Systems Engineering Cost Estimation Workbook

Raymond Madachy
Naval Postgraduate School

Table of Contents

1 Introduction

This workbook covers applications of parametric cost and schedule models for systems engineers estimating systems comprised of software and hardware. It includes models in [1] and [2] for systems and software engineering with updates, a hardware model [3], and worked-out examples for integrated systems.

Parametric cost modeling uses cost estimating relationships (CERs) as mathematical algorithms relating cost factors. The parametric models use numeric inputs for explanatory variables reflecting system characteristics to compute cost. A parametric cost model is defined for a specific aspect of a project, product or process. The models are developed using regression analysis and other methods.

A cost estimating relationship between parameters provides a logical and predictable correlation between the physical or functional characteristics of a system and the resultant cost. A parametric CER may account for factors such as cost quantity relationships, inflation, staff skills, schedules etc. They also provide decomposition of generic cost elements for work breakdown structures (WBS).

Parametric models in general are fast, easy to use and helpful early in a program. The use of parametric models serves as valuable tools for systems engineers and project managers to estimate engineering effort or perform tradeoffs between cost, schedule and functionality.

These models are often implemented in sophisticated software applications where the CER is not apparent to the user but the model parameters can be adjusted to perform a cost estimate. These tools implement the models but the underlying equations are not always visible. This workbook uses parametric cost models in the public domain, with the formulas open and available for use. They are also more valuable this way when the user knows why the models produce the results they do and be better informed for making decisions. The parametric models overviewed next are called “constructive” for these reasons.

1.1. Constructive Cost Models

A primary objective of this workbook's models is to be *constructive*, in which the job of cost and schedule estimation helps people better understand the nature of the project being estimated. This was the reason for the name of the original COnstructive COst MOdel (COCOMO) for users to understand the complexities of development lifecycle processes through the medium of a cost model [1]. A user of constructive cost models can tell why they give the estimates they do.

The underlying core effort formula used in the cost models is

$$Effort = A * Size^{B} * \prod_{i=1}^{N} EM_i \qquad (1)$$

Where

- *Effort* is in Person-Months (PM)
- *A* is a constant derived from historical project data
- *Size* is a measure of the work product
- *B* is an exponent for the diseconomy of scale
- EM_i is an effort multiplier for the i^{th} cost driver. The geometric product of *N* multipliers is an overall Effort Adjustment Factor (EAF) to the nominal effort.

Size is interpreted within the context of the specific work being estimated in the units of the work products. The effort multipliers cover factors for product, platform, personnel and project attributes that affect cost. The resulting top-level effort can be decomposed for each phase, activity or increment. A Person-Month of effort is the equivalent of an average working month with a default of 152 hours/Person-Month.

The basic schedule formula depends on the estimated effort:

$$Schedule = C * Effort^{D} \qquad (2)$$

Where

- *Schedule* is in calendar months
- *C* is a constant derived from historical project data
- *D* is a constant derived from historical project data

The schedule equation can be further adjusted for schedule compression or stretch-out. These details are provided in subsequent sections with the respective models.

The estimated staffing levels for the project duration can be determined with the effort and schedule. By dividing them we can calculate the average staffing levels with:

$$Average\,Staff = \frac{Effort}{Schedule} = \frac{Person\text{-}Months}{Months} = \#\,People \qquad (3)$$

The unit of Months cancels out and results in the number of people assuming a constant staffing level for the duration.

As an example, we will calculate the systems engineering effort, schedule and staffing for a medium size project of 100 nominal system requirements and EAF =1. Using the Constructive Systems Engineering Cost Model (COSYSMO) with the default values of A=.25 and B=1.06 we obtain effort with:

$$Effort = .254 * 100^{1.06} * 1 = 33.5\,Person\text{-}Months$$

The schedule is calculated using C=1.5 and D=.33:

$$Schedule = 1.5 * 33.5^{.33} = 4.8\,Months$$

Finally, the average staff is determined with:

$$Average\,Staff = \frac{Effort}{Schedule} = \frac{33.5\,Person\text{-}Months}{4.8\,Months} = 7\,People\,.$$

2 Systems Engineering Cost Model

The Constructive Systems Engineering Model (COSYSMO) estimates the labor cost of performing systems engineering [2]. The effort equation in COSYSMO is:

$$Effort = A \cdot \left(\sum_k (w_{e,k}\Phi_{e,k} + w_{n,k}\Phi_{n,k} + w_{d,k}\Phi_{d,k}) \right)^B \cdot \prod_{j=1}^{14} EM_j \quad \textbf{(4)}$$

Where

- *Effort* is in Person-Months (PM) for a nominal schedule
- *A* is a constant derived from historical project data
- *k* = {Requirements, Interfaces, Algorithms, Scenarios}
- w_x = weight for "easy", "nominal", or "difficult" size driver
- Φ is the quantity of "k" size driver
- *B* is an exponent for the diseconomy of scale
- EM_i is an effort multiplier for the i^{th} cost driver. The geometric product is an overall Effort Adjustment Factor (EAF) to the nominal effort.

The effort distribution in Table 1 shows the percent of total effort allocated to systems engineering activities using the ANSI/ EIA 632 standard breakdown. E.g., the activity effort for Acquisition and Supply is 7% of the total effort calculated in equation 4.

Table 1: Systems Engineering Effort Distribution

Activities	**Effort %**
Acquisition and Supply	7%
Technical Management	17%
System Design	30%
Product Realization	15%
Technical Evaluation	31%

2.1 Size

The size drivers and their complexity weights are in

Table 2. They are described in the next sections.

Table 2: COSYSMO Size Drivers

	Easy	Nominal	Difficult
# of System Requirements	0.5	1.00	5.0
# of Interfaces	1.1	2.8	6.3
# of Critical Algorithms	2.2	4.1	11.5
# of Operational Scenarios	6.2	14.4	30

Number of System Requirements

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 is such as V&V, 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. 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

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
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
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

2.1.1 Examples: Top-Level Effort, Schedule and Staffing

Below are examples that vary the top-level size, the overall EAF, and illustrate the size weighting approach.

Example 1
Let size=100 requirements, EAF=1
Effort (Person-Months) = $.254 * 100^{1.06} * 1 = .254 * 131.83 * 1 = 33.5$ PM
Schedule (Months) = $1.5 * \text{Effort}^{.33} = 1.5 * 33.5^{.33} = 1.5 * 3.186 = 4.8$ Months
Staff = 33.5 Person-Months / 4.8 Months = 7.0 people

Example 2
Let size =200 requirements, EAF=1.2
Effort = $.254 * 200^{1.06} * 1.2 = .254 * 274 * 1.2 = 83.8$ PM
Schedule = $1.5 * 83.8^{.33} = 6.5$ Months
Staff = 83.8 / 6.5 = 12.95 ~ 13

Example 3
The size weight for easy requirements is used to adjust the equivalent size in the effort equation:

Let size=100 easy requirements, EAF=1
Effort (Person-Months) = $.254 * (.5*100)^{1.06} * 1 = .254 * 63.2 * 1 = 16.1$ PM

The next section details the cost drivers and their effort multipliers that comprise the overall Effort Adjustment Factor.

2.2 Cost Drivers

Table 3 lists the multiplicative cost drivers and their effort multipliers. The detailed descriptions of the factors and their rating scales are in the next sections.

Table 3: COSYSMO Cost Drivers and Effort Multipliers

	Very Low	Low	Nominal	High	Very High	Extra High	Effort Multiplier Ratio
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	1.55	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

An example of effort multipliers for the cost driver *Level of Service Requirements* is shown in Figure 1.

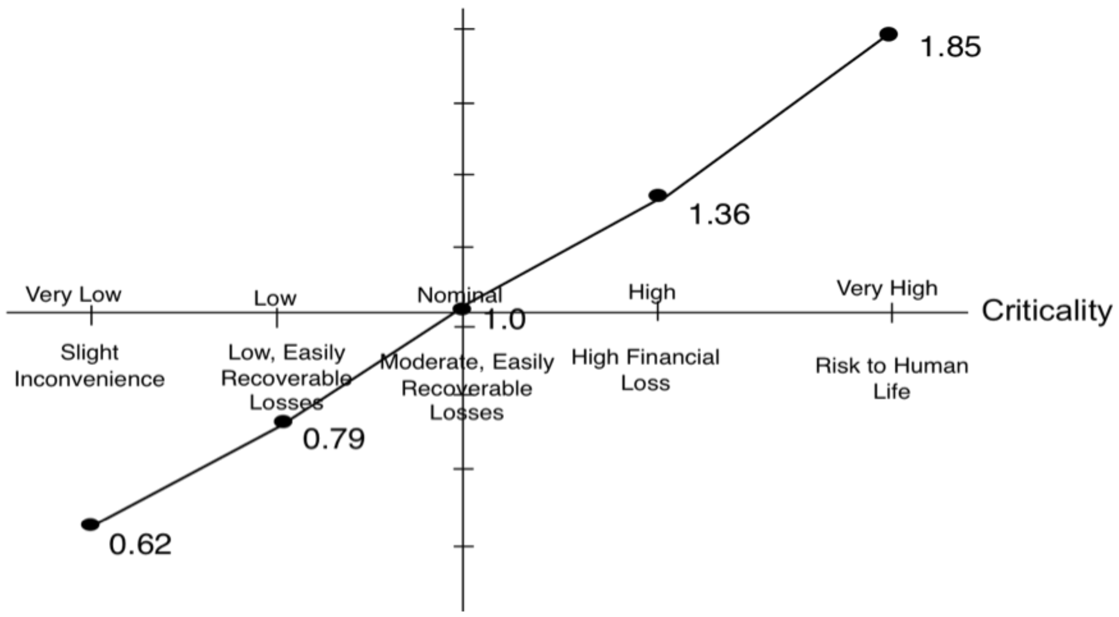


Figure 1: Example Effort Multipliers for Cost Factor *Level of Service Requirements*

The figure helps illustrate the cost tradeoffs provided by the effort multipliers. E.g. a very highly critical system costs 85% more than a nominally reliable system (1.85/1.0=1.85), or a very highly critical system costs 198% more than one rated very low (1.85/.62=2.98).

Requirements Understanding

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 systems 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 system	Minimal: many undefined areas	Reasonable: some undefined areas	Strong: few undefined areas	Full understanding of requirements, familiar system

Architecture Understanding
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 unfamilar 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	

Level of Service Requirements
The difficulty and criticality of satisfying the ensemble of level of service requirements, such as security, safety, response time, interoperability, maintainability, Key Performance Parameters (KPPs), the “ilities”, etc.

Viewpoint	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

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

Viewpoint	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 new system	Everything is new; legacy system is completely replaced or non-existent	Migration is restricted to integration only	Migration is related to integration and development	Migration is related to integration, development, architecture and design

Technology Risk

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**
	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 the near-term	- Technology is outdated and use should be avoided in new systems - Spare parts supply is scarce

Documentation Match To Life Cycle Needs

The formality and detail of documentation required to be formally delivered based on 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

# and Diversity of Installations/Platforms
The number of different platforms that the system will be hosted and installed on. The complexity in the operating environment (space, sea, land, fixed, mobile, portable, information assurance/security). For example, in a wireless network it could be the number of unique installation sites and the number of and types of fixed clients, mobile clients, and servers. Number of platforms being implemented should be added to the number being phased out (dual count).

Viewpoint	**Nominal**	**High**	**Very High**	**Extra High**
Sites/ installations	Single installation site or configuration	2-3 sites or diverse installation configurations	4-5 sites or diverse installation configurations	>6 sites or diverse installation configurations
Operating environment	Existing facility meets all known environmental operating requirements	Moderate environmental constraints; controlled environment (i.e., A/C, electrical)	Ruggedized mobile land-based requirements; some information security requirements. Coordination between 1 or 2 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/replaced	4-7 types of platforms being installed and/or being phased out/replaced	8-10 types of platforms being installed and/or being phased out/replaced	>10 types of platforms being installed and/or being phased out/replaced
	Homogeneous platforms	Compatible platforms	Heterogeneous, but compatible platforms	Heterogeneous, incompatible platforms
	Typically networked using a single industry standard protocol	Typically networked using a single industry standard protocol and multiple operating systems	Typically networked using a mix of industry standard protocols and proprietary protocols; single	Typically networked using a mix of industry standard protocols and proprietary protocols; multiple

			operating systems	operating systems

# 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	3-5	6-7	>7
Required SE effort	Focused on single product	Some vertical and horizontal coordination	More complex interdependencies coordination, and tradeoff analysis	Very complex interdependencies coordination, and tradeoff analysis	Extremely complex interdependencies coordination, and tradeoff analysis

Stakeholder Team Cohesion
Represents a multi-attribute parameter which includes leadership, shared vision, 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	Stakeholders with diverse domain experience, task nature, language, culture, infrastructure 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	Virtually homogeneous stakeholder communities Institutionalized project culture
Compatibility	Highly conflicting organizational objectives	Converging organizational objectives	Compatible organizational objectives	Clear roles & 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

Personnel/team Capability
Basic intellectual capability of a Systems Engineer (compared to the national pool of SEs) to analyze complex problems and synthesize solutions.

Very Low	**Low**	**Nominal**	**High**	**Very High**
15th percentile	35th percentile	55th percentile	75th percentile	90th percentile

Personnel experience/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.

	Very low	**Low**	**Nominal**	**High**	**Very High**
Experience	Less than 2 months	1 year continuous experience, other technical experience 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%**

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 also be based on project team behavioral characteristics, if no assessment has been performed.

	Very low	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Assessment Rating (Capability or Maturity)	Level 0 (if continuous model)	Level 1	Level 2	Level 3	Level 4	Level 5
Project Team	Ad Hoc approach to	Performed SE process,	Managed SE	Defined SE process,	Quantitatively Managed SE	Optimizing SE process,

Behavioral Characteristics	process performance	activities driven only by immediate contractual or customer requirements, SE focus limited	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	activities driven by benefit to project, SE focus is through operation, process approach driven by organizational processes tailored for the project	process, activities driven by SE benefit, SE focus on all phases of the life cycle	continuous improvement, activities driven by system engineering and organizational benefit, SE focus is product life cycle & strategic applications

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

Viewpoint	**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-locate 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 f supported by interactive, collaborative t environment

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

2.2.1 Example: Estimation Case Study

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. From past experience, your organization has a proven track-record of developing similar systems and has been operating at a CMMI Level 3 for several years.

The first step is to decompose 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 entered into the model size drivers.

At this stage, the model provides an initial systems engineering person-month estimate based solely on the systems engineering size parameters. The nominal effort without accounting for the multiplicative cost drivers is 235 Person-Months.

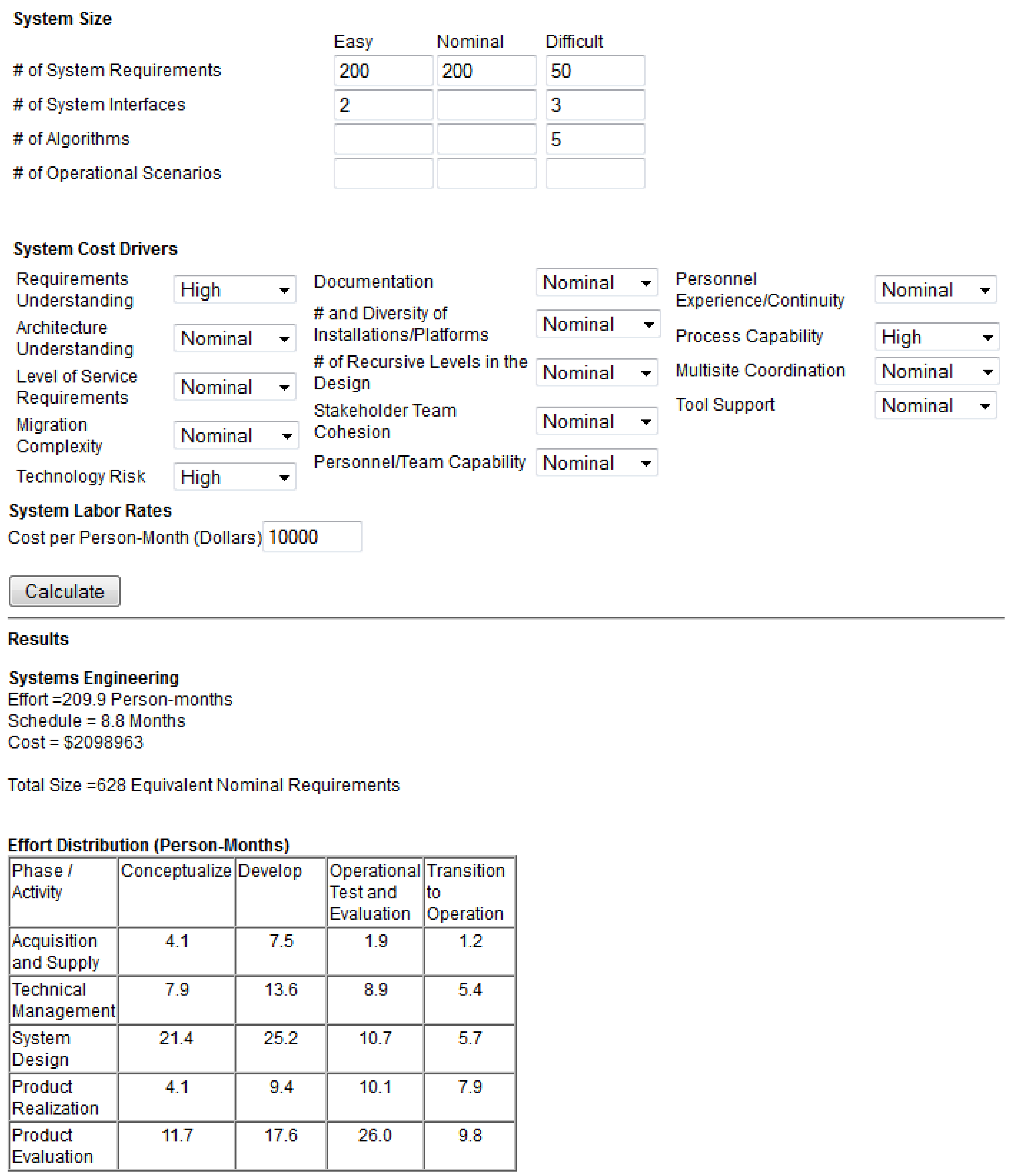
Now the estimate can be adjusted by rating the cost drivers and using their effort multipliers. Additional information about the program is available that can be used about three unique factors on this project.

