1.0

29-Jan-06

ENTER SIZE PARAMETERS FOR SYSTEM OF INTEREST

	Easy	Nominal	Difficult
# of System Requirements			
# of Interfaces			
# of Operational Scenarios			

0
0
0
0
0 } equivalent size

SELECT COST PARAMETERS FOR SYSTEM OF INTEREST

Requirements Understanding	N	1.00
Architecture Understanding	N	1.00
Level of Service Requirements	N	1.00
Migration Complexity	N	1.00
Technology Risk	N	1.00
Documentation	N	1.00
# and diversity of installations/platforms	N	1.00
# of recursive levels in the design	N	1.00
Stakeholder team cohesion	N	1.00
Personnel/team capability	N	1.00
Personnel experience/continuity	N	1.00
Process capability	N	1.00
Multisite coordination	N	1.00
Tool support	N	1.00
		1.00 composite effort multiplier

SYSTEMS ENGINEERING PERSON MONTHS 0.0

academicCOSYSMO User Manual

A Practical Guide for Industry & Government

Version 1.0
July 2006

By:
Ricardo Valerdi
MIT Lean Aerospace Initiative
rvalerdi@mit.edu

This work has been funded by the MIT Lean Aerospace Initiative Consortium, the USC Center for Software Engineering Corporate Affiliates, and the US Air Force Office Space & Missile Systems Center of the Chief Engineer.

Table of Contents

Acknowledgements
The information in this document is representative of the work accomplished since 2001 by the COSYSMO Working Group in conjunction with the INCOSE Measurement Working Group. The motivation behind the first implementation of the COSYSMO model came from Gary Thomas (Raytheon- Garland, TX). Other individuals who have contributed to this document are: Tony Hart (General Dynamics – Pittsfield, MA), Joe Emerick (Lockheed Martin – Gaithersburg, MD), and Chris Miller (SSCI – Herndon, VA).

1. Model definition, scope, and assumptions

The Constructive Systems Engineering Cost Model (COSYSMO) is a model that can help people reason about the economic implications of systems engineering on projects. Similar to its predecessor, COCOMO II[1], it was developed at the University of Southern California as a research project[2] with the help of BAE Systems, General Dynamics, Lockheed Martin, Northrop Grumman, Raytheon, and SAIC. COSYSMO follows a parametric modeling approach used to estimate the quantity of systems engineering labor, in terms of person months, required for the conceptualization, design, test, and deployment of large-scale software and hardware projects. User objectives include the ability to make Proposal estimates, investment decisions, budget planning, project tracking, tradeoffs, risk management, strategy planning, and process improvement measurement.

The academicCOSYSMO model is a simple EXCEL implementation that should be compared to other quoting methods and eventually calibrated to reflect the organization's definitions, scope of systems engineering activities, and life cycle coverage. The parameters in the model are defined in sections 1. c) and 1. d), followed by a description of the work breakdown structure used in the model which is driven by the *ANSI/EIA 632 Processes for Engineering a System*[3] and described in sections 1. d) and 3. a). The scope of COSYSMO is defined by system life cycle phases inspired by the ISO/IEC 15288 *Systems Engineering – System Life Cycle Processes*[4] standard described in section 3. b).

Before users begin to work with COSYSMO, they should be aware of the inherent assumptions embedded in the model. The most important assumptions are that the user performing an estimate with COSYSMO has a basic understanding of: (1) the definitions of the eighteen drivers, (2) the associated counting rules for the size drivers, (3) the output of the model, and (4) how COSYSMO relates to the general systems engineering context (i.e., process, labor categories) in their organization. Beyond the assumptions surrounding the user, the model has additional embedded assumptions that reflect its ancestry. These assumptions are that:

- the organization using the model defines systems engineering in a way that is compatible with the INCOSE definition (see section 1. a) for definition)
- a predetermined set of systems engineering activities and life cycle phases exist in the organization and these are closely aligned with the two aforementioned standards
- the model will be used to estimate effort from the contractor perspective
- the organization, not its subcontractors, is performing a majority of the systems engineering work
- reuse of requirements and interfaces is minimal
- the organization using the model develops systems for the defense or aerospace domain similar to those developed by the six organizations that participated in the industry calibration

The implications of these assumptions are significant and, if not carefully considered, can lead to insignificant results.

[1] Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B., Horowitz, E., Madachy, R., Reifer, D. J. and Steece, B., *Software Cost Estimation With COCOMO II*, Prentice Hall, 2000.
[2] Valerdi, R., "The Constructive Systems Engineering Cost Model (COSYSMO)," unpublished PhD Dissertation, University of Southern California, May 2005.
[3] ANSI/EIA-632-1988 Processes for Engineering a System, 1999.
[4] ISO/IEC 15288:2002(E) Systems Engineering - System Life Cycle Processes, 2002.

a) Definition of key terms

Cost driver – see “Effort Multiplier”

Cost Estimating Relationship – a mathematical expression that represents the relationship between the model’s independent variables (size drivers and cost drivers) and the dependent variable (person months).

Effort Multiplier – one of fourteen parameters in the model, also know as a cost drivers, that have a multiplicative effect on systems engineering effort either in the way of an effort savings or effort penalty. (also see “Rating”)

Effort Multiplier Ratio – the ratio of the highest valued rating multiplier and the lowest valued rating multiplier for an individual cost driver. This is an indication of the swing between the effort savings and effort penalty associated with a driver and is used to compare the relative significance between drivers.

Life cycle – a set of six phases which describe the typical evolution of a system in which systems engineers perform work. COSYSMO uses a life cycle inspired by the ISO/IEC 15288 standard.

Person Month – a generally accepted unit of measure for people effort which usually equals 152 person hours. It is the dependent variable for the COSYSMO model.

PRED – the criterion for measuring predictive accuracy of cost models, usually expressed in three different ranges: 20%, 25%, and 30%. A model with perfect predictive ability would have a PRED(20) of 100% meaning that it would estimate within 20% of the actual effort, 100% of the time.

Rating scale – an ordinal scale to represent the different attributes of a cost driver. It can have anywhere between four and seven rating levels ranging from “Very Low” to “Extra High” and whose default value is “Nominal”.

Rating – the value assigned for a particular cost driver by the model user based on their assessment of the impact of that driver on the systems engineering effort estimate.

Size drivers – one of four parameters in the model which have an additive effect on systems engineering effort.

Systems Engineering – The formal INCOSE definition of SE is “an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem.” See definitions for “WBS” and “Life Cycle” for the activities that are included in SE and the scope of SE life cycle as used in COSYSMO.

Work Breakdown Structure – the set of activities that represent specific tasks that, in this case, are performed by systems engineers and are included in COSYSMO. The thirty three activities described in the ANSI/EIA 632 standard define the scope of COSYSMO and its estimate.

b) COSYSMO Algorithm

Each parameter in the COSYSMO Algorithm is part of the Cost Estimating Relationship (CER) that was defined by systems engineering experts.

