

COTS Software Integration Cost Modeling Study

University of Southern California
Center for Software Engineering

Performed for the
USAF Electronic Systems Center
Hanscom AFB, Massachusetts
under
Contract F30602-94-C-1095

Prepared by
Christopher M. Abts

Under the direction of
Dr. Barry W. Boehm
Director, USC Center for Software Engineering

Study concluded
29 June 1997



**UNIVERSITY OF
SOUTHERN CALIFORNIA**

Period of Performance

This research was sponsored by the USAF Electronic Systems Center through Rome Laboratory under DoD contract number F30602-94-C-1095. The original period of performance was March 1, 1996 through February 28, 1997. An amendment was made to the original contract providing for a four month no cost extension to the period of performance concluding June 29, 1997.

Acknowledgments

In addition to the support of the USAF Electronic Systems Center, the USC Center for Software Engineering would like to recognize the ongoing generous support of the CSE Affiliates, without whom this work could not have been completed. The Affiliates are:

Aerospace Corp.
AT&T Bell Laboratories
Bellcore
DISA
E-Systems
Electronic Data Systems
Hughes Aircraft Company
Institute for Defense Analysis
Interactive Development Environments
Jet Propulsion Laboratory
Litton Data Systems
Lockheed Martin Corp.
Loral
Motorola
Northrop Grumman Corp.
Rational, Inc.
Rockwell International
Science Applications International Corp.
Software Engineering Institute
Software Productivity Consortium
Sun Microsystems
Teledyne, Inc.
TRW
U.S. Air Force Cost Analysis Agency
U.S. Air Force Rome Laboratory
U.S. Army Research Laboratory
Xerox

Table of Contents

<i>Period of Performance</i>	p. i
<i>Acknowledgments</i>	p. i
I. Executive Summary.....	p. 1
II. Mapping of COTS Study Contractual Requirements to Report Contents.....	p. 2
III. Introduction/Topic Background Placing COTS Research in Context.....	p. 3
IV. Scope of Project/Overview of Methodology.....	p. 5
V. Modeling Background.....	p. 8
A. Literature Review/Other Models.....	p. 8
B. Related Topic--Software Reuse.....	p. 13
C. COTS Related Definitions.....	p. 14
VI. Derivation of CSE Proposed Influence Factors/Cost Drivers.....	p. 15
A. SEI Risk Repository/Risk Taxonomy/Lexical Maps.....	p. 15
B. Loral Model Drivers.....	p. 19
C. COCOMO II Drivers.....	p. 20
D. Round 1 Survey Influence Factors.....	p. 23
E. Expert Panel Input Leading to Synthesis of Items A through D.....	p. 27
VII. Mathematical Model.....	p. 32
A. COTS Integration Life-cycle.....	p. 32
B. COTS Integration Activity Phases Covered.....	p. 32
C. Equations.....	p. 33
D. Definition of Terms/Cost Drivers.....	p. 34
E. Delphi Experiment.....	p. 39
F. Initial Delphi Derived Multiplier Values.....	p. 42
VIII. Calibration Efforts/Results.....	p. 44
A. Round 2 Survey.....	p. 44
B. Student Projects.....	p. 44
C. Industrial Projects.....	p. 56
D. Final Version 1.0 Multiplier Values.....	p. 65
IX. Cost Estimation Procedure Using the Model.....	p. 66
A. Estimation Procedure.....	p. 66
B. How to Size Glue Code via Function Points.....	p. 67
C. How to Estimate Percentage Breakage.....	p. 67
X. Relating/Using the COTS Estimate with a COCOMO II Project Estimate.....	p. 68
XI. Overall COTS Integration Cost Estimation/Mitigation guidelines.....	p. 69
XII. Conclusion/Future Directions.....	p. 75
XIII. References.....	p. 76

Appendices:

- A. COTS Integration Cost Calculator V1.0
- B. Calibration Case Data
 - B1. Student Projects
 - B2. Industrial Projects
- C. SEI Risk Repository Data
 - C1. COTS Related SERR Statements
 - C2. COTS Related Lexical Maps
- D. Data Collection Instruments
 - D1. Delphi Exercise
 - D2. Data Request Letter and Round 1 Survey
 - D3. Round 2 Survey

I. Executive Summary

This study represents a first effort towards the goal of developing a comprehensive COTS integration cost modeling tool. The approach taken was to first examine a wide variety of sources in an attempt to identify the most significant factors driving COTS integration costs, and to develop a mathematical form for such a model. These sources ranged from already existing cost models to information gathered in a preliminary high level data collection survey. Once the form and candidate drivers had been identified, the next step was to gather project level COTS integration effort data in a second round data collection exercise. This project level data was then used to calibrate and validate the proposed model. Data from both a graduate level software engineering class and from industrial sources were used in calibration attempts. The industrial data proved problematic, however, so for the purposes of this study, the final calibration of the model was based upon the student projects.

The final result was a cost model following the general form of the well-known COCOMO software cost estimation model, but with an alternate set of cost drivers. The scope of the model is also narrow, addressing only initial integration coding costs. The predictive power of the model at this stage is only fair, but it was demonstrated that with appropriate data, the accuracy of the model could be greatly improved.

Finally, the richness to the problem of capturing *all* significant costs associated with using COTS software offers many worth-while directions in which to expand the scope of this model.

II. Mapping of COTS Study Contractual Requirements to Report Contents

Task 4.1.8.1 - Sentence 1¹

Analyze SEI Risk Repository and other sources for COTS experience factors.

Reference Report: Sections IV and VI, Appendices C and D.

Task 4.1.8.1- Sentences 2 and 3

Prepare COTS experience questionnaire and data request letter for government review. Incorporate government comments and provide questionnaire to selected users.

Reference Report: Sections IV and VI.D, Appendix D2.

Task 4.1.8.2 - Sentence 1

Investigate candidate functional forms for the COTS integration cost model.

Reference Report: Sections IV and V.

Task 4.1.8.2 - Sentence 2

Collect and analyze initial questionnaire responses.

Reference Report: Sections IV and VI.D.

Task 4.1.8.2 - Sentence 3

Develop initial cost model and test on small initial data sample.

Reference Report: Sections IV, VII, VIII.A, VIII.B and IX, Appendix B1.

Task 4.1.8.3 - Sentence 1

Collect and analyze further questionnaire and cost data.

Reference Report: Sections IV and VIII.A , Appendix D3.

Task 4.1.8.3 - Sentence 2

Update initial cost model based on data collected.

Reference Report: Sections IV and VIII.C, Appendix B2.

Task 4.1.8.4

Prepare a report on guidelines for scoping COTS integration cost and schedules.

Reference Report: All Sections, with special note of Section XI.

¹ B.W. Boehm, Technical Proposal: Added Tasks for Contract F30602-94-C-1095, "Next Generation Software Processes and Their Environment Support," USC Center for Software Engineering, January 4, 1996.

III. Introduction/Topic Background Placing COTS Research in Context

One of the more significant changes in the software development market over the past twenty years is the greatly increased emphasis being placed on building systems incorporating pre-existing software, with special emphasis being placed upon the use of commercial-off-the-shelf (COTS) software components. This is especially true with respect to software systems being purchased by the United States federal government, most notably within the Department of Defense. Increasingly, new DoD procurement contracts are calling for mandated levels of COTS component use. In 1993, the Navy went so far as to establish a policy stating that the selection of a government in-house or procured software solution and not a COTS based solution was to be taken as a rejection of a comparable COTS solution. This shift in policy meant that Navy procurers now had to justify why they were *not* using COTS software.

The rationale for requiring COTS based systems is that they will involve less development time by taking advantage of existing, market proven, vendor supported products, thereby reducing overall system costs. But there is a trade-off in the COTS approach in that software development time can indeed be reduced but generally at the cost of an *increase* in software integration work. COTS software also brings with it a host of unique risks quite different from those associated with software developed in-house. Once again, the elusive software silver bullet remains just that, elusive. The use of COTS components in and of themselves will not slay the monster of upwardly spiraling software procurement costs. COTS components are not the Universal Solution.

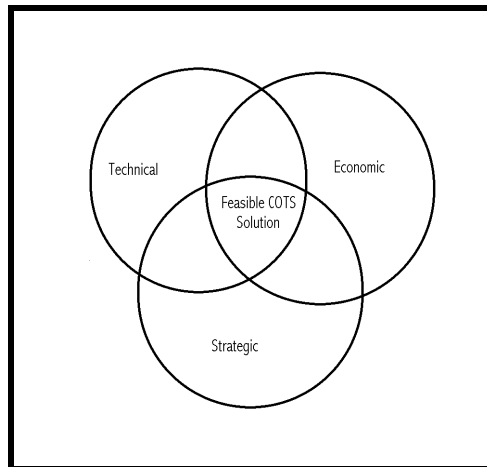


Figure 1- Considerations in evaluating the feasibility of COTS components.

However, under the correct conditions, they can still be the *right* solution, offering the most cost-effective, shortest schedule approach to assembling major software systems. They are the right solution when they lie at the intersection of the three determinants of feasibility: technical, economic, and strategic constraints. The key then to success in using COTS components is being able to identify whether they fit the current procurement situation, technically, economically, and strategically.

Technically, they have to be able to supply the desired functionality at the required level of reliability. Economically, they have to be able to be incorporated and maintained in the new system within the available budget and schedule. Strategically, they have to meet the needs of the system operating environment—which includes technical, political, and legal considerations—now, and as that environment is expected to evolve in the future.

Technical and strategic feasibility is determined during the candidate COTS products assessment phase, which occurs at the start of a COTS integration activity. How to determine the viability of a COTS product in either of these two dimensions is not a trivial question. Each of these dimensions is worthy of its own formal study, and as such generally remains outside the scope of the research detailed in this report.

It is the third dimension, determining economic feasibility, which provided the fundamental motivation for this COTS Integration Cost Modeling study. Also, this current study did not arise out of a vacuum. It was conducted as part of an overall effort currently underway at the University of Southern California to enhance the utility of the well-known COCOMO software cost estimation model first published by Dr. Barry Boehm in 1981². This broader effort is designed to update COCOMO to reflect how software development has evolved from the days when the model was first developed in the 1970s, to where modern software development practice is heading as the software industry moves into the 21st century. The release of the updated COCOMO II³ model this past year was the first major milestone in this effort. The completion of this current study and the prototype *COTS Integration Cost Calculator* which accompanies it is another.

The remainder of this report will discuss how the COTS cost model and calculator tool was developed. It will also discuss the model's relationship to the more general COCOMO software cost estimation model. (As such a familiarity in the reader with at least the basics of the COCOMO model is assumed.)

² B.W. Boehm, *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

³ B. W. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy and R. Selby, "Cost Models for Future Software Life Cycle Process: COCOMO 2.0," *Annals of Software Engineering Special Volume on Software Process and Product Management*, J. D. Arthur and S.M. Henry, Eds., J.C. Balter AG, Science Publishers, Amsterdam, The Netherlands, 1995, Vol. 1, pp.45-60.

IV. Scope of Study/Overview of Methodology

This study was performed over a sixteen-month period beginning in March of 1996. During that time the mandate was to develop a basic model form and accompanying prototype tool with the ultimate goal of being able to reasonably and consistently predict the cost of a given COTS software integration effort.

But COTS integration efforts are not all of a kind. COTS products can be used in essentially three ways: 1) as a component of a tool bed, 2) as a component of a system development infrastructure, and 3) as a component of a new application.

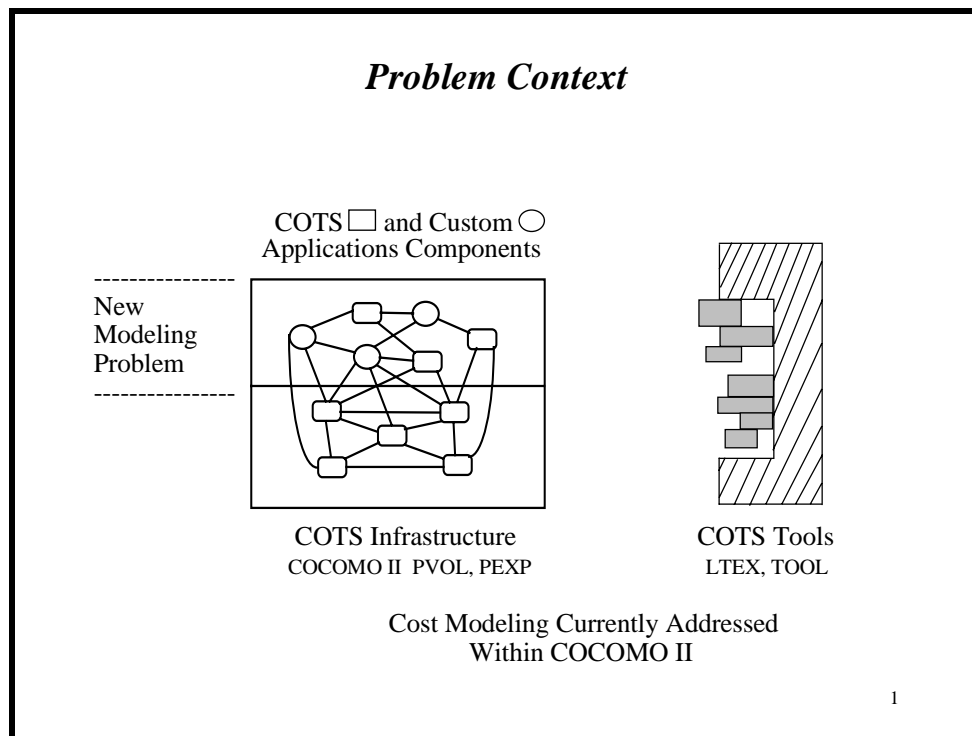


Figure 2 - COTS Integration task being addressed.