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 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 selecting the appropriate settings on 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 populate three of the fourteen cost drivers.

The resulting systems engineering estimate is 210 person months as shown in Figure 2. The top level estimate is then decomposed into standard systems engineering life cycle phases and activities in a work breakdown structure in the effort distribution table.



System Size

	Easy	Nominal	Difficult
# of System Requirements	200	200	50
# of System Interfaces	2		3
# of Algorithms			5
# of Operational Scenarios			

System Cost Drivers

Requirements Understanding	High	Documentation	Nominal	Personnel Experience/Continuity	Nominal
Architecture Understanding	Nominal	# and Diversity of Installations/Platforms	Nominal	Process Capability	High
Level of Service Requirements	Nominal	# of Recursive Levels in the Design	Nominal	Multisite Coordination	Nominal
Migration Complexity	Nominal	Stakeholder Team Cohesion	Nominal	Tool Support	Nominal
Technology Risk	High	Personnel/Team Capability	Nominal		

System Labor Rates

Cost per Person-Month (Dollars) 10000

Calculate

Results

Systems Engineering
Effort =209.9 Person-months
Schedule = 8.8 Months
Cost = $2098963

Total Size =628 Equivalent Nominal Requirements

Effort Distribution (Person-Months)

Phase / Activity	Conceptualize	Develop	Operational Test and Evaluation	Transition to Operation
Acquisition and Supply	4.1	7.5	1.9	1.2
Technical Management	7.9	13.6	8.9	5.4
System Design	21.4	25.2	10.7	5.7
Product Realization	4.1	9.4	10.1	7.9
Product Evaluation	11.7	17.6	26.0	9.8

Figure 2: COSYSMO Example

2.2.2 Example: Sensitivity Analysis

The effort multipliers are convenient for tradeoffs and sensitivity analyses. For example, what is the difference in effort for a project if the *Migration Complexity* rating is Nominal vs. High?

The Effort Multiplier at Nominal = 1.0
The Effort Multiplier at High = 1.25

Compute an effort ratio:

$$\text{Ratio} = \frac{\text{Effort @Nominal} = .254 * \text{Size}^{1.06} * 1.0}{\text{Effort @High} = .254 * \text{Size}^{1.06} * 1.25}$$

Now terms cancel out:

$$\text{Ratio} = \frac{\cancel{.254 * \text{Size}^{1.06}} * 1.0}{\cancel{.254 * \text{Size}^{1.06}} * 1.25}$$

Ratio = 1.0 / 1.25 = .80

The effort at *Migration Complexity* = Nominal is 80% of the effort at High, or 20% less effort.

➔ If size doesn't change, the effort ratio between two cases is the ratio of their effort multipliers because all other terms cancel out.

2.3 Equivalent Size and Reuse

Systems engineering reuse is the utilization of systems engineering products from previous efforts. Examples include architectures, requirements, test plans, interfaces, etc. Reuse or modification of existing assets is a means of reducing cost, schedule, and/or risk.

For effort estimation, the key element of size is the equivalent (effective) size of the product. Equivalent size is an adjustment of total size to reflect the actual degree of work involved. A method is used to make new, reused and adapted portions so they can be rolled up into an aggregate size estimate.

In addition to newly developed systems engineering products, those modified and reused from another source and used in the product also contribute to the product's equivalent size for cost models. All of the constructive models use equivalent size for calculating effort this way.

In COSYSMO the size categories and weights in Table 4 are used to calculate equivalent size.

Table 4: COSYSMO Size Categories and Weights

Category	Definition	Weight
New	Products that are completely new	1.0
Designed for Reuse	Products that require an additional upfront investment to improve the potential reusability	1.38
Modified	Products that are inherited, but are tailored	.65
Deleted	Products that are removed from the system	.51
Adopted	Products that are incorporated unmodified (a.k.a. "black box" reuse)	.43
Managed	Products that are incorporated unmodified and with minimal testing	.15

2.3.1 Example: Systems Engineering Reuse

Compare the following systems engineering reuse options. Use the reuse weights and suggest the best cost alternative by comparing the equivalent sizes.

Option 1: All New
Option 2: Modification of Legacy System
- 50% New
- 50% Modified

Calculate the total equivalent size factor by applying reuse weighting factors to product portions.

Option 1: The equivalent size factor = 100% * 1.0 = 1.0.

Option 2: The portions for both new and modified are added together:
(50% * 1.0) + (50% * .65) = 0.825.

Thus option 2 is 82.5% in equivalent size and will be cheaper.

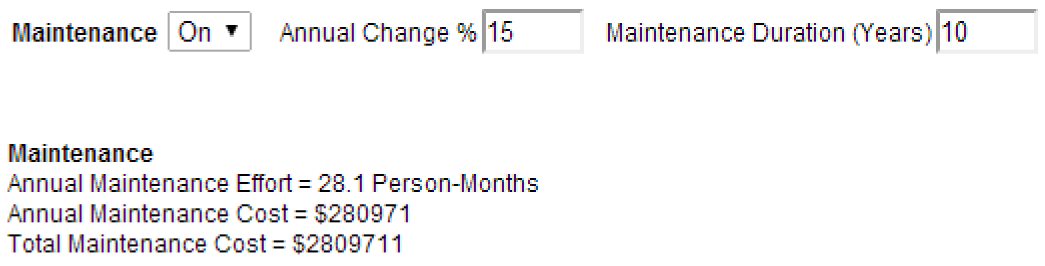
The equivalent size can then be applied in the COSYSMO effort equation to see how the costs compare. In the effort equation cancelling out the other terms (similar to the previous sensitivity analysis) the effort ratio would be:

$(.825 / 1.0)^{1.06}$ = 0.815, or 81.5% effort compared to all new. The difference from the 82.5% size ratio is due to the size exponent effect.

2.4 Systems Engineering Maintenance

Maintenance after initial system delivery can involve either updates or defect fixes. Fixes can be further segregated into corrective, adaptive, or perfective maintenance. Maintenance is typically performed as a level of effort activity, but not always. Traditional maintenance planning is for a constant change percent per annum, which can be related as a fixed percent of equivalent size.

A maintenance example will be illustrated by extending the case study in section 2.2.1. We will assume a 10 year maintenance period with a fixed 15% effective change per year. The example inputs in Figure 3 show specifying the duration in years and percent of overall size being changed every year. The corresponding result for the case study indicates 28.1 Person-Months of effort for annual maintenance.



Maintenance On ▼ Annual Change % 15 Maintenance Duration (Years) 10

Maintenance
Annual Maintenance Effort = 28.1 Person-Months
Annual Maintenance Cost = $280971
Total Maintenance Cost = $2809711

Figure 3: Systems Engineering Maintenance Example

2.5 Exercises

1. This first problem with COSYSMO should be done manually without an automated estimation tool, using the tables of effort multipliers and effort distributions. Calculate the top-level estimate for a project and then decompose it by activity.

a) What is the total effort, total schedule, and average staffing level for systems engineering labor with the following.

$$\text{Effort} = .254 * \text{Size}^{1.06} * \text{EAF}$$
$$\text{Schedule} = 1.5 * \text{Effort}^{0.33}$$

The size is 500 Nominal Requirements and the Effort Adjustment Factor =.8.

b) Now decompose the total effort for the ANSI/ EIA 632 process activities and determine their average staffing levels. Assume all activities last for the full project duration. For each activity determine how much effort and how many people are required, on average.

2. Which 3 multiplicative cost drivers in COSYSMO account for the largest variation in productivity? Use their effort multiplier ratios to determine their productivity ranges.

3. You are asked to estimate the cost impact of modifying the requirements for a project tradeoff study. The stakeholders are considering descoping Migration Complexity from Very High to High. What is the estimated percent change in systems engineering effort?

4. What is the estimated systems engineering effort, schedule and cost for the following project. Use a labor cost of $10,000/Person-Month.

   Size
   - 20 easy requirements
   - 36 nominal requirements
   - 30 difficult requirements
   - 2 nominal interfaces
   - 3 difficult interfaces
   - 5 nominal algorithms
   - 2 difficult algorithms
   - 2 easy scenarios
   - 3 nominal scenarios

   Cost Factors
   Level of Service Requirements = High
   Architecture Understanding = Low

5. Revise the project example in section 2.2.1. Estimates are like other living documents that require updates for new information. Now use the following additional information to revise the estimate based on your continuing analysis.

   - The system will be hosted and installed on 3 sites with diverse configurations. The operating environment has only moderate environmental constraints. It is planned for 5 compatible platform types that are networked together with a single protocol.
   - The team has an average of 5 years of experience working on similar systems with the customer, and a stable staff with 6% annual turnover.
   - The program stakeholders reside in the same metropolitan area, all have Internet access for email and occasional video conferencing, and have a virtual team environment with collaboration tools.
   - The systems engineering tools are strong and mature, but partially integrated with other disciplines.
   - Further discussion with your algorithms staff indicated that 8 algorithms will now be necessary - 4 are Nominal and 4 are Difficult.
   - Systems engineering labor rate = $12,000 / Person-Month

   What is the revised effort, schedule and cost? How many people are needed, on average, for the project duration?

6. Assume a systems engineering organization working on Naval C^4I projects develops 500 Equivalent Nominal System Requirements per year at $10,000/Person-Month. Funding for 30% of that current annual cost will be used for process improvements to bring their CMMI rating from Level 2 to Level 3. All other factors remain Nominal.

What is the estimated cost savings per year after the improvements? Is there a break-even point in time when the investment will pay for itself? If so, how many years after institutionalization?

The annual requirements development rate represents an aggregate of projects. Use only the COSYSMO model effort, not the schedule since it is a fixed duration of one year.

7. Use the COSYSMO reuse model in these two problems.

   a) Two configurations for a towed array system are under consideration that employ systems engineering reuse in varying degrees. The reuse categories are per the reuse model.

      <u>Option 1: Adapt Legacy System</u>
      50% Deleted from the legacy system

      The system will then be comprised of:
      50% New
      50% Modified

      <u>Option 2: Product Line Approach</u>
      25% Designed for Reuse (across product line)
      25% Modified
      50% Managed

      Use the reuse weights to compare the options and suggest the best cost alternative.

   b) A contractor is considering whether to increase the charter of a new project by using the system baseline for a product line design to be reused on future projects. This will require additional investment so that the system will be *Designed for Reuse* per the COSYSMO model instead of all *New*.

      The current estimate for systems engineering is $2.5M. What is the new estimated systems engineering cost with the product line investment?

3 Software Cost Model

The Constructive Cost Model (COCOMO) is the most widely used, thoroughly documented and calibrated software cost model [1]. “Constructive” means the model provides insight into the sources of cost variation and helps the user better understand the software job to be done. With it one can reason about the cost and schedule implications of software decisions. COCOMO is an open model tailorable for specific environments.

It uses Source Lines of Code (SLOC) or function points for sizing, five scale factors, and seventeen multiplicative cost drivers. The core effort formula in the model is:

$$Effort = A * Size^{\mathrm{B}} * \prod_{i=1}^{N} EM_i \qquad (5)$$

Where

- *Effort* is in person-months
- *A* is a constant derived from historical project data
- *Size* is in KSLOC (thousand source lines of code), or converted from other size measures
- *B* is an exponent for the diseconomy of scale dependent on additive scale drivers
- EM_i is an effort multiplier for the i^{th} cost driver. The geometric product of *N* multipliers is an overall Effort Adjustment Factor (EAF) to the nominal effort.

The schedule equation for COCOMO II is:

$$TDEV = (C * PM_{NS}{}^{(\mathrm{D}+0.2*(\mathrm{B}-.91))}) * \frac{SCED\%}{100} \qquad (6)$$

Where C=3.67, D=0.28.

Time to Develop (TDEV) is the calendar time in months between the determination of a product’s requirements baseline to the completion of an acceptance activity certifying that the product satisfies its requirements.

In equation 6, C is a schedule coefficient that can be calibrated, PM_{NS} is the estimated person-months *excluding* the SCED effort multiplier as defined in Equation 5, D is a scaling base-exponent that can also be calibrated, B is the effort scaling exponent derived as the sum of project scale drivers, and SCED% is the compression / expansion percentage in the SCED effort multiplier rating scale.

The effort and schedule are further decomposed per the phase distributions in
Table 5. The default COCOMO effort/schedule covers Elaboration and Construction at 100% for *Development*, and adds percentages for the full acquisition cost/schedule including Inception and Transition totaling > 100%.

Table 5: COCOMO Phase Distributions

Phase	Effort %	Schedule %
Inception	6	12.5
Elaboration	24	37.5
Construction	76	62.5
Transition	12	12.5

3.1 Size

The measure of software size used in COCOMO is Source Lines of Code (SLOC) or other size metrics (e.g. function points) converted to SLOC. SLOC are logical source statements consisting of data declarations and executables. Not included are commercial-off-the-shelf software (COTS), government-furnished software (GFS), other products, language support libraries and operating systems, or other commercial libraries. Code generated with source code generators is handled by counting separate operator directives as lines of source code.

The baseline size in COCOMO II is a count of new lines of code. For new and adapted code, a method is used to make them equivalent so they can be rolled up into an aggregate size estimate per the equation below and described in the next section.

$$\text{Size} = \text{New KSLOC} + \text{Equivalent KSLOC}$$

For automatically translated code, a separate translation productivity rate is used to determine effort from the amount of code to be translated. The software size types are summarized in Table 6 and adapted software is described further in the next section.

Table 6: Software Size Types

Size Type	Description
New	Original software created for the first time.
Adapted	Pre-existing software that is used as-is (Reused) or changed (Modified).
Reused	Pre-existing software that is not changed with the adaption parameter settings: • Design Modification % (DM) = 0% • Code Modification % (CM) = 0%
Modified	Pre-existing software that is modified for use by making design, code and/or test changes: • Code Modification % (CM) > 0%
Equivalent	A relative measure of the work done to produce software compared to the code-counted size of the delivered software. It adjusts the size of software relative to developing it all new. This is also sometimes called *Effective* size.
Generated	Software created with automated source code generators. The code to include for equivalent size may consist of the generator statements directly produced by the programmer, or the 3GL generated statements produced by the automated tools.
Converted	Software that is converted between languages using automated translators.
Commercial Off-The-Shelf Software (COTS)	Pre-built commercially available software components. The source code is not available to application developers. It is not included for equivalent size. Other unmodified software not included in equivalent size are Government Furnished Software (GFS), libraries, operating systems and utilities.

3.1.1 Examples: Top-Level Effort

Example 1
Using a size of 100,000 Source Lines of Code (100 KSLOC) and all nominal cost drivers such that B=1.1 (approximately) and EAF=1:

Effort = 2.94 * $KSLOC^{1.1}$ * EAF
Effort = 2.94 * $100^{1.1}$ * EAF
= 2.94 * 158.5 * 1 = 466 Person-Months (PM)

Example 2
With a size of 200,000 Source Lines of Code (200 KSLOC):

Effort = 2.94 * $KSLOC^{1.1}$ * EAF
Effort = 2.94 * $200^{1.1}$ * EAF
= 2.94 * 339.7 * 1 = 998 Person-Months (PM)

The effort for 200 KSLOC is more than double for the size of 100 KSLOC due to the diseconomy of scale represented in the size exponent.

3.2 Equivalent Size and Adapted Code

A software product's size discussed above has been for new development, but code that is taken from another source and used in the product under development also contributes to the product's equivalent size. Adapted code is pre-existing software that is used as-is (Reused) or changed (Modified). It is aggregated with new code to determine an overall equivalent size for the effort equation.

The size of reused and modified code is adjusted to be its equivalent in new code. The adjusted code is called equivalent source lines of code (ESLOC). The adjustment is based on the additional effort it takes to modify the code for inclusion in the product.

COCOMO II uses a nonlinear software reuse estimation model. It involves estimating the amount of software to be adapted, degree-of-modification factors for a linear Adaptation Adjustment Factor (AAF), with nonlinear increments for Software Understanding (SU) and Assessment and Assimilation (AA) adjusted by an Unfamiliarity (UNFM) factor.