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

Where:

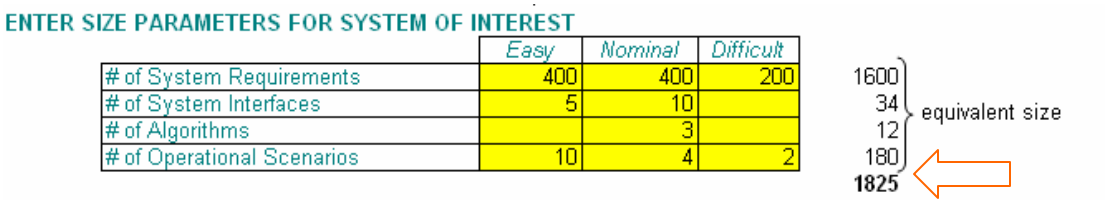
PM_{NS} = effort in Person Months (Nominal Schedule)
A = calibration constant derived from historical project data
k = {REQ, IF, ALG, SCN}
w_x = weight for "easy", "nominal", or "difficult" size driver
Φ_x = quantity of "k" size driver
E = represents diseconomies of scale
EM = effort multiplier for the j_{th} cost driver. The geometric product results in an overall effort adjustment factor to the nominal effort.

The size of the system is the weighted sum of the REQ, IF, ALG, and SCN parameters and represents the additive part of the model while the EM factor is the product of the 14 effort multipliers and represents the multiplicative part of the model. This algorithm is built into the academicCOSYSMO spreadsheet in cell E29 where the systems engineering person month estimate is displayed.

Detailed definitions for these parameters are provided in the following sections.

c) Size Drivers

The size drivers should be entered first because they require the user to think about the quantitative parameters that determine size of the system in terms of systems engineering. The value of the size drivers can be entered in the yellow cells show below. The spreadsheet will keep a running total of the number of equivalent requirements in cell F9 which is a weighted sum of the four size drivers.



Although there are twelve available cells for data entry, an estimate can be obtained by entering information into only one cell. This is not recommended because the absence of project size drivers typically means that incomplete information exists which is not a good time to do an estimate.

Number of System Requirements

This driver represents the number of requirements for the system-of-interest at a specific level of design. 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 may have effort associated with it such as verification and validation, functional decomposition, functional allocation, etc. System requirements can typically be quantified by counting the number of applicable shalls/wills/shoulds/mays in the system or marketing specification. Note: some work is involved in decomposing requirements so that they may be counted at the appropriate system-of-interest.

Easy	Nominal	Difficult
- Simple to implement	- Familiar	- 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

Number of System Interfaces

This driver represents the number of 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 external and internal system interfaces among ISO/IEC 15288-defined system elements.

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

Number of System-Specific Algorithms

This driver represents 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. 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 brand new 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

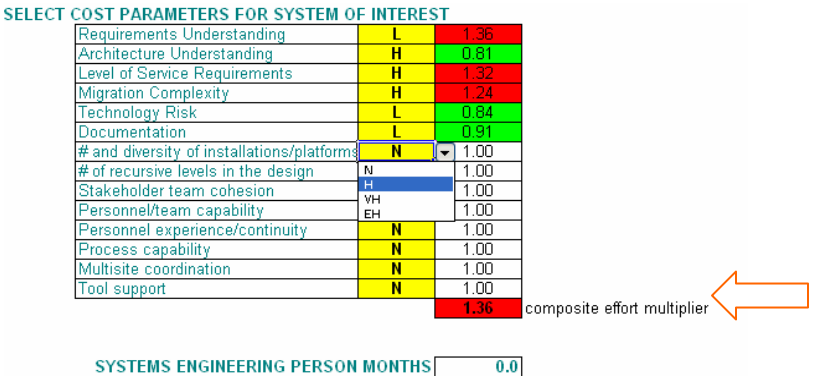
Number of Operational Scenarios

This driver represents the number of operational scenarios that a system must satisfy. Such scenarios include both the nominal stimulus-response thread plus all of the off-nominal threads resulting from bad or missing data, unavailable processes, network connections, or other exception-handling cases. The number of scenarios can typically be quantified by counting the number of system test thread packages or unique end-to-end tests used to validate the system functionality and performance or by counting the number of use cases, 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

d) Cost Drivers

The cost drivers in the model represent the multiplicative part of the model introduced. These drivers are also referred to as effort multipliers since they affect the entire systems engineering effort calculation in a multiplicative manner. Assigning ratings for these drivers is not as straight forward as the size drivers mentioned previously. The difference is that most of the cost drivers are qualitative in nature and require subjective assessment in order to be rated. Provide a rating for each of the cost drivers that apply to your project/system of interest by using the drop-down box in the yellow cells of the spreadsheet. As values are selected, the cells will change colors to represent either a cost savings (green) or a cost penalty (red).



The model displays the composite effort multiplier in cell D25 which is a running total of the product of the fourteen cost drivers. It is an indicator of the overall environment in which the systems engineering is being performed.

1. Requirements Understanding

This cost driver rates the level of understanding of the system requirements by all stakeholders including the systems, software, hardware, customers, team members, users, etc. Primary sources of added system engineering effort are unprecedented systems, unfamiliar domains, or systems whose requirements are emergent with use.

Very Low	*Low*	*Nominal*	*High*	*Very High*
Poor: emergent requirements or unprecedented systems	Minimal: many undefined areas	Reasonable: some undefined areas	Strong: few undefined areas	Full: understanding of requirements, familiar systems

2. Architecture Understanding

This cost driver rates the relative difficulty of determining and managing the system architecture in terms of platforms, standards, components (COTS, GOTS, NDI, new), connectors (protocols), and constraints. This includes tasks like systems analysis, tradeoff analysis, modeling, simulation, case studies, etc.

Very low	Low	Nominal	High	Very High
Poor understanding of architecture and COTS, unprecedented system	Minimal understanding of architecture and COTS, many unfamilar areas	Reasonable understanding of architecture and COTS, some unfamiliar areas	Strong understanding of architecture and COTS, few unfamiliar areas	Full understanding of architecture, familiar system and COTS
>6 level WBS	5-6 level WBS	3-4 level WBS	2 level WBS	

3. Level of Service Requirements

This cost driver rates the difficulty and criticality of satisfying the ensemble of level of service requirements, such a security, safety, response time, interoperability, maintainability, Key Performance Parameters (KPP's), the "ilities", etc.

	Very low	Low	Nominal	High	Very High
Difficulty	Simple; single dominant KPP	Low, some coupling among KPPs	Moderately complex, coupled KPPs	Difficult, coupled KPPs	Very complex, tightly coupled KPPs
Criticality	Slight inconvenience	Easily recoverable losses	Some loss	High financial loss	Risk to human life

4. Migration Complexity

This cost driver rates the extent to which the legacy system affects the migration complexity, if any. Legacy systems components, databases, workflows, environments, etc., may affect the new system implementation due to new technology introductions, planned upgrades, increased performance, business process reengineering, etc.

	Nominal	High	Very High	Extra High
Legacy contractor	Self; legacy system is well documented. Original team largely available	Self; original development team not available; most documentation available	Different contractor; limited documentation	Original contractor out of business; no documentation available
Effect of legacy system on	Everything is new; legacy system is	Migration is restricted to integration only	Migration is related to integration and	Migration is related to integration,

new system	completely replaced or non-existent		development	development, architecture and design

5. Technical Risk

This represents the maturity, readiness, and obsolescence of the technology being implemented. Immature or obsolescent technology will require more Systems Engineering effort.