Currently the problem of COTS software being integrated as infrastructure or as part of a tool bed can be addressed within the COCOMO II model itself via the following drivers: Platform Volatility (PVOL) and developer Platform Experience (PEXP) for COTS as infrastructure; and Use of Software Tools (TOOL) and developer Language and Tool Experience (LTEX) for COTS as tools. The problem which remains currently *unaddressed* by COCOMO is that depicted in the upper half of Figure 2, in which COTS components are being integrated as part of an application. It is this problem which was addressed by the COTS integration model developed in this study.

The course of the study ran along traditional lines. It began with a general review of the available literature to learn what ideas others may have developed regarding this

COTS integration issue. The interesting result was the discovery that many people are talking about COTS, a number are even offering good ideas on how to approach the problem from a risk management perspective, but there is a dearth of information in the public domain on true empirical models, on the problem of actually predicting the cost of performing a COTS integration task. One important exception, however, is a COTS integration cost model developed by Loral Federal Systems, which offered a good jumping off point for the development of the USC COTS integration cost model.⁴ In addition to the general literature review, an examination was done on information related to COTS integration found in the Risk Repository of the Software Engineering Institute at Carnegie-Mellon University. Of particular help was information that was available in the form of lexical maps. These maps were derived from dozens of statements collected from software professionals around the country concerning the risks involved with COTS integration, visually emphasizing the relative importance of the key concerns relating to COTS integration which were to be found in the professionals' statements.

The information extracted from the repository became the first part of the kernel of ideas, which ultimately lead to the concepts captured in the cost drivers defined for the USC COTS integration model.

A synthesis of the ideas gleaned from the literature review, including the Loral model, the COCOMO model, and the SEI Risk Repository lead to the creation of a first round data collection survey. The approach to data collection in this study was two fold. The intention was to conduct a first round of data collection within industry with the goal of both prioritizing a candidate set of COTS integration effort influence factors, and also identifying potential sources of COTS integration experience. The first round survey was very successful with regard to the former, but only marginally so in regard to the latter. None-the-less, enough information was learned from the first survey--along with the considerable help of a panel of industry experts who examined that information--to refine those candidate effort influence factors into the cost drivers which appear in the version 1.0 of the USC model.

At this point a second round industry survey was drafted asking for specific project level information, including actual effort and sizing data of past COTS integration projects. In addition, the survey asks each cost driver defined for the model to be rated on a five point scale from very low through very high according to the development conditions that obtained for the COTS project being reported. This project specific data was needed to actually calibrate the parameter values of the model.

The second round data collection effort is still on-going at USC, but enough industry data was procured within the time frame of this study, combined with other sources of data, to attempt an experimental calibration of the USC COTS model. The other data used to help calibrate the model came from two sources. The first was sizing and effort data obtained from a set of recently completed student COTS integration

⁴ T. Ellis, "COTS Integration in Software Solutions—A Cost Model," in *Systems Engineering in the Global Marketplace*, NCOSE International Symposium, St. Louis, MO, July 24-26, 1995. (While the details of the Loral model, including its specific functional form, remain proprietary, the general parameters of the model have been published and will be discussed in Section VI. Another proprietary COTS integration model has been developed by Seer Technologies, Los Angeles, CA., but was unavailable for review during this study.)

projects. The second was a two round Delphi⁵ exercise conducted with the participation of several industry professionals with the intent of establishing preliminary parameter values for the model drivers.

Ultimately, it was in fact these latter two sources of data, which formed the basis for the current calibration offered in version 1.0 of the USC COTS modeling tool.

⁵ O. Helmer, *Social Technology*, Basic Books, NY, 1966. (The Delphi technique, named for the mystical oracle at Delphi in Greece, is a methodology for arriving at group consensus. It was originally developed at the RAND Corp.)

V. Modeling Background

A. Literature Review/Other Models

An examination of the public literature revealed an interesting phenomenon. There has been much discussion in recent years about the topic of COTS integration, but most of it has been restricted to framing the issue in qualitative terms. Several good offerings along these lines help to define key issues of concern pertaining to COTS integration, flagging risks in using COTS software and suggesting strategies for avoiding or mitigating those risks.

However, quantitative COTS integration models available for review in the public domain were almost non-existent. Of the few out there, one interesting model has been proposed by Dr. Richard Stutzke⁶ of SAIC. It is still bare bones, but it is centered on the issue of COTS volatility, that is, the frequency with which a COTS vendor releases new versions of its software. (This has been identified as one of two primary determinants in the cost of using COTS software, the other being the actual size of the interface or “glue” code needed to integrate a given COTS product.⁷) His model suggests a way of quantifying the added cost associated with using a COTS product that has a significant volatility.

In brief, Dr. Stutzke proposes the following formula:

$$\text{Extra Cost due to Volatility} = CV * AC * IS * (CS + CC)$$

where

CV = component volatility (number of new releases of the COTS component over life of the project).

AC = architectural coupling (number of other components which interface with the given COTS component).

IS = apparent interface size in terms of the number of entry points, procedures, functions or other methods used to access the COTS component, weighted by the number of arguments passed.

CS = cost of screening the COTS component and all the other components with which it interfaces to determine the impact of a new release.

CC = cost of making changes to impacted components.

As of this writing no attempt has yet been made to implement this model. It also addresses only one aspect associated with integrating COTS software, but it is an important aspect.

⁶ R. Stutzke, “Costs Impact of COTS Volatility,” *Knowledge Summary: Focused Workshop on COCOMO 2.0*, USC Center for Software Engineering, May 16-18, 1995.

⁷ See Ellis, footnote 4, p6.

* * *

Still another approach to modeling COTS integration costs has been taken at SAIC⁸. This second model addresses more the end user costs of using COTS software. The model takes this form:

$$\text{COTS integration cost} = [(\text{Cost of COTS product license}) * (\text{Number of licenses})] \\ + (\text{COTS product Training cost}) + (\text{COTS interface or glue code cost})$$

Again, this model highlights some important sources of cost, but ignores the details of determining the last term, the cost of developing the COTS product glue code.

* * *

An alternate model that it attempts to address this very issue of estimating the cost of developing COTS interface code has been described by Mr. Tim Ellis⁹ of Loral Federal Systems. Mentioned previously, this model is also well into the implementation stage, having been calibrated to a number of internal Loral COTS integration projects, with more continually being added to the model's calibration database. As of May 1995, an accuracy of plus or minus 15% was being claimed for its effort predictions against the Loral database.

Mr. Ellis describes the COTS integration model in these general terms:

$$\text{Work Units} = f_n (\text{Size, Drivers}) \quad (1)$$

$$\text{Productivity} = \text{Labor-months/Work Unit} \quad (2)$$

$$\text{Estimated Effort in LM} = \text{WU} * \text{P} \quad (3)$$

where

Size = the size of the COTS interface or glue code in function points.

Drivers = a set of seventeen COTS integration cost drivers.

LM = labor-months.

P = productivity.

WU = work units.

The greatest influence the Loral model had on the development of the USC COTS integration cost model came from its seventeen cost drivers, which served as one of the starting points in the definition of the USC COTS model drivers. As such, the Loral drivers are discussed more fully in section VI.B.

* * *

The COCOMO II model itself was examined as a potential modeling source. It too influenced the USC COTS model via its drivers, which are discussed in section VI.C. In the end COCOMO also provided the USC COTS model its basic form, primarily

⁸ M. Karpowich, T. Sanders and R. Verge, "An Economic Analysis Model for Determining the Custom vs. Commercial Software Tradeoff," in T. Gullledge and W. Hutzler, Analytical Methods in Software Engineering Economics, Springer-Verlag, 1993.

⁹ See Ellis, footnote 4, p6.

because this form is well understood at USC, both in terms of its behavior, and in the approach needed to calibrate such a model.

The post-architecture model of COCOMO II takes the following general form¹⁰:

$$PM = A * \{ [(Size) * (1 + Brak/100)]^{(1.01 + .01 \sum_{j=1}^5 SF_j)} \} * \prod_{i=1}^{17} EM_i + (\text{auto adaptation effort})$$

where

PM = person-months.

A = linear scaling constant.

Size = size of coding effort as a function of new and adapted code.

Brak = percentage of code discarded due to requirements volatility.

SF_j = five non-linear scaling factors.

EM_i = seventeen effort multipliers.

* * *

Before the option of a stand-alone model using the basic COCOMO form was decided upon (number 4 below), several approaches were considered for the USC COTS model in relation to the COCOMO II model:

- 1) Use the COCOMO II Reuse model.

The advantage here is that unmodified, so-called “black box” COTS software and reuse software could be handled by the COCOMO model in the same fashion. The disadvantage is that the total size of the COTS component is generally thought to be irrelevant, and the Assessment & Assimilation (AA) and Percentage of Integration & Test Modification (IM) COCOMO parameters are not adequate descriptors of the factors affecting COTS integration effort.

- 2) Include a COTS integration effort multiplier or exponent factor in COCOMO II.

The advantage here is that such parameters capture the fact that some COTS integration effects scale with the size of the overall system being developed. The disadvantage is that some COTS integration effects do *not* scale with the size of the overall system. Also, a single factor is again not really enough to capture the range of factors affecting a COTS integration effort.

¹⁰ See Boehm, et al, footnote 3, p.4.

- 3) Imbed a COTS integration effort submodel within COCOMO II and add its output to the other development effort estimated by COCOMO.

The advantage here is that such a submodel offers the flexibility to tailor portions of the COCOMO model to COTS integration phenomenology. The disadvantage is that COCOMO II is already fairly complex. Also, it may be hard to separate out COTS integration effort from other development effort.

- 4) Develop a stand-alone COTS integration estimation model and relate its estimate externally to COCOMO effort estimates.

The advantage here is that such a model is not bound by COCOMO II constraints. Also, data analysis is likely to be much cleaner, separating COTS integration effort from other development effort. The disadvantage is that the proper relation of such a model to COCOMO II is not necessarily obvious, particularly for projects incorporating a significant mix of new and COTS components.

These options and their pros and cons are summarized in table V.1 on the following page. As was indicated, approach 4 was the one finally selected, because a stand-alone model at this stage in USC's COTS integration modeling efforts seemed the simplest yet most complete approach. It also had the advantage of not perturbing the COCOMO model with elements that were not yet well understood. Addressing the proper way to relate such a model to COCOMO II is discussed in section X.

Finally, note that approach 3 has not been permanently ruled out. Now that an independent COTS model has been developed, future effort at USC will explore the feasibility of associating the COTS integration model more directly with COCOMO II.

Option	Pro	Con
<ul style="list-style-type: none"> • Use COCOMO II reuse model 	<ul style="list-style-type: none"> • Consistent: Black Box SW reuse handled same way 	<ul style="list-style-type: none"> • Total size may be irrelevant • Assessment & Assimilation (AA), Integration & Test Modification (IM) insufficient descriptors
<ul style="list-style-type: none"> • COTS-integration effort multiplier or exponent factor 	<ul style="list-style-type: none"> • Some COTS integration effects scale with size of product being developed 	<ul style="list-style-type: none"> • Many COTS integration effects do not: equally expensive for small or large self-originated SW • Single factor not enough
<ul style="list-style-type: none"> • COTS-integration effort added to other development effort 	<ul style="list-style-type: none"> • Flexibility to tailor portions of model to COTS integration phenomenology 	<ul style="list-style-type: none"> • COCOMO II getting pretty complex already • Data: may be hard to separate out COTS integration effort
<ul style="list-style-type: none"> • Standalone COTS-driven estimation model 	<ul style="list-style-type: none"> • Flexibility: no COCOMO II constraints • Ease of data analysis 	<ul style="list-style-type: none"> • Relation to COCOMO II unclear, particularly for mixed development and COTS integration projects

Table V.1 - COTS Integration modeling options in relation to COCOMO II.

B. Related Topic—Software Reuse

Closely related to the topic of COTS software is software that is being *reused*. In fact, it might be proper to class COTS software components as a subset—or at least an alternate set—of reuse software components. But reuse software differs from COTS software in three significant ways: 1) reuse components are not necessarily able to operate as stand-alone entities (as is assumed to be the case with most components defined as COTS software); 2) reuse software generally is acquired internally within the software developing organization (by definition COTS components *must* come from outside); and 3) reuse software usually requires access to the source code, whereas with COTS components access to the source code is rare (references to so-called “white box” COTS notwithstanding).

In light of the preceding, COTS software and reuse software share similar—but not identical—benefits and risk factors.

The advantages touted for reuse software are familiar ones:

- Reduced effort and development time, translating to reduced costs.
- Increased system quality.
- Added functionality not otherwise achievable.

The pitfalls associated with reuse software are also familiar:

- Lack of functionality (does less than advertised).
- Never does exactly what is needed.
- Unable to interoperate with other software components.
- Lack of support/documentation from originating developers.

However, pitfalls probably *not* associated often with reuse software that pertain to COTS software are the following:

- Component volatility/frequent product upgrades.
- licensing issues.

Bottom line, the sources of costs associated with reuse software are also similar to those of COTS software:

- Potential Reuse software components must be identified (parallels COTS component screening).
- The feasibility of reuse components in the context of the overall system must be determined (parallels COTS component assessment).
- The reuse component must be integrated and tested (parallels integration and test of COTS components).

C. COTS Related Definitions

There is controversy in defining exactly just what is meant by the term “COTS” software. Some software practitioners insist that the term must apply only to products which are truly used off-the-shelf “as is,” with no tailoring of the COTS product source code to the particular application into which it is to be integrated allowed. Others give a nod to the reality that as much as 30% of procured off-the-shelf products must be modified in some fashion before being suitable for use¹¹, and thus allow these modified products also to be referred to as COTS components. This study favors defining COTS components as those in which no source code is provided with the product (separate from any required API software). Commercial software components that include modifiable source code are considered to be reuse components. In either case, the following terms are frequently associated with COTS software:

API – application program interface.

Black Box COTS - internal code modifications not allowed.

COTS - commercial-off-the-shelf.

COTS Assessment/Qualifying - determining the feasibility of potential COTS components for the current application.

GFE - government furnished equipment.

GFS - government furnished software (see GOTS).

GOTS - government-off-the-shelf (see GFS).

MOTS - modified-off-the-shelf (see White Box COTS).

