legacy system definition in engineering a system and introduces multiple reuse categories for classifying each of the four COSYSMO size drivers – requirements, interfaces, algorithms, and operational scenarios. It modifies the counting rules for the COSYSMO size drivers and updates the definition of system size in COSYSMO. It provides an enabling framework for estimating a system under incremental and spiral development.

In this paper, we present (1) the definition of the COSYSMO reuse extension and the approach employed to define this extension; (2) the updated COSYSMO size driver definitions that are consistent with the reuse model; (3) the method applied to defining the reuse weights used in the modified parametric relationship; (4) a practical implementation example that instantiates the reuse model by an industry organization and the empirical data that provided practical validation of the extended COSYSMO model; and (5) recommendations for organizational implementation and deployment of this extension.

***Index Terms*—systems engineering, reuse, metrics, cost estimation**

I. Introduction

ALMOST all systems have a legacy. Today, more often than not, systems are developed based on an evolution of previous systems. New releases of software are developed by modifying and enhancing a previous release. Similarly, new generations of airplanes, ships, and automobiles are developed by improving the functionality and performance of previous models. In many situations, product lines are managed through incremental improvement of previous system definitions. In other situations, existing systems are modernized through technology insertions and obsolescence management. Although system-level examples where previously developed components and capabilities have been leveraged can generally be easily identified, the current literature fails to address how such instances of reuse at the system-level should be appropriately quantified [1]. In other domains, such as software, reuse is a much better documented concept [2].

When a "new system" is developed, it commonly uses existing components, proven functionality, or established architecture. Across industry, we have witnessed ever increasing trends of COTS-integration in system development and ever diminishing endeavors of constructing something completely new from a "clean slate". However, incorporating disparate elements together into a developing system is often no easy task, as the integration of legacy elements can negatively impact project resources, sometimes substantially [3]. Knowing this, addressing reuse from a systems perspective, particularly the impact on the systems engineering effort required throughout the life cycle, is critical to accurately estimating the development cost of a system.

One method for estimating the amount of systems engineering effort required for a project is the Constructive Systems Engineering Cost Model (COSYSMO) [4]. Developed at the University of Southern California with the support of a consortium of academic, industry, and government organizations, COSYSMO 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. COSYSMO is part of the COCOMO II suite of models and developed through a similar methodology. COCOMO, originally developed in 1981 and revised as COCOMO II in 2000, is the most widely used, thoroughly documented and well calibrated software cost model [5]. Applying a similar approach in COCOMO to the systems domain, COSYSMO defines a parametric relationship that estimates systems engineering effort under nominal schedule, in person months, based on four size drivers – system requirements, system interfaces, system algorithms, and operational scenarios – and adjusted by fourteen effort multipliers, which capture the product and project environment and complexity factors. COSYSMO defines a sizing quantity called "system size" from a weighted sum of

Manuscript received August 9, 2008. This work was supported in part by BAE Systems, the Industrial Affiliates of the Center for Systems and Software Engineering, and the Consortium members of the Lean Advancement Initiative.

G. Wang is with BAE Systems, Reston, VA 20190 USA (phone: 703-668-4259; fax: 703-668-4224; e-mail: gan.wang@baesystems.com).

R. Valerdi is with the Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: rvalerdi@mit.edu).

J. Fortune is a doctoral student in the Industrial and Systems Engineering Department, University of Southern California, Los Angeles, CA 90089 USA (email: fortune@usc.edu).

the four size drivers. The parametric relationship is shown below (1):

$$PM_{NS} = A \cdot \left(\sum_{k} (w_{e,k}\Phi_{e,k} + w_{n,k}\Phi_{n,k} + w_{d,k}\Phi_{d,k}) \right)^{E} \cdot \prod_{j=1}^{14} EM_j \quad (1)$$

Where,

PM_{NS} = effort in Person Months (Nominal Schedule)

A = calibration constant derived from historical project data

k = {REQ, IF, ALG, SCN}

w_x = weight for "easy", "nominal", or "difficult" size driver

Φ_x = quantity of "k" size driver

E = represents (dis)economies of scale

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

Being the first of its kind, the model has, in a very short period of time, caught the attention of the systems engineering community, industry and academia alike. It has demonstrated the potential to bridge a long-time gap between system complexity and its corresponding systems engineering effort estimate. Organizations have made various attempts to pilot the model and apply it to practical applications [6, 7].

Early application of COSYSMO, however, has revealed that the model did not recognize the concept of reuse in systems engineering [8, 9]. It assumes that all of its four size drivers – system requirements, system interfaces, system algorithms, and operational scenarios – are new entities when defining the system size. In other words, the model is based on a "build from scratch" philosophy and assumes all systems are developed from a "clean slate".

This differs from how systems are typically built today since requirements for a new system may be "adopted" from an existing system. Furthermore, some of the new system's requirements may be "modified" from a prior system. Moreover, the evolution of system requirements over the system life cycle may result in "deleted" requirements from the initial configuration baseline. The same situations may apply to the other three size drivers – interfaces, algorithms, and operational scenarios. As the result, the calculated system size does not reflect reusing elements of the system definition and, consequently, can result in inaccurate estimate of systems engineering effort required to realize such a system. This problem intensifies when dealing with the incremental and spiral development.

Therefore, we propose incorporating the concept of reuse for estimating the size of a system, in order for COSYSMO to more accurately estimate the systems engineering effort.