Viewpoint	*Very low*	*Low*	*Nominal*	*High*	*Very High*
Lack of Maturity	Technology proven and widely used throughout industry	Proven through actual use and ready for widespread adoption	Proven on pilot projects and ready to roll-out for production jobs	Ready for pilot use	Still in the laboratory
Lack of readiness	Mission proven (TRL 9)	Concept qualified (TRL 8)	Concept has been demonstrated (TRL 7)	Proof of concept validated (TRL 5 & 6)	Concept defined (TRL 3 & 4)
Obsolescence			Technology is the state of the practice, Emerging technology could compete in future	Technology is stale. New and better technology is on the horizon in near -term	Technology is outdated and use should be avoided in new system. Spare parts supply is scarce.

6. Documentation match to life cycle needs

This represents the formality and detail of the documentation required to be formally delivered based upon the life cycle needs of the system.

Viewpoint	Very low	Low	Nominal	High	Very High
Formality	General goals, stories	Broad guidance, flexibility is allowed	Risk-driven degree of formality	Partially streamlined process, largely standards-driven	Rigorous, follows strict standards and requirements
Detail	Minimal or no specified documentation and review requirements relative to life cycle needs	Relaxed documentation and review requirements relative to life cycle needs	Risk-driven degree of formality, amount of documentation and reviews in sync and consistent with life cycle needs of the system	High amounts of documentation, more rigorous relative to life cycle needs, some revisions required	Extensive documentation and review requirements relative to life cycle needs, multiple revisions required

7. Number and Diversity of Installations or Platforms

The number of different platforms that will host the system and number of installations required. The complexity of the operating environment (space, sea, land, mobile, portable, information assurance / security) must be considered in weighting your answer. In a wireless network environment it could be the number of unique installation sites and the number of or types of fixed clients, mobile clients, and servers. The number of platforms being implemented should be added to the number being phased out (dual count), in order to account for total life cycle labor.

Viewpoint	*Nominal*	*High*	*Very High*	*Extra High*
Sites & installations	Single installation site or configuration	2-3 site or diverse installation configurations	4-5 sites or diverse installation configurations	> 6 sites or diverse installation configuration
Operating environment	Existing facility meets all known environmental operating requirements	Moderate environmental constraints. Controlled environment HVAC constraints or electrical power constraints	Ruggedized mobile land-based requirements. Some information security requirements. Coordination several regulatory or cross functional agencies required.	Harsh environment (space, sea, airborne), sensitive information security requirements. Coordination between 3 or more regulatory or cross functional agencies required.
Platforms	< 3 types of platforms being installed and or being phased out or replaced	4-7 types of platforms being installed and or being phased out or replaced.	8-10 types of platforms being installed and or being phased out or replaced	> 10 types of platforms being installed and or being phased out or replaced
	Homogeneous platform	Compatible platforms	Heterogeneous, but compatible platform	Heterogeneous, incompatible platforms
	Typically networked using a single industry standard protocol	Typically networked using a single industry standard protocol and multiple operating systems	Typically network using mix of industry standard protocols and proprietary protocols with single operating systems	Typically networked using a mix of industry standard protocols and proprietary protocols with multiple operating systems.

8. Number of Recursive Levels in the Design

The number of levels of design related to the system-of-interest (as defined by ISO/IEC 15288) and the amount of required SE effort for each level.

Viewpoint	*Very Low*	*Low*	*Nominal*	*High*	*Very High*
Number of levels	1	2	3 to 5	6 to 7	**>** 7
Required SE Effort	Focused on single product	Some vertical and horizontal coordination	More complex interdependencies coordination and trade-off analysis	Very complex interdependencies coordination and trade-off analysis	Extremely complex interdependencies coordination and trade-off analysis

9. Stakeholder Team Cohesion

This represents a multi-attribute parameter which includes leadership, shared vision and diversity of stakeholders, approval cycles, group dynamics, IPT framework, team dynamics and amount of change in responsibilities. It further represents the heterogeneity in stakeholder community of the end users, customers, implementers, and development team.

Viewpoint	*Very Low*	*Low*	*Nominal*	*High*	*Very High*
Culture	Stake holders with diverse domain experience, task nature, language, culture, infrastructure of highly heterogeneous stakeholder communities	Heterogeneous stakeholder community. Some similarities in language and culture.	Shared project culture.	Strong team cohesion and project culture. Multiple similarities in language and expertise.	Virtual homogeneous stake holder communities. Institutionalized project culture.
Compatibility	Highly conflicting organizational objectives	Converging organizational objectives	Compatible organizational objectives	Clear roles and responsibilities.	Strong mutual advantage to collaboration.
Familiarity	Unfamiliar-never worked together	Willing to collaborate-little experience	Some familiarity	High level of familiarity	Extensive successful collaboration

10. Personnel Experience and Continuity

The applicability and consistency of the staff at the initial stage of the project with respect to the domain, customer, user, technology, tools, etc.

Viewpoint	Very Low	Low	Nominal	High	Very High
Experience	Less than 2 months	1 yr continuous experience or other similar technical tasks in similar job	3 years of continuous experience	5 years of continuous experience	10 years of continuous experience
Annual Turnover	48%	24%	12%	6%	3%

11. Process Capability

The consistency and effectiveness of the project team at performing SE processes. This may be based on assessment ratings from a published process model (e.g., CMMI, EIA-731, SE-CMM, ISO/IEC15504). It can alternatively be based on project team behavioral characteristics, if no assessment has been performed.

	Very Low	Low	Nominal	High	Very High	Extra High
Assessment Rating	Level 0 (if continuous model)	Level 1	Level 2	Level 3	Level 4	Level 5
Project Team Behavioral Characteristics	Ad Hoc approach to process performance	Performed SE process, activities driven only by immediate contractual or customer requirements, SE focus limited	Managed SE process, activities driven by customer and stakeholder needs in a suitable manner, SE focus is requirements through design, project-centric approach – not driven by organizational processes	Defined SE process, activities driven by benefit to project, SE focus is through operation, process approach driven by organizational processes tailored for the project	Quantitatively Managed SE process, activities driven by SE benefit, SE focus on all phases of the life cycle	Optimizing SE process, continuous improvement, activities driven by system engineering and organizational benefit, SE focus is product life cycle & strategic applications
SEMP Sophistication	Management judgment is used	SEMP is used in an ad-hoc manner only on portions of the project that require it	Project uses a SEMP with some customization	Highly customized SEMP exists and is used throughout the organization	The SEMP is thorough and consistently used; organizational rewards are in place for those that improve it	Organization develop best practices for SEMP; all aspects of the project are included in the SEMP; organizational rewards exist for those that improve it

12. Multisite Coordination

Location of stakeholders, team members, resources, corporate collaboration barriers.