Equation 7 is used to determine the equivalent number of new lines of code based on the adapted product size and the Adaptation Adjustment Modifier (AAM) that accounts for the effort involved in fitting adapted code into an existing product. AAM uses the factors for Software Understanding (SU), Programmer Unfamiliarity (UNFM), and Assessment and Assimilation (AA) with AAF which contains Design Modified (DM), Code Modified (CM), and Integration Required for Modified (IM).

$$AAF = (0.4 * DM) + (0.3 * CM) + (0.3 * IM)$$

$$AAM = \begin{cases} \frac{AA + AAF * (1 + (0.02 * SU * UNFM)}{100}, & \text{for } AAF \leq 50 \\ \frac{AA + AAF + (SU * UNFM)}{100}, & \text{for } AAF > 50 \end{cases} \quad (7)$$

Equivalent KSLOC = Adapted KSLOC * AAM

Where

- DM is Percent Design Modified. The percentage of the adapted software's design which is modified in order to adapt it to the new objectives and environment.
- CM is Percent Code Modified. The percentage of the adapted software's code which is modified in order to adapt it to the new objectives and environment.
- IM is Percent of Integration Required for Modified Software. The percentage of effort required to integrate the adapted software into an overall product and to test the resulting product as compared to the normal amount of integration and test effort for software of comparable size.
- AA is Assessment and Assimilation effort needed to determine whether a fully-reused software module is appropriate to the application, and to integrate its description into the overall product description. See Table 8.
- SU - Software Understanding. Effort increment as a percentage. Only used when code is modified (zero when DM=0 and CM=0). See Table 7.
- UNFM - Unfamiliarity. The programmer's relative unfamiliarity with the software which is applied multiplicatively to the software understanding effort increment (0-1). Only used when code is modified. See Table 9.

The Software Understanding increment (SU) is obtained from Table 7. SU is expressed quantitatively as a percentage. SU is determined by taking the subjective average of the three categories. If there is no DM or CM (the component is being used unmodified) then there is no need for SU. If the code is being modified then SU applies.

Table 7: Software Understanding Rating

	Very Low	**Low**	**Nominal**	**High**	**Very High**
Structure	Very low cohesion, high coupling, spaghetti code.	Moderately low cohesion, high coupling.	Reasonably well-structured; some weak areas.	High cohesion, low coupling.	Strong modularity, information hiding in data / control structures.
Application Clarity	No match between program and application world views.	Some correlation between program and application.	Moderate correlation between program and application.	Good correlation between program and application.	Clear match between program and application world-views.

Self-Descriptiveness	Obscure code; documentation missing, obscure or obsolete	Some code commentary and headers; some useful documentation.	Moderate level of code commentary, headers, documentation.	Good code commentary and headers; useful documentation; some weak areas.	Self-descriptive code; documentation up-to-date, well-organized, with design rationale.
SU Increment	50	40	30	20	10

The other nonlinear reuse increment deals with the degree of Assessment and Assimilation (AA) needed to determine whether a reused software module is appropriate to the application, and to integrate its description into the overall product description. Table 8 provides the rating scale and values for the assessment and assimilation increment. AA is a percentage.

Table 8: Assessment and Assimilation Rating

AA Increment	Level of AA Effort
0	None
2	Basic module search and documentation
4	Some module test and evaluation, documentation
6	Considerable module test and evaluation, documentation
8	Extensive module test and evaluation, documentation

The amount of effort required to modify existing software is also a function of the programmer's relative unfamiliarity with the software (UNFM). The UNFM factor is applied multiplicatively to the software understanding effort increment. If the programmer works with the software every day, the 0.0 multiplier for UNFM will add no software understanding increment. If the programmer has never seen the software before, the 1.0 multiplier will add the full software understanding effort increment. The rating of UNFM is in Table 9.

Table 9: Programmer Unfamiliarity Rating

UNFM Increment	Level of Unfamiliarity
0.0	Completely familiar
0.2	Mostly familiar
0.4	Somewhat familiar
0.6	Considerably unfamiliar
0.8	Mostly unfamiliar
1.0	Completely unfamiliar

3.3 Cost Drivers

Cost drivers are used to capture characteristics of the software development that affect the effort. A cost driver is a model factor that "drives" the cost estimated by the model. All COCOMO II cost drivers have rating levels that express the impact of the driver from Extra Low to Extra High. Scale drivers affect effort exponentially via the B factor in Equation 5 and the other cost drivers are linear factors with effort multipliers.

SCALE DRIVERS

The size exponent B in Equation 8 is an aggregation of five *scale drivers* that account for the relative economies or diseconomies of scale encountered for software projects of different sizes. When B > 1.0, the project exhibits diseconomies of scale generally due to two main factors: growth of interpersonal communications overhead and growth of large-system integration overhead. Larger projects will have more personnel, and thus more interpersonal communications paths consuming overhead. Integrating a small product as part of a larger product requires not only the effort to develop the small product, but also the additional overhead effort to design, maintain, integrate, and test its interfaces with the remainder of the product.

Each scale driver has a range of rating levels, from Very Low to Extra High. Each rating level has a weight. The specific value of the weight is called a *Scale Factor* (SF). The project's scale factors, the selected scale driver ratings, are summed and used to determine a scale exponent via Equation 8.

$$B = .91 + 0.01 * \sum_{j=1}^{5} SF_j \qquad \textbf{(8)}$$

These scale drivers are described next.

Precedentedness (PREC)

Precedentedness is the degree to which the system is new and past experience applies. If a product is similar to several previously developed products, then the precedentedness is high.

	Very Low	Low	Nominal	High	Very High	Extra High
Precedentedness	thoroughly un-precedented	largely un-precedented	somewhat un-precedented	generally familiar	largely familiar	thoroughly familiar

The rating scale is elaborated below. You can use a weighted average of the underlying ratings to obtain the overall rating.

Feature	Very Low	Low	Nominal	High	Extra High
Organizational understanding of product objectives	General (60%)	Moderate (70%)	Considerable (80%)	Large (90%)	Thorough (100%)
Experience in working with related software systems	6 months	1 year	2 years	3 years	6 years
Concurrent development of associated new hardware and operational procedures	Extensive	Considerable	Moderate	Little	Minimal

Need for innovative data processing architectures, algorithms	Considerable	Large	Some	Little	Minimal

Development Flexibility (FLEX)
Development Flexibility is the need to conform to specified requirements.

	Very Low	Low	Nominal	High	Very High	Extra High
Development Flexibility	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals

You can use a weighted average of the underlying ratings to obtain the overall rating.

Feature	Very Low	Low	Nominal	High	Extra High
Need for software conformance with pre-established requirements	Full (100%)	Large (95%)	Considerable (90%)	Moderate (85%)	Basic (80%)
Need for software conformance with external interface specifications	Full (100%)	Large (95%)	Considerable (90%)	Moderate (85%)	Basic (80%)
Combination of inflexibilities above with premium on early completion	High (critical path + 10%)	Considerable (critical path)	Medium (critical path – 5%)	Some critical path - 10%)	Low (critical path - 15%)

The next three scale factors, Architecture / Risk Resolution, Team Cohesion, and Process Maturity, identify management controllables by which projects can reduce diseconomies of scale by reducing sources of project turbulence, entropy, and rework.

Architecture / Risk Resolution (RESL)
Architecture/Risk Resolution is the degree of design thoroughness and risk elimination. The percentages below refer to a weighted average of the percent of significant module interfaces specified and the percent of significant risks eliminated.

	Very Low	Low	Nominal	High	Very High	Extra High
Architecture/Risk Resolution	little 20%	some 40%	often 60%	generally 75%	mostly 90%	full 100%

The rating is the weighted average of the following characteristics.

Characteristic	Very Low	Low	Nominal	High	Very High	Extra High
Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by CDR.	None (0%)	Little (20%)	Some (40%)	Generally (60%)	Mostly (80%)	Fully (100%)
Schedule, budget, and internal milestones through CDR compatible with Risk Management Plan.	None (0%)	Little (20%)	Some (40%)	Generally (60%)	Mostly (80%)	Fully (100%)
Percent of development schedule devoted to establishing	5	10	17	25	33	40

architecture, given general product objectives.						
Percent of required top software architects available to project.	20	40	60	80	100	120
Tool support available for resolving risk items, developing and verifying architectural specs.	None (0%)	Little (20%)	Some (40%)	Good (60%)	Strong (80%)	Full (100%)
Level of uncertainty in key architecture drivers: mission, user interface, COTS, hardware, technology, performance. Percentages reflect the level of certainty.	Extreme (<50% certain)	Significant (60% certain)	Considerable (70% certain)	Some (80% certain)	Little (90% certain)	Very Little (99% certain)
Number and criticality of risk items.	> 10 Critical	5-10 Critical	2-4 Critical	1 Critical	> 5Non-Critical	< 5 Non-Critical

Team Cohesion (TEAM)

Team Cohesion is the need to synchronize stakeholders and minimize conflict. This scale driver accounts for the sources of project turbulence and entropy because of difficulties in synchronizing the project's stakeholders: users, customers, developers, maintainers, interfacers, others. These difficulties may arise from differences in stakeholder objectives and cultures; and stakeholders' lack of opportunities in operating as a team.

	Very Low	Low	Nominal	High	Very High	Extra High
Team Cohesion	strongly adversarial	occasionally cooperative	moderately cooperative	largely cooperative	highly cooperative	seamless

The overall rating is the weighted average of the following characteristics.

Characteristic	Very Low	Low	Nominal	High	Very High	Extra High
Consistency of stakeholder objectives and cultures	Little (50%)	Some (60%)	Basic (70%)	Considerable (80%)	Strong (90%)	Full (100%)
Ability, willingness of stakeholders to accommodate other stakeholders' objectives	Little (50% of time)	Some (60% of time)	Basic (70% of time)	Considerable (80% of time)	Strong (90% of time)	Full (100% of time)
Experience of stakeholders in operating as a team	None	Little (6 months)	Little (1 year)	Basic (2 years)	Considerable (3 years)	Extensive (6 years)
Stakeholder teambuilding to achieve shared vision and commitments measured as % of objectives reconciled	None (0%)	Little (20%)	Little (40%)	Basic (60%)	Considerable (80%)	Extensive (100%)

Process Maturity (PMAT)

Process Maturity is the Software Engineering Institute's Capability Maturity Model (CMM) process maturity rating. The time period for rating Process Maturity is the time the project starts.

	Very Low	Low	Nominal	High	Very High	Extra High
Process Maturity	CMM Level 1 (lower half)	CMM Level 1 (upper half)	CMM Level 2	CMM Level 3	CMM Level 4	CMM Level 5

MULTIPLICATIVE COST DRIVERS

The multiplicative cost drivers are defined below by rating levels and a corresponding set of *Effort Multipliers* (EM) associated with each rating level. The EM value assigned to a cost driver's nominal rating is 1.0. If a cost driver's rating level causes more software development effort than nominal, then its corresponding EM is above 1.0. Conversely, if the rating level reduces the effort then the corresponding EM is less than 1.0.

PRODUCT ATTRIBUTES

Required Software Reliability (RELY)

This is a measure of the extent to which the software must perform its intended function over a period of time. If the effect of a software failure is only a slight inconvenience then Required Software Reliability is very low. If a failure would risk human life then it is very high.

	Very Low	Low	Nominal	High	Very High	Extra High
Required SW Reliability	effect: slight inconvenience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	

Data Base Size (DATA)

This factor captures the effect large data requirements have on product development. The rating is determined by calculating D/P, the ratio of bytes in the database to SLOC in the program. The size of the database is important to consider because of the effort required to generate the test data that will be used to exercise the program. This factor accounts for the effort to assemble the data required to complete test of the program.

	Very Low	Low	Nominal	High	Very High	Extra High
Data Base Size		DB bytes/Prog. SLOCS $<$ 10	10 $\leq$D/P $<$100	100 $\leq$D/P $<$1000	D/P $\geq$ 1000	

Product Complexity (CPLX)

Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. The overall rating is the weighted average of the areas that best characterize the product or component being rated.

Product Complexity Rating

	Very Low	Low	Nominal	High	Very High	Extra High
Control Operations	Straight-line code with a few non-nested	Straightforward nesting of structured programming	Mostly simple nesting. Some	Highly nested structured programming operators with	Reentrant and recursive coding. Fixed-priority interrupt	Multiple resource scheduling with

	structured programming operators: DOs, CASEs, IF-THEN-ELSEs. Simple module composition via procedure calls or simple scripts.	operators. Mostly simple predicates	intermodule control. Decision tables. Simple callbacks or message passing, including middleware-supported distributed processing	many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real-time control.	handling. Task synchronization, complex callbacks, heterogeneous distributed processing. Single-processor hard real-time control.	dynamically changing priorities. Microcode-level control. Distributed hard real-time control.
Computational Operations	Evaluation of simple expressions: e.g., A=B+C*(D-E)	Evaluation of moderate-level expressions: e.g., D=SQRT(B**2-4.*A*C)	Use of standard math and statistical routines. Basic matrix/vector operations.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, round-off concerns.	Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations. Simple parallelization.	Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data. Complex parallelization.
Device-dependent Operations	Simple read, write statements with simple formats.	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level.	I/O processing includes device selection, status checking and error processing.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc.). Optimized I/O overlap.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance-intensive embedded systems.	Device timing-dependent coding, micro-programmed operations. Performance-critical embedded systems.
Data Management Operations	Simple arrays in main memory. Simple COTS-DB queries, updates.	Single file subsetting with no data structure changes, no edits, no intermediate files. Moderately complex COTS-DB queries, updates.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple triggers activated by data stream contents. Complex data restructuring.	Distributed database coordination. Complex triggers. Search optimization.	Highly coupled, dynamic relational and object structure\s. Natural language data management.
User Interface Management Operations	Simple input forms, report generators.	Use of simple graphic user interface (GUI) builders.	Simple use of widget set.	Widget set development and extension. Simple voice I/O,	Moderately complex 2D/3D, dynamic graphics, multimedia.	Complex multimedia, virtual reality, natural language

				multimedia.		interface.

Developed for Reuse (RUSE)

This cost driver accounts for the additional effort needed to construct components intended for reuse on current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications. "Across project" could apply to reuse across the modules in a single financial applications project. "Across program" could apply to reuse across multiple financial applications projects for a single organization. "Across product line" could apply if the reuse is extended across multiple organizations. "Across multiple product lines" could apply to reuse across financial, sales, and marketing product lines.

	Very Low	Low	Nominal	High	Very High	Extra High
Required Reusability		none	across project	across program	across product line	across multiple product lines

Documentation Match to Life-Cycle Needs (DOCU)

The rating scale for this cost driver is evaluated in terms of the suitability of the project's documentation to its life-cycle needs.

	Very Low	Low	Nominal	High	Very High	Extra High
Documentation match to lifecycle needs	many lifecycle needs uncovered	some lifecycle needs uncovered	right-sized to lifecycle needs	excessive for lifecycle needs	very excessive for lifecycle needs	

PLATFORM ATTRIBUTES

Execution Time Constraint (TIME)

This is a measure of the execution time constraint imposed upon a software system. The rating is expressed in terms of the percentage of constrained execution time expected to be used by the system or subsystem consuming the execution time resource (the project constraint may be less than the total physically available execution time).

	Very Low	Low	Nominal	High	Very High	Extra High
Execution Time Constraint			$\leq$ 50% use of available execution time	$\leq$ 70% use of available execution time	$\leq$ 85% use of available execution time	$\leq$ 95% use of available execution time

Main Storage Constraint (STOR)

This rating represents the degree of main storage constraint imposed on a software system or subsystem. It is expressed in terms of percentage used of the constrained direct random access storage (the project constraint may be less than the total physically available storage).

	Very Low	Low	Nominal	High	Very High	Extra High

	Very Low	Low	Nominal	High	Very High	Extra High
Main Storage Constraint			≤ 50% use of available storage	≤ 70% use of available storage	≤ 85% use of available storage	≤ 95% use of available storage

Platform Volatility (PVOL)
Platform refers to the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks. If the software to be developed is an operating system then the platform is the computer hardware. If a database management system is to be developed then the platform is the hardware and the operating system. If a network text browser is to be developed then the platform is the network, computer hardware, the operating system, and the distributed information repositories. The platform includes any compilers or assemblers supporting the development of the software system.

The ratings are defined in terms of the relative frequency of major and minor changes. A major change significantly affects roughly 10% of routines under development, while a minor change significantly affects about 1% of routines under development.

	Very Low	Low	Nominal	High	Very High	Extra High
Platform Volatility		major change every 12 mths. minor: 1 mth.	major: 6 mths. minor: 2 weeks	major: 2 mths. minor: 1 week	major: 2 weeks minor: 2 days	

PERSONNEL ATTRIBUTES

Analyst Capability (ACAP)
Analysts are personnel who work on requirements, high-level design and detailed design. This cost driver is based on the capability of the analysts as a team rather than as individuals. The major attributes that should be considered are analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate. The rating should not consider the level of experience of the analysts.

	Very Low	Low	Nominal	High	Very High	Extra High
Analyst Capability	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	

Programmer Capability (PCAP)
This cost driver is based on the capability of the programmers as a team rather than as individuals. Major factors which should be considered are ability, efficiency and thoroughness, and the ability to communicate and cooperate. The experience of the programmers should not be considered.

	Very Low	Low	Nominal	High	Very High	Extra High
Programmer Capability	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	

Personnel Continuity (PCON)

The rating scale for Personnel Continuity is in terms of the project's annual personnel turnover.

	Very Low	Low	Nominal	High	Very High	Extra High
Personnel Continuity	48%/year	24%/year	12%/year	6%/year	3%/year	

Applications Experience (APEX)
The rating for this cost driver depends on the level of applications experience of the project team developing the software system or subsystem. The ratings are defined in terms of the project team's equivalent level of experience with this type of application.