II. Development Approach

The approach taken to represent the system size is analogous to that used to represent software code size in those frequent instances in which there are several categories of code, including new and different levels of reuse, as well as deleted [10, 11, 12]. In the case of software, the size of the code is often represented as "ESLOC," or "Equivalent New Source Lines of Code." ESLOC is computed as the weighted sum of the new, the reused, the modified, and the deleted code. Similarly, we define "Equivalent Requirements" or "eReqs" in COSYSMO as a function of the weighted sum of the new, reused, modified, and deleted requirements in a system. This reuse approach COSYSMO is motivated by the similar model defined by COCOMO II [5]. Therefore, additional consideration has been given so that the definitions of reuse are conceptually consistent between COSYSMO and COCOMO II since they belong to the same family of models.

When defining the concept of reuse for COSYSMO, the following basic principles hold true.

1. COSYSMO is an open model, developed by the community and for the community. Users are free to change and/or extend the model.

2. On the other hand, it is beneficial for the industry to agree on the basic definition, relationship and parameters to better communicate basis of estimates.

COSYSMO has been developed as an open model by the community of academia, industry and government for the use of the general public. This implies that everyone is free to adopt, modify, and/or extend the model. In fact, it is intended for organizations to adapt to their own engineering processes and business models, and to develop local, tailored estimating tools. In defining reuse as another aspect of the model, it should not, in any way, restrict or hinder individual applications or adaptations of this model. In fact, it should help to facilitate such a local implementation.

On the other hand, similar to other cost estimating models, COSYSMO provides a methodology for the industry at large to measure and communicate productivity and basis of estimate. For the industry to best communicate such a measure of productivity and basis of estimate, it is important that the same basic definitions are consistently understood and applied. This includes definition of terms and nomenclatures, the basic estimating relationship, and the guidelines for measurement.

However, the above two principles could inherently be conflicting with each other. Free adaptation of the model could lead to individual inconsistent definitions and interpretations of the model. While over-restriction of the model definition could limit the application of the model by organizations, care must be taken to strike a fine balance to preserve both of the above principles, so that it does not over-constrain its application and adaptation across the industry and, at the same time, preserve the integrity of the model.

The approach taken to develop the reuse concept was to

summarize the leading organizational definitions and to aggregate them based on a "minimum common denominator" strategy. Several organizations including BAE Systems, Lockheed Martin and Raytheon have been piloting and implementing this reuse concept. There are some subtle differences in the definitions used, but the basic concept is the same. For this effort, the following guidelines for the industry definition were defined.

A. Establish a minimum set of reference categories for each size driver (i.e., system requirements, system interfaces, system algorithms, and operational scenarios), so that organizations can either directly apply and/or expand to additional reuse categories.

B. Provide minimum common-denominator definitions so that organizations can use, refine, and/or instantiate as appropriate to fit their operational needs.

Goal "A" indicates that we define only those reuse categories that all stakeholders can agree on as the common denominator for the community, and these categories are intended as the reference for an organization in its own implementation to either directly apply or expand to additional categories if it deems necessary. In other words, the intent is to always maintain this set of categories, but one may add other categories, generally as subdivisions, if appropriate. Goal "B" states that the definitions are intentionally structured with the minimum common denominator language so that organizations can either directly use or, if necessary, may refine and substantiate the definitions to fit their operational use. In other words, the industry definitions are intentionally termed at a high level to cover wide ranges of applications.

Several industry-level workshops have been conducted under the stated guidelines to achieve the community agreement on the pilot implementation and validation of the reuse model in leading organizations. The following reuse extension has been defined through such as a community agreement.

III. The COSYSMO Reuse Extension

The COSYSMO reuse extension defines five categories for counting its size drivers, termed: *new*, *modified*, *adopted*, *deleted*, and *managed*. The quantities of each of the four COSYSMO size drivers, i.e., number of requirements, number of interfaces, number of algorithms, and number of operational scenarios, may be classified into the follow five categories of reuse:

1. **New**: Items that are completely new.
2. **Modified**: Items that are inherited, but are tailored.
3. **Adopted**: Items that are incorporated unmodified. Also known as "black box" reuse.
4. **Managed**: Items that are incorporated unmodified and untested.
5. **Deleted**: Items that are removed from a system.

As an example, a requirement can be *new*, which no precedence can be found for the system to be developed. New items are generally unprecedented and may be associated with a low level of familiarity. A requirement may be *modified* in the sense that a heritage element is reused, but needs a limited level of modification or tailoring for it to be fully incorporated in the new system. A requirement may also be *adopted* where the indicated functionality and performance has previously been developed and can therefore be incorporated without any changes. This is commonly referred to as "black box" reuse since the requirement is literally copied from a previous effort remaining unchanged. A requirement that is considered *managed* is incorporated typically as a turn-key component where there is minimal development effort, except for possibly engineering management, on the part of the contractor since a subcontractor may be responsible for its delivery. *Deleted* requirements are those that are already in the legacy system or architecture design, but need to be removed from the current system definition based on customer need or contractual commitments. Consequently, each of these five types of reuse requires different degrees of systems engineering effort.