NDI – non-developmental item (not developed in-house).

NOTS - not-off-the-shelf.

OTS - off-the-shelf.

Reuse software - reusable software components built in-house, or obtained from outside, and for which the source code is available.

ROTS - research-off-the-shelf.

White Box COTS - some internal code modifications permitted (see MOTS).

¹¹ Mr. Marvin Carr, SEI.

VI. Derivation of CSE Proposed Influence Factors/Cost Drivers

The thirteen cost drivers which appear in the version 1.0 of the USC COTS integration cost model have been derived as the result of a synthesis of four major sources: the SEI Risk Repository, the Loral COTS integration cost model, COCOMO II, and the USC Round 1 COTS data collection survey. Over a two day period in November, 1996, an expert industry panel gathered as part of a conference on COTS integration sponsored by the USC Center for Software Engineering, reviewed the information available from these four sources. The panel then set about the task of reducing this information into a workable set of sufficient and reasonable drivers for the USC COTS model by prioritizing the various factors influencing COTS integration effort as indicated in the various sources. Once prioritized, the most important concepts were then combined and refined until the current set of thirteen drivers was established.

Sections VI. A through D discuss each source individually, with section VI.E describing how the synthesis was achieved.

A. SEI Risk Repository/Risk Taxonomy/Lexical Maps

The SEI Software Engineering Risk Repository (SERR) is an archive of statements that have been methodically collected from highly experienced software professionals that identify potential risks facing software development efforts. A subset numbering 77 of the SERR statements directly addresses risks related to COTS integration.

As an aid to analyzing the SERR statements, the SEI also has a tool that produces lexical mappings based upon the concepts found in the statements. These maps are able to provide quick and visual emphasis to the most important concepts captured in the SERR.

In the case of the COTS related statements, thirteen lexical maps were produced, one of which is reproduced in Figure 3. (The complete set of COTS related SERR statements and lexical maps is found in Appendix C.) The boxes in the figure represent key concepts. The lines between boxes represent associations of concepts with other concepts. The thicknesses of the lines represent the strength of the association between concepts, the thicker the line, the stronger the association. Strength in this case is defined as the relative percentage of occurrences of like pairs of concepts within the SERR statements. For example, if two concepts are paired within 50 out of the 77 SERR statements related to COTS integration, the relative percentage of occurrences of that pairing is $50/77$ equal to 65%, which is considered a strong association, and thus would be represented in the map by a thick line between those concepts. The maps thus provide qualitative flags to quantitative measures, allowing for quick divination of the most important concepts relating to the risks of COTS integration.

Two other figures related to the lexical maps have also been reproduced here. Figure 4 is called the Results Network. Each node in the figure represents one of the thirteen COTS based lexical maps. The arrows between some of the nodes indicate a two way occurrence of paired key words or concepts in the connected maps above a given frequency level.

Figure 5 is called the Results Distribution. The quantity appearing on the increasing Y axis called “coupling” represents the strength of internal connections appearing in the lexical maps. The quantity appearing on the increasing X axis called “cohesion” represents the strength of external connections between the maps. Thus, concepts appearing in maps falling in quadrant I of Figure 5 represent the best candidates for potential COTS integration cost factors.

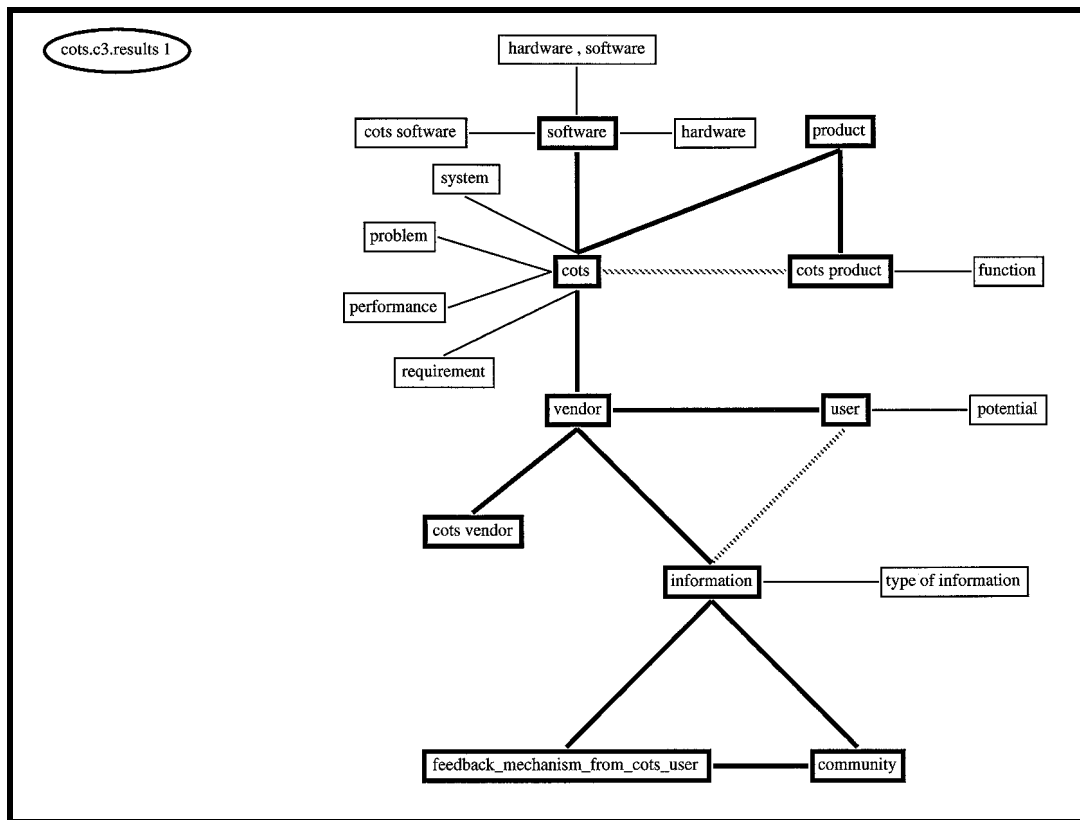


Figure 3 - SEI lexical map.

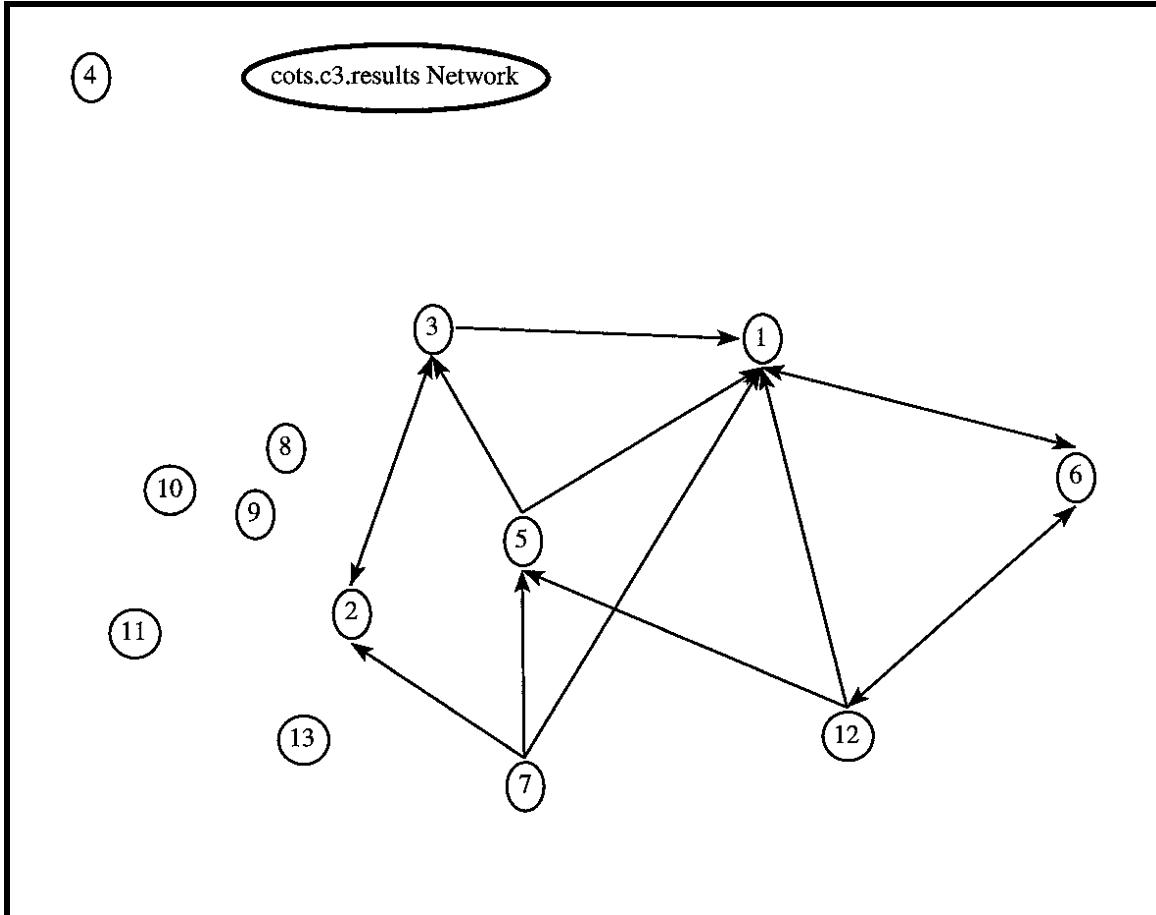


Figure 4 - Results Network.

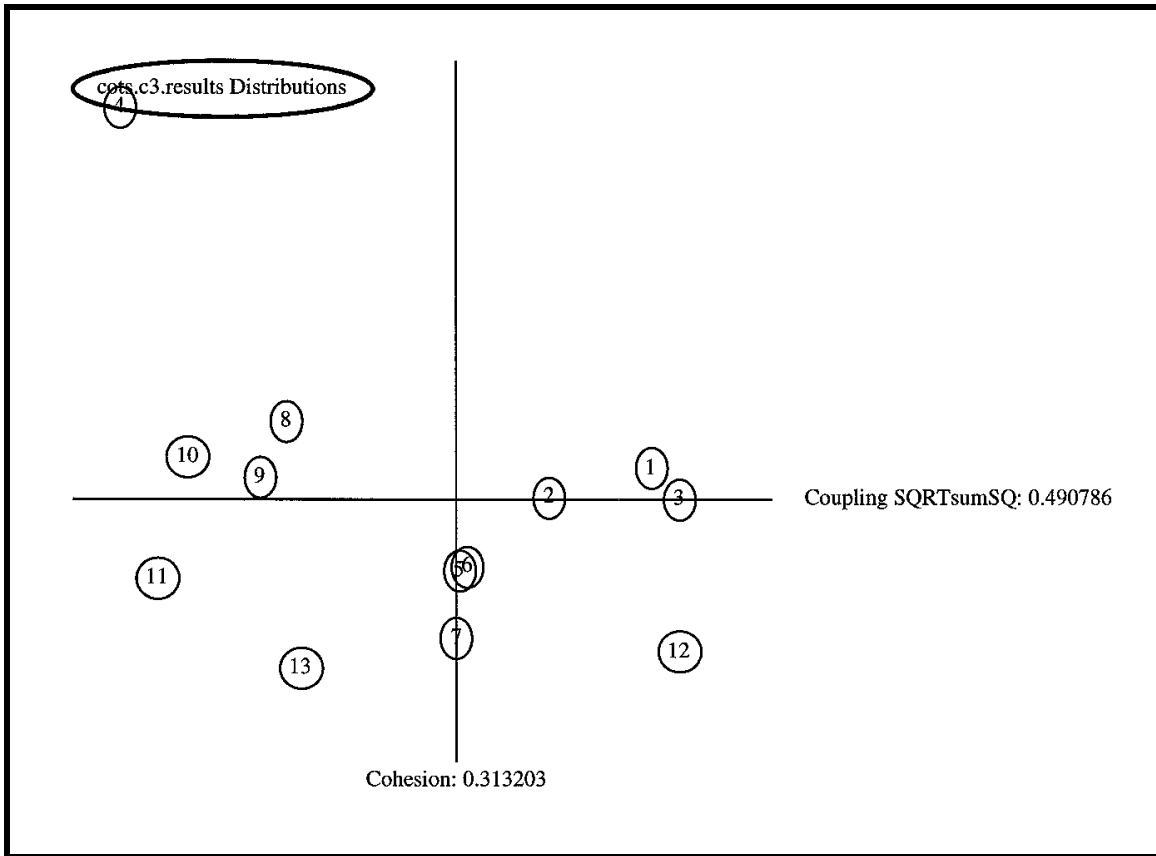


Figure 5 - Results Distribution.

Based upon an analysis of these maps and the associated SERR statements, the following items were identified as important COTS integration risk concerns:

- Documentation
- Performance
- Maturity
- Interface
- Verification
- Vendor Support/Upgrades
- Data Rights

These items represented the first set of potential COTS integration effort drivers.

B. Loral Model Drivers

The next source of potential drivers came from the Loral COTS integration model described in section V.A. The Loral drivers were examined in terms of the utility of their definitions and universality of their potential application. The more a given driver seemed to apply to COTS integration situations likely to be encountered by a broad base of developers, the more important the concepts being captured by that driver were potentially assumed to be.

The complete set of seventeen Loral drivers¹² is as follows:

- Product Maturity: Measures the length of time the product has been in the marketplace, existence of extensive alpha/beta testing programs, size of the market segment, number of bug fixes per release, and adherence to industry standards.
- Vendor Maturity: Measures the length of time the vendor has been in the business, vendor reputation, and size of product line.
- Configurability/Customization: Number of configuration options, and effort needed to customize the product.
- Installation Ease: Effort needed for product installation.
- Ease to Upgrade: Measures the level of difficulty to upgrade the COTS software from one release to the next and the impact to the applications being developed.
- Vendor Cooperation: Represents willingness of vendors to modify their product based on suggestions or enhancements recommended by the user. The more cooperative the vendor, the more functionality is provided thus reducing new development and glue code.
- Product Support Services: Types of services offered by the vendor to support the product (i.e., 24-hour hotline, seminars, trouble ticketing, etc.).
- Product Support Quality: Responsiveness of the vendor to answer user questions.
- User, Administrator, & Installation Documentation: Quality of documentation offered.
- Ease of Use for End User: How intuitive is the product for the end user.
- Ease of Use for Administrator: How intuitive is the product for the administrator.