	Very Low	Low	Nominal	High	Very High	Extra High
Collocation	International, severe time zone impact	Multi-city and multi-national, considerable time zone impact	Multi-city or multi-company, some time zone effects	Same city or metro area	Same building or complex, some co-located stakeholders or onsite representation	Fully co-located stakeholders
Communications	Some phone, mail	Individual phone, FAX	Narrowband e-mail	Wideband electronic communication	Wideband electronic communication, occasional video conference	Interactive multimedia
Corporate collaboration barriers	Severe export and security restrictions	Mild export and security restrictions	Some contractual & Intellectual property constraints	Some collaborative tools & processes in place to facilitate or overcome, mitigate barriers	Widely used and accepted collaborative tools & processes in place to facilitate or overcome, mitigate barriers	Virtual team environment fully supported by interactive, collaborative tools environment

13. Tool Support

Coverage, integration, and maturity of the tools in the Systems Engineering environment.

Very low	Low	Nominal	High	Very High
No SE tools	Simple SE tools, little integration	Basic SE tools moderately integrated throughout the systems engineering process	Strong, mature SE tools, moderately integrated with other disciplines	Strong, mature proactive use of SE tools integrated with process, model-based SE and management systems

e) Work Breakdown Structure

The definition of systems engineering used in COSYSMO hinges on the 33 activities defined in the ANSI/EIA 632 standard shown in Table 1. These activities are rolled up into Process Categories and Fundamental Processes. The latter categories will be used later as a general framework for categorizing systems engineering effort.

Table 1. Systems Engineering Work Breakdown Structure per ANSI/EIA 632

Fundamental Processes	Process Categories	Activities
Acquisition and Supply	Supply Process	(1) Product Supply
	Acquisition Process	(2) Product Acquisition, (3) Supplier Performance
Technical Management	Planning Process	(4) Process Implementation Strategy, (5) Technical Effort Definition, (6) Schedule and Organization, (7) Technical Plans, (8)Work Directives
	Assessment Process	(9) Progress Against Plans and Schedules, (10) Progress Against Requirements, (11) Technical Reviews
	Control Process	(12) Outcomes Management, (13) Information Dissemination
System Design	Requirements Definition Process	(14) Acquirer Requirements, (15) Other Stakeholder Requirements, (16) System Technical Requirements
	Solution Definition Process	(17) Logical Solution Representations, (18) Physical Solution Representations, (19) Specified Requirements
Product Realization	Implementation Process	(20) Implementation
	Transition to Use Process	(21) Transition to use
Technical Evaluation	Systems Analysis Process	(22) Effectiveness Analysis, (23) Tradeoff Analysis, (24) Risk Analysis
	Requirements Validation Process	(25) Requirement Statements Validation, (26) Acquirer Requirements, (27) Other Stakeholder Requirements, (28) System Technical Requirements, (29) Logical Solution Representations
	System Verification Process	(30) Design Solution Verification, (31) End Product Verification, (32) Enabling Product Readiness
	End Products Validation Process	(33) End products validation

2. Model Usage

a) Determining size via REQ, INTF, ALG, SCN

Requirements

Different systems will exhibit different levels of requirements decomposition depending on the application domain, customer's ability to write good system requirements, and the functional size of the system. The following rules should increase the reliability of requirements counting by different organizations on different systems regardless of their application domain:

1. **Determine the system of interest.** For an airplane, the system of interest may be the avionics subsystem or the entire airplane depending on the perspective of the organization interested in estimating systems engineering. This key decision needs to be made early on to determine the scope of the COSYSMO estimate and identify the requirements that are applicable for the chosen system.
2. **Decompose system objectives, capabilities, or measures of effectiveness into requirements that can be tested, verified, or designed.** The decomposition of requirements must be performed by the organization using COSYSMO because the initial set of requirements provided by the customer may not be representative of the actual systems engineering effort required for the contractor to deliver the system. The level of decomposition of interest for COSYSMO is the level in which the system will be designed and tested; which is equivalent to the TYPE A, System/Segment Specification (MIL-STD 490-A 1985). For some organizations, these are referred to as "systems engineering requirements" because they reflect the level at which systems engineers do their job.
3. **Provide a graphical or narrative representation of the system of interest and how it relates to the rest of the system.** This step focuses on the hierarchical relationship between the system elements. This information can help describe the size of the system and its levels of design. It serves as a sanity check for the previous two steps.
4. **Count the number of requirements in the system/marketing specification or the verification test matrix for the level of design in which systems engineering is taking place in the desired system of interest.** The focus of the counted requirements needs to be for systems engineering. Lower level requirements may not be applicable if they have no effect on systems engineering. Requirements may be counted from the Requirements Verification Trace Matrix (RVTM) – or an equivalent construct – that is used for testing system requirements. The same rules apply as before: all counted requirements must be at the same design or bid level and lower level requirements must be disregarded if they do not influence systems engineering effort.
5. **Determine the volatility, complexity, and reuse of requirements.** Once the quantity of requirements has been determined, the three adjustment factors can be applied. Currently three complexity factors have been determined: easy, nominal, and difficult. These weights for these factors were determined using expert opinion through the use of a Delphi survey. The volatility and reuse factors are optional and depend on the version of COSYSMO implementation being used.

The objective of the five steps is to lead users down a consistent path of similar logic when determining the number of system requirements for the purposes of estimating systems engineering effort in COSYSMO. It has been found that the level of decomposition described in step #2 may be the most volatile step as indicated by the data collected thus far. To alleviate this, a framework for software use case decomposition[5] was adopted. The basic premise behind the framework is that different levels exist for specific system functions. Choosing the appropriate level can provide a focused basis for describing the customer and developer needs. A metaphor is used to describe four levels: *sky level, kite level, sea level,* and *underwater level*. The *sea level* goals represent a user level task that is the target level for counting requirements in COSYSMO.

[5] Cockburn, A., *Writing Effective Use Cases,* Boston, Addison-Wesley, 2001.

Interfaces

Similar challenges of decomposition exist for the # of interfaces driver because interfaces are often defined at multiple levels of the system hierarchy. The target level for counting interfaces involves the following rules:

1. **Focus on technical interfaces only.** Other parameters in the model address organizational interfaces.
2. **Identify the interfaces that involve systems engineering for your system of interest.** Counting interfaces at the integrated circuit or software subroutine level is often too low. Sometimes there may be multiple levels of interfaces connecting higher system elements, lower system elements, and elements at the same level of the system hierarchy. Identify which level is driving the amount of systems engineering effort in your organization and focus on it.
3. **Determine the number of unique interface types.** If twenty interfaces exist but there are only two types of unique interfaces, then the relevant number to count is two. This is especially true if there is systems engineering effort involved with developing a unique test procedure for each of the unique interfaces.
4. **Focus on the logical aspects of the interface to determine complexity.** This provides a better indicator of the complexity of each interface from a systems engineering standpoint. Counting the number of wires in an interface may not be a good indicator. Instead, the protocol used or the timing requirement associated with the interface will be a better indicator of complexity.
5. **Consider directionality of the interface.** Bidirectional interfaces count as two interfaces because they require coordination on both ends.