It is important to note that the *modified* category spans a wide range of possible effort. Modification may entail a simple change in an interface connection to a complete revamp of the entire system architecture. The intent of the *modified* category is to capture those elements that involve tailoring changes only, with no changes to the internal architecture. Therefore, those items that are inherited but require a significant amount of architecture or implementation changes should be counted as *new*.

For each size driver, three complexity levels are defined in COSYSMO: *easy*, *nominal*, and *difficult*. These levels are invariant in the context of reuse. Depending upon the point of view, within each category of reuse, there are three levels of complexity. Alternatively, within each level of difficulty, there can be five categories of reuse. Conceptually, the two notions – *categories of reuse* and *levels of difficulty* – form a two dimensional classification framework for size drivers that provide adequate level of granularity in determining system size. Operationally, when classifying drivers in terms of reuse, there are two alternative sequences, purely depending upon user preferences. When counting requirements, for example, they can first be classified into the five reuse categories. Next, within each reuse category, they can be further divided into three levels of complexity. This additional dimension yields a revised COSYSMO parametric relationship that incorporates the concept of reuse (2):

$$PM_{NS} = A \cdot \left[\sum_{k} \left(\sum_{r} w_r \left(w_{e,k} \Phi_{e,k} + w_{n,k} \Phi_{n,k} + w_{d,k} \Phi_{d,k} \right) \right) \right]^{E} \cdot \prod_{j=1}^{14} EM_j \quad (2)$$

Where:

PM_{NS} = effort in Person Months (Nominal Schedule)

A = calibration constant derived from historical project data

k = {REQ, IF, ALG, SCN}

r = {New, Modified, Adopted, Deleted, Managed}

w_r = weight for defined degrees of reuse

w_x = weight for "easy", "nominal", or "difficult" size driver

Φ_x = quantity of "k" size driver

E = represents diseconomies of scale

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

In contrast to the COSYSMO equation shown earlier (1), the revised relationship (2) introduces reuse as an additional dimension for each of the size drivers.

IV. Modified Size Driver Definitions

With the concept of reuse incorporated into its parametric relationship, the COSYSMO size driver definitions require their amendments. As an example, the original definition for algorithm contains the verbiage: "This driver represents the number of *newly defined or significantly altered* functions…", which directly conflicts with the concept of reuse such as "adopted". The modifications to the definitions are underlined.

These proposed changes in size driver definitions were obtained with industry feedback at the COSYSMO working group meeting at the Practical Software & Systems Measurement User Group Meeting in Denver in July 2007. As the result, the amended COSYSMO size driver definitions, consistent with the reuse extension, are given as below.

A. Number of Systems Requirements

This driver represents the number of new, modified, adopted, managed, and deleted requirements for the system-of-interest at the system level or the level of "sell-off" to customer, which may include derived requirements at the same level. The quantity of requirements includes those related to the effort involved in system engineering the system interfaces, system specific algorithms, and operational scenarios. Requirements may be functional, performance, feature, or service-oriented in nature depending on the methodology used for specification. They may also be defined by the customer or contractor. Each requirement must have systems engineering effort associated with it such as verification and validation (V&V), functional decomposition, functional allocation, etc. System requirements can typically be quantified by counting the number of applicable "shalls" in the system or marketing specification.

Easy	Nominal	Difficult
- Simple to implement	- Moderately difficult to implement	- Complex to implement or engineer
- Traceable to source	- Can be traced to source with some effort	- Hard to trace to source
- Little requirements overlap	- Some overlap	- High degree of requirements overlap

B. Number of System Interfaces

This driver represents the number of new, modified, adopted, managed, and deleted shared physical and logical boundaries between system components or functions (internal interfaces) and those external to the system (external interfaces). These interfaces typically can be quantified by counting the number of unique external and internal system interfaces among ISO/IEC 15288-defined [13] system elements at the system level for the system-of-interest.

Easy	Nominal	Difficult
- Simple	- Moderate complexity	- Complex protocol(s)
- Uncoupled	- Loosely coupled	- Highly coupled
- Strong consensus	- Moderate consensus	- Low consensus
- Well behaved	- Predictable behavior	- Poorly behaved

C. Number of System-Specific Algorithms

This driver represents the number of new, modified, adopted, managed, and deleted mathematical algorithms to be derived in order to achieve the system functional and performance requirements. As an example, this could include a complex aircraft tracking algorithm like a Kalman Filter being derived using existing experience as the basis for the all aspect search function. Another example could be a discrimination algorithm being derived to identify friend or foe function in space-based applications. The number can be quantified by counting the number of unique algorithms needed to realize the requirements specified in the system specification or mode description document.

Easy	Nominal	Difficult
-Algebraic	- Straight forward calculus	- Complex constrained optimization; pattern recognition
-- Straightforward structure	- Nested structure with decision logic	- Recursive in structure with distributed control
- Simple data	- Relational data	- Noisy, ill-conditioned

		data
- Timing not an issue	- Timing a constraint	- Dynamic, with timing and uncertainty issues
- Adaptation of library-based solution	- Some modeling involved	- Simulation and modeling involved

D. Number of Operational Scenarios

This driver represents the number of new, modified, adopted, managed, and deleted operational scenarios that a system must satisfy in order to accomplish its intended mission. An operational scenario must be end-to-end and triggered by an operational event. Such scenarios include both the nominal stimulus-response thread plus all of the off-nominal threads resulting from bad or missing data, unavailable processes, or other exceptional conditions. The number of scenarios can typically be quantified by counting the number of use cases or operational modes captured in the user manual, including off-nominal extensions, developed as part of the operational architecture.