¹² See Ellis, footnote 4, p6.

- End User & Administrative Training: Types and quality of training available.
- Administrative Effort: Amount of time spent by system administrator to regularly maintain the system.
- Portability: Portability of the product between platforms.
- Previous Product Experience: Amount of experience that personnel have had developing/using/integrating the product.
- Expected Release Frequency: Amount of time between product upgrades and releases. For every product upgrade, testing must be performed to ensure that no new incompatibilities have been introduced. This is a key cost driver that can adversely affect the integration phases of COTS products.
- Application or System COTS Package: Is the COTS software an application or system type of product.

C. COCOMO II Drivers

The cost drivers and scaling factors appearing in the post-architecture model of COCOMO II were also examined in terms of their potential applicability to COTS integration activities. Again, appropriateness of definition and broad-based applicability were the key items of concern.

The COCOMO II post-architecture model drivers¹³ are as follows:

Non-linear scale factors:

- Precedentedness: If the product is similar to several that have been developed before then the precededtedness is high.
- Development Flexibility: Captures the amount of constraints the product has to meet. The more flexible the requirements, schedules, interfaces, etc., the higher the rating.
- Architecture/Risk Resolution: Captures the thoroughness of definition and freedom from risk of the software architecture used for the product.

¹³ See Boehm, et al, footnote 3, p.4.

- Team Cohesion: Accounts for the sources of project turbulence and extra effort due to difficulties in synchronizing the project's stakeholders: users, customers, developers, maintainers, interfacers, others.
- Process Maturity: Based upon the SEI's Capability Maturity Model (CMM) ratings of organization-wide software development process maturity.

Linear Effort Multipliers:

Product Drivers

- Required Software Reliability: Measure of the extent to which the software must perform its intended function over a period of time.
- Database Size: Measure of the affect large data requirements has on product development.
- Required Reusability: Accounts for the additional effort needed to construct components intended for reuse on the current or future projects.
- Documentation Match to Life-cycle Needs: Measures the suitability of the project's documentation to its life-cycle needs.
- Product Complexity: Measures complexity of software under development in five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations.

Platform Drivers

- Execution Time Constraint: Measure of the execution time constraint imposed upon a software system.
- Main Storage Constraint: Measures the degree of main storage constraint imposed on a software system or subsystem.
- Platform Volatility: Measure of the degree of volatility/rate of change in the complex of hardware and software (operating system, DBMS, etc.) that the product under development calls upon to perform its tasks.

Personnel Drivers

- **Analyst Capability**: Analysts are personnel that work on requirements, high level design, and detailed design.
- **Programmer Capability**: Measure of the capability of the programmers as a team rather than as individuals, and considers ability, efficiency, thoroughness, and the ability to communicate and cooperate.
- **Applications Experience**: Measure of the project team's overall level of experience building the current type of product under development.
- **Platform Experience**: Measures the project team's experience with modern and powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities.
- **Language and Tool Experience**: Measure of the level of programming language and software tool experience of the project team.
- **Personnel Continuity**: Measure of the development project's annual personnel turnover rate.

Project Drivers

- **Use of Software Tools**: Measure of the extent advanced software development tools are used during development.
- **Multi-site Development**: Measure of the nature of project development site locations (from fully collocated to international distribution), and communication support between those sites (from surface mail and phone access to full interactive multimedia).
- **Required Development Schedule**: Measure of the schedule constraint imposed on the project; defined in terms of the percentage schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort.

D. Round 1 Survey Influence Factors

The final initial source of potential COTS model cost drivers came from the first round COTS data collection survey conducted as part of this study. This survey was distributed to an industry contact list supplied by the ESC, and asked participants to prioritize a set of twenty proposed COTS integration cost influence factors into three groups, from most significant, to intermediately significant, to least significant, with roughly an equal number of factors assigned to each category. The survey responses were then tabulated to determine the overall vote for what industry experts felt were the most, to least, significant (potential) factors affecting COTS integration costs. (A total of some 800 surveys were distributed, with a return rate of about 4.5%. A copy of the survey form can be found in Appendix D.)

The proposed factors as they were defined in the survey are provided below, followed by a discussion of the survey results:

The Vendor

- **Vendor Maturity:** How strongly is effort/productivity affected by how long the COTS product vendor has been in business? Are they a new start-up without a track record? Or have they been around awhile and established a reputation for quality, reliability and customer support?
- **Vendor Cooperation:** How strongly is effort/productivity affected by the extent to which the COTS product vendor provides technical, training and other support as needed specifically to help you incorporate their product into your system? Does the vendor offer a lot of assistance? Or no assistance?
- **Vendor Restrictions:** How strongly is effort/productivity affected by the extent to which the vendor demands special licensing, royalty or copyright arrangements for the use of its COTS product? Does the vendor impose significant restrictions? Or no restrictions?

The Developer

- **General COTS Software Integration Experience:** How strongly is effort/productivity affected by the extent of the experience the development staff has with incorporating COTS products into new systems? Have they done this kind of job before? Or have they no experience with this kind of job?

- Specific COTS Product Experience: How strongly is effort/productivity affected by the extent of the experience the development staff has with the particular COTS product (or products) being considered for incorporation into the new system? Do they have a lot of experience with the given COTS products? Or no experience with those products?

The User

- User Restrictions: How strongly is effort/productivity affected by the extent to which the user demands special licensing, royalty or copyright arrangements from the vendor to accept the use of a COTS product? Does the user impose significant restrictions? Or no restrictions?
- User COTS Product Experience: How strongly is effort/productivity affected by the extent to which the user has experience working with the particular COTS product (or products) being considered for incorporation into the new system? Does the user have a lot of experience with the given COTS products? Or no experience with those products?

The New System

- New System Complexity: How strongly is effort/productivity required to incorporate available COTS products into its design affected by the complexity of the new system under development? (For example, would a hard real-time system be more conducive to the use of COTS transaction processing software than an interactive query system?)

The COTS Software

- COTS Product Technical Complexity: How strongly is effort/productivity affected by the technical complexity of the COTS product (or products) selected for incorporation into a new larger system? Are the COTS products simple or complex?
- COTS Product Maturity: How strongly is effort/productivity affected by how long the COTS product has been available? How many copies have been sold? Has the product established a reputation for utility and reliability? Or have only a few copies been sold, leaving the product without a known track record?
- COTS Product Volatility: How strongly is effort/productivity affected by how often new releases of the COTS product are issued by the vendor? Does the product undergo frequent and significant updates? Or is it stable and remain relatively non-changing during the life of the larger system being developed?

- COTS Product Documentation: How strongly is effort/productivity affected by the extent to which the COTS product comes with the necessary documentation to install, maintain and use the product? Does the software come with extensive and well-written documentation? Or does it come with little documentation?
- COTS Product Vendor Support: How strongly is effort/productivity affected by the extent to which the vendor offers technical support for the COTS product? Does the vendor provide extensive support for its products? Or no support?
- COTS Product Ease of Installation: How strongly is effort/productivity affected by the ease or difficulty anticipated to install and integrate the COTS product? Are the interfaces required between the COTS product and the larger system simple or complex?
- COTS Product Ease of Maintenance or Upgrade: How strongly is effort/productivity affected by the ease or difficulty anticipated to maintain or upgrade the COTS product, particularly after it has been integrated into the larger system? Are upgrades to the COTS product simple to perform, or difficult?
- COTS Product Ease of Customization: How strongly is effort/productivity affected by the ease or difficulty anticipated to customize or modify the COTS product to make it suitable for use in the larger system if adaptation is necessary? Is customization simple, or difficult?
- COTS Product Portability: How strongly is effort/productivity affected by the portability of the COTS product across platforms? Is the product easily portable, or difficult to port?
- COTS Product Ease of Use: How strongly is effort/productivity affected by the ease or difficulty anticipated for the user to operate the COTS product, particularly after it has been integrated into the larger system? Is the product easy, or difficult to use?
- COTS Product Training: How strongly is effort/productivity affected by the extent of the training the user will require learning to operate the COTS product? Will the user need a lot of training, or little training?
- COTS Product Dedicated Database: How strongly is effort/productivity affected by the extent to which the COTS product has specialized data needs? Does the product require a specialized database? Or require the population of new elements within an existing database? Or are the product's specialized data needs minimal?

Round 1 Survey Results

800+ surveys were mailed, requesting the prioritization of twenty candidate cost drivers (top seven, middle seven, bottom six in influence on COTS integration cost). Thirty-six responses came back, giving about a 4.5% return, which is slightly better than typical for such mailings. Of those surveys returned, however, the responses were reasonably consistent:

Cost Influence Factor	Votes for Degree of Influence		
	Most	Intermediate	Least
COTS S/W Volatility	20	10	3
COTS S/W Technical Complexity	19	15	0
Vendor Cooperation	17	15	1
New System Complexity	17	10	3
COTS S/W Vendor Support	16	15	2
Vendor Maturity	12	16	6

Table VI.1 - Round 1 survey results indicating most influential COTS cost drivers.

Cost Influence Factor	Votes for Degree of Influence		
	Most	Intermediate	Least
COTS S/W Documentation	4	26	2
COTS S/W Ease of Installation	4	20	9
General COTS S/W Integration Experience	9	18	8
COTS S/W Ease of Maintenance or Upgrade	10	17	6
COTS S/W Training	4	17	13
Vendor Maturity	12	16	6
COTS S/W Ease of Use	9	16	7
Specific COTS Product Experience	12	15	7
COTS S/W Ease of Customization	9	14	9

Table VI.2 - Round 1 survey results indicating intermediately influential COTS cost drivers.

Cost Influence Factor	Votes for Degree of Influence		
	Most	Intermediate	Least
COTS S/W Portability	6	7	19
User Restrictions	6	9	18
Vendor Restrictions	6	10	17
COTS S/W Dedicated Database	6	11	15
User COTS Product Experience	5	13	15

Table VI.3- Round 1 survey results indicating least influential COTS cost drivers.

E. Expert Panel Input Leading to Synthesis of Items A through D

As part of a conference on COTS integration issues hosted by USC in November, 1996, a panel of experienced software professionals¹⁴ met over a two day period and examined the information presented in sections VI.A through VI.D. Out of discussion of this information, the panel identified five major sources of COTS integration effort (COTS assessment; COTS tailoring, tuning, and installation; COTS glue code development; and application volatility due to the presence of COTS products). The COCOMO and Loral drivers were then assessed in terms of their relation to these five sources of effort.

Next, from all the sources listed previously of potential COTS integration cost drivers, and while keeping the source of effort assessments just performed on the COCOMO and Loral drivers in mind, the panel mixed and matched cost drivers and their definitions, eliminating some, combining others, until a new set of potential cost drivers was created which represented a synthesis of the drivers from all the previous sources. Then again, the panel made an assessment of these new drivers relative to their impact on the five previously identified sources of COTS integration effort.

Finally, those drivers deemed to have the most impact across those five effort sources were selected as the set of drivers that would appear in version 1.0 of the USC COTS integration cost model.

The Five Identified Sources of COTS Integration Effort

- **COTS Assessment:** Refers to the activity required to determine which COTS components are viable candidates for integration., based upon the technical, economic, and strategic considerations discussed in section III.
- **COTS Tailoring, Tuning and Installation:** many COTS products can't be tailored at all (some would argue that a true COTS product can never be tailored), but problems with tailoring are often compensated for by adding more functionality in the glue code; however good documentation often makes tailoring, configuring, etc., much simpler.
- **COTS Glue Code Development:** This effort can be small or large, depending upon the cleanliness and openness of the COTS product external interface elements, and how much additional functionality must be added to the glue code for the reasons noted above under COTS tailoring. This effort is usually large, however, and in fact is almost invariably the source of greatest required effort during the integration task.

¹⁴ Panel members: Christopher Abts (USC), Barry Boehm (USC), Marvin Carr (SEI), Sunita Devnani (USC), Roger Dziegiel (Rome Laboratories), Gary Thomas (Raytheon E-Systems), and Peggy Wells (USAF/ESC).

- Application Volatility due to COTS Products: This can be a very large effort to manage and/or contain, because the developer rarely has much control over when and how often the COTS vendor releases new versions of its product. This can become particularly acute if the overall software development project encompasses a large system whose development is spread out over a significant period of time. During that same period, the COTS products vendors, in order to stay competitive within their markets, have likely released multiple updates of their own products, which by default can lead to significant volatility in the main application software as the developer struggles to keep in step with the COTS vendors.
- Added Application IV&V Effort: COTS products usually come with more functionality than is needed, but the developer dare not forego doing IV&V on the *entirety* of functionality offered by the COTS product to avoid unexpected interactions and problems.

Driver Assessments by The Five Sources of COTS Integration Effort

COCOMO II Cost Factor	Source of COTS Integration Effort				
	COTS Assessment	COTS Tailoring	Glue Code Development	Application Volatility	System IV&V
Reliability, Data, Complexity, Docum'n	+		++	+	++
Required Reuse			+		+
Platform Difficulty	+	+	++	+	+
Personnel Capability	++	+	++	++	++
Process (tools, sites, etc.)			+	+	+
Schedule	+		+		+
Architecture/Risk Resolution	+		++	++	++

Blank = minimal driver contribution to named source of effort; + = moderate contribution; ++ = strong contribution.

