COSYSMO: A Systems Engineering Cost Model

RICARDO VALERDI AND BARRY W. BOEHM

Abstract: Building on the synergy between Systems engineering and Software Engineering, we have developed a parametric model to estimate systems engineering costs. The goal of this model, called COSYSMO (Constructive Systems Engineering Cost Model), is to more accurately estimate the time and effort associated with performing the system engineering tasks in complex systems. This article describes how COSYSMO was developed and summarizes its size drivers and effort multipliers. We conclude with an example estimate to illustrate the usage of the model to estimate Systems engineering cost.

Key-words: systems engineering, cost estimation, model, complex systems

1. INTRODUCTION

The management of Systems engineering tasks in large projects, both commercial and military, has become increasingly important especially as systems become larger and more complex. To reduce the danger of cost and schedule overruns, organizations must coordinate and manage a diverse set of activities throughout the total systems life cycle. Similarly, large projects cannot succeed without applying good Systems engineering practices and principles in such a way that unify systems and software engineering efforts [3]. The advantages of integrating the two disciplines are clear, but little work has been done to weave them together in a manner that synergistically capitalizes on their joint strengths.

Some research has been done to qualify the complex relationship between software and Systems engineering and to describe the crucial role of software in future systems [3]. This study concluded that large Software Engineering projects cannot exist without considering their linkages with the Systems engineering effort. Current Systems engineering estimation techniques like Cost As an Independent Variable (CAIV), Design To Cost (DTC), and Total Life Cycle Costing (TLCC) embrace activity-based costing techniques. As such, they do not fully address the linkages with software engineering. This creates cost estimates that can be off by an order of magnitude because tasks that require the interaction of engineering disciplines are excluded.

2. MODEL DEVELOPMENT

COSYSMO is part of a trend to improve cost estimating accuracy and increase domain understanding that can potentially lead to increased productivity [4]. The model estimates the effort and duration of such projects based on a variety of parametric drivers that have been shown to have an influence on cost. It uses standard processes for engineering a system [1] as a basis for Systems engineering activities and system life cycle processes [6] to describe the phases in which these activities are performed.

Using a model development process similar to its predecessor, COCOMO II [4], COSYSMO aims at providing an approach for reasoning about the systems engineering decisions on a project as shown in Figure 1.

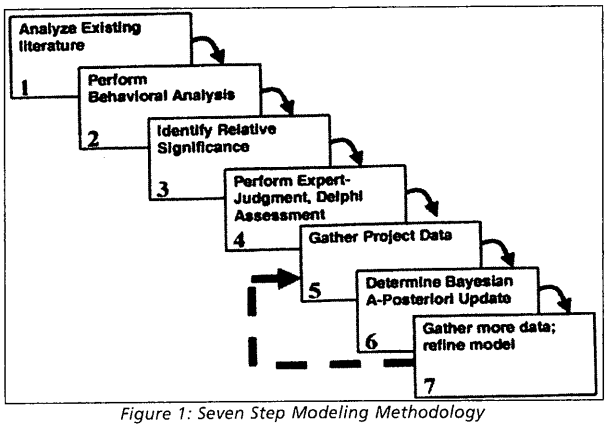

Figure 1: Seven Step Modeling Methodology

The first step involved investigating related literature on how existing cost models addressed Systems engineering effort. What we found was that most of the modeling work done to date relied on heuristics or rules of thumb to cost Systems engineering activities (e.g., allocate 15% of total program effort to systems engineering). Alternatively, most of the firms we interviewed during this fact-finding period did bottoms-up costing that relied on engineers to provide task level estimates based upon their experience. These estimates were validated based on past experience and then summed to form the top-level estimates. Little was done to factor into these estimates the synergy, dynamics, or confusion that occurs on large projects as teams are formed and organizations energized to do the work. As a result, managers often put little confidence in these estimates.