Easy	**Nominal**	**Difficult**
- Well defined	- Loosely defined	- Ill defined
- Loosely coupled	- Moderately coupled	- Tightly coupled or many dependencies/conflicting requirements
- Timelines not an issue	- Timelines a constraint	- Tight timelines through scenario network
- Few, simple off-nominal threads	- Moderate number or complexity of off-nominal threads	- Many or very complex off-nominal threads

V. Weight Definition For Reuse Categories

We present in this section the approach used to define the weights for the reuse categories in the COSYSMO equation (2). It is important to note that the approach outlined below is designed to capture the statistical behavior of a group of the projects, rather than individual behavior of a particular project. In fact, on an individual basis, a project may exhibit a vastly different pattern of labor distribution relatively to reuse. An *adopted* or *modified* element could prove to be more costly than a brand-new element in terms of life cycle systems engineering effort.

The approach taken is bottoms-up activity-based, by which we define the reuse weights by evaluating life cycle systems engineering activities. In particular, we examined the 33 systems engineering activities in five activity groups defined by the ANSI/EIA 632 standard [14] relative to four life cycle phases derived from (but not exactly the same as) the stages defined in ISO/IEC 15288, Systems Life Cycle Processes.

The result of this analysis is presented in the matrix in Figure 1. Along the x-axis, the four life cycle phases are repeated for each defined reuse category, namely, *Conceptualize*, *Develop*, *Operational Test & Evaluation*, and *Transition to Operation*.

Along the y-axis are the 33 systems engineering activities in the five activity groups. For each defined reuse category, we identify the applicable activities by life cycle, which derive the sparsely populated matrix in Figure 1. The analysis involves the determination of applicability of an activity across the life cycle for a particular reuse category. As an example, realizing a new requirement into a product would in general incur all of the activities as specified by EIA-632. A reused requirement, on the other hand, would likely exclude some of the activities. The underlying assumption is that a reused element generally saves systems engineering effort compared to a new element. This matrix allows qualitative distinction between relative scales for *reused* and *new*.

ISO/IEC 15288-Based Life Cycle Phases		Conceptualize	Develop	Operate Test & Eval	Transition to Operation	Conceptualize	Develop	Operate Test & Eval	Transition to Operation	Conceptualize	Develop	Operate Test & Eval	Transition to Operation	Conceptualize	Develop	Operate Test & Eval	Transition to Operation	Conceptualize	Develop	Operate Test & Eval	Transition to Operation
		Reuse Categories																			
EIA 632-Reuse Activity Cross Walk		SE Activities For NEW				SE Activities For MANAGED				SE Activities For ADOPTED				SE Activities For MODIFIED				SE Activities For DELETED			
Acquisition and Supply	1. Product Supply	X	X	X	X			X													
	2. Product Acquisition	X	X	X	X			X													
	3. Supplier Performance	X	X	X	X			X													
Technical Management	4. Process Implementation Strategy	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X
	5. Technical Effort Definition	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X
	6. Schedule and Organization	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X
	7. Technical Plans	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X
	8. Work Directives	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X
	9. Progress Against Plans and Schedules	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X
	10. Progress Against Requirements	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X
	11. Technical Reviews	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X
	12. Outcomes Management	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X
	13. Information Dissemination	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X
System Design	14. Acquirer Requirements	X	X	X	X					X				X				X			
	15. Other Stakeholder Requirements	X	X	X	X					X				X				X			
	16. System Technical Requirements	X	X	X	X					X				X				X			
	17. Logical Solution Representations	X	X	X	X					X				X				X			
	18. Physical Solution Representations	X	X	X	X					X				X				X			
	19. Specified Requirements	X	X	X	X					X				X				X			
Product Realization	20. Implementation	X	X	X	X									X	X			X	X		
	21. Transition to Use	X	X	X	X									X	X			X	X		
Technical Evaluation	22. Effectiveness Analysis	X	X	X	X					X		X	X	X	X	X	X			X	X
	23. Tradeoff Analysis	X	X	X	X					X		X	X	X	X	X	X			X	X
	24. Risk Analysis	X	X	X	X					X		X	X	X	X	X	X			X	X
	25. Requirements Statements Validation	X	X	X	X					X		X	X	X	X	X	X			X	X
	26. Acquirer Requirements Validation	X	X	X	X					X		X	X	X	X	X	X			X	X
	27. Other Stakeholder Requirements Validation	X	X	X	X					X		X	X	X	X	X	X			X	X
	28. System Technical Requirements Validation	X	X	X	X					X		X	X	X	X	X	X			X	X
	29. Logical Solution Representations Validation	X	X	X	X					X		X	X	X	X	X	X			X	X
	30. Design Solution Verification	X	X	X	X					X		X	X	X	X	X	X			X	X
	31. End Product Verification	X	X	X	X					X		X	X	X	X	X	X			X	X
	32. Enabling Product Readiness	X	X	X	X					X		X	X	X	X	X	X			X	X
	33. End Products Validation	X	X	X	X					X		X	X	X	X	X	X			X	X

Fig. 1. Systems engineering activity vs. life cycle phase mapping by reuse categories

The next step is to turn the qualitative relationship to a quantitative one. This is done with an effort distribution table derived from an industry wide-band Delphi survey [15], as shown in Figure 2. Similarly, the four life cycle phases from ISO/IEC 15288 and the five systems engineering activities from ANSI/EIA 632 are presented. The value in each cell of the matrix represents the percentage of the total effort applied to a particular activity in a particular life cycle phase. The total sums to 100%, which corresponds to the life cycle effort of developing a *new* system or system element, from concept to delivery.