Algorithms

Since the influence of algorithms can vary by organization, the process of identifying an algorithm for COSYSMO can also be different. Ultimately the sources from which the number of algorithms can be obtained change as the system definition matures. For example, during the conceptual stage of a system, where there is a limited amount of information available, the only indicators may be functional block diagrams. As the system design evolves and more uncertainties are resolved, there are more sources available to aid in the estimation of algorithms. Table 2 includes examples of the entities that are available at different stages of the system life cycle and their corresponding attributes that can be used to estimate the number of algorithms. They are listed in typical order of availability; the first entities are typically available during the conceptual stages while the latter ones are available as the system design evolves.

Table 2. Candidate Entities and Attributes for Algorithms

Entities	Attributes
Historical database	# of algorithms
Functional block diagram	# of functions that relate to algorithms
Mode description document	algorithms
Risk analysis	algorithm related risks
System specification	algorithms
Subsystem description documents	algorithms
Configuration baseline	technical notes

The attributes may provide more detailed information about the functions that the algorithms perform. This can aid in determining the complexity of that algorithm, an important step in estimating size for COSYSMO.

The approach for determining the quantity of algorithms in a system is unavoidably different for each organization. System algorithms are unique in the sense that they are highly related to the "# of Requirements" and "# of Interfaces" size drivers. If not explicitly defined up front, the number of algorithms can be derived from a system-level requirement or deduced from the properties of an interface. In terms of systems engineering effort, the existence of an algorithm

introduces additional work related to simulation, implementation, test cases, documentation, and support. These activities are illustrated in Figure 1.

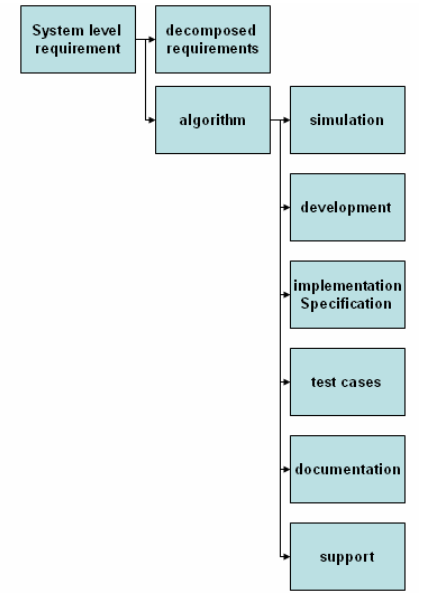


Figure 1. Effort Decomposition Associated With an Algorithm

There exists an entire process in which the general types of algorithms needed are determined, math is developed to implement them, algorithm-related requirements are communicated to other designers (subsystems, hardware, software, etc.) for what data and data quality requirements, and algorithm trade-offs are performed. These activities are within the scope of systems engineering and are covered in COSYSMO.

In some cases, a significant amount of effort associated with systems engineering as related to algorithms will involve reuse which can reduce the complexity of algorithms and in turn the effort associated with their implementation. Conversely, there may be situations where algorithms are unprecedented and loosely defined. From an implementation standpoint, the number of design constraints – such as timing restrictions or processor limitations – may influence the complexity of software algorithms when compared to hardware algorithms. In either case, both types of algorithms should be counted and assigned a level of complexity for input into COSYSMO.

To demonstrate the process of identifying and counting an algorithm an example is provided from the field of signal processing. For purposes of this example it is assumed that a system specification has been developed. From this specification, the following system level requirement is obtained: *All images captured by the sensor shall be compressed in compliance with MPEG-4 coding standard.* This requirement triggers several possible solutions that meet the required standard. A developer may decide to implement the requirement with a well-known algorithm used for compressing visual images: *MPEG-4 Visual Texture Coding (VTC)*. As illustrated in Figure 1, this algorithm generates products associated with it which lead to increased systems engineering effort that is estimated by COSYSMO. Other effort generated by the implementation specification, such as software engineering, is not estimated by COSYSMO. Models such as COCOMO II should be used to estimate the software development effort. For purposes of COSYSMO, the MPEG-4 VTC algorithm counts as one distinct algorithm even if it is used multiple times in the same system. Since this is a well known algorithm with predictable behavior it qualifies as an “easy” algorithm.

Operational Scenarios
In a similar way requirements were defined at *sea level*, operational scenarios must also be identified at a level that is of interest to systems engineering. Operational scenarios are often obtained via test cases or system use cases since they represent end-to-end system functionality or independent capabilities of a system. For example, an operational scenario for a Windows XP computer is to operate in "safe mode". Use case diagrams in UML are also helpful for determining the number and complexity of use cases in a system.

b) How to avoid double counting between size drivers

Each size driver takes the form of both a continuous and categorical variable. As a continuous variable it can represent quantities for "requirements" or interfaces", which can range from small values to very large ones; with most cases falling within an expected range (i.e., most systems have hundreds of systems engineering requirements). As a categorical variable it can be represented in terms of discrete categories such as "easy", "nominal", or "difficult" to describe the complexity associated with each requirement, interface, etc. It is expected that most systems have requirements that fall into at least one of these categories. The categorical scales are presented next and the counting rules for determining the values of the continuous variables are provided in the following sections.

Three strategies exist for counting size drivers in COSYSMO. They are presented here by most ideal to least ideal, although the strategy employed will be primarily determined by the phase of the life cycle in which the system is in.

Strategy 1: Pure size drivers
The ideal situation is when the size drivers are readily available to be counted because they are well documented in a database such as DOORS or are easily identified in system documentation. This gives the user the highest level of confidence when determining the number of requirements, interfaces, algorithms, and operational scenarios in a system.

Step 1: Determine which of the four size drivers are relevant to the system of interest. Since COSYSMO calculates the functional size of the system as weighted sum of requirements, interfaces, algorithms, and scenarios, any combination of the four drivers is an acceptable input. However, too much information presents its own challenges. If duplicated information exists in two different sources, such as interfaces also being described as requirements, use only one of them and be sure to

- favor the driver that provides the most reliable information; and
- favor the driver that is most representative of systems engineering effort

Step 2: Distribute the size driver quantities among categories of easy, nominal, and difficult. Since there are four size drivers and three levels of complexity for each, there user has twelve fields where size information can be entered.

Step 3: Identify whether any of the effort represented by multiple size drivers is being double counted. There nature of double counting frequently stems from situations where an interface is described by both an interface control document and a system specification. This can subsequently lead to the same technical effort being counted in terms of number of interfaces and number of system requirements, leading to an inflated count of functional size and thereby and overestimate of systems engineering effort. Whenever the user needs to choose between one driver or the other, the following criteria should be helpful in deciding which will provide more reliable information:

- favor information that comes from clearly defined project documentation that contains information that is easy to count
- if requirements are not available early in the life cycle, then favor information that is available such as operational scenarios
- favor information that is less likely to change over the course of the project
- favor information that is most representative of systems engineering effort

Guidance on how to identify the best level of system decomposition (also referred to as "sea level") at which to count requirements is discussed in section 2. a). Guidance on how to allocate requirements among the three complexity categories is discussed in section 2. c). Additional approaches to facilitate the size driver counting process are provided in section 2. e). and include the use of systems engineering tools (i.e., DOORS) and frameworks (i.e., DODAF).