The second step pursued was to perform a behavioral analysis. We asked experts from the U.S. aerospace industry to help us identify the parameters which they felt Systems engineering costs were most sensitive to and the range of variation. We received a wide range of inputs, but could achieve only limited consensus primarily because of the assortment of definitions of Systems engineering terminology. In order to have a shared definition of Systems engineering we adopted the ANSI/EIA 632 standard so that all organizations would relate to the same Work Breakdown Structure.

While our analysis showed that the number of requirements was an indicator of the size of the systems engineering effort, they were a necessary but not sufficient predictor. Another parameter involved was the effort needed to understand the interface requirements and the operational concept document. However, determining the weight of these other indicators was not so straight-forward a task. We found that the domain in which systems engineering was performed influenced the selection of these parameters (e.g., sensor systems were driven by the number of sensor modes while command and control systems were driven by the number of operational modes) as did the organizations performing the task. The COSYSMO size drivers are listed in Table I. These are equivalent to metrics used to determine the size of software projects such as *software lines of code or function points*.

Number of System Requirements The number of requirements taken from the system specification. A requirement is a statement of capability or attribute containing a normative verb such as shall or will. It may be functional or system service-oriented in nature depending on the methodology used for specification. System requirements can typically be quantified by counting the number of applicable shall's or will's in the system or marketing specification.
Number of Major Interfaces The number of shared major 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 interfaces identified in either the system's context diagram and/or by counting the significant interfaces in applicable Interface Control Documents.
Number of Unique Algorithms The number of newly defined or significantly altered functions that require unique mathematical algorithms to be derived in order to achieve the system performance requirements.
Number of Operational Scenarios The number of operational scenarios that a system is specified to satisfy. Such threads typically result in end-to-end test scenarios that are developed to validate the system satisfies its requirements. The number of scenarios can typically be quantified by counting the number of end-to-end tests used to validate the system functionality and performance. They can also be calculated by counting the number of high-level use cases developed as part of the operational architecture.

Table I: COSYSMO Size Drivers

Requirements Understanding The level of understanding of the system requirements by all stakeholders including the systems, software, hardware, customers, team members, users, etc...
Architecture Understanding The relative difficulty of determining and managing the system architecture in terms of IP platforms, standards, components (COTS/GOTS/NDI/new), connectors (protocols), and constraints. This includes systems analysis, tradeoff analysis, modeling, simulation, case studies, etc...
Level of Service Requirements The difficulty and criticality of satisfying the Key Performance Parameters (KPP). For example: security, safety, response time, the "illities", etc...
Migration Complexity The complexity of migrating the system from previous system components, databases, workflows, etc, due to new technology introductions, planned upgrades, increased performance, business process reengineering etc...
Technology Risk The relative readiness of the key technologies for operational use.
Documentation to Match Lifecycle Needs The formality and detail of documentation required to be formally delivered based on the life cycle needs of the system.
 # and Diversity of installations/platforms The number of different platforms that the system will be hosted and installed on.

Table II: COSYSMO Effort Multipliers – Application Factors

Stakeholder Team Cohesion Leadership, frequency of meetings, shared vision, approval cycles, group dynamics (self-directed teams, project engineers/managers), IPT framework, and effective team dynamics.
Personnel Capability Systems engineering's ability to perform in their duties and the quality of human capital.
Personnel Experience/Continuity The applicability and consistency of the staff over the life of the project with respect to the customer, user, technology, domain, etc...
Process Capability Maturity per EIA/IS 731, SE CMM or CMMI.
Multisite Coordination Location of stakeholders, team members, resources (travel).
Tool Support Use of tools in the System Engineering environment.

Table III: COSYSMO Effort Multipliers – Team Factors

In parallel with the behavioral analysis, we proceeded to identify the effort multipliers that experts felt significantly impacted Systems engineering cost and schedule. These are shown in Tables II and III. As in the COCOMO II model [Boehm et al 2000], such effort multipliers are used to modify the amount of effort to reflect product, platform, personnel, and project factors that have been shown to influence cost and schedule.