Each systems engineering process yields a unique effort profile. For example, the Acquisition and Supply activity typically represents 7% of the total systems engineering effort across four phases of the life cycle. By combining the results in Fig 1 with the data in Fig 2, an approximation of the weight of a particular activity can be prorated for different reuse categories. For example, in the *adopted* category, the effort for System Design process is not significant in the Development phase. Hence, we will assign the value of 0% or remove the original effort value (12%) for that cell. On the other hand, the Technical Evaluation effort is significant and comparable to that in the *new* category for the Operational Test & Evaluation phase. We retain the original percent effort value (12.4%) for that cell. An example of such a exercise yields a series of weights as shown in Figure 3, defined for each reuse category, which aggregates into a set of reuse weight values for size drivers of nominal complexity (i.e., levels of difficulty), as shown in Figure 4.

	Phases				
EIA 632 Fundamental Process	Conceptualize	Develop	Operational Test & Eval.	Transition To Operation	Fundamental Process Total
Acquisition & Supply	1.96%	3.57%	0.91%	0.56%	7.00%
Technical Management	3.74%	6.46%	4.25%	2.55%	17.00%
System Design	10.20%	12.00%	5.10%	2.70%	30.00%
Product Realization	1.95%	4.50%	4.80%	3.75%	15.00%
Technical Evaluation	5.58%	8.37%	12.40%	4.65%	31.00%
Percentage of Total Systems Engineering Effort Per Phase	23.43%	34.90%	27.46%	14.21%	**100.00%**

Fig. 2. Life cycle systems engineering effort distribution [15]

Num. Requirements

	New	Managed	Adopted	Modified	Deleted
	100.00%	15.36%	43.37%	64.65%	50.70%
Easy	0.5	0.0768	0.21685	0.32325	0.2535
Nominal	1	0.1536	0.4337	0.6465	0.507
Difficult	5	0.768	2.1685	3.2325	2.535

Num. Interfaces

	New	Managed	Adopted	Modified	Deleted
	100.00%	15.36%	43.37%	64.65%	50.70%
Easy	1.1	0.16896	0.47707	0.71115	0.5577
Nominal	2.8	0.43008	1.21436	1.8102	1.4196
Difficult	6.3	0.96768	2.73231	4.07295	3.1941

Num. Algorithms

	New	Managed	Adopted	Modified	Deleted
	100.00%	15.36%	43.37%	64.65%	50.70%
Easy	2.2	0.33792	0.95414	1.4223	1.1154
Nominal	4.1	0.62976	1.77817	2.65065	2.0787
Difficult	11.5	1.7664	4.98755	7.43475	5.8305

Num. Scenarios

	New	Managed	Adopted	Modified	Deleted
	100.00%	15.36%	43.37%	64.65%	50.70%
Easy	6.2	0.95232	2.68894	4.0083	3.1434
Nominal	14.4	2.21184	6.24528	9.3096	7.3008
Difficult	30	4.608	13.011	19.395	15.21

Fig. 3. Activity-based weight derivation for reuse categories

New	Managed	Adopted	Modified	Deleted
100.0%	15.4%	43.4%	64.7%	50.7%

Fig. 4. Aggregated weights for the reuse categories

This analysis was accomplished through a series of key stakeholder round-table sessions. Alternatively, the same outcome can be achieved through either Delphi surveys or collection and analysis of historical program data.

The reuse weights are summarized along a continuum in Figure 5 to illustrate two additional points. First, it should be noted that the weight values in Figure 4 represent the nominal values, or the mode, for the respectively categories. The exact weights may fall within a range of possible values that may be greater than or less than the suggested values or a set of distributions whose mode is represented by the values in Figure 4. This presents an opportunity for further tailoring by each organization that wishes to incorporate reuse into their COSYSMO implementation and to more accurately capture organizational productivity. For example, in the *modified* case the corresponding weight may be lower than 0.65 in situations where there is very little modification taking place. Such a situation may arise when the color of an airplane is changed from a Forest Green to Sea Grey. This is a simple modification of a requirement that does not demand critical changes in systems engineering effort. On the other hand, significant modifications may emerge which can result in a higher weight for the *modified* parameter. This may arise when the previous requirement is modified to work in a new environment that was previously considered. Such a scenario frequently arises when companies attempt to modify system components from commercial helicopters to military helicopters. Different operational and performance criteria apply when such components are incorporated into the military domain.