Strategy 2: Pure requirements
The requirements specification document often emerges as the dominant source for functional size because requirements are commonly accepted vehicles for describing detailed system functionality. Equivalent requirements can serve as a baseline measure of functional size which could otherwise be represented by the three other size drivers.
In cases where interfaces, algorithms, or operational scenarios are not available, functional size can be estimated by using requirements only. The translation of system attributes across easy/nominal/difficult requirements may be less reliable because the complexity aspects of the other size drivers may be lost in translation.

Strategy 3: Equivalent requirements
In some situations, characteristics of the system may be described in documents where requirements are not mentioned. These include interface control documents, functional block diagrams, and operational concept documents. In this case, interfaces, algorithms, or high-level operational scenarios may be the only way to quantify systems engineering effort due to the absence of requirements. This situation is common during the early stages of system conceptualization, and ironically, the time when COSYSMO is most useful despite the high degree of uncertainty of the system definition. Nevertheless, the quantities of interfaces, algorithms, and scenarios should be entered into the model. Mathematically, these quantities are converted into equivalent requirements in the model since all parameter weights are relative to Nominal # of System Requirements.

Despite which strategy is used, COSYSMO assumes that the systems engineering effort being counted has life cycle considerations that include works associated with conceptualize, develop, OT&E, and transition to operation.

c) Deciding between Easy, Medium, and Difficult

As described in section 1. c), the quantities for each size driver can be distributed across easy, nominal, and difficult complexity categories. It is not expected that all requirements are easy nor are they all difficult. The user must make an assessment of the distribution of each size driver is among the complexity categories based on the nature of the system. The viewpoints provided for each size driver should serve as guides to make this decision but ultimately the assessment of complexity should be made with the impact of systems engineering effort in mind.

Fore example, a system with 1,000 requirements could have 40% of them be considered easy because they are straightforward and have been implemented successfully before, 40% of them considered to be nominal because they are moderately complex and require some effort, and 20% of them could be difficult because they are very complex and have a high degree of overlap with other requirements. The distribution would then be:

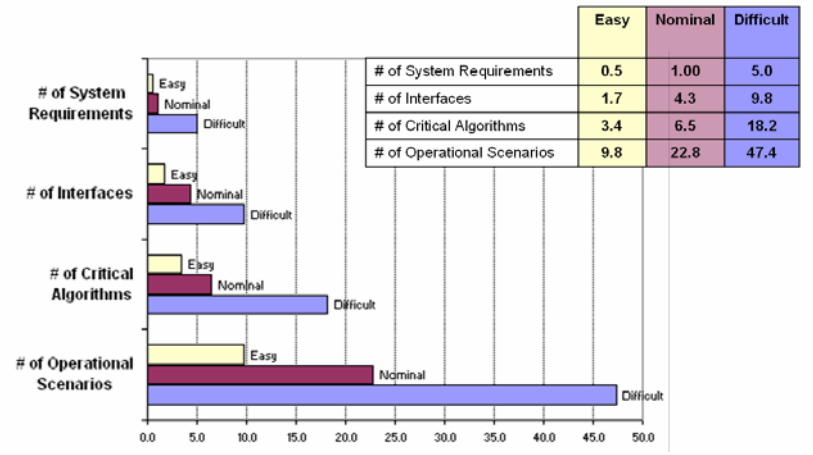
400	**400**	**200**
Easy	Nominal	Difficult
- Simple to implement	- Familiar	- Complex to implement or engineer
- Traceable to source	- Can be traced to source with some effort	- Hard to trace to source
- Little requirements overlap	- Some overlap	- High degree of requirements overlap

The model algorithm will apply the appropriate weights, from Figure 2, and calculate the equivalent number of requirements as follows:

400 easy requirements * 0.5 weight for easy = 200 equivalent requirements
400 nominal requirements * 1.0 weight for nominal = 400 equivalent requirements
200 difficult requirements * 5.0 weight for difficult = 1,000 equivalent requirements

TOTAL = 1,600 equivalent requirements

Rather than the 1,000 original requirements defined in the system specification, COSYSMO will estimate systems engineering effort based on 1,600 equivalent requirements due to the distribution of complexity selected by the user.



	Easy	Nominal	Difficult
# of System Requirements	0.5	1.00	5.0
# of Interfaces	1.7	4.3	9.8
# of Critical Algorithms	3.4	6.5	18.2
# of Operational Scenarios	9.8	22.8	47.4

Figure 2. Size Driver Weights

The same process is followed for the three other size drivers and eventually they are all converted into equivalent requirements.

d) Adjusting nominal effort (rating cost drivers)

After determining the size of the system by assigning values to the size drivers, the user must give an assessment of the system of interest with respect to the understanding, risk, implementation difficulty, development environment, and people capability with respect to systems engineering. This is done through the cost drivers, also referred to as effort multipliers, because of their multiplicative effect on the systems engineering effort calculation. Assigning ratings for these drivers is not as straight forward as the size drivers mentioned previously. The difference is that most of the cost drivers are qualitative in nature and require subjective assessment in order to be rated.

In addition to a definition, each driver has a corresponding rating scale that describes different attributes that could be used to rate the degree of impact on systems engineering effort. Rating levels include: Very Low, Low, Nominal, High, Very High, and in some cases Extra High. The Nominal level is assigned a multiplier of 1.0 and therefore represents no impact on the systems engineering effort estimate. Levels above and below nominal are assigned multipliers above or below 1.0 according to their individual impact on systems engineering effort. The incremental impact of each step along a multiplier's rating scale depends on the polarity of each driver. For example, the requirements understanding multiplier is defined in such a way that *Very Low* understanding will have a productivity penalty on systems engineering. As a result, it will have a multiplier of greater than 1.0, such as 1.85, to reflect an 85% productivity penalty. The rating scale values for the cost drivers are provided in Table 3.

Table 3. Rating Scale Values for Cost Drivers

	Very Low	Low	Nominal	High	Very High	Extra High	*EMR*
Requirements Understanding	1.85	1.36	1.00	0.77	0.60		*3.08*
Architecture Understanding	1.62	1.27	1.00	0.81	0.65		*2.49*
Level of Service Requirements	0.62	0.79	1.00	1.32	1.74		*2.81*
Migration Complexity			1.00	1.24	1.54	1.92	*1.92*
Technology Risk	0.70	0.84	1.00	1.32	1.74		*2.49*
Documentation	0.82	0.91	1.00	1.13	1.28		*1.56*
# and diversity of installations/platforms			1.00	1.23	1.51	1.86	*1.86*
# of recursive levels in the design	0.80	0.89	1.00	1.21	1.46		*1.83*
Stakeholder team cohesion	1.50	1.22	1.00	0.81	0.66		*2.27*
Personnel/team capability	1.48	1.22	1.00	0.81	0.66		*2.28*
Personnel experience/continuity	1.46	1.21	1.00	0.82	0.67		*2.18*
Process capability	1.46	1.21	1.00	0.88	0.77	0.68	*2.15*
Multisite coordination	1.33	1.15	1.00	0.90	0.80	0.72	*1.85*
Tool support	1.34	1.16	1.00	0.85	0.73		*1.84*

For example, the *Requirements Understanding* driver is worded positively since there is an effort savings associated with High or Very High understanding of the requirements. This is indicated by multipliers of 0.77 and 0.60, respectively representing a 23% and 40% savings in effort compared to the nominal case. Alternatively, the *Technology Risk* driver has a cost penalty of 32% for High and 74% for Very High. Not all rating levels apply to all of the drivers. Again, it is a matter of how the drivers are defined. The *Migration Complexity* driver, for example, only contains ratings at Nominal and higher. The rationale behind this is that the more complex the legacy system migration becomes, the more systems engineering work will be required. Not having a legacy system as a concern, however, does not translate to a savings in effort. The absence of a legacy system is the Nominal case which corresponds to a multiplier of 1.0.