Table VI.4 -COCOMO II cost factors by source of COTS integration effort

Loral Cost Factor	Source of COTS Integration Effort				
	COTS Assessment	COTS Tailoring	Glue Code Development	Application Volatility	System IV&V
Product Maturity	+	+	++	++	+
Vendor Maturity	+		+	+	
Configurability		++	+		
Installation Ease		+			
Vendor Cooperation	+	+	++	+	++
Product Support Service	+		+		++
Product Support Quality	+		+		++
Documentation Quality		+	+	+	++
Ease of Use for End User			+	+	
Ease of Use for Administrator				+	+
Training			+	+	+
Administration Effort			+		
Portability		+			
Previous Product Experience		+			++

Blank = minimal driver contribution to named source of effort; + = moderate contribution; ++ = strong contribution.

Table VI.5 -Loral cost factors by source of COTS integration effort

Major Significance

Factor	COTS Ass.	COTS Tailor	Glue Code	App. Volatil.	Integ. V&V
COTS Product and Documentation Maturity	++	++	++	++	++
Vendor Extension Responsiveness	+	+	++	+	++
Integrator Experience with COTS Product	++	++	++	++	++
Reliability*	++		++	+	++
Complexity of COTS Product and Application	+		++	++	++
Integrator Personnel Capability	++	+	++	++	++
Integrator Architecture/Risk Resolution	+		++	++	++
COTS Compliance with Open Interface Standards	+	++	++	+	++
Performance*	++	+	++	++	++

* COTS reliability/performance relative to required system reliability/performance.

Two "++" indicates significant influence of the given factor under the given activity.

One "+" indicates moderate influence of the given factor under the given activity.

A blank indicates minor influence of the given factor under the given activity.

Table VI.6 - Revised candidate COTS integration cost drivers: perceived influence by activity.

Intermediate Significance

Factor	COTS Ass.	COTS Tailor	Glue Code	App. Volatil.	Integ. V&V
Integrator Experience with COTS Integration	+	+	+	+	+
Vendor Maturity and Product Support	+	+	+	+	++
Vendor Provided Training		+	+	+	+
Portability*	+	+	++		+

Minor Significance

Integrator Process Maturity			+	+	+
COTS Configurability, Customization, & Ease of Installation		++	+		

* COTS portability relative to required system portability.

Two "++" indicates significant influence of the given factor under the given activity.

One "+" indicates moderate influence of the given factor under the given activity.

A blank indicates minor influence of the given factor under the given activity.

Table VI.7 - Revised candidate COTS integration cost drivers: perceived influence by activity.

The final set of cost drivers chosen for version 1.0 of the USC COTS integration cost model were those factors determined to be of major and intermediate significance by the panel according to the assessments by effort source indicated in tables VI.6 and VI.7.

VII. Mathematical Model

A. COTS Integration Life-cycle

COTS integration activities follow their own unique life-cycle, adding the additional step of pre-qualifying or assessing COTS components to the traditional software development cycle of determine requirements, design, code, integrate, test, and deliver. Figure 6 shows this assessment activity occurring prior to the main project development phase, but in fact can occur during the opening stages of the project as well. The key concept is that sometimes system requirements dictate which COTS components can feasibly be used, and sometimes it is the availability (or lack there-of) of certain COTS components which determine the final system requirements.

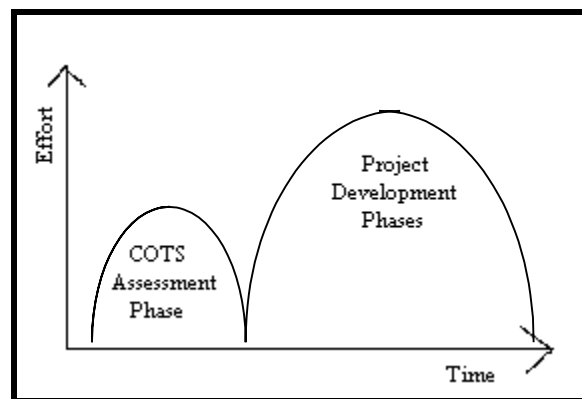


Figure 6 - COTS integration life-cycle.

B. COTS Integration Activity Phases Covered

The USC COTS integration cost model covers the following phases of COTS integration activity:

- Preliminary code design
- Detailed code design
- Code and Unit Test
- Integration and Test

Note the effort resulting from COTS assessment activities lies outside the scope of this model.

C. Equations

The USC COTS integration cost model version 1.0 takes the following form:

$$ESIZE = UFP * (1.0 + BRAK / 100)$$

$$PM = A * (ESIZE)^B * \prod_{i=1}^{13} (EM_i)$$

$$COST = (PM) * (\$/PM)$$

where

UFP = estimated sizing of the COTS glue code in Unadjusted Function Points.

BRAK = estimated percentage of glue code breakage during development. This is code that must be reworked due to changes in requirements or release of an updated COTS product.

ESIZE = effective size of the developed glue code.

A = a linear scaling constant calibrated to provide an accurate effort estimate when all effort multipliers are nominal.

B = a nonlinear scaling constant that accounts for the influence of factors that have exponential rather than multiplicative affects. This is temporarily set = 1 for modeling simplicity, with the expectation that later versions of the model will offer the opportunity to more precisely calibrate this parameter.

EM = the thirteen effort multipliers or cost drivers, each of which assumes one of five possible values based upon the following ratings: very low, low, nominal, high, and very high. Nominal ratings always have a multiplier value of 1.0. The other ratings typically have a multiplier value some small percentage above or below 1.0. Derived as explained in section VI.E, these cost drivers are defined explicitly in the following section VII.D.

PM = the estimated effort in person-months for the COTS integration task.
(Person-months is the preferred reporting unit, but as will be seen later on, for reasons of scale it became necessary to provide the effort estimate under the calibrated version 1.0 of this model in terms of person-hours.)

\$/PM = estimated average labor rate per person-month.

D. Definition of Terms/Cost Drivers

Thirteen Effort Multipliers (EM_i)

- **CPDM** - *COTS Product and Documentation Maturity*
- **CVEW** - *COTS Vendor Product Extension Willingness*
- **CIEP** - *COTS Integrator Experience with Product*
- **CREL** - *COTS Reliability*
- **CPAX** - *COTS Product and Application Complexity*
- **CIPC** - *COTS Integrator Personnel Capability*
- **CIAR** - *COTS Integrator Architecture/Risk Resolution*
- **CCOS** - *COTS Compliance with Open Interface Standards*
- **CPER** - *COTS Performance*
- **CIXI** - *COTS Integrator Experience with COTS Integration*
- **CVMS** - *COTS Vendor Maturity and Product Support*
- **CVPT** - *COTS Vendor Provided Training*
- **CPRT** - *COTS Portability*

Cost Driver Definitions

CPDM - COTS Product and Documentation Maturity: How many copies of the COTS product have been sold? How long has it been on the market? Has the product established a reputation for utility and reliability, i.e., a known track record? Does the product come with the necessary, well-written documentation to install, maintain, and use the package?

CVEW - COTS Vendor Product Extension Willingness: How willing is the vendor of the COTS product to modify the design of their software to meet your specific needs, either by adding or removing functionality or by changing the way it operates?

CIEP - COTS Integrator Experience with Product: How much experience does the development staff have with running, integrating, and maintaining the COTS product?

CREL - COTS Reliability: Does the COTS product meet or exceed the same standards of reliability as is required of the system as a whole into which the product is being integrated?

CPAX - COTS Product and Application Complexity: What kind of system are you building? Pushing software technology to state-of-the-art? Real time transaction monitoring, or basic file maintenance? Are there difficult synchronization issues? Does the system have to balance conflicting criteria (e.g., security, safety, accuracy, ease of use, speed)?

CIPC - COTS Integrator Personnel Capability: What are the overall software development skills and abilities that your personnel bring to the COTS product integration task?

CIAR - COTS Integrator Architecture/Risk Resolution: How much effort is expended by your integration staff in ensuring that potential risks to the COTS integration task are identified and mitigated, including through the examination of potential architectural mismatches between the COTS components and the overall system, or between the COTS components themselves? How thorough is the project's Software Architecture Review?

CCOS - COTS Compliance with Open Interface Standards: How well does the COTS product comply with accepted industry external and internal interface standards?

CPER - COTS Performance: How well does the COTS product meet or exceed the same standards of performance as is required of the system as a whole into which the product is being integrated?

CIXI - COTS Integrator Experience with COTS Integration: How much experience does the development staff have with assessing, integrating, and adapting to upgrades of COTS products in general?

CVMS - COTS Vendor Maturity and Product Support: How long has the vendor been in business? Are they a known quantity, or are they a new start-up? Have their products established a reputation for reliability? Even if they have been in business for awhile, how well do they provide technical support for their products (either directly or through third parties)?

CVPT - COTS Vendor Provided Training: How much training will the vendor provide (either directly or through third parties)?

CPRT - COTS Portability: How well does the COTS product meet or exceed the same standards of portability as is required of the system as a whole into which the product is being integrated?

Rating Scales

Each of the drivers needs to be rated on a scale ranging from Very Low to Very High as they apply to the circumstances of a given COTS component integration effort. The criteria for making these determinations is listed in the table below:

Driver	VL	L	N	H	VH
CPDM	Product in pre-release beta test.	Product on market less than 1 year.	Product on market between 1 and 2 years.	Product on market between 2 and 5 years.	Product on market more than 5 years.
CVEW	Vendor will not change the product.	Vendor will make minor changes only.	Vendor will make one major change or a few minor ones, but not both.	Vendor will make one or two major changes and any minor ones desired.	Vendor will change the product any way you desire essentially without restriction.
CIEP	Staff on average has no experience with the product.	Staff on average has less than 1 year's experience with the product.	Staff on average has between 1 and 2 years' experience with the product.	Staff on average has between 2 and 5 years' experience with the product.	Staff on average has more than 5 years' experience with the product.
CREL	Product does not meet system level reliability standards, mandating extensive added effort per line of glue code to compensate.	Product has some moderate reliability shortfalls, mandating moderate added effort per line of glue code to compensate.	Product meets system level reliability standards, but does not exceed them, requiring essentially no added effort per line of glue code to compensate.	Product moderately exceeds system level reliability standards, allowing moderate effort savings per line of glue code.	Product well exceeds system level reliability standards, allowing further effort savings per line of glue code.
CPAX	Product and/or system greatly lags mainstream levels of currently achievable modern design; i.e., this use of technology is "primitive."	Product and/or system moderately lags mainstream levels of currently achievable modern design; i.e., it is not innovative.	Product and/or system achieves mainstream levels of currently accepted modern design; i.e., it is up to date.	Product and/or system approaches or reaches state-of-the-art, without breaking new ground in software design.	Product and/or system pushes beyond state-of-the-art, breaking new ground in software design.

Driver	VL	L	N	H	VH
CIPC	Staff on average has well below average capability as compared to industry accepted standards for skill levels expected of personnel according to time on the job.	Staff on average has below average capability as compared to industry accepted standards for skill levels expected of personnel according to time on the job.	Staff on average has average capability as compared to industry accepted standards for skill levels expected of personnel according to time on the job.	Staff on average has above average capability as compared to industry accepted standards for skill levels expected of personnel according to time on the job.	Staff on average has well above average capability as compared to industry accepted standards for skill levels expected of personnel according to time on the job.
CIAR	No risk mitigation is done.	Little risk mitigation is done, with no addressing of architectural issues.	Standard risk mitigation/architectural assessment is performed.	Marginally more risk mitigation is done, with some assessment of architectural issues.	Extensive risk mitigation is done, with particular and special attention paid to architectural issues.
CCOS	Product uses non-standard, proprietary interfaces.	Product uses non-standard, non-proprietary interfaces.	Product uses a mix of standard and non-standard interfaces, some of which may be proprietary.	Product uses a mix of standard and non-standard interfaces, none of which are proprietary.	Product uses exclusively open industry standard interfaces.
CPER	Product does not meet system level performance standards, mandating extensive added effort per line of glue code to compensate.	Product has some moderate performance shortfalls, mandating some additional effort per line of glue code to compensate.	Product meets system level performance standards, but does not exceed them, requiring essentially no additional effort per line of glue code to compensate.	Product moderately exceeds system level performance standards, allowing some savings in effort per line of glue code.	Product well exceeds system level performance standards, allowing further effort savings per line of glue code.

Driver	VL	L	N	H	VH
CIXI	Staff on average has no experience with COTS integration.	Staff on average has less than 1 year's experience with COTS integration.	Staff on average has between 1 and 2 years' experience with COTS integration.	Staff on average has between 2 and 5 years' experience with COTS integration.	Staff on average has more than 5 years' experience with COTS integration.
CVMS	Vendor in business less than 6 months; weak product support.	Vendor in business between 6 months and 2 years; basic product support.	Vendor in business between 2 and 5 years; reasonable product support.	Vendor in business between 5 and 10 years; strong product support.	Vendor in business more than 10 years; excellent product support.
CVPT	Vendor provides no training.	Vendor provides roughly ¼ of the needed training.	Vendor provides roughly ½ of the needed training.	Vendor provides roughly ¾ of the needed training.	Vendor provides as much training as needed.
CPRT	Product does not meet system level portability standards, mandating extensive added effort per line of glue code to compensate.	Product has some moderate portability shortfalls, mandating moderate additional effort per line of glue code to compensate.	Product meets system level portability standards, but does not exceed them, requiring essentially no additional effort per line of glue code to compensate.	Product moderately exceeds system level portability standards, allowing some savings in effort per line of glue code.	Product well exceeds system level portability standards, allowing further effort savings per line of glue code.

Table VII.1 - Parameter rating criteria.

E. Delphi Experiment

The Delphi Technique¹⁵ is a means of guiding a group of informed individuals to a consensus of opinion on some issue. Participants are asked to make some assessment regarding an issue, individually in a preliminary round, without consulting the other participants in the exercise. The first round results are then collected, tabulated, and then returned to each participant for a second round, during which the participants are again asked to make an assessment regarding the same issue, but this time with knowledge of what the other participants did in the first round. The second round usually results in a narrowing of the range in assessments by the group, pointing to some reasonable middle ground regarding the issue of concern.

This is a useful technique for coming to some conclusion regarding an issue when the only information available is based more on “expert opinion” than hard empirical data.