	Very Low	Low	Nominal	High	Very High	Extra High
Application Experience	≤ 2 months experience	6 months	1 year	3 years	6 years	

Platform Experience (PLEX)
This driver reflects the project team's equivalent duration of experience with the platform to be used. See the Platform Volatility cost driver for the definition of platform.

	Very Low	Low	Nominal	High	Very High	Extra High
Platform Experience	≤ 2 months experience	6 months	1 year	3 years	6 years	

Language and Tool Experience (LTEX)
This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. It includes the use of tools that perform requirements and design representation and analysis, configuration management, document extraction, library management, program style and formatting, consistency checking, planning and control, etc.

	Very Low	Low	Nominal	High	Very High	Extra High
Language & Toolset Experience	≤ 2 months experience	6 months	1 year	3 years	6 years	

PROJECT ATTRIBUTES

Use of Software Tools (TOOL)
The tool rating ranges from simple edit and coding tools to integrated life-cycle management tools. See the Language and Tool Experience driver for a list of software tools.

	Very Low	Low	Nominal	High	Very High	Extra High

	Very Low	Low	Nominal	High	Very High	Extra High
Use of SW Tools	edit, code, debug	simple front-end, backend CASE, little integration	basic lifecycle tools, moderately integrated	strong, mature lifecycle tools, moderately integrated	strong, mature, pro-active lifecycle tools, well integrated with processes, methods, reuse	

Multisite Development (SITE)
Determining this cost driver rating involves the assessment and averaging of two factors: site collocation and communication. For example, if a team is fully collocated, it doesn't need interactive multimedia to achieve an Extra High rating. Narrowband e-mail would usually be sufficient.

	Very Low	Low	Nominal	High	Very High	Extra High
Multi-site Development: Collocation	international	multi-city or multi-company	multi-city and multi-company	same city or metro area	same building or complex	fully collocated
Multi-site Development: Communications	some phone, mail	individual phone, fax	narrowband email	widebank electronic communicatio n	wideband elect. comm., occasional video conf.	interactive multimedia

Required Development Schedule (SCED)
This rating measures the project schedule constraint imposed on the team developing the software. It is defined in terms of the percentage of schedule stretch or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. It is a ratio of the available schedule to the nominal schedule. The nominal schedule can be calculated by inputting all the factors to COCOMO except for SCED (set to nominal).

	Very Low	Low	Nominal	High	Very High	Extra High
Required Development Schedule	75% of nominal	85%	100%	130%	160%	

Summaries of the COCOMO II cost factors are below. The multiplicative cost factor rating scales, scale factors and effort multipliers are in Table 10, Table 11 and Table 12 respectively.

Table 10: COCOMO II Cost Factor Ratings

Cost Factor	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
SCALE DRIVERS						
Precedentedness	thoroughly un-precedented	largely un-precedented	somewhat un-precedented	generally familiar	largely familiar	thoroughly familiar
Development Flexibility	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals

Cost Factor	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
Architecture/Risk Resolution	little 20%	some 40%	often 60%	generally 75%	mostly 90%	full 100%
Team Cohesion	strongly adversarial	occasionally cooperative	moderately cooperative	largely cooperative	highly cooperative	seamless
Process Maturity	CMM Level 1 (lower half)	CMM Level 1 (upper half)	CMM Level 2	CMM Level 3	CMM Level 4	CMM Level 5
PRODUCT ATTRIBUTES						
Required SW Reliability	effect: slight inconvenience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Data Base Size		DB bytes/Prog. SLOCS < 10	$10 \leq D/P < 100$	$100 \leq D/P < 1000$	$D/P \geq 1000$	
Product Complexity	See Product Complexity Rating Table					
Required Reusability		none	across project	across program	across product line	across multiple product lines
Documentation match to lifecycle needs	many lifecycle needs uncovered	some lifecycle needs uncovered	right-sized to lifecycle needs	excessive for lifecycle needs	very excessive for lifecycle needs	
PLATFORM ATTRIBUTES						
Execution Time Constraint			≤ 50% use of available execution time	≤ 70%	≤ 85%	≤ 95%
Main Storage Constraint			≤ 50% use of available storage	≤ 70%	≤ 85%	≤ 95%
Platform Volatility		major change every 12 mths. minor: 1 mth.	major: 6 mths. minor: 2 weeks	major: 2 mths. minor: 1 week	major: 2 weeks minor: 2 days	
PERSONNEL ATTRIBUTES						
Analyst Capability	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Programmer Capability	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Personnel Continuity	48%/year	24%/year	12%/year	6%/year	3%/year	
Application Experience	≤ 2 months experience	6 months	1 year	3 years	6 years	
Platform Experience	≤ 2 months experience	6 months	1 year	3 years	6 years	
Language & Toolset Experience	≤ 2 months experience	6 months	1 year	3 years	6 years	

COCOMO II Cost Factor Ratings (Cont.)

	Very Low	Low	Nominal	High	Very High	Extra High
PROJECT ATTRIBUTES						
Use of Software Tools	edit, code, debug	simple front-end, backend CASE, little integration	basic lifecycle tools, moderately integrated	strong, mature lifecycle tools, moderately integrated	strong, mature, pro-active lifecycle tools, well integrated with processes, methods, reuse	
Multi-site Development: Collocation	international	multi-city or multi-company	multi-city and multi-company	same city or metro area	same building or complex	fully collocated
Multi-site Development: Communications	some phone, mail	individual phone, fax	narrowband email	widebank electronic communicatio n	wideband elect. comm., occasional video conf.	interactive multimedia
Required Development Schedule	75% of nominal	85%	100%	130%	160%	

Table 11: COCOMO II Scale Scale Factors

Scale Driver	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
SCALE DRIVERS						
Precedentedness	6.20	4.96	3.72	2.48	1.24	0.00
Development Flexibility	5.07	4.05	3.04	2.03	1.01	0.00
Architecture/Risk Resolution	7.07	5.65	4.24	2.83	1.41	0.00
Team Cohesion	5.48	4.38	3.29	2.19	1.10	0.00
Process Maturity	7.80	6.24	4.68	3.12	1.56	0.00

Table 12: COCOMO II Effort Multipliers

Cost Driver	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
Required SW Reliability	0.82	0.92	1.00	1.10	1.26	
Data Base Size		0.90	1.00	1.14	1.28	
Product Complexity	0.73	0.87	1.00	1.17	1.34	1.74
Required Reusability		0.95	1.00	1.07	1.15	1.24
Documentation match to lifecycle needs	0.81	0.91	1.00	1.11	1.23	
Execution Time Constraint			1.00	1.11	1.29	1.63
Main Storage Constraint			1.00	1.05	1.17	1.46
Platform Volatility		0.87	1.00	1.15	1.30	
Analyst Capability	1.42	1.19	1.00	0.85	0.71	
Programmer Capability	1.34	1.15	1.00	0.88	0.76	
Personnel Continuity	1.29	1.12	1.00	0.90	0.81	
Application Experience	1.22	1.10	1.00	0.88	0.81	
Platform Experience	1.19	1.09	1.00	0.91	0.85	

Language & Toolset Experience	1.20	1.09	1.00	0.91	0.84	
Use of Software Tools	1.17	1.09	1.00	0.90	0.78	
Multi-site Development	1.22	1.09	1.00	0.93	0.86	0.80
Required Development Schedule	1.43	1.14	1.00	1.00	1.00	

3.3.1 Example: Effort and Schedule with Modified Software

Calculate the estimated effort and schedule for the following software project:

The newly developed software size is 300,000 SLOC and modified software size is 75,000 SLOC. Other information on the project includes:
- the schedule compression ratio is 75%
- the effect of a software failure is a large equipment loss
- average language and tool experience is 6 years
- there is expected to be a major change in the operating system once per year
- use of a comprehensive toolset fully integrated across all activities.

For the modified software:
- 40 out of 100 design modules were modified
- 30 KSLOC out of the total 75 KSLOC were modified
- the integration of the modified code is estimated to be 2/3 of the effort compared to if it was all new
- to understand the software, it is described as:
    - structure - high cohesion, low coupling
    - application clarity - moderate correlation between program and application
    - self-descriptiveness - moderate level of comments, headers, documentation.
- only basic module search and documentation was required for its assessment and assimilation
- the team did not develop the software being modified and are unfamiliar with it.

Cost drivers:
Required Development Schedule = Very Low (EM=1.43)
Required Software Reliability = High (EM=1.1)
Language & Toolset Experience = Very High (EM=.84)
Platform Volatility = Low (EM=.87)
Use of Software Tools = Very High (EM=.75)

EAF = 1.43*1.1*.84*.87*.75=.90

For the modified code:
DM = 40/100 = 40%, CM = 30/75 = 40%, IM = 67%, SU = 26.6% (averaged), AA=2 per table, UNFM= 1.0 per table (overall adjustment factor = 48.1 %)

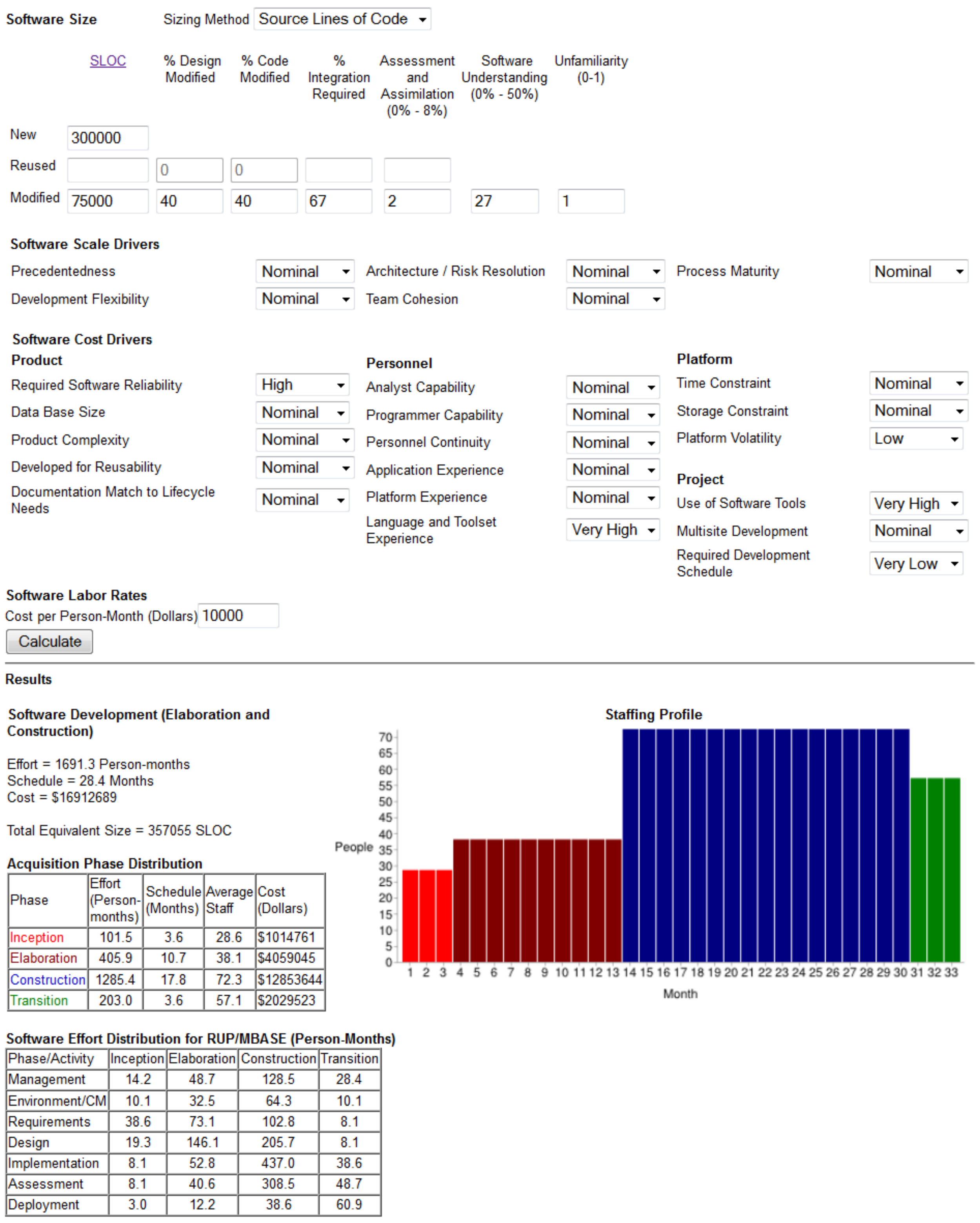
AAF = .481, AAM = .761 per the adapted software model

Equivalent size of 75KSLOC modified code = .761*75K = 57 KSLOC

Total equivalent size = 300 + 57 = 357 KSLOC

Effort = 1691 Person-Months
Schedule = 28 Months

An estimate using the COCOMO tool for this case using these inputs is shown in Figure 4. Note in the results the total equivalent size of 357 KSLOC is displayed. It is the aggregate size used in the effort equation accounting for both the new and modified software.



Software Size Sizing Method: Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	300000						
Reused		0	0				
Modified	75000	40	40	67	2	27	1

Software Scale Drivers

Precedentedness	Nominal	Architecture / Risk Resolution	Nominal	Process Maturity	Nominal
Development Flexibility	Nominal	Team Cohesion	Nominal		

Software Cost Drivers

Product		**Personnel**		**Platform**	
Required Software Reliability	High	Analyst Capability	Nominal	Time Constraint	Nominal
Data Base Size	Nominal	Programmer Capability	Nominal	Storage Constraint	Nominal
Product Complexity	Nominal	Personnel Continuity	Nominal	Platform Volatility	Low
Developed for Reusability	Nominal	Application Experience	Nominal	**Project**	
Documentation Match to Lifecycle Needs	Nominal	Platform Experience	Nominal	Use of Software Tools	Very High
		Language and Toolset Experience	Very High	Multisite Development	Nominal
				Required Development Schedule	Very Low

Software Labor Rates

Cost per Person-Month (Dollars) 10000

Results

Software Development (Elaboration and Construction)

Effort = 1691.3 Person-months
Schedule = 28.4 Months
Cost = $16912689

Total Equivalent Size = 357055 SLOC

Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	101.5	3.6	28.6	$1014761
Elaboration	405.9	10.7	38.1	$4059045
Construction	1285.4	17.8	72.3	$12853644
Transition	203.0	3.6	57.1	$2029523

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	14.2	48.7	128.5	28.4
Environment/CM	10.1	32.5	64.3	10.1
Requirements	38.6	73.1	102.8	8.1
Design	19.3	146.1	205.7	8.1
Implementation	8.1	52.8	437.0	38.6
Assessment	8.1	40.6	308.5	48.7
Deployment	3.0	12.2	38.6	60.9

Figure 4: Modified Software Example

3.3.2 Examples: Sensitivity Analysis

Below are additional illustrations of using the software cost model for sensitivity analyses.

Example 1
What is the difference in effort for a project if the *Use of Software Tools* rating is High vs. Low?

The Effort Multiplier at High = 0.9
The Effort Multiplier at Low = 1.09

Compute an effort ratio (with like terms cancelling per section 2.2.2):

Ratio = .9 / 1.09 = .83

The effort at *Use of Software Tools* = High is 83% of the effort at Low, or 17% less effort.

Alternatively, calculating the percent difference gives the same answer:

$(2.94 * KSLOC^{1.1} * .9 - 2.94 * KSLOC^{1.1} * 1.09) / (2.94 * KSLOC^{1.1} * 1.09)$
= (.9-1.09) / 1.09 = -.17 (-17% effort going from Low to High)

Example 2
The estimated effort for a project is 30 person-months. The project manager is considering the purchase of an advanced tool environment with better integration across the lifecycle. This would increase the Use of Software Tools rating from nominal to very high. What is the new estimated effort if the tool environment is purchased?

Use of Software Tools effort multiplier ratio = .78 / 1.0 = .78

New effort = .78*30 = 23.4 person-months

Example 3
The scale drivers are all nominal for a project so the size exponent in the COCOMO effort equation is 1.1. What would be the relative change in effort if the size of a software application was 300,000 SLOC vs. 250,000 SLOC?

$300^{1.1} / 250^{1.1}$ = 1.23, or 23% additional effort
In this case, all effort terms cancel out again except size.

3.4 Software Maintenance

Software maintenance is the process of modifying existing software while not changing its primary functions. The assumption made by COCOMO II is that software maintenance cost generally has the same cost driver attributes as software development costs. Maintenance includes redesign and recoding of small portions of the original product, redesign and development of interfaces, and minor modification of the product structure. Maintenance can be classified as either updates or repairs. Product repairs can be further segregated into corrective, adaptive, or perfective maintenance.

The maintenance size is normally obtained via Equation 9, when the base code size is known and the percentage of change to the base code is known.

$$Size_M = \left(\text{Base Code Size} * MCF\right) * MAF \tag{9}$$

The percentage of change to the base code is called the Maintenance Change Factor (MCF) which represents the ratio in Equation 10:

$$MCF = \frac{\text{Size Added + Size Modified}}{\text{Base Code Size}} \tag{10}$$

The Maintenance Adjustment Factor (MAF) in Equation 11 is used to adjust the effective maintenance size to account for software understanding and unfamiliarity effects, as with reuse. COCOMO II uses the Software Understanding (SU) and Programmer Unfamiliarity (UNFM) factors from the reuse model to handle the effects of well or poorly structured/understandable software on maintenance effort.

$$MAF = 1 + \left(\frac{SU * UNFM}{100}\right) \tag{11}$$

There are also special cost driver considerations for using COCOMO II in software maintenance:

- *Required Development Schedule* is not used in the estimation of effort for maintenance. This is because the maintenance cycle is usually of a fixed duration.
- *Required Reusability* is not used in the estimation of effort for maintenance. This is because the extra effort required to maintain a component's reusability is roughly balanced by the reduced maintenance effort due to the component's careful design, documentation, and testing.
- *Required Software Reliability* has a different set of effort multipliers for maintenance. For maintenance the RELY cost driver depends on the required reliability under which the product was developed. If the product was developed with low reliability it will require more effort to fix latent faults. If the product was developed with very high reliability, the effort required to maintain that level of reliability will be above nominal. Table 13 shows the maintenance effort multipliers.