The cost drivers are compared to each other in terms of their range of variability, or Effort Multiplier Ratio. The EMR column in Table 3 is representative of an individual driver's possible influence on systems engineering effort. The four most influential cost drivers are: *Requirements Understanding, Level of Service Requirements, Technology Risk,* and *Architecture Understanding*. The least influential, *Documentation, # of Installations, Tool Support*, and # *of Recursive Levels in the Design* were kept because users wanted to have the capability to estimate their impacts on systems engineering effort. The relatively small influence of these four drivers does not mean that the model users felt they were insignificant. Their presence gives users the ability to quantify their impact on systems engineering.

e) Additional tools to facilitate size driver counting process

Early in the system life cycle, well defined project documentation may not be available to organizations due to the evolutionary nature of systems. In this case surrogate sources of data must be obtained or derived in order to capture leading indicators related to the four size drivers. Possible sources are the DoD Architecture Framework products shown in Table 4 because of their availability early in the program life cycle.

Table 4. Useful DODAF Products for COSYSMO Sizing

Driver Name	*Useful Architecture Products*
Number of System Requirements	The architecture development process and the requirements engineering process are interrelated and fairly loopy. The cost modeler can benefit from this iterative exercise and extract quantifiable values.
Number of Major Interfaces	As mentioned earlier, the number of interfaces in the Interface Control Document helps provide a ballpark figure for this size driver. DoDAF products, notably, **SV-1** ("System Interface Description") and **SV-2** ("System Communication Description"), the **OV-3** ("Operational Information Exchange Matrix") can also help in identifying the total number of major interfaces.
Number of Critical Algorithms	The operational and the system view have specific architectural products which can help in estimating the number of critical algorithms. Notably, from the operational view, the **OV-6a, b and c** ("Operational Rules, State Transition and Event-Trace Description") and the **SV-10a, b and c** ("System Rules, State Transition and Event-Trace Description") from the system view can help in understanding and estimating the number of critical algorithms.
Number of Operational Scenarios	The **OV-1** "High-level Operational Concept Graphic" maps to use-cases when architecture products are developed using the Object-Oriented methodology. The total number of use-cases can be used in estimating this size driver.

Requirements management tools such as DOORS are also helpful in populating the # of system requirements driver. An additional step is needed to distribute the quantity of requirements across the three complexity categories; easy, nominal, and difficult. This is a manual process because of the expert judgment needed to assess the relative complexity of the requirements and their impact on systems engineering effort.

3. Model output

a) Project effort distribution across activities

The academicCOSYSMO model provides a single point person month output which requires some interpretation. As show earlier in Table 1, one of the assumptions of the model is that a standard set of systems engineering activities are being performed throughout certain phases in the life cycle. These 33 activities are distributed across 5 fundamental processes as shown in Table 5. This distribution is not universal but it provides a typical spread of effort that is characteristic of systems engineering projects.

Table 5. Effort Distribution Across ANSI/EIA 632 Fundamental Processes

ANSI/EIA 632 Fundamental Process	Typical effort
Acquisition & Supply	7%
Technical Management	17%
System Design	30%
Product Realization	15%
Technical Evaluation	31%

The effort distribution across fundamental processes, P_x, is helpful in determining how to allocate the estimated systems engineering resources in COSYSMO. The sum of the 5 fundamental processes equal the total systems engineering estimate, that is:

$$P_1 + P_2 + P_3 + P_4 + P_5 = 100\%$$

Therefore, the COSYSMO estimate *x* can be allocated to each of the 5 processes.

x * 0.07 = effort required for P1
x * 0.17 = effort required for P2
x * 0.30 = effort required for P3
x * 0.15 = effort required for P4
x * 0.31 = effort required for P5

TOTAL = x

The breakdown of effort by systems engineering process is helpful not only for planning purposes but also when an organization is only interested in estimating part of the systems engineering activities. For example, if the systems engineering organization is only responsible for system design, product realization, and technical evaluation then the typical effort is:

$P_3 + P_4 + P_5$ = adjusted effort factor
0.30 + 0.15 + 0.31 = adjusted effort factor
0.76 = adjusted effort factor

The initial estimate provided by COSYSMO, x, should be adjusted by a factor of 0.76 to reflect the absence of acquisition & supply and technical management activities assumed in the estimate.

b) Project effort distribution across phases

The single point person month output can also be distributed over time. The assumption in the model is that a standard set of systems engineering activities are being performed throughout certain phases in the life cycle. These 4 life cycle phases are: Conceptualize, Develop, Operational Test & Evaluation, and Transition to Operation as shown in

Table 6. This distribution is not universal but it provides a typical spread of effort that is characteristic of systems engineering projects.

Table 6. Systems Engineering Effort Distribution % Across ISO/IEC 15288 Phases

Conceptualize	Develop	Operational Test & Evaluation	Transition to Operation
23	35	28	14

The distribution across life cycle phases, A_x, is helpful in determining how to allocate the estimated systems engineering resources in COSYSMO. The sum of the 4 life cycle phases equal the total systems engineering estimate, that is:

$$A_1 + A_2 + A_3 + A_4 = 100\%$$

Therefore, the COSYSMO estimate *x* can be allocated across each of the 4 life cycle phases.

x * 0.23 = effort needed in A1
x * 0.35 = effort needed in A2
x * 0.28 = effort needed in A3
x * 0.14 = effort needed in A4

TOTAL = x

The breakdown of effort by systems engineering life cycle phase is helpful not only for planning purposes but also when an organization is only interested in estimating part of the systems engineering life cycle. For example, if the systems engineering organization is only responsible for the conceptualization and development of the system then the typical effort is:

$A_1 + A_2$ = adjusted effort factor
0.23 + 0.35 = adjusted effort factor
0.58 = adjusted effort factor

The initial estimate provided by COSYSMO, x, should be adjusted by a factor of 0.58 to reflect the absence of the operational test & evaluation and transition to operation life cycle phases assumed in the estimate.

c) Potential overlap with COCOMO II

The danger with model overlap is that it can lead to unnecessary double-counting of effort because it is expected that systems engineering and software engineering are highly coupled in most organizations. On the surface, COCOMO II and COSYSMO appear to be similar. However, there are fundamental differences between them that should be highlighted. These are apparent when the main features of the model are considered:

- Sizing. COCOMO II uses software size metrics while COSYSMO uses metrics at a level of the system that incorporates both hardware and software
- Life cycle. COCOMO II, based on a software tradition, focuses exclusively on software development life cycle phases defined by MBASE[6] while COSYSMO follows the system life cycle provided by ISO/IEC 15288
- Cost Drivers. Each model includes drivers that model different phenomena. The overlap between the two models is minimal since very few of the COCOMO II parameters are applicable to systems engineering

Other differences are highlighted in Table 7.