As applied in this study, it was used as means of obtaining consensus on what might be reasonable initial values for the thirteen effort multiplier parameters. These Delphi derived parameter values were then used as a starting point for the model calibration activity described in section VIII.

Specifically, for each of the thirteen parameters, the participants were asked to provide a *productivity range* (PR) value. (The actual Delphi instrument used to conduct the exercise can be found in Appendix D.) This PR value represents the greatest range in impact a given driver might be reasonably expected to have on overall integration effort between the driver’s most favorable and least favorable settings. For example, one driver might cause a 150% increase in effort between its most and least favorable settings, while another driver might cause only a 75% increase in effort between its most and least favorable settings. (PR values of 150% and 75% would be indicated by values of 2.5 and 1.75 respectively in the participant responses.)

The final PR values resulting from the concluded Delphi exercise were then plugged into a formula as described in section VII.F to derive the initial set of parameter values used to begin the model calibration.

The results of the Delphi experiment are shown in tables VII.2 and VII.3 on the following pages.

¹⁵ See Helmer, footnote 5, p.7.

Round 1 COTS Delphi Analysis

		PR - Productivity Range												
		initial	resp. 1	resp. 2	resp. 3	resp. 4	resp. 5	resp. 6	resp. 7	mean	median	mode	range	
1	CPDM COTS Product and Documentation Maturity	1.83	2.20	2.50	N/C	N/C	1.79	10.00	2.00	3.00	1.92	1.83	1.79 - 10.00	
2	CVEW COTS Vendor Product Extension Willingness	1.61	1.50	1.28	1.80	2.00	1.76	0.00	1.50	1.43	1.56	1.50	0 - 2.00	
3	CIEP COTS Integrator Experience with Product	1.77	N/C	1.50	1.50	2.66	1.64	3.00	2.90	2.09	1.77	1.50&1.77	1.50 - 3.00	
4	CREL COTS Reliability	1.56	2.00	1.67	N/C	1.20	1.40	3.00	N/C	1.74	1.56	1.56	1.20 - 3.00	
5	CPAX COTS Product and Application Complexity	1.77	2.50	2.25	N/C	2.66	N/A*	10.00	1.80	2.84	2.03	1.77	0 - 10.00	
6	CIPC COTS Integrator Personnel Capability	2.78	N/C	2.13	N/C	3.33	1.97	1.38	1.80	2.37	2.46	2.78	1.38 - 3.33	
7	CIAR COTS Integrator Architecture/Risk Resolution	2.04	2.20	2.46	N/C	2.00	2.11	N/C	2.00	2.11	2.04	2.04	2.00 - 2.46	
8	CCOS COTS Compliance with Interface Standards	1.39	N/C	1.39++	1.50	1.20	N/C	N/C	1.40	1.38	1.39	1.39	1.20 - 1.50	
9	CPER COTS Performance	1.49	1.75	1.65	1.20	1.20	1.52	N/C	1.50	1.48	1.50	1.20&1.49	1.20 - 1.75	
10	CIXI COTS Integrator Experience with COTS Integration	1.49	1.35	1.33	N/C	3.33	1.39	N/C	1.70	1.70	1.49	1.49	1.33 - 3.33	
11	CVMS COTS Vendor Maturity and Product Support	1.69	2.25	N/A	N/C	1.33	1.94	N/C	1.70	1.54	1.69	1.69	0 - 2.25	
12	CVPT COTS Vendor Provided Training	1.39	1.60	N/C	N/C	1.85	1.42	N/C	1.40	1.48	1.40	1.39	1.39 - 1.85	
13	CPER COTS Portability	1.29	1.20	1.43	N/C	1.21	1.41	N/C	1.50	1.33	1.29	1.29	1.20 - 1.43	

Table VII.2 - Round 1 Delphi Analysis.

Key:

N/C - no change

N/A - not applicable (covered by other factors)

N/A* - felt proper focus of question was missed

++ - within the given range if there is information on this ahead of time;
potentially exceedingly large if an unknown quantity until integration is underway.

(In the analysis, N/A answers were assigned a value of zero for purposes of averaging and finding the range.)

Round 2 COTS Delphi Analysis

			PR - Productivity Range (Revised Values)											
			(initial)								mean	median	mode	range
			resp. 0	resp. 1	resp. 2	resp. 3	resp. 4	resp. 5	resp. 6	resp. 7				
1	CPDM	COTS Product and Documentation Maturity	1.83	2.25	2.50	3.00	3.00	2.00	1.83	1.90	2.29	2.13	1.83 & 3.00	1.83 - 3.00
2	CVEW	COTS Vendor Product Extension Willingness	1.61	1.50	1.28	1.80	1.80	1.50	N/A	1.50	1.37	1.50	1.50	0 - 1.80
3	CIEP	COTS Integrator Experience with Product	1.77	1.90	1.75	2.00	2.66	1.75	1.77	1.80	1.93	1.79	1.75 & 1.77	1.75 - 2.66
4	CREL	COTS Reliability	1.56	2.00	1.67	1.56	1.40	1.40	2.00	1.60	1.65	1.58	1.40 & 1.56 & 2.00	1.40 - 2.00
5	CPAX	COTS Product and Application Complexity	2.04	2.35	2.38	2.00	4.00	1.5*	2.00	2.00	2.28	2.02	2.00	1.50 - 4.00
6	CIPC	COTS Integrator Personnel Capability	2.78	2.55	2.13	2.13	3.00	2.00	3.00	2.50	2.51	2.53	2.13 & 3.00	2.00 - 3.00
7	CIAR	COTS Integrator Architecture/Risk Resolution	2.04	2.10	2.14	2.30	2.00	2.10	2.00	2.10	2.10	2.10	2.10	2.00 - 2.30
8	CCOS	COTS Compliance with Interface Standards	1.39	1.40	N/C	N/C	1.40	N/C	1.40	1.40	1.40	1.40	1.39 & 1.40	1.39 - 1.40
9	CPER	COTS Performance	1.49	1.65	1.65	1.20	1.20	1.50	1.50	1.50	1.46	1.50	1.50	1.20 - 1.65
10	CIXI	COTS Integrator Experience with COTS Integration	1.49	1.55	1.65	N/C	3.00	1.50	2.00	1.50	1.77	1.53	1.49 & 1.50	1.49 - 3.00
11	CVMS	COTS Vendor Maturity and Product Support	1.69	1.85	1.53	1.53	2.00	2.00	1.50	1.70	1.73	1.70	1.53 & 2.00	1.50 - 2.00
12	CVPT	COTS Vendor Provided Training	1.39	1.45	N/C	1.50	1.75	1.50	1.40	1.40	1.47	1.43	1.39 & 1.40 & 1.50	1.39 - 1.75
13	CPER	COTS Portability	1.29	1.25	1.43	1.20	1.21	1.40	1.30	1.30	1.30	1.30	1.30	1.20 - 1.43

Table VII.3 - Round 2 Delphi Analysis.

Key:

N/C - no change

N/A - felt not applicable by definition (*"Once the vendor makes changes it's no longer COTS."*)

* - still felt proper focus of question was missed

(In the analysis, N/A answers were assigned a value of zero for purposes of averaging and finding the range.)

F. Initial Delphi Derived Multiplier Values

(Note: the values in table VII.4 on the following page were used to *begin* the calibration effort. They are NOT the calibrated parameter values appearing in the final version 1.0 of the model.)

Associated with each of the five possible ratings for all the cost drivers are numerical values that serve as the multiplicative adjustment factors away from the starting nominal effort in the effort equation. These are the model parameter values upon which regressions are performed and which ideally individual organizations should tailor to their own practices.

But the model must have some initial parameter values to serve as a starting point. Table VII.4 shows the initial parameter values used in calibrating the USC COTS integration cost model version 1.0. They were derived using the Delphi exercise round 2 median values for PR found in table VII.3.

The values in the table were obtained using the following procedure¹⁶:

For all drivers except #5 (CPAX), the Very Low parameter value is defined to be greater than the Very High parameter value, and $PR = VL/VH$. The individual parameter values are then found with these formulae:

$$VL = 1+2X \quad L = 1+X \quad N = 1 \quad H = 1-X \quad VH = 1-2X$$

$$\text{where} \quad X = [(PR_{\text{median}} - 1)/(PR_{\text{median}} + 1)] * \frac{1}{2} .$$

For driver #5 (CPAX), the Very Low parameter value is defined to be less than the Very High parameter value, and $PR = VH/VL$. The individual parameter values are then found with these formulae:

$$VL = 1+2X \quad L = 1+X \quad N = 1 \quad H = 1-X \quad VH = 1-2X$$

$$\text{where} \quad X = [(1 - PR_{\text{median}})/(1 + PR_{\text{median}})] * \frac{1}{2} .$$

¹⁶ The procedure described ensures that the parameter values are linearly proportional along the ratings for a given cost driver. This was done to simplify this first modeling effort. The drawback to this approach, however, is that the relative differences in the absolute person-month values produced by the model will *not* linearly reflect the implied productivity range. A better approach that avoids this problem is to make the parameter values geometrically proportional along the rating scale.

Driver	VL	L	N	H	VH
1) CPDM	1.36	1.18	1.00	0.82	0.64
2) CVEW	1.20	1.10	1.00	0.90	0.80
3) CIEP	1.28	1.14	1.00	0.86	0.72
4) CREL	1.22	1.11	1.00	0.89	0.77
5) CPAX	0.66	0.83	1.00	1.17	1.34
6) CIPC	1.43	1.22	1.00	0.78	0.57
7) CIAR	1.36	1.18	1.00	0.82	0.65
8) CCOS	1.17	1.08	1.00	0.92	0.83
9) CPER	1.20	1.10	1.00	0.90	0.80
10) CIXI	1.21	1.11	1.00	0.90	0.79
11) CVMS	1.26	1.13	1.00	0.87	0.74
12) CVPT	1.18	1.09	1.00	0.91	0.82
13) CPRT	1.13	1.07	1.00	0.94	0.87

Table VII.4 - Initial Delphi derived parameter values.

VIII. Calibration Efforts/Results

A. Round 2 Survey

The second round COTS data collection survey was a detailed survey designed to capture project level data which could then be used to calibrate the COTS integration cost model as described previously in sections VI and VII. (A copy of the round 2 survey form can be found in Appendix D.)

The survey consists of seven sections. Section 1 contains general background information regarding this research effort. Section 2 provides definitions of terms. Section 3 asks for identifying information from the survey respondent. Section 4 asks for project level technical information such as project domain, current project phase, development type, overall effort, schedule, sizing, languages, etc. Section 5 asks for similar technical information but at the COTS component level. Section 6 asks the respondent to evaluate the development conditions obtaining during the COTS integration activities according to the model cost drivers and driver rating criteria described in section VII.D of this report. Finally, section 7 provides some follow-up information.

By the conclusion of the study period, the round 2 survey had been used to gather data from eighteen projects, six projects from a graduate level software engineering design class, and twelve projects from an industrial source. Separate model calibrations were then attempted for each of the two sets of data. The results of those calibration efforts are described in the following two sections of this report.

B. Student Projects

The student projects represented a controlled sample data set from a recent USC graduate software engineering class that involved multimedia software projects that used COTS components as part of the design. The value of these projects was three-fold: 1) real-time effort data could be retrieved weekly as development was in progress, avoiding the problem of having to estimate or reconstruct effort data after the fact, which is often the case in industrial settings; 2) the projects had a fixed, known schedule for completion (one academic semester); and 3) effort data could be quickly paired with final actual development size. Accurate figures for all three elements (effort, schedule, and size) are crucial to any calibration effort. Being able to control the data being collected along these lines made these projects ideal for attempting an initial COTS integration cost model test calibration and validation.

Results from the student data collection are summarized in the tables on the following pages.

ACTIVITY	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Total Pers-hrs	% Total Pers-hrs
	Edgar Cpr	Med. Mscrp	Tech Rpts	LAPIS	CNTV Archv	Hancock PH	by Activity	by Activity
General Activity								
Determine Requirements:	16.00	49.50	86.50	26.50	5.50	38.50	222.50	4.99
Prepare, update plans :	107.00	142.00	209.50	39.00	83.50	134.75	715.75	16.06
Design product :	99.00	3.00	103.50	63.50	13.00	96.00	378.00	8.48
Code product :	161.00	20.50	190.00	168.00	67.50	115.00	722.00	16.20
Participate in formal design/code reviews:	14.00	8.00	21.00	21.00	22.50	24.00	110.50	2.48
Integrate and test :	70.00	94.50	85.50	6.50	13.00	29.50	299.00	6.71
Fix defects found in testing:	60.00	27.50	61.00	2.00	15.00	71.00	236.50	5.31
COTS Related Activity								
Understand and qualify COTS:	2.00	6.00	98.50	10.00	61.00	19.50	197.00	4.42
Design COTS glue code :	0.00	0.00	7.50	0.00	0.30	9.00	16.80	0.38
Code COTS glue code :	0.00	0.00	4.00	0.00	16.80	30.50	51.30	1.15
Fix defects found in COTS testing:	5.00	0.00	2.50	1.00	1.50	4.00	14.00	0.31
Administrative Activity								
Management:	8.50	34.00	33.50	13.50	10.00	25.00	124.50	2.79
Documentation :	52.50	449.50	38.00	59.50	68.00	126.00	793.50	17.81
Other:	114.00	239.00	31.50	8.00	100.00	82.50	575.00	12.90
TOTAL WEEKLY Person-Hours	709.00	1073.50	972.50	418.50	477.60	805.25	4456.35	99.99

Table VIII.1- Effort hours by activity for graduate software engineering class projects incorporating COTS products.