As our fourth step, we conducted a Delphi exercise to reach group consensus and validate our initial findings. The Wideband Delphi technique has been identified as being a powerful tool for achieving group consensus on decisions involving unquantifiable criteria [2]. We used it to circulate our initial findings and reach consensus on the parametric ratings of the size drivers and effort multipliers. The COSYSMO Delphi survey was designed to (1) reach consensus from a sample of Systems engineering experts, (2) determine the distribution of effort across effort categories, (3) determine the most influential predictors of Systems engineering size, (4) identify the cost drivers to which effort was most sensitive to, and (5) help us refine the scope of the model elements. Part of the Delphi process involved multiple distributions of the surveys to arrive at the values that experts could converge on. The model that evolved as a product of this Delphi effort is of the following regression equation form:

$$PM_{NS} = A \cdot (Size)^{E} \cdot \prod_{i=1}^{n} EM_i$$

Where:

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

A = calibration constant derived from historical project data

Size = determined by computing the weighted sum of four size drivers (requirements, interfaces, algorithms and operational scenarios)

E = represents economy/diseconomy of scale; default is 1.0

n = number of cost drivers (14)

EM_i = effort multiplier for the i_{th} cost driver. Default value for all effort multipliers is 1.0.

We then developed a method for developing rating scales for the effort multipliers, as shown in Table IV. The polarity of each individual effort multiplier determined the direction of the rating scale. For instance, the *requirements understanding* effort multiplier is positively phrased and therefore has a rating scale with effort savings for higher rating levels. Alternatively, the *level of service requirements* effort multiplier is negatively phrased and has a rating scale with increasing values (effort penalty) for higher rating levels.

The Effort Multiplier Ratio (EMR) is the ratio between the highest value and the lowest value in each rating scale. For example, in the case of

	Very Low	Low	Nominal	High	Very High	Extra High	EMR
Requirements Understanding	1.87	1.37	1.00	0.77	0.60		3.12
Architecture Understanding	1.64	1.28	1.00	0.81	0.65		2.52
Level of Service Requirements	0.62	0.79	1.00	1.36	1.85		2.98
Migration Complexity			1.00	1.25	1.65	1.93	1.93
Technology Risk	0.67	0.82	1.00	1.32	1.75		2.61
Documentation	0.78	0.88	1.00	1.13	1.28		1.64
# and diversity of installations/platforms			1.00	1.23	1.52	1.87	1.87
# of recursive levels in the design	0.76	0.87	1.00	1.21	1.47		1.93
Stakeholder team cohesion	1.50	1.22	1.00	0.81	0.65		2.31
Personnel/team capability	1.50	1.22	1.00	0.81	0.65		2.31
Personnel experience/continuity	1.48	1.22	1.00	0.82	0.67		2.21
Process capability	1.47	1.21	1.00	0.88	0.77	0.68	2.16
Multisite coordination	1.39	1.18	1.00	0.90	0.80	0.72	1.93
Tool support	1.39	1.18	1.00	0.85	0.72		1.93

Table IV: Rating Scales for COSYSMO Effort Multipliers [7]