Table 13: *Required Software Reliability* Maintenance Cost Driver

	Very Low	**Low**	**Nominal**	**High**	**Very High**
Required Software Reliability	effect: slight inconvenience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life
Effort Multipliers	1.35	1.15	1.00	0.98	1.10

3.5 Examples: Using Software Cost Model for Decisions

We will use the COCOMO framework and cost drivers to illustrate how one can use them for software decision analyses, even without exercising the full model. The example organization is the Defense Electronics Supply Center (DESC) which retains a software engineering staff of 200 people to perform mission-critical software functions.

Making Investment Decisions and Mission/Business-Case Analyses

DESC is considering the development of an Inventory Control System (ICS) to support just-in-time inventory operations. DESC expects that the reductions in inventory carrying costs (25% of the value of the inventory per year) will justify the expenditure to develop the ICS, but would like to do a return-on-investment (ROI) analysis to make sure.

Conservatively, DESC estimates that the new system will reduce its current average manufacturing inventory value of $80 M by 20%. The corresponding savings in annual carrying costs is $80M * 20% * 25% = $4M/Year.

A similar conservative COCOMO II software cost analysis assumes that the ICS system will require 100 KSLOC of software, that all the cost driver and scale driver ratings will be nominal, and that the burdened cost of a software engineer is $8K per person-month. The nominal cost drivers are all 1.0. The nominal scale factors in Table 2.10 lead to a scaling exponent B = 0.91+.01(3.72+3.04+4.24+3.29+4.68) = 1.10. The resulting estimated effort is E = $2.94 * 100^{1.10}$ = 466 Person-Months (PM). The corresponding estimated development cost is 466 * $8K = $3.728M.

However, the development cost is not the full ICS project cost. Table 14 indicates that an added 6% is needed for the project's Inception phase, and an added 12% is needed for the Transition phase. This extra 18% means that the estimated project cost is 1.18 * $3.728M = $4.4M.

For software maintenance, the ratings are again assumed to be nominal, and the annual amount of modified software is estimated to be 20% or 20 KSLOC. The estimated annual cost of software maintenance is $2.94 * 20^{1.10}$ * $8K = $635K; the net savings per year is thus $4M-.635M = $3.365M. The annual return on investment is thus net savings/project costs or $3.365M/$4.4M = 76%. For 5 years of operation, the net savings are 5 * $3.365M = $16.8M, and the ROI is $16.8M/$4.4M = 380%.

Setting Project Budgets and Schedules

The development schedule is estimated to be TDEV = $3.67 * PM^{(0.28+0.2(E-0.91))} = 3.67 * 466^{0.318}$ = 26 months. With the estimated effort in person-months and the schedule in months for each phase, we can divide them to calculate the average personnel level for each phase. These calculations are done in Table 14.

Thus, if the ICS project were all-nominal with respect to its cost drivers and scale drivers, it should expect to employ an average of 11.5 people during its Elaboration phase. The manager should also plan for the Elaboration phase to take almost 10 months and cost $896K.

Table 14: ICS Project Phase Distributions

Phase	Effort		Schedule		# Personnel	Cost @ $8K/PM
	%	PM	%	Months		
Inception	6	28	12.5	3.25	8.6	$224K
Elaboration	24	112	37.5	9.75	11.5	$896K
Construction	76	354	62.5	16.25	21.8	$2832K
Transition	12	56	12.5	3.25	17.2	$448K
Development Total	100	466	100.0	26.0	17.9	$3728K
Project Total	118	550	125	32.5	16.9	$4400K

Performing Tradeoff Analyses

The COCOMO II cost drivers and scale drivers can be used directly to analyze their effects on a project's cost and schedule. For example, we can use the *Required Reliability* effort multipliers for software development and maintenance to analyze the effects of different *Required Reliability* levels on software life cycle costs of the ICS project. A good planning factor is to assume that life cycle maintenance costs will be about twice as high as the original project costs for a Nominal level.

The resulting analysis is shown in Table 15. For development, it appears that lower Reliability levels can save money. But these lower levels cause more maintenance effort to be spent on debugging and error correction, making their life cycle costs higher than the Nominal level of *Required Reliability*.

Table 15: Tradeoffs of *Required Reliability* Level on Life Cycle Costs

***Required Reliability* Rating**	**Very Low**	**Low**	**Nominal**	**High**	**Very High**
Development Effort Multiplier	0.82	0.92	1.0	1.10	1.26
Development Cost	$3608K	$4048K	$4400K	$4840K	$5544K
Maintenance Effort Multiplier	1.35	1.15	1.0	0.98	1.10
Maintenance Cost (*2 for Nominal)	$11,880K	$10,120K	$8,800K	$8,624K	$9,680K
Life Cycle Cost	$15,488K	$14,168K	$13,200K	$13,464K	$15,224K

When the costs of operational failures are also included, the higher levels will turn out to be lower-cost for high cost-of-failure systems. This is shown in Table 16 where it is assumed the cost of system downtime is $3B. The value-based analysis indicates the lowest ownership cost is for *Required Reliability* = High.

Table 16: Value-Based Tradeoffs of *Required Reliability* Level on Ownership Costs

***Required Reliability* Rating**	**Very Low**	**Low**	**Nominal**	**High**	**Very High**
Mean Time to Failure (Hours)	1	10	300	10,000	300,000

***Required Reliability* Rating**	**Very Low**	**Low**	**Nominal**	**High**	**Very High**
Mean Time to Repair (Hours)	1	1	1	1	1
Avail=MTBF/(MTBF+MTTR)	.50	.90	.997	0.9999	0.999997
Downtime	.50	.10	.003	0.0001	0.000003
Cost=$3B*Downtime	$1500M	$300M	$10M	$0.3M	$0.01M
SW Life-cycle Cost	$15.5M	$14.2M	$13.2M	$13.5M	$15.2M

Cost Risk Management

If there are uncertainties in key cost drivers, it is a good idea to establish a risk management reserve to cover the resulting possibilities of cost growth. For example, suppose that there is a chance that new developments will cause a 15% requirements evolution and volatility in the amount of software to be developed. This changes the effective size of the product to 1.15 * 100 KSLOC = 115 KSLOC, and the estimated cost to 2.94 * $115^{1.1}$ * 1.18 * $8K = $5130K, an increase of $730K over the original $4400K estimate. In this case the project's financial sponsor could establish a risk management reserve of $730K to cover any added costs due to requirements evolution or volatility.

The project may other common risks to assess. For example, the project may not get all the experienced people it has asked for. Rather than having overall Nominal ratings of *Applications Experience* and *Platform Experience*, the project personnel may average halfway between Nominal and Low for these cost drivers. The effort multipliers for *Applications Experience* (1.05) and *Platform Experience* (1.045) would be halfway between the Nominal and Low values. The resulting estimated cost would be $4400K * 1.05 * 1.045 = $4828K and the resulting potential overrun risk would be $428K.

Development vs. Reuse Decisions

Many of the COCOMO II cost drivers provide opportunities to reduce costs; one of the most attractive is to reduce the effective product-development size via software reuse or commercial-off-the-shelf (COTS) products.

The COCOMO II reuse model includes factors which help you to reason about reuse decisions. In the context of the ICS system, for example, suppose we find a 40 KSLOC component of a related project which we might reuse. However, it will need a good deal of redesign and recoding. It is also not well-structured or well-documented, and there are no people available for its modification who are familiar with its internals.

We would evaluate the COCOMO II reuse factors as follows:

DM = 40	(% design modified)
CM = 50	(% code modified)
IM = 100	(% integration into new system required)
SU = 50	(poorly structured and documented)
UNFM = 1.0	(complete unfamiliarity with software)
AA = 5	(average assessment and adaptation effort)

The resulting equivalent size of the reuse component is:

$$\begin{aligned} AAF &= (0.4*DM)+(0.3*CM)+(0.3*IM) \\ &= (0.4*40)+(0.3*30)+(0.3*100) \\ &= 61\% \end{aligned}$$

$$\begin{aligned} AAM &= \frac{AA+AAF+(SU*UNFM)), \text{for } AAF>50}{100} \\ &= \frac{(5+61+(50*1.0))}{100} \\ &= 1.16 \end{aligned}$$

$$\begin{aligned} \text{ESLOC} &= \text{Adapted SLOC } *AAM \\ &= 40\,\text{KSLOC}*1.16 \\ &= 46.4\,\text{KSLOC} \end{aligned}$$

In this situation we can see that the reuse component would be a poor decision: its equivalent size and thus its contribution to cost and schedule is larger than the 40 KSLOC option of developing a new component. A similar life cycle decision situation, legacy software phase-out, is discussed next.

<u>**Legacy Software Phase-out Decisions**</u>

A separate program related to the ICS program is a military property value accounting system. It is an old 50 KSLOC COBOL program which is becoming increasingly difficult to maintain. Its key factors are:

Annual MCF = 0.20	(20% of the code –10KSLOC—is changed each year)
SU = 50	(very poorly structured and documented)
UNFM = 0.7	(few people familiar with the code)

Thus, its equivalent size to be changed each year is:

$$10\,\text{KSLOC}*\left(1+\left(\frac{50*0.7}{100}\right)\right)=13.5\,\text{KSLOC/Year}$$

If the program were redeveloped, it could use the same database management system and graphic user interface software as the ICS system is using, leaving only 20 KSLOC of new applications software to be built. Conservatively, we assume that the new software will reduce the SU penalty from 50 to 25, and that the UNFM factor will be reduced from 0.7 to 0.4. Then the equivalent size to be changed each year is:

$$20\,\text{KSLOC}*0.2*\left(1+\left(\frac{25*0.4}{100}\right)\right)=4*1.1=4.4\,\text{KSLOC/Year}$$

If we add up the equivalent sizes of developing and maintaining the new software for 3 years we get:

$$20 + (3*4.4) = 32.2 \text{ KSLOC}$$

Maintaining the old legacy software for 3 years generates a larger equivalent size of

$$3*13.5= 40.5 \text{ KSLOC}$$

Given that we will need to do property value accounting for at least 3 years, and that the size and cost reductions for the new software continue to improve after 3 years, it is a good decision to build the new system and phase out the old one.

Process Improvement Decisions

COCOMO has several cost drivers and scale drivers which can be used to evaluate candidate strategies for process and productivity improvement. The DESC software organization has achieved Level 2 on the CMMI rating scale, and wishes to perform a return-on-investment analysis for the investment required to achieve Level 3. DESC's analysis of required investments identifies process definition and training as the major investment items. This cost over the normal 2 years required to go from Level 2 to Level 3 is estimated as 2 years * 4 persons * $96K/person-year = $768K.

DESC assumes that all 200 software people will require the 3 weeks of training. This is 3/52 of their annual personnel cost of 200 persons * $8K/PM * 12 months = $19,200K, or $1108K. DESC budgets another $124K for contingencies to make a total investment of $768K + $1108K + $124K = $2000K, or $2M.

In the COCOMO II scale drivers the effect of going from Level 2 (Nominal) to Level 3 (High) involves a reduction in the scale factor from 4.68 to 3.12, or a reduction of .0156 in the exponent B used to relate size to effort. For the typical 100 KSLOC, all-nominal ICS project, the baseline exponent B = 1.10 yields an estimated effort of $2.94 * 100^{1.10} = 466$ PM. The reduced exponent for Level 3 yields an estimated effort of $2.94 * 100^{1.0844} = 434$ PM, roughly a 7% improvement in productivity.

The annual cost of DESC's 200-person software organization is $19.2M. A 7% improvement in productivity corresponds with an annual savings of .07 * $19.2M = $1.344M. Over 5 years the ROI would be (5 * $1.344M - $2M) / $2M = 2.36. This conservative top-level analysis provides strong support that investments in software process maturity are worthwhile.

Decision Analysis Summary

This section showed how COCOMO or other parametric cost models can support decision analysis situations. All of the analyses could be done by hand and calculator via use of the open internal equations, tables, and definitions. One of the major advantages of COCOMO is its ability to support such analyses, based on a full understanding of the underlying cost drivers and their effects, and to support reasoned discussion and negotiation of software cost and schedule tradeoffs, based on the project stakeholders' shared understanding of the underlying model.

3.6 Exercises

1. This problem with COCOMO is to be done manually without an automated estimation tool. Calculate effort and schedule using the following simplified equations that cover Elaboration and Construction:

   Effort (Person-Months) = 2.94 * $\text{KSLOC}^{1.1}$ * EAF
   Schedule (Months) = 3.67 * $\text{Effort}^{0.32}$

   a) The project size is 250,000 SLOC (250 KSLOC) and all cost drivers are rated Nominal so the Effort Adjustment Factor (EAF) = 1.0. What are the total effort and schedule?

   b) For the same project size, what are the effort and schedule with an EAF of 1.3?

   c) Use the effort and schedule phase distributions below to decompose the above total effort and schedule for 250 KSLOC and EAF =1.3 into the two phases and determine their staffing levels.

COCOMO Phase Distributions

Phase	Effort %	Schedule %
Elaboration	24	37.5
Construction	76	62.5

   What are the estimated effort, schedule and average staff per phase?

2. Estimate software effort and productivity at the following sizes of New Source Lines of Code (SLOC): 2000 SLOC, 10,000 SLOC, 50,000 SLOC, 100,000 SLOC, 500,000 SLOC, 1,000,000 and 10,000,000 SLOC. All other inputs are set to their defaults.

   For productivity use SLOC/Person-Month and produce a graph of productivity vs. size. What phenomenon related to scale is represented in the results? How is software production interpreted according to the concept of "returns to scale"?

   You can use an estimation tool or the simplified effort equation:

$$\text{Effort (Person-Months)} = 2.94 * \text{KSLOC}^{1.1} * \text{EAF}$$

3. Conduct the effort vs. size analysis per above and compare for two cases: 1) scale factors are all rated Nominal and 2) scale factors are all rated Very Low. What are the percent differences between these cases at the varying size levels? Describe this phenomenon covered in the model.

4. Which three multiplicative cost drivers account for the largest variation in productivity? Use the effort multiplier ratios to determine their productivity ranges and show their values.

5. The nominal schedule for a software project is 4 years. What will be the estimated increase in effort if the customer mandates a schedule of 3 years?

6. For the problem above, assume the nominal schedule required 20 people for 4 years. What will be the new estimated staffing level with the constrained schedule?

7. The estimated effort for a project is 350 person-months. The developer is trying to negotiate a relaxation on timing requirements that will change the *Execution Time Constraint* rating from Extra High to Very High. What is the reduced effort if the change is accepted?

8. You are asked to do a sensitivity analysis for staffing a software project with different teams. What is the change in effort if the average applications experience was 6 months vs. 3 years? What would be the total change in effort if they also had language and toolset experience of 6 months vs. 3 years in addition to the applications experience difference?

9. The scale drivers are rated very low for a project so the size exponent in the COCOMO effort equation is 1.226. What would be the relative change in effort if the size of a 500 KSLOC software application grew to 800 KSLOC?

10. A project is developing mobile communications software for U.S. Navy SEALs that is estimated to be 350,000 SLOC. Cost drivers are all Nominal except for the following aspects:

    - A software failure would put the SEALs' lives in jeopardy
    - *Product Complexity* is rated very high according to the programmers
    - There is a large data base size of 200 MB.

    The cost per person-month is $12K.

    a) What is the estimated cost and schedule of the software development, and how many people are needed on average for the Elaboration and Construction phases?

    b) What is the cost if the Navy mandates 41 months for the project?

11. The candidate reusable component analyzed in Section 3.5 would not have paid off for the ICS project. Suppose that DESC has identified another 40 KSLOC component with better reuse parameters:

- DM = 25
- CM = 30
- IM = 40
- AA = 4
- SU = 20

- UNFM = 0.5

For this component, compute its equivalent SLOC (ESLOC), and determine if it should be used in place of the newly developed 40 KSLOC instead.

12. The ICS project manager is concerned the reuse parameters above may be optimistic. Use the more conservative set of reuse parameters below to establish a risk reserve:

    - DM = 25
    - CM = 30
    - IM = 40
    - AA = 5
    - SU = 30
    - UNFM = 0.6

    Determine the resulting ESLOC, effort and cost using these parameters. The difference in cost should be used as risk reserve.

13. Three Open Source repositories for data analysis software are being considered for integrating with a 500 KSLOC product per the following data:

    Open Source #1 - 50 KSLOC, overall adaptation factor of .3 (accounting for AA, SU, UNFM, DM, CM and IM)
    Open Source #2 - 80 KSLOC, overall adaptation factor of .15
    Open Source #3 - 20 KSLOC, overall adaptation factor of .7

    Based on cost effectiveness and assuming all other factors being equal, which option should be chosen?