Key:Group 1 - EDGAR Corporate Data
Group 2 - Medieval Manuscripts
Group 3 - Technical Reports
Group 4 - Latin American Pamphlets
Group 5 - CNTV Moving Image Archive
Group 6 - Hancock Photo Archive

ACTIVITY	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
	Edgar Cpr	Med. Mscrp	Tech Rpts	LAPIS	CNTV Archv	Hancock PH
General Activity						
Determine Requirements:	2.3	4.6	8.9	6.3	1.2	4.8
Prepare, update plans :	15.1	13.2	21.5	9.3	17.5	16.7
Design product :	14.0	0.3	10.6	15.1	2.7	11.9
Code product :	22.7	1.9	19.5	40.1	14.1	14.3
Participate in formal design/code reviews:	2.0	0.7	2.2	5.0	4.7	3.0
Integrate and test :	9.9	8.8	8.8	1.6	2.7	3.7
Fix defects found in testing:	8.5	2.6	6.3	0.5	3.1	8.8
COTS Related Activity						
Understand and qualify COTS:	0.3	0.6	10.1	2.4	12.8	2.4
Design COTS glue code :	0.0	0.0	0.8	0.0	0.1	2.4
Code COTS glue code :	0.0	0.0	0.4	0.0	3.5	3.8
Fix defects found in COTS testing:	0.7	0.0	0.3	0.2	0.3	0.5
Administrative Activity						
Management:	1.2	3.2	3.4	3.2	2.1	3.1
Documentation :	7.4	41.9	3.9	14.2	14.2	15.6
Other:	16.1	22.6	3.2	1.9	20.9	10.2
TOTAL WEEKLY Person-Hours	100.2	100.4	99.9	99.8	99.9	101.2

Table VIII.2- Percentage of effort by activity for graduate software engineering class projects incorporating COTS products.

Project	CPDM	CVEW	CIEP	CREL	CPAX	CIPC	CIAR	CCOS	CPER	CIXI	CVMS	CVPT	CPRT
1	N	N	VL	N	N	N	N	N	H	L	N	VL	H
3A	N	H	L	N	H	H	H	VH	N	N	N	VL	H
4	H	VL	H	N	N	H	H	H	H	H	N	L	VH
5	N	VL	VL	L	N	H	L	L	N	L	VL	VL	VH
6A	L	L	L	N	H	VL	VL	H	H	VL	VL	VL	VH
6B	L	VL	VL	H	H	VL	VL	H	H	VL	VL	VL	H

Table VIII.3- COTS model effort driver ratings as reported for each student project.

Project*	COTS Glue Code			
	UFP	BRAK	ESIZE	ACT_PH
1	10	0%	10	7.00
3A	12	0%	12	61.36
4	10	0%	10	11.00
5	10	0%	10	79.60
6A	3	0%	3	14.54
6B	10	0%	10	48.46

*A and B signify more than one COTS integration activity per overall project. Project 2 deleted because COTS product was removed from final design.

Table VIII.4- Student project COTS integration sizing and effort data.

Table VIII.4 shows the sizing of the student project COTS glue code in unadjusted function points. For these projects the effective glue code size was the same because no breakage estimate was reported. The actual person-hours associated with the integration effort are also reported. It is against these values that the validation of the model calibration was performed.

Calibration Efforts

Seven cases were run on the student project data in order to arrive at a set of calibration values for this first pass at the COTS integration cost model that provide a reasonable effort estimate when compared to the actual effort reported for each project.

The cases were as follows:

- 1) Initial provisional model parameter values derived from the Delphi exercise as shown in table VII.4.
- 2) 1st revision values based upon regression on a natural log transformation¹⁷ of the parameters and data in case 1.
- 3) 2nd revision values based upon a blending of the initial provisional values in case 1 and 5% of the difference between the initial values and the values derived via regression in case 2¹⁸.
- 4) 3rd revision values based upon the engineering judgment of Dr. Barry Boehm as manifest in his round 2 values for the Delphi exercise.
- 5) 4th revision values based upon the case 4 values but with the linear scaling constant “A” reduced from 2.00 to 1.00.
- 6) 5th revision values based upon the case 3 values but with the linear scaling constant “A” reduced from 2.00 to 1.00.
- 7) 6th revision values based upon the case 4 values but with the linear scaling constant “A” set to 1.20.

The parameter sets derived for the cases described above appear on the following pages.

¹⁷ The basic COTS model takes the form $PM = A * (SIZE)^B * EM_1 * EM_2 * EM_3 * \dots * EM_{n-1} * EM_n$. The log transform of this is then $\ln(PM) = B * \ln(SIZE) + \ln(EM_1) + \ln(EM_2) + \ln(EM_3) + \dots + \ln(EM_{n-1}) + \ln(EM_n)$. Regression on the latter formula will produce an intercept term B_0 , coefficients B_x for each EM term, and a residual for each observation. Then $EM_{revised} = (EM_{initial})^B x$.

¹⁸ $EM_{revised} = \{[(EM_{initial})^B x - EM_{initial}] * (.05)\} + EM_{initial}$.

Student Data

Parameter	VL	L	N	H	VH	PR ¹
1 CPDM	1.36	1.18	1.00	0.82	0.64	2.13
2 CVEW	1.20	1.10	1.00	0.90	0.80	1.50
3 CIEP	1.28	1.14	1.00	0.86	0.72	1.79
4 CREL	1.22	1.11	1.00	0.89	0.77	1.58
5 CPAX	0.66	0.83	1.00	1.17	1.34	2.02
6 CIPC	1.43	1.22	1.00	0.78	0.57	2.53
7 CIAR	1.36	1.18	1.00	0.82	0.65	2.10
8 CCOS	1.17	1.08	1.00	0.92	0.83	1.40
9 CPER	1.20	1.10	1.00	0.90	0.80	1.50
10 CIXI	1.21	1.11	1.00	0.90	0.79	1.53
11 CVMS	1.26	1.13	1.00	0.87	0.74	1.70
12 CVPT	1.18	1.09	1.00	0.91	0.82	1.43
13 CPRT	1.13	1.07	1.00	0.94	0.87	1.30

Table VIII.5 - Initial provisional parameter values (Case 1).

Parameter	VL	L	N	H	VH	PR ¹
1 CPDM	3.74	2.03	1.00	0.43	0.15	24.93
2 CVEW	1.14	1.07	1.00	0.93	0.85	1.34
3 CIEP	1.80	1.37	1.00	0.70	0.45	3.97
4 CREL	1.38	1.19	1.00	0.83	0.66	2.08
5 CPAX	0.08	0.31	1.00	2.65	6.17	77.13
6 CIPC	1.06	1.03	1.00	0.96	0.92	1.15
7 CIAR	1.10	1.05	1.00	0.94	0.87	1.26
* 8 CCOS	0.93	0.96	1.00	1.04	1.09	1.17
9 CPER	12.22	3.70	1.00	0.24	0.05	244.40
10 CIXI	2.04	1.45	1.00	0.66	0.41	4.98
11 CVMS	4.41	2.19	1.00	0.41	0.14	31.50
12 CVPT	10.54	3.41	1.00	0.26	0.06	175.67
* 13 CPRT	0.77	0.87	1.00	1.15	1.35	1.75

* This change in parameters is counter-intuitive.

Table VIII.6 - First recalibration of parameter values (Case 2).

¹ PR (Productivity Ratio) = VL/VH except for parameter 5 (CPAX) where PR = VH/VL.

Student Data

Parameter	VL	L	N	H	VH	PR ¹
1 CPDM	1.48	1.22	1.00	0.80	0.61	2.43
2 CVEW	1.20	1.10	1.00	0.90	0.80	1.50
3 CIEP	1.31	1.15	1.00	0.85	0.70	1.87
4 CREL	1.23	1.12	1.00	0.89	0.77	1.60
5 CPAX	0.63	0.80	1.00	1.24	1.58	2.51
6 CIPC	1.42	1.21	1.00	0.79	0.58	2.45
7 CIAR	1.34	1.17	1.00	0.83	0.66	2.03
* 8 CCOS	<i>1.17</i>	<i>1.08</i>	<i>1.00</i>	<i>0.92</i>	<i>83.00</i>	<i>1.41</i>
9 CPER	1.75	1.23	1.00	0.87	0.76	2.30
10 CIXI	1.25	1.12	1.00	0.92	0.77	1.62
11 CVMS	1.42	1.18	1.00	0.85	0.71	2.00
12 CVPT	1.64	1.20	1.00	0.88	0.79	2.08
* 13 CPRT	<i>1.13</i>	<i>1.06</i>	<i>1.00</i>	<i>0.94</i>	<i>0.87</i>	<i>1.30</i>

* Initial provisional value restored as change in first revision was counter-intuitive.

Table VIII.7 - Second recalibration of parameter values (Cases 3 and 6).

Parameter	VL	L	N	H	VH	PR ¹
1 CPDM	1.30	1.15	1.00	0.85	0.70	1.86
2 CVEW	1.24	1.12	1.00	0.88	0.76	1.63
3 CIEP	1.28	1.14	1.00	0.86	0.72	1.78
4 CREL	1.22	1.11	1.00	0.89	0.78	1.56
5 CPAX	0.66	0.83	1.00	1.17	1.34	2.03
6 CIPC	1.48	1.24	1.00	0.76	0.52	2.85
7 CIAR	1.34	1.17	1.00	0.83	0.66	2.03
8 CCOS	1.16	1.08	1.00	0.92	0.84	1.38
9 CPER	1.20	1.10	1.00	0.90	0.80	1.50
10 CIXI	1.20	1.10	1.00	0.90	0.80	1.50
11 CVMS	1.26	1.13	1.00	0.87	0.74	1.70
12 CVPT	1.16	1.08	1.00	0.92	0.84	1.38
13 CPRT	1.12	1.06	1.00	0.94	0.88	1.27

Table VIII.8 - Third recalibration of parameter values (Cases 4, 5 and 7).

Student Data

Project	Actual Effort	Estimated Effort (Prs-hrs)						
		Initial Parm	1st Rev.	2nd Rev.	3rd Rev.	4th Rev.	5th Rev.	6th Rev.
1	7.00	28.4	151.80	39.40	27.60	13.80	19.70	16.56
3A	61.36	17.04	966.00	25.92	16.32	8.16	12.96	9.79
4	11.00	7.60	4.60	8.20	8.00	4.00	4.10	4.80
5	79.60	48.60	4300.00	79.80	47.80	23.90	39.90	28.68
6A	14.54	25.98	1762.74	44.28	26.10	13.05	22.14	15.66
6B	48.56	102.00	5815.40	176.40	103.00	51.50	88.20	61.80

(A) - Estimated Effort

Project	Difference Between Estimated and Actual Effort (Prs-hrs)						
	Initial Parm	1st Rev.	2nd Rev.	3rd Rev.	4th Rev.	5th Rev.	6th Rev.
1	21.40	144.80	32.40	20.60	6.80	12.70	9.56
3A	-44.32	904.64	-35.44	-45.04	-53.20	-48.40	-51.57
4	-3.40	-6.40	-10.80	-3.00	-7.00	-6.40	-6.20
5	-31.00	4220.40	0.20	-31.80	-55.70	-39.70	-50.92
6A	11.44	1748.20	29.74	11.56	-1.49	7.60	1.12
6B	53.54	5766.94	127.94	54.54	3.04	39.74	13.34

(B) - Difference = {Estimate - Actual}

Project	Percentage Error in Estimate Relative to Actual Effort						
	Initial Parm	1st Rev.	2nd Rev.	3rd Rev.	4th Rev.	5th Rev.	6th Rev.
1	306%	2069%	463%	294%	97%	181%	137%
3A	-72%	1474%	-58%	-73%	-87%	-79%	-84%
4	-31%	-58%	-25%	-27%	-64%	-63%	-56%
5	-39%	5302%	0%*	-40%	-70%	-50%	-64%
6A	77%	12023%	205%	80%	-10%	52%	8%
6B	110%	11876%	263%	113%	6%	82%	28%

* percentage error was negligible

(C) - Percentage Error = {[(Estimate - Actual) / Actual] . 100}

Table VIII.9 - Relative error in COTS integration effort estimates for each student project under each calibration of the model parameter values.

Student Data

Spread in Percentage Error (PE) for Estimates Across All Six Projects							
	Initial Parm	1st Rev.	2nd Rev.	3rd Rev.	4th Rev.	5th Rev.	6th Rev.
most positive PE	306%	12023%	463%	294%	97%	181%	137%
most negative PE	-72%	-58%	-58%	-73%	-87%	-79%	-84%
delta range in PE	378%	12081%	521%	367%	184%	260%	221%
largest absolute PE	306%	12023%	463%	294%	97%	181%	137%
smallest absolute PE	31%	58%	0*	27%	6%	50%	8%
combined sum	337%	12081%	463%	321%	103%	231%	145%

* percentage error was negligible

Table VIII.10 - Range in error and combined sum of the largest and smallest absolute relative error in COTS integration effort estimates across all student projects for each calibration of the model parameters.

No. & Percentage of Estimates with PE's Above and Below Initial PE's Across All Projects						
	1st Rev.	2nd Rev.	3rd Rev.	4th Rev.	5th Rev.	6th Rev.
larger abs. PE	6 : 100%	3 : 50%	4 : 67%	3 : 50%	3 : 50%	3 : 50%
smaller abs. PE	0 : 0%	3 : 50%	2 : 33%	3 : 50%	3 : 50%	3 : 50%

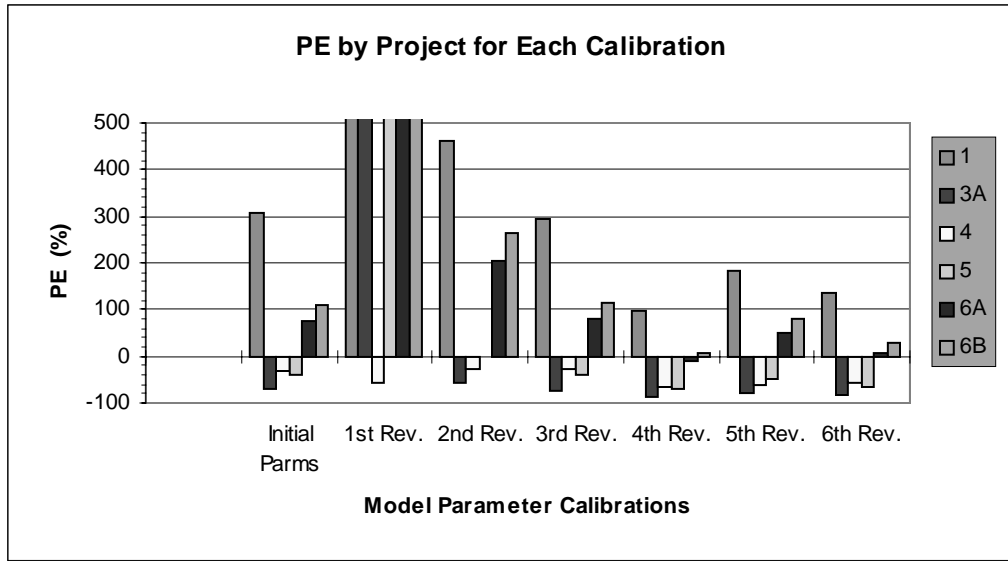
Table VIII.11 - Simple tally of how many estimates under each revised calibration had larger and smaller absolute PE's relative to the estimates produced using the initial provisional values of the model parameters without regard to *how much* greater or smaller those revised PE's were relative to the initial PE's.