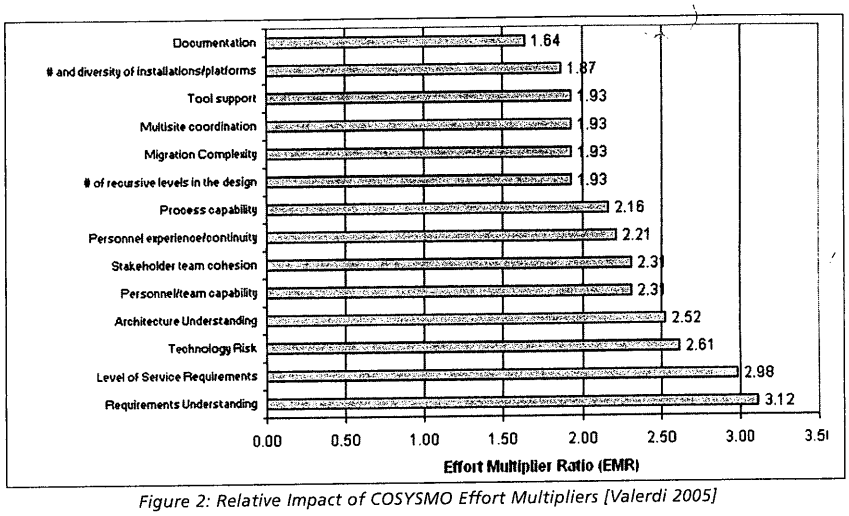

Figure 2: Relative Impact of COSYSMO Effort Multipliers [Valerdi 2005]

requirements understanding the value for Very Low and Very High is 1.87/0.60 = 3.12. The results in Table IV are shown in graphic form in Figure 2 to illustrate their order of importance using EMR values. The most influential drivers of Systems engineering effort are *requirements understanding, level of service requirements, and technology risk.*

These parameters form the basis of the fifth step of gathering historical data from companies that collected Systems engineering effort information on projects. Steps six and seven – refining the model using a Bayesian statistical approach and refining the model based on usage experience – are ongoing activities that have helped identify potential improvements to COSYSMO [7]. The ultimate objective was to implement the COSYSMO tool so that companies could begin estimating systems engineering effort. For more information on COSYSMO and available tools for download visit: *http://cosysmo.mit.edu.*

3. Example estimate

The following example is designed to demonstrate the use of COSYSMO. It is based on a real scenario from the U.S. aerospace industry.

Your customer has provided a system specification that contains 200 requirements, 5 interfaces, and 5 algorithms for a new system based on a newly developed technology. The first step is to decompose[1] the requirements provided by the customer down to the appropriate level for systems engineering. After decomposition, it is determined that the 200 requirements provided by the customer yield 450 requirements at the systems engineering level.

The next step is to allocate the decomposed requirements into the available complexity levels in COSYSMO. Through additional dialog with the customer, a review of the system specification, and discussion with experts in your organization that have worked on similar systems it is determined that the 450 decomposed requirements can be allocated as follows: *200 easy, 200 nominal,* and *50 difficult.*

The 5 interfaces provided by the customer must also be allocated into complexity levels. After comparing the system specification to similar systems built by your organization, it is determined that 2 of the interfaces are easy and 3 are *difficult*. Similarly, the 5 algorithms are reviewed and assigned a complexity level of *difficult.*

These quantities are the COSYSMO model size drivers. At this stage, an initial systems engineering person-month estimate can be obtained based solely on the systems engineering size drivers. However, additional information about the program is available that can be used to adjust the estimate. In particular, three things are known to be unique about this project. The first is that the contractor has done similar projects and therefore has a *high* degree of *requirements understanding*. Second, it is determined that a critical technology used in the project is relatively immature and requires a significant degree of research & development to make it usable. This translates to a *high technology risk.* Third, the contractor responsible for this project has relatively mature systems engineering processes and has obtained a CMMI (Capability Maturity Model Integrated® [5]) rating of 3. This translates to a *high process capability.*

Together, these three characteristics of the project being estimated can be captured in the COSYSMO cost drivers by rating the following three effort multipliers: *requirements understanding, technology risk,* and *process capability*. This additional project information also provides deeper insight into the project's potential performance and possible risk factors that may introduce schedule or cost variation.

In summary, the information obtained from the system specification – supplemented by additional dialog with the customer and discussion with experts familiar with similar efforts – provided the necessary information to populate three of the four size drivers in COSYSMO. The familiarity and process maturity of the contractor combined with an assessment of the technology risk provided the necessary information to rate three of the fourteen cost drivers. The resulting Systems engineering estimate provided by COSYSMO is 195 *person months* as shown in Figure 3.