14. What is the Equivalent SLOC for a product consisting of the following language components:

    Java: 410 Unadjusted Function Points
    C++: 600 Unadjusted Function Points
    Unix Shell Scripts: 20 Unadjusted Function Points

    Use the total Equivalent SLOC to determine the nominal software development effort.

15. What is the estimated software development effort, schedule, and average staff for a drone project. The newly developed software size is 250,000 SLOC and modified software size is 35,000 SLOC.

    For the modified software:
    - 20 out of 200 design modules were modified
    - 10 KSLOC out of the total 35 KSLOC were modified
    - the integration of the modified code is estimated to be 2/3 of the effort compared to if it was all new
    - to understand the software, it is described as:

        - structure - high cohesion, low coupling
        - application clarity - good correlation between program and application
        - self-descriptiveness – good code commentary and headers, useful documentation, some weak areas
    - only basic module search and documentation was required for its assessment and assimilation
    - the team is considerably familiar with the software being modified.

16. A top Army project is for an application that enables soldiers to find the location of resources for Forward Operating Bases (FOBs). To achieve the Geo-Mapping capability, a geographic software vendor is offering a tailored version of Google Maps at an up-front license price of $750K.

    However, an alternate proposal is to use a dedicated team with expertise in geographic applications and Google Maps to build a better and cheaper map system. Prepare a COCOMO estimate of the custom development cost using these inputs:

    Adapt the Google Map software of 80 KSLOC with the adaptation parameters:
    - % Design Modified = 35
    - % Code Modified = 20
    - % Integration Required = 60
    - Assessment and Assimilation = 0
    - Software Understanding = 10
    - Unfamiliarity = 0.2

    Cost drivers are all Nominal except for the following:
    - *Programmer Capability* and *Analyst Capability*: 75th percentile
    - *Personnel Continuity*: stable staff with 3% annual turnover
    - *Multi-site Development*: entire team is co-located together
    - *Use of Software Tools*: strong toolset, moderately integrated

    The cost per person-month is $8K.

    Additionally, the manager believes that they would have no case if they delivered the software system too late. Thus they must have the software completed 6 weeks ahead of the deadline, which would suppress the schedule to 85% of the nominal schedule.

    Determine the estimated cost of adapting the Geo-Mapping capability for the application. Is it a better option from a cost standpoint than the vendor's tailored product?

17. The baseline ICS software cost analysis in Section 3.5 assumes that all the cost driver ratings are Nominal. Relative to the baseline case, perform a return on investment analysis to calculate

$$\text{ROI} = (\text{Savings} - \text{Costs}) / \text{Costs}$$

for an investment in software tools covering development and maintenance lifecycle phases (not the ROI of undertaking the project). Use the following information:

- The cost of the tools is $250K. This is the investment cost under consideration,
- The investment in tools will improve the ICS project's *Use of Software Tools* rating from Nominal to High for both the Development and Maintenance phases.
- Using the tools during development will reduce the ICS projects' *Language and Toolset Experience* rating from Nominal to Low. During maintenance, the experience with the tools will return the rating to Nominal.

You can assume that the maintenance cost would be twice the development cost at Nominal similar to the reliability tradeoff analysis in Section 3.5.

What is the ROI for the tool investment and your recommendation?

18. Investigate the tradeoffs between opportunity costs and delivery time for military capability against time-sensitive threats. Electronic warfare software is needed and timing is important to the mission. With longer delivery time mission value decreases because more adversaries with anti-measures enter the theater. More resources will then be required to counteract. The table below specifies the mission value losses as a function of development time.

Development Time	**Lost Mission Value**
2 years	0
2.5 years	$1M
3 years	$3M
3.5 years	$6M
4 years	$10M

Sum the costs of software development and opportunity costs (lost mission value) for the 5 schedule options shown in the table. Find the optimum schedule with respect to the total sum costs. It is estimated that the software will be 150 KSLOC of new development, and use a software labor rate of $8000 per person-month.

Development cost should be for elaboration and construction only. Inception is over because the requirements are already defined, and the system will be fielded immediately with no interim transition.

4 Hardware Cost Model

Hardware cost is defined as all program cost, excluding software development, for all components, assemblies, subsystems, and systems for sea, ground, airborne or space applications. It includes mechanical, electromechanical, electrical/electronic, structural or pneumatic equipment, and may also include system test, or system operations. It is the cost to develop, design, build, test, launch, and operate a system.

The Advanced Missions Cost Model (AMCM) predicts development, recurring and mission operations costs of ground vehicles, ships, aircraft, helicopters, missiles, launch vehicles, spacecraft, and human explorations missions [3]. It is a system level cost model appropriate for large scale programs requiring many different systems that will be integrated to perform a complex mission. The methodology is most useful in the pre-conceptual and conceptual design phases of a program when the actual design of the systems is not known and many factors are being traded off.

The AMCM is a two equation, multi-variable cost estimating relationship. The first equation predicts the development and production cost of the system based of various technical and programmatic factors. The second equation predicts the basic mission operations cost of the system. Both equations are fitted to a historical database for ground vehicles, ships, aircraft, helicopters, and missiles as well as launch vehicles and spacecraft.

The combined CER is shown in Equation 12. Model inputs are described in Table 17. The model parameter values are shown in Table 18. The system types and their specification values for the equation are in Table 19.

$$\text{Cost} = a \bullet Q^{b} \bullet W^{c} \bullet d^{S} \bullet e^{(1/(IOC-1900))} \bullet B^{f} \bullet g^{D} \qquad \textbf{(6)}$$

Where

- Q is the quantity
- W is the weight
- S is the Specification
- B is the Block Number
- D is the Difficulty Factor
- a-g are model parameters.

Table 17: AMCM Model Inputs

Input	**Description**
Quantity	The quantity is the total number of units to be produced. This includes prototypes, test articles, operational units, and spares.
Dry weight	The dry weight is the total empty weight of the system in pounds, not including fuel, payload, crew, or passengers.

Input	**Description**
Mission type	The mission type classifies the type of system by the operating environment and the type of mission to be performed. Select one that best describes the system you wish to estimate.
IOC Year	The IOC is the year of Initial Operating Capability. For space systems, this is the year in which the spacecraft or vehicle is first launched.
Block Number	The block number represents the level of design inheritance in the system. If the system is a new design, then the block number is 1. If the estimate represents a modification to an existing design, then a block number of 2 or more may be used. For example, block 5 means that this is the 5th in a series of major modifications to an existing system.
Difficulty	The difficulty factor represents the level of programmatic and technical difficulty anticipated for the new system. This difficulty should be assessed relative to other similar systems that have been developed in the past. For example, if the new system is significantly more complex than previous similar systems, then a difficulty of high or very high should be selected.

The model outputs the development, production and operations costs in millions of dollars. For space systems, the cost estimates are for prime contractor cost only and do not include fee, program support, or other government costs. For non-space systems, the costs are all inclusive.

The calibrated model parameters are listed in Table 18. Table 19 shows the available system types, number of historical data points for each, and their specification values used for S in Equation 12.

Table 18: AMCM Parameters

Parameter	**Value**
a	0.000504839
b	0.594183076
c	0.653947922
d	76.99939424
e	1.68051e-52
f	0.355322218
g	1.554982942
inf91	1.414

Table 19: AMCM System Type Specifications

System Type	**Number**	**Specification**
Aircraft - Attack	8	1.97
Aircraft - Bomber	7	1.99
Aircraft - Commercial	3	1.75

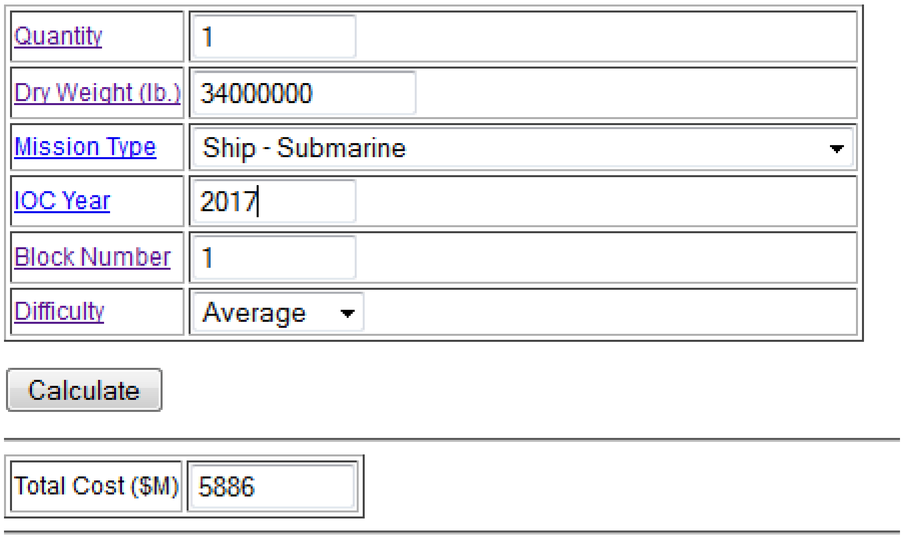
Aircraft - Fighter	16	1.91
Aircraft - Patrol	5	1.93
Aircraft - Recon (SR-71)	1	2.35
Aircraft - Rotary Attack	5	1.92
Aircraft - Rotary Transport/Utility	4	1.75
Aircraft - Trainer	3	1.48
Aircraft - Transport	9	1.67
Land Vehicle - APC	2	0.99
Land Vehicle - Artillery	0	1.12
Land Vehicle - Automobile	2	0.98
Land Vehicle - Rifle	3	1.63
Land Vehicle - Tank	4	1.29
Land Vehicle - Truck	7	0.87
Missile - Air-Air	14	2.02
Missile - Air-Orbit	1	2.02
Missile - Air-Surface	15	1.8
Missile - Anti-Tank	3	1.8
Missile - ICBM	11	1.91
Missile - ICBM (Sub Launched)	4	1.89
Missile - Rocket	5	1.63
Missile - Ship-Air	9	1.73
Missile - Surface-Air	12	1.95
Missile - Surf-Surf Land Mobile	8	1.87
Missile - Surf-Surf Other	4	2.05
Ship - AC Carrier	5	1.18
Ship - Amphib Assault	5	0.96
Ship - Cruiser	4	1.25
Ship - Destroyer	5	1.32
Ship - Frigate	3	1.19
Ship - Submarine	7	1.29
Space Transport - Centaur Fairing	2	1.6
Space Transport - Launch Vehicle Stage	3	2.01
Space Transport - Liquid Rocket Engine - Lox/Lh	3	2.19
Space Transport - Liquid Rocket Engine - Lox/RP-1	2	1.84
Space Transport - Payload Fairing	3	1.15
Space Transport - Unmanned Reentry	4	1.91
Space Transport - Upper Stage	5	2.07
Spacecraft - Communication	9	2.22
Spacecraft - Earth Observation	3	2.16
Spacecraft - Lunar Rover	1	2.14

Spacecraft - Manned Habitat	4	2.13
Spacecraft - Manned Reentry	6	2.27
Spacecraft - Physics & Astronomy	11	2.17
Spacecraft - Planetary	11	2.39
Spacecraft - Planetary Lander	3	2.46
Spacecraft - Weather	6	2.18

4.1 Examples

4.1.1 Submarine

Calculate the estimated cost for a 17 ton submarine. It is the first design of its type, average difficulty with an IOC year of 2017. The weight needs to be converted from tons to pounds as specified in the model. The Mission Type is chosen as shown in Figure 4, one submarine with the corresponding Dry Weight, Block Number 1 because it's the initial design, and Difficulty of average. The resulting cost for a single submarine is displayed as $5886M.



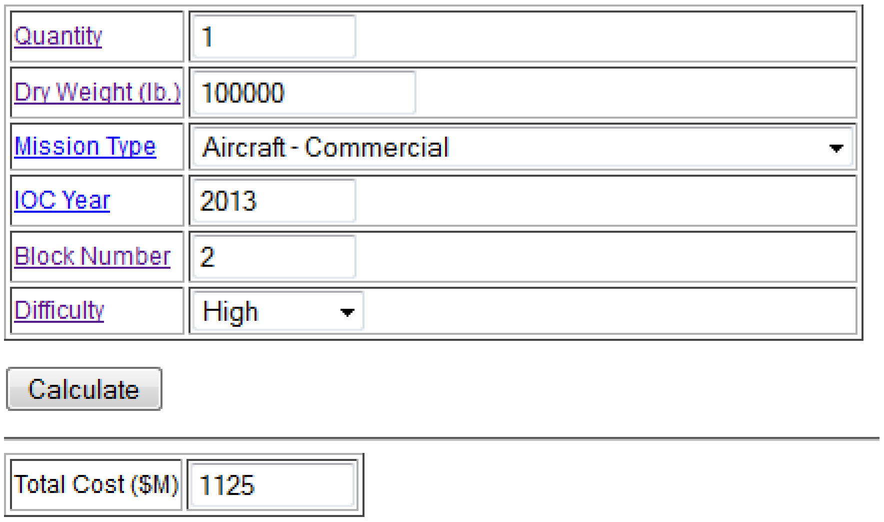
Quantity	1
Dry Weight (lb.)	34000000
Mission Type	Ship - Submarine
IOC Year	2017
Block Number	1
Difficulty	Average

Total Cost ($M)	5886

Figure 4: Submarine Example

4.1.2 Aircraft Fleet

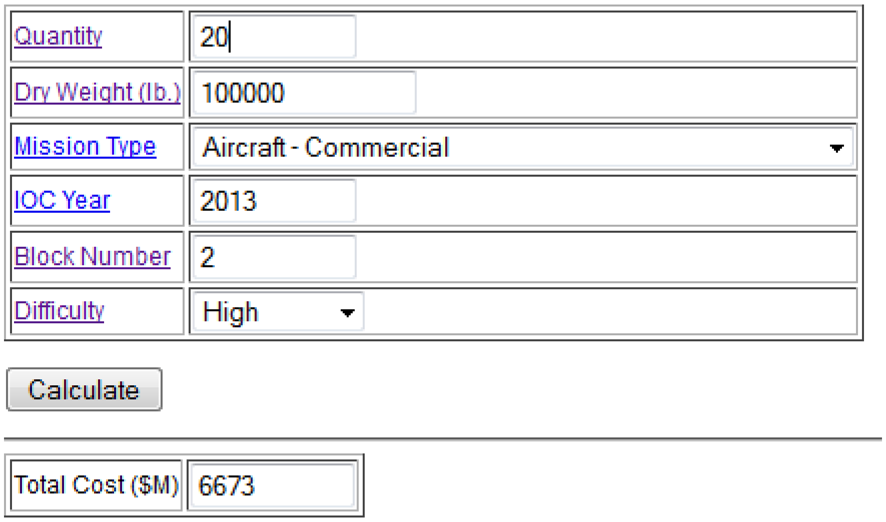
Calculate the cost of a fleet of 20 commercial aircraft. The aircraft is high difficulty, but modifies an initial design and is thus Block Number 2. Assume each weighs about 100,000 pounds. For reference the first unit cost is shown in Figure 5 as $1125M. Subsequent planes have lower unit costs, and the total for 20 aircraft comes to $6673M per Figure 6 for an average unit cost of $334M per plane.



Quantity	1
Dry Weight (lb.)	100000
Mission Type	Aircraft - Commercial
IOC Year	2013
Block Number	2
Difficulty	High

Total Cost ($M)	1125

Figure 5: Aircraft First Unit Cost Example



Quantity	20
Dry Weight (lb.)	100000
Mission Type	Aircraft - Commercial
IOC Year	2013
Block Number	2
Difficulty	High

Total Cost ($M)	6673

Figure 6: Aircraft Fleet Example

4.1.3 Ship Total Ownership Cost

The Navy requires a new Amphibious Assault Ship with a planned IOC of 2019. The program office wants to assess the Total Ownership Cost for a 15 year operational lifetime and set a conservative budget for it. First derive a point estimate for the full lifecycle, then apply uncertainty analysis to determine a 70% confidence level budget.

a) Use the following information for the initial point estimate covering RDT&E and 15 years of operations.

The ship will be new design with a most likely weight of 5000 long tons. The combined subsystems for the ship total to the following sizes and degrees of difficulty:

	Easy	Nominal	Difficult
# of System Requirements	120	185	48
# of System Interfaces	12	67	45
# of Algorithms	19	125	58
# of Operational Scenarios	3	14	8

Other system-level cost factors are:

- The requirements understanding is strong with only a few undefined areas.
- There is a similar strong understanding of the system architecture and COTS.
- The level of service requirements involve complex, coupled Key Performance Parameters and are critical due to a risk to human life.

The software will be comprised of the following:

- The total new software to develop is estimated to be 850 KSLOC.
- There will be reused combat system software that is 225 KSLOC. This reused software will require substantial integration and testing that is 50% of the effort compared to new software.
- Other subsystems will include 400 KSLOC of modified software. It is estimated that 10% of the modified software design needs to change, about 15% of its code will change and will require 60% integration and testing compared to new. The software baseline is described as:
    - Moderate level of code commentary, headers, documentation.
    - High cohesion, low coupling.
    - Clear match between program and application world-views.

Additional software cost factors include:

- There will be some development flexibility allowed to the contractors, and occasional relaxation of the need to conform to specified requirements.
- The overall team of organizations is fairly cohesive and largely cooperative with each other.
- The required software reliability of nearly all systems must be very high, since failure could be a risk to human life.
- The software complexity is very high with many subsystems interfacing with each other.
- The software development starts in FY14 and must be completed in five years to meet the IOC of 2019.