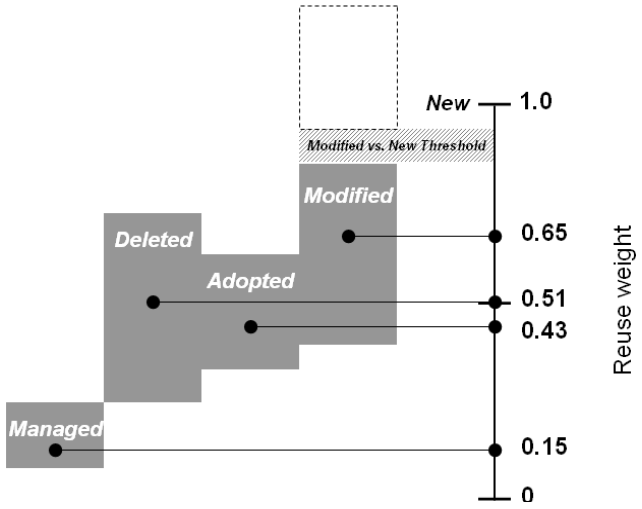


Fig. 5. Reuse Continuum

The second point illustrated by the continuum in Figure 5 is the existence of the *Modified vs. New Threshold*. This is relevant in cases where extreme modification of requirements causes the original reused requirement to be more complex than a new requirement. In this situation, the systems engineer must make a tradeoff decision to determine whether it is better to "throw away" the old requirement and start with a new one or keep the old requirement in spite of its extra expense. The range of possible weights for *modified* requirements may theoretically exceed the weight for *new*, but the exploration of such values was beyond the scope of this analysis.

The approach as presented above can be followed to derive organization-specific reuse weights. Operationally, it is important to note that these weights, once defined or derived in an organization, should be applied to all data points consistently, between the calibration data and new estimates. It is not to be redefined for each data point or new estimate.

VI. A Practical Application Example

For the past two years, BAE Systems has been developing a systems engineering estimating tool based on COSYSMO to locally calibrate the model to its product lines. During the course of this project, a significant amount of historical data was collected to calibrate the model to leading products and platforms. The COSYSMO development at BAE Systems provided the first organizational implementation and validation of the reuse model. In fact, BAE Systems has been part of the core stakeholder group and has led the industry's effort in defining the reuse extension. In order to achieve a practical and deployable implementation, the reuse definition was elaborated and additional specifications were added to better adapt to its engineering process and product lines and

to ensure higher level of correlations in data collection for the organization, while maintaining a more generic definition for the industry model for a broader community.

To develop BAE Systems instantiation of the reuse model, we established the following objectives or guidelines:

- Be consistent with industry definition. Define such reuse categories by instantiating and refining the industry definitions so as to avoid any potential conflicts and inconsistencies.
- Provide clear and consistent *operational guidelines* for driver counting and classification, by using unambiguous verbiage in the reuse definition.
- Establish clear boundaries between categories to ensure easy separation and consistency.
- Enable further extension of reuse, if necessary, and facilitate customer definition of additional reuse levels.

COSYSMO innately is a subjective model, which leads its driver (size and cost) definitions to individual interpretation and, consequently, possible inconsistent sizing of the systems and estimation of effort. One of the challenges for the operational use has been the guidance in understanding the driver definitions and their classification categories. With the instantiated model, clear boundaries between the reuse categories can be established so that there are limited degrees of freedom for individual interpretation in counting the size drivers. In other words, clear steps rather than ramps between the reuse categories help delineate the differences using characteristics that are common to all programs.

The approach is to define a classification framework for counting size drivers with two orthogonal dimensions to enable finer grain estimation of these drivers: *reuse* by systems engineering activities and *levels of difficulties* by relative effort (i.e., easy, nominal and difficult, as defined by COSYSMO). Six high-level, signature life-cycle systems engineering activities were identified. Easy to apply in practice and can be related by most systems engineers, they were used as the key discriminators in delineating the reuse categories: 1) Technical Management; 2) Requirement Definition; 3) Design Analysis; 4) Architecture & Implementation Changes; 5) Tailoring and Interface Changes; 6) Verification & Validation. The definition for the reuse categories is based on a determination of required number of these activities to realize a size drive in an end-to-end development life cycle.

Therefore, at the BAE Systems, we further instantiated reuse model and provided more specific definitions for the five categories, as follows (differences are *italicized*):

1. **New**: Items that are completely new.
2. **Modified**: Items that are incorporated *but require tailoring or interface changes, and verification and validation testing.*
3. **Adopted**: Items that are incorporated unmodified *but require verification and validation testing*. Also known as "black box" reuse.
4. **Managed**: Items that are incorporated unmodified and untested, *and require no additional SE effort other than technical management.*
5. **Deleted**: Items that are removed from a *legacy* system, *which require design analysis, tailoring or interface changes, and verification and validation testing*.

Several points are worth noting for the above definitions. First, these definitions are directly instantiated from the industry version and inherited all its defined categories: *new*, *managed*, *adopted*, *managed*, and *deleted*. Additional clarification of the base definitions was added with the designated systems engineering activities in mind. These activity-based clarifications such as technical management, variation and validation, provide further discriminators and boundary conditions for operational use. For this purpose, a reference table was created, as shown in Figure 6, to serve as a Rosetta Stone between the industry definitions and the ones used at BAE Systems.

