COSYSMO: A Systems Engineering Cost Model

Barry W. Boehm, Donald J. Reifer, Ricardo Valerdi
University of Southern California – Center for Software Engineering
941 W. 37th Place, SAL Room 328
Los Angeles, CA 90089-0781
(213) 740-8163
boehm@sunset.usc.edu, dreifer@earthlink.net, rvalerdi@sunset.usc.edu

Abstract
Building on the synergy between Systems Engineering and Software Engineering, the Center for Software Engineering (CSE) at the University of Southern California (USC), has initiated an effort to develop 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 defined by ISO/IEC 15288.

This paper describes the work done during the past two years by USC, in collaboration with its Corporate Affiliates and the INCOSE Measurement Working Group, to develop the initial version of COSYSMO. The paper focuses on the need for the model, describes how it was developed, summarizes its initial architecture, and identifies the size and cost drivers that participants in the model design thought to be significant. These were identified through a series of Delphi surveys where experts were asked to provide their opinions. The paper concludes with a summary of the results of these Delphi surveys and a discussion of future steps that need to be taken to guarantee COSYSMO’s successful adoption by the Systems Engineering community.

1. Introduction
The management of Systems Engineering tasks in large projects, both commercial and defense, has become increasingly important especially as systems become larger and more complex. To reduce the danger of cost and schedule overruns, Systems Engineering 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 [Boehm 1994]. The recent release of the Capability Maturity Model Integration v1.1 (CMMI®) represents one such effort that is aimed towards integrating Systems and Software Engineering processes. 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 [Boehm 1994]. 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 estimates that can be off by an order of magnitude because tasks that require the interaction of engineering disciplines are left out.

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 [Boehm, et al 2000]. 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. Based on stakeholder inputs, it uses ISO/IEC 15288 Standard for System Life Cycle Processes as a basis for Systems Engineering activities.

Using a proven model development process, the team created COSYSMO to be the newest member of the COCOMO family of software cost models. Like other COCOMO-based models created before it, COSYSMO has been developed using the seven-step modeling methodology outlined in the book Software Cost Estimation With COCOMO II (Boehm, Reifer, et al. 2000) and illustrated in Figure 1.

In 2000, USC Affiliates identified accurate Systems Engineering cost estimation as their most pressing need. In response, USC formed a task team to develop the model using volunteer experts from commercial firms (Motorola, Rational, Xerox, etc.), aerospace companies (Lockheed, Raytheon, TRW, etc.) and government organizations (DOD, FAA, NASA, etc.) who followed the process in Figure 1. The first step involved investigating related literature on how existing cost models addressed Systems Engineering costing. 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. 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 our affiliated organizations 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. To develop a joint vision and realize closure on this task, we conducted the first of many workshops at USC in the Fall of 2000. At this first workshop, we tried to reach agreement on definitions, scope of the model, identify the applicable activities, and reach consensus on the cost and size drivers, and their ranges of variation. The workshop was successful in developing an initial framework for the model and enabled the team to devise a migration path (see Figure 2 which illustrates a layered view of systems engineering activities) by which we could evolve the model as data became available to confirm or deny the opinions of the experts.

During the next year, members of the team have collaborated to solidify the definitions of the parameters and focused on the most difficult drivers, the size parameters. While our experts agreed that the number of requirements was an indicator of the size of the Systems Engineering effort, most agreed they were a necessary but not sufficient predictor. Another parameter involved was the effort needed to pin down the interface requirements and the operational concept document. As the team dug deeper, it found that determining the weight of these other indicators was not so straightforward 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 organization performing the task. Not surprisingly, both the number of drivers and the evolution path for the model changed as new people and organizations became affiliated with the effort.

In parallel with the behavioral analysis, we proceeded to identify the parameters that the experts felt significantly impacted Systems Engineering cost and schedule. The challenge here was managing the different perspectives of Systems Engineering. To address this issue, we resorted to using a specification for an exemplar system, a satellite ground station [Aerospace 1996], to act as a common reference for discussion. Using an exemplar successfully
enabled us to come to consensus on the parameters which members of the task team felt most significantly impacted Systems Engineering cost, schedule and, risk exposure. The team further assisted us in defining the parameters in such a way that users could rate them for their systems.

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 [Boehm 1981]. We used it to circulate our initial findings and reach consensus on our parametric ratings with our experts. The COSYSMO Delphi survey, conducted in December 2001, received 28 responses. The purpose of the survey was to (1) reach consensus from a sample of systems engineering experts, (2) determine the distribution of effort across effort categories, (3) determine the 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 engineers could converge on. The model that evolved as a product of this Delphi effort is of the following regression equation form:

$$\text{Effort} = A \prod c_i (\text{Size})^p$$

Where:
- $A$ = calibration constant
- $c_i$ = calculated as the product of $i$ cost drivers
- $P$ = power law (represents economies/diseconomies of scale)
- $\text{Size}$ = determined by computing the weighted average of the size predictors

The results of this Delphi are illustrated in Figure 3. The predictors are arranged in order of influence, with those that have the largest influence on the size of Systems Engineering work (Number of Algorithms and Number of Interfaces) on the right side of Figure 3(a). This was surprising to us because we initially felt that requirements would have the largest impact. Size in the model, like in function points for software [Boehm, Reifer, et al. 2000], is determined by taking the weighted average of the predictors.

In Figures 3(b) and (c), the four cost drivers that the experts felt could affect the cost of Systems Engineering effort the most are shown on the right side (Level of Service Requirements, Personnel Capability, Requirements Understanding, and Architecture Understanding).

As in the COCOMO II model, such drivers 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. It should be noted that drivers described later in this paper differ slightly from the ones used in the initial Delphi survey because the model has evolved based on inputs from other members of the Systems Engineering community since the time of the survey.