For purposes of this example, it is assumed that the project being estimated includes the standard systems engineering life cycle phases per ►

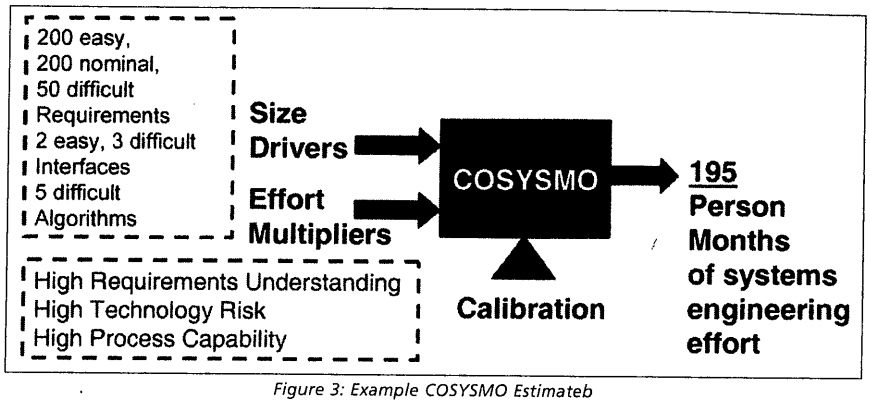

Figure 3: Example COSYSMO Estimateb

ISO/IEC 15288 and a standard systems engineering Work Breakdown Structure per ANSI/EIA 632. Individual organizations may wish to tailor this predetermined scope based on their own definition of Systems engineering.

4. Conclusion

As systems are built, engineering disciplines don't act alone. Instead, they interact in order to solve the customer's real problem. Estimating how much these tasks cost is equally important. To reduce the danger of cost and schedule overruns, the total engineering effort must be orchestrated, coordinated, and managed throughout the total system development life cycle.

The intent of this paper was to provide insight into the development of COSYSMO, its parameter definitions, and an example estimate. Our efforts have confirmed that Systems Engineers and Program Managers desire models that they can use in developing accurate estimates of Systems engineering effort. Our contribution, COSYSMO, represents such a model. Its parametric formulation differs from others currently available which are heuristic-based. Being based on actual project data enables engineers and managers to use COSYSMO to justify forecasted expenditures and reduce risk.

We have had tremendous support in our efforts but still have a number of significant challenges ahead of us. We hope to continue collaborating with systems engineering practitioners to produce an accurate and flexible model. Their insights and suggestions have helped us solidify concepts and make the terminology we use more acceptable. We plan to continue reporting the status and progress of our efforts as we pursue model development with the intent to make an impact in the fields of cost estimation and systems engineering.

Acknowledgements

The authors would like to thank the Consortium Members of the MIT Lean Advancement Initiative and the Corporate Affiliates of the USC Center for Systems and Software Engineering for supporting the development of COSYSMO. Especially The Aerospace Corporation, BAE Systems, Boeing, General Dynamics, L-3 Communications, Lockheed Martin, Northrop Grumman, Raytheon and SAIC.

5. Références

[1] ANSI/EIA-632-1988 *Processes for Engineering a System*. New York, NY: American National Standards Institute, 1999.

[2] B. W. Boehm: *Software Engineering Economics*, Prentice-Hall, 1981.

[3] B. W. Boehm:*Integrating Software Engineering and Systems Engineering*; The Journal of NCOSE, Volume I, No. 1, July-September 1994, pp. 147-151.

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

[5] *Capability Maturity Model Integration - CMMI-SE/SW/IPPD/SS*, V1.1. Pittsburg, PA, Carnegie Mellon - Software Engineering Institute, 2002

[6] ISO/IEC 15288:2002(E), *Systems Engineering – System Life Cycle Processes*, First Edition, 2002.

[7] R. Valerdi: *The Constructive Systems Engineering Cost Estimation Model (COSYSMO)*, PhD Dissertation, University of Southern California, May 2005.

Note

1 It is important to distinguish between decomposed and derived requirements. For purposes of COSYSMO, we focus on decomposed requirements because they are a better proxy for systems engineering effort.