	Tech. Management	Requirement Definition	Design Analysis	Architecture & Implementation Change	Tailoring/Interface Change	V&V
New	X	X	X	X	X	X
Managed	X					
Adopted	X	X				X
Modified	X	X	X		X	X
Deleted	X	X	X		X	X

Fig. 6 - Activities-based classification wizard for reuse classification

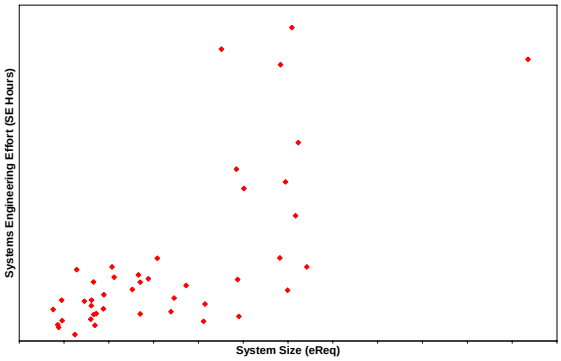
Secondly, we strongly advocated and recommended the category called "*managed*" to the industry definition, which we believe is important in capturing the intricacies of today's evolutionary and spiral development, as well as prevalent teaming arrangement between industry partners. This category is intended for two main circumstances. The first is when a new system incorporates legacy elements that have already been developed and verified and validated from a prior system, the systems engineering activities now are mostly limited to technical management. The second situation is when a part of the system under development is subcontracted out or uses COTS/GOTS-based components that are "turn-key" or "plug-and-play". The requirements and other drivers related to these subtracted parts have already been verified and validated by the providers. To the prime contractor, the majority of the activities required are technical (subcontract) management in nature.

Finally, the *new* items by definition are new and generally unprecedented. However, the modified category can cover such a wide range of spectrum in terms of degrees of change or modification that it is difficult to maintain consistency. To mitigate this problem, we further confine the category to those items that are basically reused as-is and that only allow the degree of modification limited to that of tailoring or interface changes. As a result, any legacy elements that require higher-degree modifications involving architecture and implementation changes are

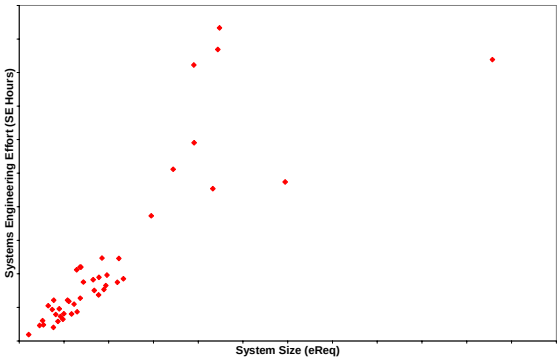
classified as *new*.

The validate the reuse model, an application to a set of historical programs from several lines of business (LoB) and major sites in the Electronics and Integrated Solutions (E&IS) Operating Group was performed. The result is significantly improved data correlation and calibrations with higher-degree of accuracy and confidence level than before. Figure 7a shows the data before applying the reuse model and Figure 7b shows the same data points after applying the reuse model, with everything else in the COSYSMO model held constant.

It is evident from the heteroscedasticity of the data that the reuse model significantly improved the predictive accuracy of the COSYSMO model. This improvement is a result of adjusting the model closer to reality, as it has been proved by practical experience that reuse is more the rule rather than the exception.



(a) Before applying reuse model, all drivers are counted as new



(b) After apply the reuse model

Fig. 7 – Distribution of the same data set, before (a) and after (b) applying the BAE Systems extended COSYSMO reuse model

VII. Conclusion and Recommendations

In this paper, we have defined a reuse extension model for the COSYSMO size drivers. We presented an industry definition, as well as an organizational implementation as the practical validation and implementation example of the reuse model. We discussed the approach applied to this development and the method used for deriving the reuse weights. We also presented the updated COSYSMO size driver definitions that are consistent and compatible with the reuse extension.

To implement this extension for the operational use, an organization can directly apply the method presented in this paper. It may consider further instantiating the definitions to establish more concrete and defined boundaries that are tailored to the business model, product lines, and engineering process of the respective organizations. Organizations may also find it necessary to add additional reuse categories. If so, it is recommended that the original reuse categories be preserved rather than changing the established categories.

The weights for reuse, once defined, should be consistently applied across all data points and over time, between calibration data and new estimates. They should not be changed for a single estimate and calibration point. Any change to these definitions may require recollection of all the calibration data points, and change to the derived weights may require recalculation of all the system sizes. This is required for the necessary level of consistency between programs and between calibrations and new estimates. In other words, this is to ensure that consistency of requirements is realized across programs and system size is measured with the same scale and counting rules.

As a community of systems engineers interested in cost estimation, we cannot dictate each individual organization's extension of the reuse model, but we should, however, agree on a set of values for reuse weights. This is desirable to ensure consistent understanding of estimate system size and better communicate basis of estimates. This will be a continuing effort in the refinement of the reuse approach which will involve feedback from key stakeholders from leading organizations.

The initial version of COSYSMO has established a frontier for systems engineering cost estimation. However, as with any other methodology in its early stage, it requires continuous improvement so that it can gain the level of maturity required by operational use and potentially as a new industry standard. The authors are also engaged in other enhancement efforts to further improve the fidelity of the model. One of these areas is the cost drivers or the effort multipliers used in the model to scale the estimate effort based on the system size. Another is the extension of the reuse concepts to other systems engineering artifacts such as knowledge [16, 17], documentation, and test procedures. We are following a similar strategy in combining expert opinion and historical data to develop the most realistic and accurate model possible and will report the progress of these activities in the near future.