In the Spring of 2002, we began collaborating with the INCOSE Measurement Working Group. At the time, INCOSE felt that the COSYSMO model we were developing was too software-oriented and did not portray an appropriate scope for most Systems Engineering tasks. However, they also felt that the effort was important and helped us refine the model to overcome the deficiencies they observed. These changes helped us in explaining concepts in terms more acceptable to the Systems Engineering community. In addition to endorsing the effort, INCOSE CAB members are actively pursuing getting INCOSE member companies to support COSYSMO with people, resources, and data.

3. Model Architecture

The current operational form of the COSYSMO model is shown in Figure 4. As previously noted, the size drivers (predictors) and cost drivers (Effort Multipliers) were determined via a Delphi exercise by a group of experts in the fields of Systems Engineering, Software Engineering, and Cost Estimation. The definitions for each of the sixteen drivers, while not final, attempt to cover those activities which have the greatest impact on estimated Systems Engineering effort and duration. Each
driver reflects the range of impact and variation assigned by the experts during our refinement exercises. The group of size drivers includes a volatility factor that accounts for the amount of change that is involved in the four factors. For example, it can be used to adjust the number of requirements should they be ill-defined, changing, or unknown at the time the estimate is being formulated. The group of cost drivers, also called Effort Multipliers, includes a schedule driver that accounts for the amount of schedule compression or expansion that might occur in a project. This schedule driver allows the user to increase the effort to reflect compressions made in the schedule for project related reasons.

<Figure 4. COSYSMO Operational Concept>

As shown in Figure 4, sixteen drivers impact the effort and duration of Systems Engineering activities. Four of these drivers are used to predict the size of the effort, while the other twelve are used to adjust effort based on parameters that are sensitive to cost. The four size drivers that are currently in the model are defined in Table I. It should be noted that we started with seven size drivers but reduced this list to four to eliminate overlap and confusion.

<Table I. Current COSYSMO Size Drivers>

As summarized in Tables II and III, the twelve cost drivers (five application and seven team factors) were defined to represent Effort Multipliers. This list was modified several times in order to realize consensus on the parameters which the experts felt costs and schedules were most significantly impacted by.

<Table II. Current COSYSMO Cost Drivers – Application Factors>

<Table III. Current COSYSMO Cost Drivers – Team Factors>

We are currently beginning Step 5 of the model development process we outlined in Figure 1. In this step, we will gather project data and use it to confirm or deny the opinions of experts. This is an important step because it enables us to calibrate the model and its size and cost drivers. Our goal this year is to gather data on at least 30 projects that have collected Systems Engineering information. This is the minimum data set that we believe is necessary to develop a reasonable calibration.

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 driver definitions, and future challenges. Our efforts have confirmed that Systems Engineers and Program Managers desire models that they can use in developing accurate estimates of Systems Engineering effort and duration. 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 win the battle of the budget.

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 and the INCOSE community at large 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.

References

http://sunset.usc.edu/research/reference_architecture/

**Acknowledgements**

The authors would like to thank the Corporate Affiliates of the USC Center for Software Engineering that have participated in the development of COSYSMO. Especially: Marilee Wheaton (Aerospace Corp.), Gary Thomas (Raytheon), John Rieff (Raytheon), Tony Jordano (SAIC), Garry Roedler (Lockheed Martin), Gary Hafen (Lockheed Martin), Karen Owens (Aerospace Corp.), Evin Stump (Golarath), Chris Miller (Software Productivity Consortium), Cheryl Jones (US Army). The authors are also grateful for the support of the International Council on Systems Engineering – Measurement Working Group, International Council on Systems Engineering – Corporate Advisory Board, and the International Society of Parametric Analysts.

**Figures and Tables**

Figure captions:
**Figure 1.** Seven Step Modeling Methodology
**Figure 2.** COSYSMO Evolution Path
**Figure 3.** COSYSMO Delphi Round 1 Results
**Figure 4.** COSYSMO Operational Concept

---

**Figure 1. Seven Step Modeling Methodology**
Figure 2. COSYSMO Evolution Path

Figure 3. COSYSMO Delphi Round 1 Results
Table 1. Current COSYSMO Size Drivers

<table>
<thead>
<tr>
<th>Size Drivers</th>
<th>Volatility Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td># Requirements</td>
<td># Interfaces</td>
</tr>
<tr>
<td># Scenarios</td>
<td># Algorithms</td>
</tr>
</tbody>
</table>

**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 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.

**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.

---

Table II. Current COSYSMO Cost Drivers – Application Factors

<table>
<thead>
<tr>
<th>Application factors (5)</th>
<th>Team factors (7)</th>
</tr>
</thead>
</table>

**Requirements Understanding**
The level of understanding of the system requirements by all stakeholders including the systems, software, hardware, customers, team members, users, etc…

**Architecture Complexity**
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 Maturity**
The relative readiness of the key technologies for operational use.
**Table III.** Current COSYSMO Cost Drivers – Team Factors

<table>
<thead>
<tr>
<th>Stakeholder Team Cohesion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leadership, frequency of meetings, shared vision, approval cycles, group dynamics (self-directed teams, project engineers/managers), IPT framework, and effective team dynamics.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Personnel Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systems Engineering’s ability to perform in their duties and the quality of human capital.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Personnel Experience/Continuity</th>
</tr>
</thead>
<tbody>
<tr>
<td>The applicability and consistency of the staff over the life of the project with respect to the customer, user, technology, domain, etc…</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process Maturity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maturity per EIA/IS 731, SE CMM or CMMI.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multisite Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location of stakeholders, team members, resources (travel).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formality of Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>The breadth and depth of documentation required to be formally delivered.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tool Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of tools in the System Engineering environment.</td>
</tr>
</tbody>
</table>