For RDT&E include the costs of systems engineering, software development, hardware development and production. For maintenance and upgrades over 15 years of operations assume:

- Systems engineering modifications to requirements, interfaces, algorithms, and scenarios are estimated to be 10% per year.
- For software maintenance 150 Equivalent KSLOC is expected to change per year. The software understanding will be improved from RDT&E due to the new software, with an estimated 10% less penalty.
- The likely operational costs of the ship hardware, associated maintenance and spare parts are $30M annually for this ship class.

Further ground rules and assumptions for the lifecycle costs and phasing are:

- All systems, software and hardware activities commence in parallel.
- Hardware monthly costs are constant before IOC, and constant within each year after IOC.
- The software Transition phase occurs once after IOC.

All labor rates are $10,000 per Person-Month.

For systems engineering, the total size is 2650 Equivalent Nominal Requirements. The off-nominal cost factors are described as:

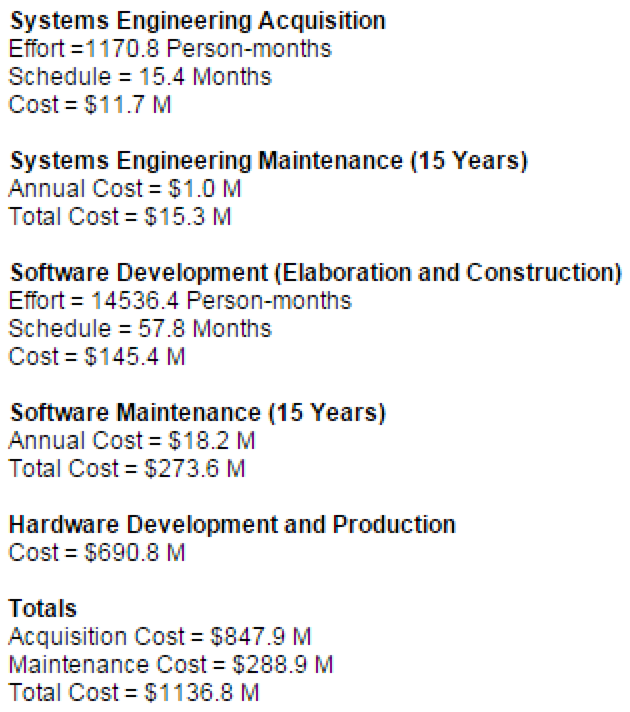
- *Requirements Understanding* is High
- *Architecture Understanding* is High
- *Level of Service Requirements* is Very High.

The software size requires inputs for new, modified and reused categories. The total aggregate size is 1031710 Equivalent SLOC (the detailed inputs are shown later in this example). The off-nominal cost factors are described as:

- *Development Flexibility* is Low
- *Team Cohesion* is High
- *Required Software Reliability* is Very High
- *Product Complexity* is Very High
- *Required Development Schedule* is Very Low.

The ship weight of 5000 long tons at 2240 pounds/long ton is the equivalent of 11,200,000 pounds for the AMCM model input. The new design indicates Block #1.

The collective inputs for systems, software and hardware above result in the point estimate in Figure 5 across acquisition and maintenance.



Systems Engineering Acquisition
Effort =1170.8 Person-months
Schedule = 15.4 Months
Cost = $11.7 M

Systems Engineering Maintenance (15 Years)
Annual Cost = $1.0 M
Total Cost = $15.3 M

Software Development (Elaboration and Construction)
Effort = 14536.4 Person-months
Schedule = 57.8 Months
Cost = $145.4 M

Software Maintenance (15 Years)
Annual Cost = $18.2 M
Total Cost = $273.6 M

Hardware Development and Production
Cost = $690.8 M

Totals
Acquisition Cost = $847.9 M
Maintenance Cost = $288.9 M
Total Cost = $1136.8 M

Figure 5: Point Estimate Summary

However, the ship maintenance needs to added manually since the AMCM model currently has no maintenance provisions. The annual maintenance and spare parts would cost:

15 years * $30M = $450M.

The TOC point estimate totals $1136.8M + $450M = $1586.8M.

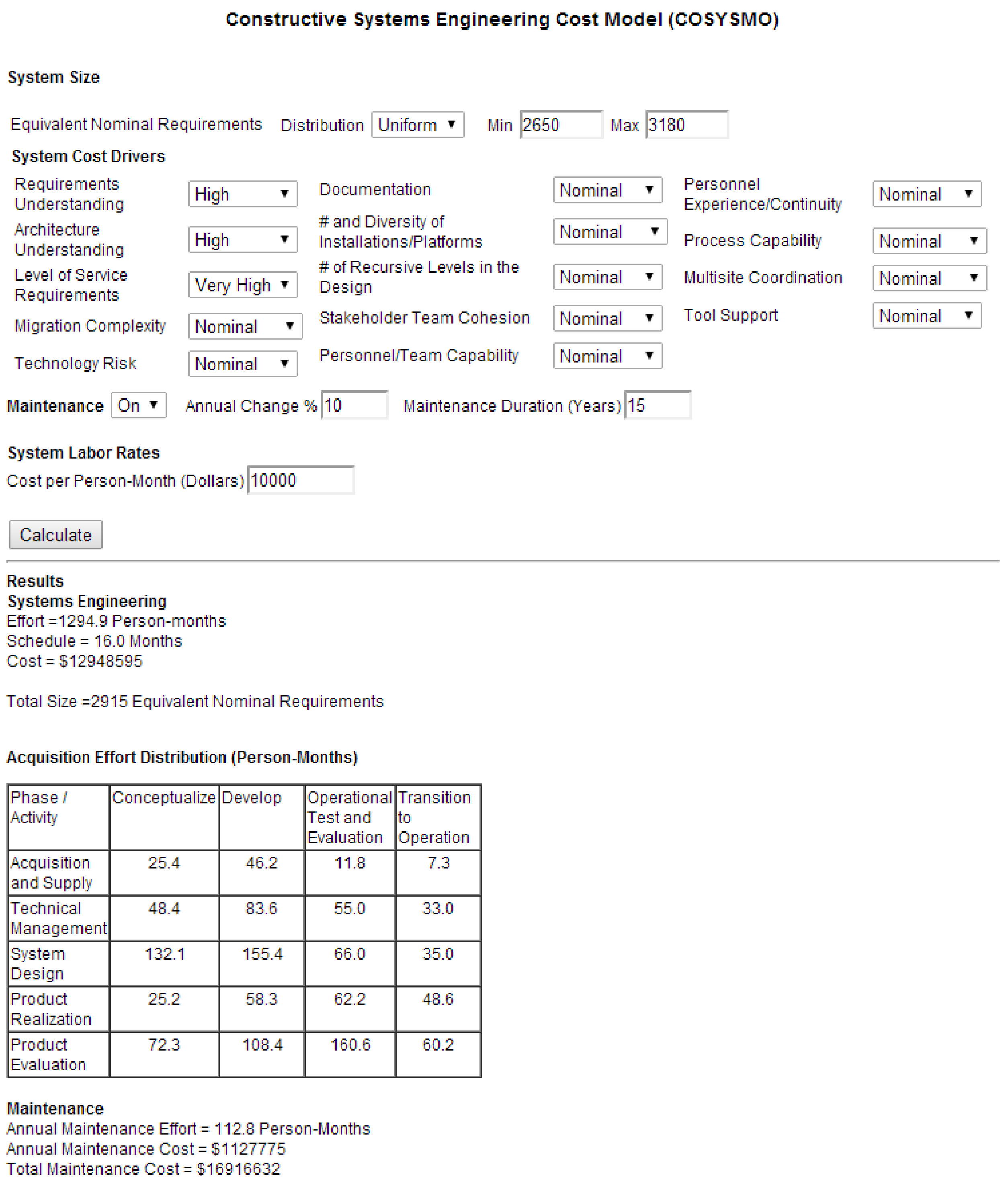
b) Next develop a safe 70% confidence level estimate for budgeting purposes and present your detailed results.

Use a symmetrical triangular probability distribution for the ship weight. The ship architects say it might be reduced to 2500 long tons, 5000 is still most likely, but it might need to be 7500 long tons to fit all mission systems on-board.

Model system requirements volatility with a uniform probability distribution for systems engineering size. Use your point estimate for total equivalent size as the minimum of a uniform distribution with a maximum that is 20% additional size.

For software size, choose a normal probability distribution using the point estimate equivalent size as the mean with a standard deviation of 15%. Assume that all other parametric cost factors stay fixed.

The complete systems engineering inputs including the probability distribution is shown next in Figure 6.



Constructive Systems Engineering Cost Model (COSYSMO)

System Size

Equivalent Nominal Requirements Distribution Uniform ▼ Min 2650 Max 3180

System Cost Drivers

Requirements Understanding	High ▼	Documentation	Nominal ▼	Personnel Experience/Continuity	Nominal ▼
Architecture Understanding	High ▼	# and Diversity of Installations/Platforms	Nominal ▼	Process Capability	Nominal ▼
Level of Service Requirements	Very High ▼	# of Recursive Levels in the Design	Nominal ▼	Multisite Coordination	Nominal ▼
Migration Complexity	Nominal ▼	Stakeholder Team Cohesion	Nominal ▼	Tool Support	Nominal ▼
Technology Risk	Nominal ▼	Personnel/Team Capability	Nominal ▼		

Maintenance On ▼ Annual Change % 10 Maintenance Duration (Years) 15

System Labor Rates

Cost per Person-Month (Dollars) 10000

Results
Systems Engineering
Effort =1294.9 Person-months
Schedule = 16.0 Months
Cost = $12948595

Total Size =2915 Equivalent Nominal Requirements

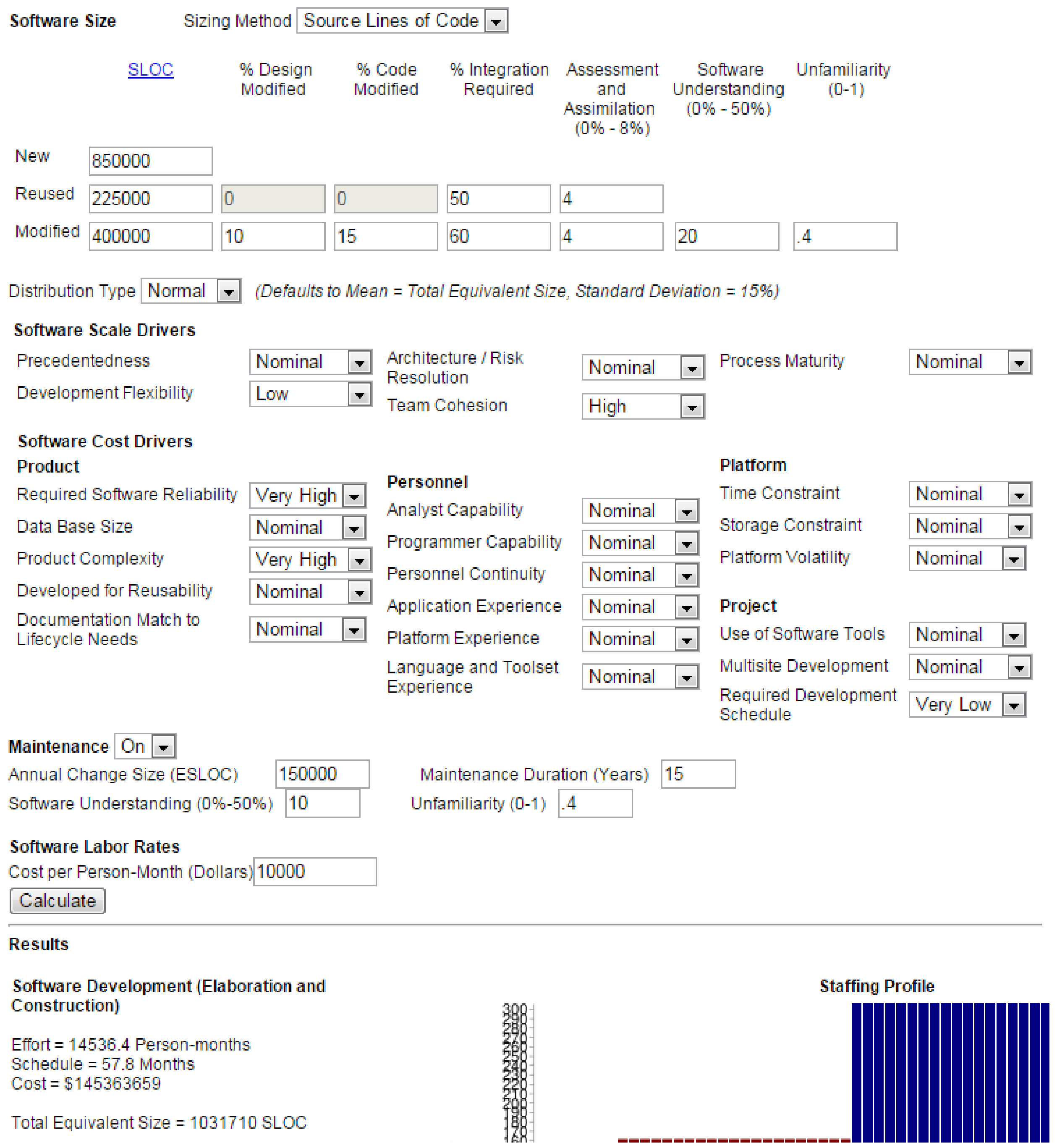
Acquisition Effort Distribution (Person-Months)

Phase / Activity	Conceptualize	Develop	Operational Test and Evaluation	Transition to Operation
Acquisition and Supply	25.4	46.2	11.8	7.3
Technical Management	48.4	83.6	55.0	33.0
System Design	132.1	155.4	66.0	35.0
Product Realization	25.2	58.3	62.2	48.6
Product Evaluation	72.3	108.4	160.6	60.2

Maintenance
Annual Maintenance Effort = 112.8 Person-Months
Annual Maintenance Cost = $1127775
Total Maintenance Cost = $16916632

Figure 6: Systems Engineering Inputs

The software inputs are in Figure 7 below.



Software Size Sizing Method: Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	850000						
Reused	225000	0	0	50	4		
Modified	400000	10	15	60	4	20	.4

Distribution Type: Normal *(Defaults to Mean = Total Equivalent Size, Standard Deviation = 15%)*

Software Scale Drivers

Precedentedness	Nominal
Development Flexibility	Low
Architecture / Risk Resolution	Nominal
Team Cohesion	High
Process Maturity	Nominal

Software Cost Drivers

Product

Required Software Reliability	Very High
Data Base Size	Nominal
Product Complexity	Very High
Developed for Reusability	Nominal
Documentation Match to Lifecycle Needs	Nominal

Personnel

Analyst Capability	Nominal
Programmer Capability	Nominal
Personnel Continuity	Nominal
Application Experience	Nominal
Platform Experience	Nominal
Language and Toolset Experience	Nominal

Platform

Time Constraint	Nominal
Storage Constraint	Nominal
Platform Volatility	Nominal

Project

Use of Software Tools	Nominal
Multisite Development	Nominal
Required Development Schedule	Very Low

Maintenance On

Annual Change Size (ESLOC)	150000
Maintenance Duration (Years)	15
Software Understanding (0%-50%)	10
Unfamiliarity (0-1)	.4

Software Labor Rates

Cost per Person-Month (Dollars): 10000

Results

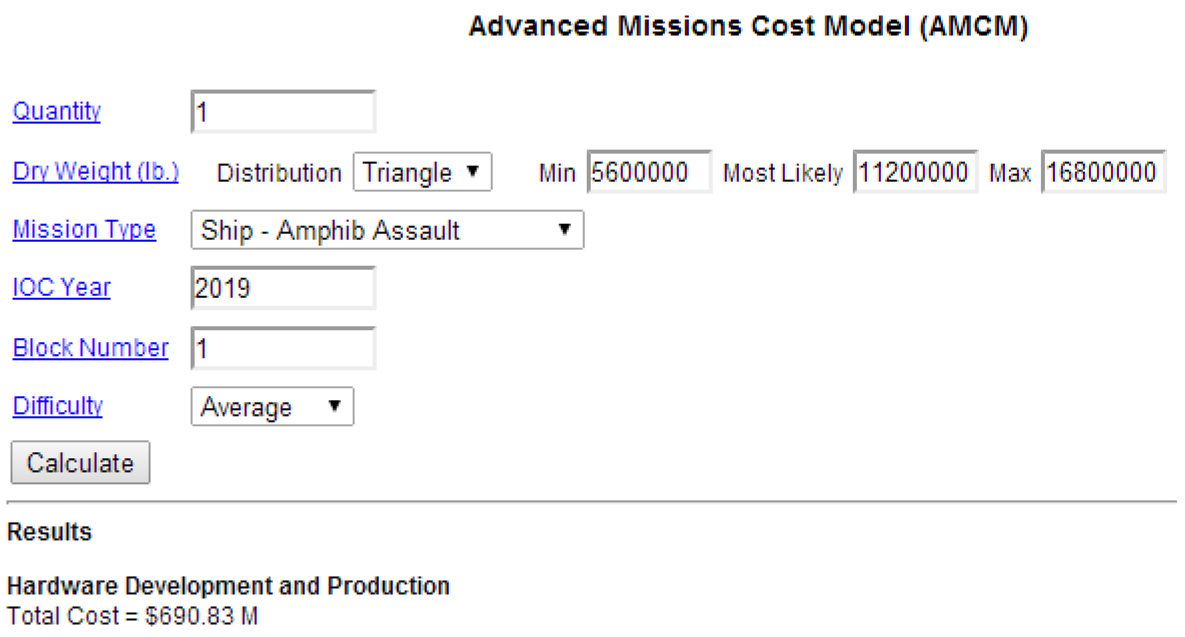
Software Development (Elaboration and Construction)

Effort = 14536.4 Person-months
Schedule = 57.8 Months
Cost = $145363659

Total Equivalent Size = 1031710 SLOC

Figure 7: Software Engineering Inputs

The hardware development and production input factors are shown in Figure 8.