ACKNOWLEDGMENT

The authors wish to thank other collaborators that have provided valuable insight and helpful discussions on this work. Garry Roedler and John Gaffney of Lockheed Martin, John Rieff of Raytheon, Dan Ligett of Softstar Systems, and Barry Boehm of USC were influential in the direction and outcome of this work.

REFERENCES

[1] A. Sage, "Systems Engineering and Systems Management for Reengineering", *Journal of Systems and Software*, vol. 30, no. 1, 1995.

[2] J. Poulin, J. Caruso, D. Hancock, "The business case for software reuse", *IBM Systems Journal*, vol. 32, no. 4, 1993.

[3] D. Garlan, R. Allen, J. Ockerbloom, "Architectural Mismatch or Why it's hard to build systems out of existing parts", *17th International Conference on Software Engineering*, Seattle, WA, April 1995.

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

[5] B. Boehm, C. Abts, A.W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer and B. Steece, *Software Cost Estimation with COCOMO II*, Upper Saddle River, NJ, Prentice-Hall, 2000.

[6] R. Valerdi, J. Rieff, G. Roedler, M. Wheaton and G. Wang, "Lessons Learned from Industrial Validation of COSYSMO," *17th INCOSE Symposium*, San Diego, CA, June 2007.

[7] J. Reiff, J. Gaffney and G. Roedler, "2007: The Breakout Year for COSYSMO," *Practical System and Software Measurement Users Group Conference*, Golden, CO, 2007.

[8] R. Valerdi, J. Gaffney, G. Roedler and J. Rieff, "Extensions of COSYSMO to Represent Reuse," *21st International COCOMO Forum*, Los Angeles, CA, October 2006.

[9] G. Wang, R. Valerdi, R., Ankrum, A., Millar, C. and Roedler, G., "COSYSMO Reuse Extension," *18th INCOSE Symposium, Utrecht, The Netherlands, June 2008.*

[10] "Special Issue on Software Reusability," *IEEE Trans. Software Eng.* T. Biggerstaff and A. Perlis, eds., vol. 10, no. 5, Sept. 1984.

[11] H. Mili, F. Mili, A. Mili, Reusing Software: Issues and Research Directions, *IEEE Trans. Software Eng.*, vol. 21, no. 6, pp. 528-562, June 1995.

[12] "Enabling Reuse-Based Software Development of Large-Scale Systems." *IEEE Trans. Software Eng.* R. Selby, vol. 31, no. 6, June 2005.

[13] ISO/IEC. *ISO/IEC 15288:2002(E) Systems Engineering - System Life Cycle Processes*, 2002.

[14] ANSI/EIA. *ANSI/EIA-632-1988 Processes for Engineering a System*, 1999.

[15] R. Valerdi, M. Wheaton, "ANSI/EIA 632 As a Standard WBS for COSYSMO," *AIAA 1st Infotech@Aerospace Conference*, Arlington, VA, September 2005.

[16] C. Huang, Explication and sharing of design knowledge through a novel product design approach, *IEEE Trans. Sys., Man and Cybernetics, Part C: Applications and Reviews*, vol. 36, pp. 426-438, 2006.

[17] A. Satyadas, Knowledge management tutorial: an editorial overview, *IEEE Trans, Sys., Man and Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 31, pp. 429-437, 2001.

BIOGRAPHIES

Gan Wang (BS'81-MS'87–PhD'91–MBA'02) is a Principal Investigator for system-of-systems engineering and integration strategic initiatives at BAE Systems. He obtained his BS in Electrical Engineering from Harbin Institute of Technology, his MS in Electrical Engineering from George Mason University, his PhD in Electrical Engineering from the University of Virginia, and his MBA from the University of Maryland.

He has been engaged in the research and development of decision support methods and life cycle cost modeling and practice for systems engineering and enterprise-level, system-of-systems engineering and management. Prior to joining BAE Systems, he spent many years developing real-time geospatial data visualization applications for mission planning and rehearsal, battlefield command and control (C2), flight simulation and aircrew training systems. He has over 20 years of experience in systems engineering, software development, research and development, and engineering management involving complex, software-intensive systems.

Dr. Wang is a member of IEEE, INCOSE and PMI.

Ricardo Valerdi (BS'99-MS'02-PhD'05) is a Research Associate at the Lean Advancement Initiative at Massachusetts Institute of Technology and a founding member of the Systems Engineering Advancement Research Initiative. He is also a Visiting Associate at the Center for Systems and Software Engineering at University of Southern California (USC) and a Senior Member of the Technical Staff at the Aerospace Corporation in the Economic & Market Analysis Center. Formerly he worked as a Systems Engineer at Motorola and at General Instrument Corporation.

He earned his BS in Electrical Engineering from the University of San Diego, MS and PhD in Industrial & Systems Engineering from USC.

Dr. Valerdi became a member of IEEE in 1995 and is a member of INCOSE and serves on its Board of Directors.

Jared Fortune (BS'05-MS'06) is a Doctoral Student at the University of Southern California in the Industrial and Systems Engineering Department. His research topic is on reuse considerations in architecture tradeoffs in space systems. He is also an Advanced Degree Fellow at the Aerospace Corporation in the Economic & Market Analysis Center, where he has worked for the past three years.

He earned his BS and MS from the University of Southern California in Industrial and Systems Engineering as well as a Minor in Business.

Mr. Fortune is a member of IEEE and INCOSE.