Table 7. Differences between COCOMO II and COSYSMO

	COCOMO II	COSYSMO
Estimates	Software development	Systems engineering
Estimates size via	Thousands of Software Lines of Code (KSLOC), Function Points, or Application Points	Requirements, Interfaces, Algorithms, and Operational Scenarios
Life cycle phases	MBASE/RUP Phases: (1) Inception, (2) elaboration, (3) construction, and (4) transition	ISO/IEC 15288 Phases: (1) Conceptualize, (2) Develop, (3) Operation, Test, and Evaluation, (4) Transition to Operation, (5) Operate Maintain or Enhance, and (6) Replace or dismantle.
Form of the model	1 size factor, 5 scale factors, and 18 effort multipliers	4 size factors, 1 scale factor, 14 effort multiplier
Represents diseconomy of scale through	Five scale factors	One exponential system factor

[6] Model Based System Architecting and Software Engineering

Despite the differences between the two models there is potential overlap between the two whenever the models are being used in parallel to estimate the effort involved with delivering a software-intensive system. Part of understanding the overlap between the two models involves deciding which activities are considered "system engineering" and which are considered "software engineering/development" and how each estimation model accounts for these activities.

COCOMO II is designed to estimate the software effort associated with the analysis of software requirements and the design, implementation, and test of software. COSYSMO estimates the system engineering effort associated with the development of the software system concept, overall software system design, implementation and test. The COCOMO II estimate of the software effort will surely account for the additional effort required by any additional testing of the software system; at the same time, the COSYSMO effort will account for additional test development and management since the systems engineers are required to perform additional validation and verification of the system. Either model can account for this effort based on how users wish to allocate the testing activity. Each organization's unique relationship between these two disciplines needs to be reconciled when using COSYSMO and COCOMO II together. One approach for accomplishing this is to examine the Work Breakdown Structures of each discipline.

COSYSMO uses the WBS defined in EIA/ANSI 632 while COCOMO II uses the one defined in MBASE/RUP. The typical effort distribution of a software-intensive system is provided in Table 8 together with the activities that could potentially overlap when using both models during an effort estimation exercise. The numbers in the cells represent the typical percentage of effort spent on each activity during a certain phase of the software development life cycle as defined by COCOMO II. Each column adds up to 100 percent.

Table 8. COCOMO II and COSYSMO Overlaps

Project Stage		Software Development		
	Inception	Elaboration	Construction	Transition
Management	14	12	10	14
Environment/CM	10	8	5	5
Requirements	38	18	8	4
Design	19	36	16	4
Implementation	8	13	34	19
Assessment	8	10	24	24
Deployment	3	3	3	30

COCOMO II only
COSYSMO
Possible COCOMO II/COSYSMO overlap

The scope of COCOMO II includes the elaboration and construction activities generally defined as software development. The gray cells indicate the systems engineering activities that are estimated in COSYSMO. The diagonally shaded cells indicate the COCOMO II/COSYSMO overlap activities that may be double counted when using the models simultaneously. The exact amount of effort being double counted will vary for each organization based on the way they define systems engineering relative to software engineering.

4. Local calibration

The best way to ensure that COSYSMO is accurate to your context is to perform a local calibration. This involves collecting data on completed programs that have a significant systems engineering component and using it to calibrate the model constant, A, described in section 1. b).

a) Data collection

The focus is on collecting data for completed programs since it is necessary to know how much systems engineer effort was actually expended on a program. This information – together with the total # of system requirements, # of interfaces, # of algorithms, and # of operational scenarios – is used to calibrate the model based on the historical performance of the organization. A data collection form is available for download at http://www.valerdi.com/cosysmo in the "Downloads" section. It is recommended that the data collection be done in person hours since this is a uniformly accepted metric. If effort data is maintained in person months then the user must make sure that every project has interpreted person months as 152 person hours. This is an important issue for European users since person months are usually considered to be 138 person hours. In which case, 1 European person month = 1.1 U.S. person months.

b) Measurement process

The recommended systems engineering cost estimation life cycle for COSYSMO is shown in Figure 3. Several verification and validation opportunities exist along the way to tailor the model to an organization.

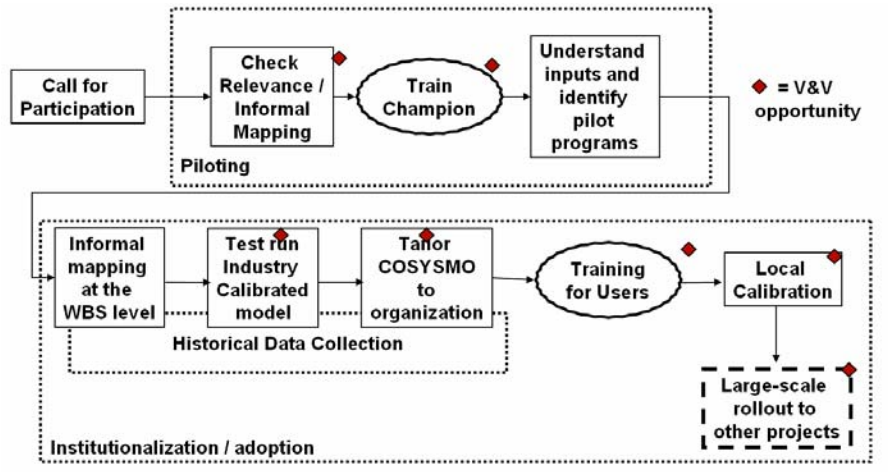


Figure 3. Estimation Life Cycle

The success of this process is dependent on the identification, training, and consistency of an internal COSYSMO champion throughout the piloting and institutionalization phases. Organizations that have identified a committed, reliable, authorized, and knowledgeable champion have been successful in adopting COSYSMO and tailoring it to their needs.

c) Data analysis

Once data on completed programs has been collected, the simplest way to perform a local calibration is to use the Calico tool that accompanies the SystemStar implementation of COSYSMO which is developed by SoftStar systems. For more information visit http://www.cosysmo.com.

d) Modification of model spreadsheet

Upon completion of the local calibration, the new value for A can be entered into the academicCOSYSMO spreadsheet in cell H28. The current value of the industry calibration is 38.55 and assumes that the calibration has been done in person hours. Alternatively, if the calibration was done in person months then the value entered in cell H28 must be multiplied by 152 (this is done to offset the subsequent division by 152 embedded in the algorithm). Regardless of whether the value of A is entered as months (and multiplied by 152) or hours, academicCOSYSMO will provide the estimate in person months in order to be compatible with other cost estimation models.

e) Customization

In addition to the local calibration, further opportunities exist for tailoring the model to a specific organization. These include:

- Adjustment of (Dis)economy of scale constant, E
- Clarification of size driver counting rules (i.e., sea level) and system-of-interest
- Mapping to internal Work Breakdown Structure
- Adjustment of life cycle scope
- Distribution of effort over time
- Addition of new cost or size drivers