Advanced Missions Cost Model (AMCM)

Quantity: 1

Dry Weight (lb.) Distribution: Triangle; Min: 5600000; Most Likely: 11200000; Max: 16800000

Mission Type: Ship - Amphib Assault

IOC Year: 2019

Block Number: 1

Difficulty: Average

Calculate

Results

Hardware Development and Production
Total Cost = $690.83 M

Figure 8: Hardware Inputs

The detailed results in Figure 9 and Figure 10 show the summary results and separate CDFs for RDT&E, operations and the combined TOC. When adding in the 70% confidence level cost for ship hardware operations/maintenance, one can manually adjust the $30M by the same proportion as the AMCM hardware point estimate to its 70% confidence level.

Systems Engineering Acquisition
Effort =1294.9 Person-months
Schedule = 16.0 Months
Cost = $12.9 M

Systems Engineering Maintenance (15 Years)
Annual Cost = $1.1 M
Total Cost = $16.9 M

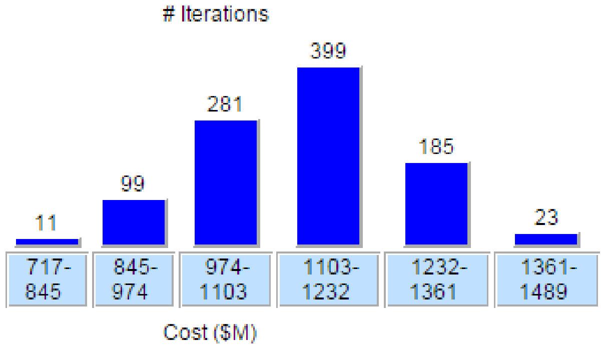
Software Development (Elaboration and Construction)
Effort = 14536.4 Person-months
Schedule = 57.8 Months
Cost = $145.4 M

Software Maintenance (15 Years)
Annual Cost = $18.2 M
Total Cost = $273.6 M

Hardware Development and Production
Cost = $690.8 M

Totals
Acquisition Cost = $849.1 M
Maintenance Cost = $290.5 M
Total Cost = $1139.6 M

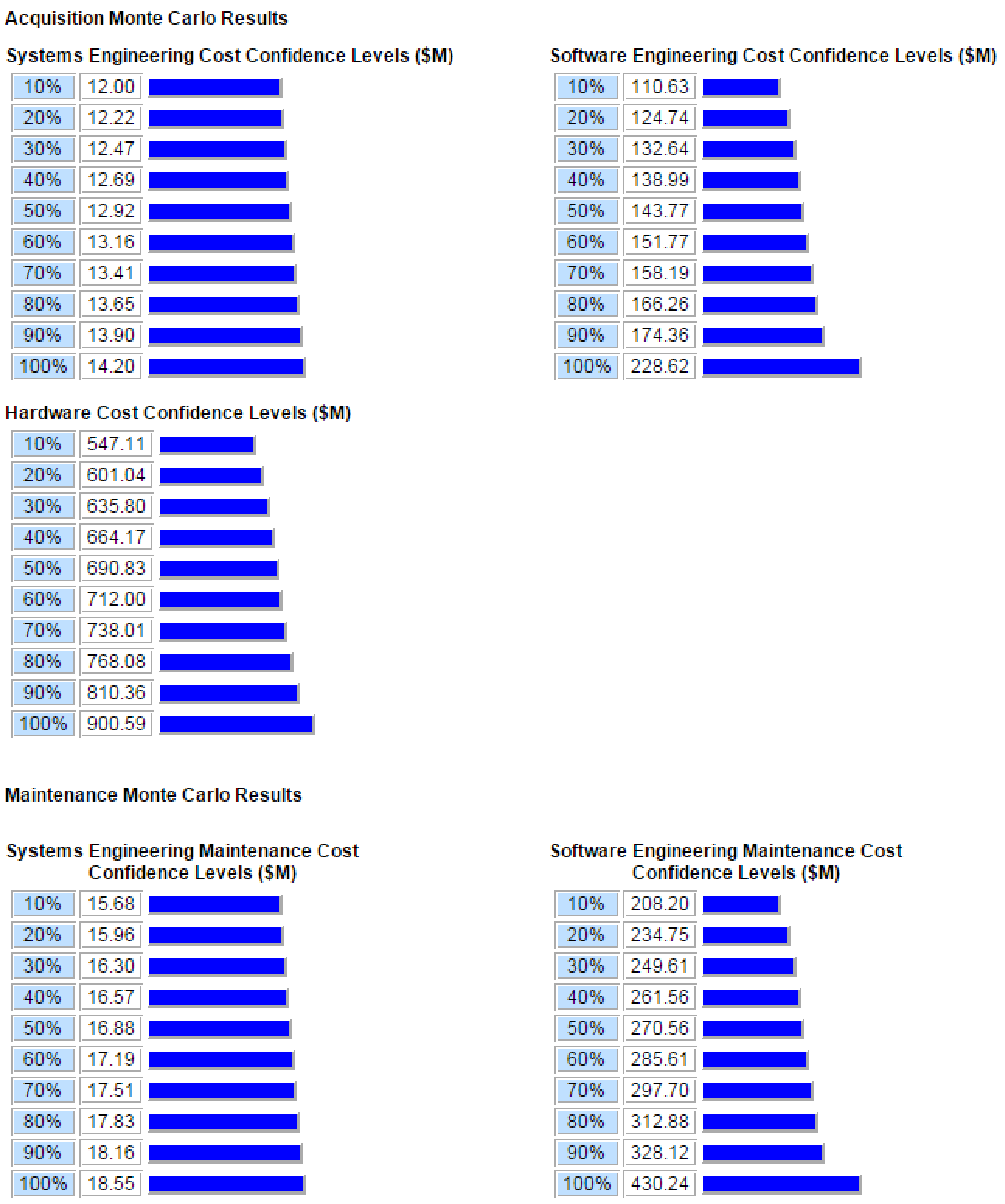
Total Cost Distribution Function



Total Cost Confidence Levels ($M)

10%	968.10	█
20%	1,026.48	█
30%	1,071.08	█
40%	1,106.85	█
50%	1,139.05	█
60%	1,168.84	█
70%	1,203.61	█
80%	1,235.54	█
90%	1,286.77	█
100%	1,489.79	█

Figure 9: Project Summary with Monte Carlo Analysis



Acquisition Monte Carlo Results

Systems Engineering Cost Confidence Levels ($M)

10%	12.00
20%	12.22
30%	12.47
40%	12.69
50%	12.92
60%	13.16
70%	13.41
80%	13.65
90%	13.90
100%	14.20

Software Engineering Cost Confidence Levels ($M)

10%	110.63
20%	124.74
30%	132.64
40%	138.99
50%	143.77
60%	151.77
70%	158.19
80%	166.26
90%	174.36
100%	228.62

Hardware Cost Confidence Levels ($M)

10%	547.11
20%	601.04
30%	635.80
40%	664.17
50%	690.83
60%	712.00
70%	738.01
80%	768.08
90%	810.36
100%	900.59

Maintenance Monte Carlo Results

Systems Engineering Maintenance Cost Confidence Levels ($M)

10%	15.68
20%	15.96
30%	16.30
40%	16.57
50%	16.88
60%	17.19
70%	17.51
80%	17.83
90%	18.16
100%	18.55

Software Engineering Maintenance Cost Confidence Levels ($M)

10%	208.20
20%	234.75
30%	249.61
40%	261.56
50%	270.56
60%	285.61
70%	297.70
80%	312.88
90%	328.12
100%	430.24

Figure 10: Detailed Monte Carlo Results

4.2 Exercises

The following exercises involve hardware cost estimation, and integrated estimates for total cost of systems engineering, software engineering and hardware.

1. Amphibious Assault Ships are being procured by the Navy. Two ships are needed that weigh 24 tons each, and three additional ships in a smaller variation of 20 tons. IOC is planned in four years using a second time design that is high difficulty. What is the estimated cost to the Navy for all five ships?

2. The Army is upgrading a fleet of Armored Personnel Carriers (APCs). Modification of an existing design is being compared to a new architecture using composite structural materials and smaller electronics. The cost profiles will be used in tradeoff analyses with mission effectiveness estimates. The current APC design has already been through 4 blocks, and the upgrade design remains average difficulty. The carriers weigh 28,000 pounds each. The new APC architecture is complex, and of very high difficulty but the APCs weights are reduced to 17,000 pounds each.

   It is desired to quickly field both first APC units for evaluation. The current design can be ready in 1 year but the new one will take 2 years for IOC. What are the respective first APC unit costs and average unit costs for 50 APCs?

3. A program for patrolling a U.S region for illicit activities using aircraft is being assessed. An associated estimate of the total economic benefits of the interdiction vs. the number of patrol aircraft is shown below. Apply marginal reasoning to decide the optimal number of aircraft, by expanding the number of aircraft as long as the marginal benefit is greater than the marginal cost.

# Patrol Aircraft	Benefit ($M)
1	1995
2	3458
3	4522
4	5320
5	5852
6	6155
7	6358
8	6517
9	6610
10	6677

   Estimate the program costs with the Advanced Missions Cost Model (AMCM). The patrol aircraft will be a new design of average difficulty. Each aircraft is estimated to weigh about 70,000 pounds each.

4. The Navy desires to develop information dominance capabilities to satisfy national security strategy objectives for Maritime Domain Awareness (MDA). To support their decision you are to estimate the annual cost profiles over time of two different Courses of Action (COAs).

   These MDA capabilities will be developed through the use of new and emerging commercial software technologies. By combining these technologies, different COAs for the capability of "Develop Awareness" were developed for the OPNAV Capability Based Assessment. COA 1 assumes that the technologies will be inserted within 13 existing systems. COA 2 creates a new architecture that combines the systems into one overall system, so the software to be developed is reduced. The ground rules and assumptions for these options are listed below.

   Ground Rules and Assumptions for the Materiel Solutions

   1. Annual maintenance costs are 25% of the total software cost and do not start until the software development is complete.
   2. Use $10,000 cost per person month for systems and software engineers.
   3. COAs are costed independently.
   4. Assume both COAs start at the beginning of next fiscal year.

	COA 1	**COA 2**
System Size	75 easy requirements 100 nominal requirements 45 difficult requirements 10 nominal interfaces 6 difficult interfaces 15 nominal algorithms 3 difficult algorithms 1 easy scenario 5 nominal scenarios	40 easy requirements 125 nominal requirements 60 difficult requirements 12 nominal interfaces 3 difficult interfaces 10 nominal algorithms 8 difficult algorithms 1 easy scenario 5 nominal scenarios
Software Size	New: 218 KSLOC Adapted: 55 Equivalent KSLOC	New: 96 KSLOC Adapted: 80 Equivalent KSLOC
Technology Risk	Lack of Maturity-Proven on pilot projects and ready to roll-out for production jobs Lack of Readiness -Concept qualified (TRL 8) Obsolescence-New and better technology is on the horizon in the near-term	Lack of Maturity-Proven on pilot projects and ready to roll-out for production jobs Lack of Readiness -Concept has been demonstrated (TRL 7) Obsolescence-Emerging technology could compete in future
System Level of Service Requirements	- Difficulty low - Criticality low	- Difficulty nominal - Criticality nominal
Requirements Understanding	Strong: few undefined areas	Reasonable: some undefined areas
# of Recursive Levels in the System Design	Some vertical and horizontal coordination	More complex interdependencies coordination, and tradeoff analysis
System Architecture Understanding	Strong understanding of architecture with few unfamiliar areas	Reasonable understanding of architecture and COTS, some unfamiliar areas

Use the above information to estimate an annual cost of COAs 1 and 2 for the first five years. List the costs per year in dollars.

Assume that systems and software engineering are sequential activities, with software engineering not starting until the systems engineering is complete. Only the software Elaboration and Construction phases are to be estimated, since requirements will already be defined and the system will be fielded immediately after IOC. Software maintenance costs start incurring when software engineering is complete (after Construction). It can be assumed that all systems engineering costs will occur in Year 1 for both COAs.

5. Compare the RDT&E costs between two system development approaches for a ship-air missile planned for IOC in five years. Include the costs of systems engineering, software and hardware using the COSYSMO, COCOMO and AMCM models through the first missile production. System option 1 is all new development and option 2 adapts a previous system. At this stage only product characteristics are known, as described.

<u>Systems Engineering</u>

For systems engineering effort, the overall system has high level of service requirements and the following:

- Requirements: 20 easy, 30 nominal, 10 difficult
- Interfaces: 16 nominal, 4 difficult
- Algorithms: 14 easy, 56 nominal, 20 difficult
- Scenarios: 2 easy, 4 nominal, 2 difficult

Because option 1 is all new development, it has very high technical risk and a low architecture understanding. Option #2 adapts a previous system and thus has a high architecture understanding. The systems engineering size inputs can be categorized as "Modified" in the COSYSMO reuse model for the same counts of requirements, interfaces, algorithms and scenarios as option 1.

<u>Software Engineering</u>
For the missile software, there will be a high financial loss if it fails and the product complexity in both designs is very high.

In option 1, the software is estimated to be 500 KSLOC (new) with low precedentedness and low architecture/risk resolution. In option 2, the software precedentedness and architecture/risk resolution are both high from the adapted system. The software will comprise 300 KSLOC of new code plus modified software. The modified software portion is 100 KSLOC that will require half of its design and code to be changed with full integration and testing compared to new.

<u>Hardware Engineering</u>
The missile hardware is estimated to be 200 pounds and high difficulty. The missile hardware for option 2 will be the 3rd in a series of major modifications.

The labor costs are $10,000/Person-Month across the board. Each estimate should indicate the total cost and a decomposition for the separate disciplines. Use constant year dollars directly from the cost models as the funds to be obligated now. Which option is the lowest cost?

6. Continue the ship-air missile RDT&E costing above using the lowest cost option and augment that point estimate in the following ways:
    - Apply uncertainty analysis to determine the probability of achieving the contractor's bid of $38M for RDT&E only.
    - Calculate the Total Ownership Cost (TOC) including the RDT&E costs in the initial five years to IOC, plus five years of operations accounting for system upgrades/maintenance and additional missile production.
    - Develop a TOC conservative budget by fiscal year at the 70% confidence level in Then-Year dollars.

    a) The largest sources of cost variance and project uncertainty are in the size and weight drivers. Model their probability distributions, keep other factors fixed and assess the resulting cost spread.

Account for possible system requirements volatility with a uniform distribution for systems engineering size. It is not expected that the requirements count will decrease but new and modified requirements are likely. Use your point estimate for total equivalent size as the minimum of a uniform distribution, and choose a maximum that represents 15% extra size.

There is a fair degree of uncertainty in the software size implementation, as well as probable volatility of requirements in tandem with systems engineering. Model software size with a normal distribution using the point estimate as the mean with a standard deviation of 15%.

Vary the missile weight with a triangular distribution. The engineers say it might be reduced to 150 pounds, 200 pounds is still most likely, but it might grow to 300 pounds.

Present and interpret your final RDT&E Cost Cumulative Distribution Functions from the combined systems, software and hardware development models. What is the likelihood of the cost being less than or equal to the contractor's bid of $38M for RDT&E only?

b) For maintenance and upgrades in the first 5 years of operations assume:

- Systems engineering modifications to requirements, interfaces, algorithms, and scenarios are estimated to be 15% per year.
- For software maintenance a corresponding 15% of the software is expected to change per year. For *Software Understanding* the organization adheres to rigorous documentation standards, the software structure exhibits high cohesion, low coupling and the application clarity is very high. The *Unfamiliarity* of the staff will probably be an even combination of people familiar with and unfamiliar with the software baseline, especially as people transition through the project.
- Annual missile production is two each year after IOC.
- Annual upgrade/maintenance hardware cost for each missile owned is $50,000.

Further ground rules and assumptions for the lifecycle costs and phasing are:

- All systems, software and hardware activities commence in parallel.
- Use the software cost model phase estimates, except Inception effort is not included because it is covered in systems engineering.
- Hardware monthly costs are constant before IOC for the first missile, and constant within each year after IOC.
- The software Transition phase occurs once after IOC.

Develop a safe 70% confidence level TOC estimate for budgeting purposes and present your detailed results.

5 Estimation Tools

Web-based tools for these parametric cost models are available at mirror sites. The *System Cost Model Suite* tool is at:

http://csse.usc.edu/tools/cost_model_suite.php

http://softwarecost.org/tools/cost_model_suite

An NPS tool location is currently being updated.

6 References

[1] B. Boehm, C. Abts, W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, B. Steece, *Software Cost Estimation with COCOMO II*, Prentice-Hall, 2000

[2] R. Valerdi, *Systems Engineering Cost Estimation with COSYSMO*, Wiley, 2016

[3] NASA, *Parametric Cost Estimating Handbook*, 1995

[4] QSM, Function Point Programming Languages Table, http://www.qsm.com/FPGearing.html, 2000

Last Update 5/2/2017