No. & Percentage of Estimates Over and Under Actual Reported Effort Across All Projects														
	Initial Parm		1st Rev.		2nd Rev.		3rd Rev.		4th Rev.		5th Rev.		6th Rev.	
over	3	50%	5	83%	3	50%	3	50%	2	33%	3	50%	3	50%
under	3	50%	1	17%	2	33%	3	50%	4	67%	3	50%	3	50%
on target	0	0%	0	0%	1*	17%	0	0%	0	0%	0	0%	0	0%

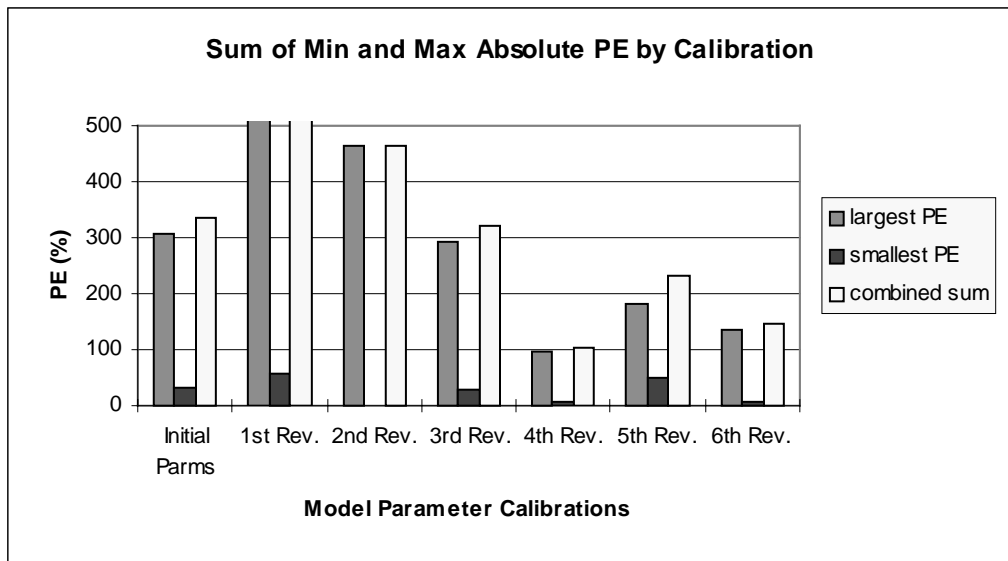
* difference in estimate was negligible

Table VIII.12 - Simple tally of how many effort estimates under each calibration were over and under the actual effort reported for each project without regard to *how much* greater or smaller those estimates were relative to the actual effort.

Student Data



Graph VIII.1 - Percentage error by project for each calibration (see Table VIII.9.C).



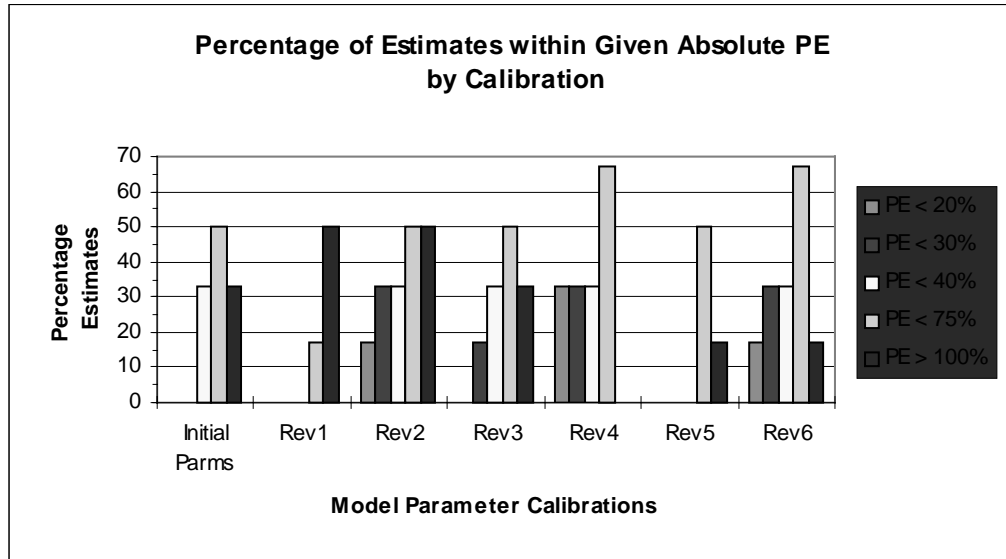
Graph VIII.2 - Combined sum of the minimum and maximum absolute percentage error for each calibration (see Table VIII.10).

Student Data

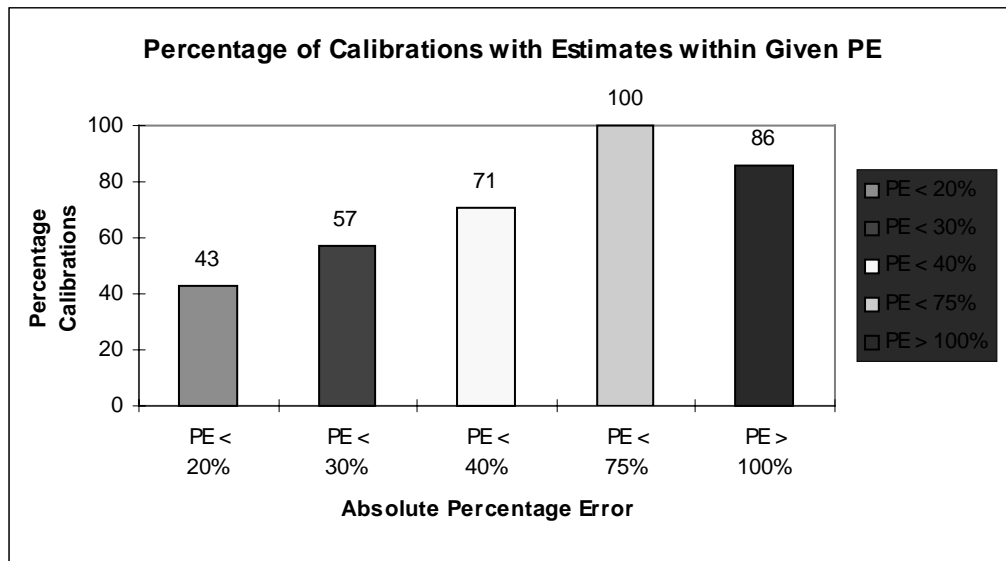
No. & Percentage of Estimates within the given Absolute PE										
Calibration	PE < 20%		PE < 30%		PE < 40%		PE < 75%		PE > 100%	
Initial Parm's	0	0%	0	0%	2	33%	3	50%	2	33%
1	0	0%	0	0%	0	0%	1	17%	5	50%
2	1	17%	2	33%	2	33%	3	50%	3	50%
3	0	0%	1	17%	2	33%	3	50%	2	33%
4	2	33%	2	33%	2	33%	4	67%	0	0%
5	0	0%	0	0%	0	0%	3	50%	1	17%
6	1	17%	2	33%	2	33%	4	67%	1	17%

Table VIII.13 - Simple tally of how many estimates under each calibration of the model parameters had absolute PE's less than the given absolute PE's.

Student Data



Graph VIII.3 - Percentage of effort estimates under a given model parameter calibration that had absolute percentage errors within the given PE range (see Table VIII.13).



Graph VIII.4 - Percentage of the seven model parameter calibrations that produced COTS integration effort estimates within the given absolute PE ranges.

Calibration Results

An examination of the data in the preceding tables and graphs reveals that the fourth revision in parameter values (case 5) had the best convergence around the actual effort reported for the various student projects. It was the revision with the most estimates within 20% PE and the only revision with no estimates above 100% PE. Still, the overall results of this calibration can at best be considered only *fair*. The goals of the calibration on the student projects were realized, however, in that it demonstrated an ability to improve the accuracy of the model if the correct kind of data is available.

Improvement Recommendation:

The overall accuracy of this initial model was low: at best, only 33% of the projects were estimated to within 40% of actuals (table VIII.13). We found that the main reason was the preponderance of effort going into COTS assessment (Understand and Qualify COTS) activities for Groups 3 and 5 (table VIII.1). This led us to recommend separating out the estimation of this effort from the estimation of COTS integration effort during project development (figure 6, section VII.A).

C. Industrial Projects

The industrial data, even though it represented twice as many data points as the student projects (twelve projects reported vs. six), was found to be problematic. There were three main reasons for this. First, the data was incomplete, with cost driver ratings, and the sizing of glue code in function points, often missing. The result was that too many assumptions had to be made in order to make the data serviceable. Second, the data was inconsistent, being a mix of COTS and Reuse projects; thus assumptions made were less likely to be valid across different projects. Third and finally, the data was old, with most of the reported projects completed in the early 1980s. Thus the factors affecting effort when these projects were performed are likely different from many of the cost drivers which appear in the current COTS model, making the projects poor candidates for use in calibration purposes.

The sizing and effort data derived from the reported data for the industrial projects appears on the following page:

Project	A	B	SLOC	UFP	BRAK	ESIZE	ACTPM
F1	2.00	1.00	9000	86	5%	90	24.00
F2	2.00	1.00	9000	86	5%	90	48.00
F3	2.00	1.00	6000	57	5%	60	24.00
F4	2.00	1.00	2000	19	5%	20	8.00
S	2.00	1.00	24000	229	5%	240	84.00
R	2.00	1.00	200000	1905	5%	200	264.00
H	2.00	1.00	42000	400	5%	420	240.00
C	2.00	1.00	16000	152	5%	160	36.00
E1	2.00	1.00	7000	67	5%	70	24.00
E2	2.00	1.00	10000	95	5%	100	20.00
T1	2.00	1.00	7000	67	5%	70	48.00
T2	2.00	1.00	5000	48	5%	50	24.00

Table VIII.14 - Industrial project COTS integration data.

Calibration Efforts

Because it became obvious very quickly that the data was not proving very useful, only three calibration cases were run on the industrial project data.

The cases were as follows:

- 1) Initial provisional model parameter values derived from the Delphi exercise as shown in table VII.4.
- 2) 1st revision values based upon regression on a natural log transformation of the parameters and data in case 1 (see footnote 17).
- 3) 2nd revision values based upon the engineering judgment of Dr. Barry Boehm as manifest in his round 2 values for the Delphi exercise.

The parameter sets derived for the cases described above appear on the following pages.

Industrial Data

Parameter	VL	L	N	H	VH	PR ¹
1 CPDM	1.36	1.18	1.00	0.82	0.64	2.13
2 CVEW	1.20	1.10	1.00	0.90	0.80	1.50
3 CIEP	1.28	1.14	1.00	0.86	0.72	1.79
4 CREL	1.22	1.11	1.00	0.89	0.77	1.58
5 CPAX	0.66	0.83	1.00	1.17	1.34	2.02
6 CIPC	1.43	1.22	1.00	0.78	0.57	2.53
7 CIAR	1.36	1.18	1.00	0.82	0.65	2.10
8 CCOS	1.17	1.08	1.00	0.92	0.83	1.40
9 CPER	1.20	1.10	1.00	0.90	0.80	1.50
10 CIXI	1.21	1.11	1.00	0.90	0.79	1.53
11 CVMS	1.26	1.13	1.00	0.87	0.74	1.70
12 CVPT	1.18	1.09	1.00	0.91	0.82	1.43
13 CPRT	1.13	1.07	1.00	0.94	0.87	1.30

Table VIII.15 - Initial provisional parameter values (Case 1).

Industrial Data

Parameter	VL	L	N	H	VH	PR ¹
1 CPDM	1.00	1.00	1.00	1.00	1.00	1.00
* 2 CVEW	0.87	0.93	1.00	1.08	1.18	1.36
3 CIEP	1.25	1.12	1.00	0.87	0.75	1.67
4 CREL	1.00	1.00	1.00	1.00	1.00	1.00
* 5 CPAX	2.86	1.60	1.00	0.67	0.48	5.96
6 CIPC	1.57	1.29	1.00	0.73	0.48	3.27
7 CIAR	1.71	1.33	1.00	0.71	0.47	3.64
8 CCOS	1.00	1.00	1.00	1.00	1.00	1.00
9 CPER	1.94	1.41	1.00	0.68	0.44	4.41
10 CIXI	1.00	1.00	1.00	1.00	1.00	1.00
11 CVMS	1.01	1.00	1.00	1.00	0.99	1.02
12 CVPT	1.00	1.00	1.00	1.00	1.00	1.00
13 CPRT	1.00	1.00	1.00	1.00	1.00	1.00

* This change in parameter is counter-intuitive.

Table VIII.16 - First recalibration of parameter values (Case 2).

¹ PR (Productivity Ratio) = VL/VH except for parameter 5 (CPAX) where PR = VH/VL.

Parameter	VL	L	N	H	VH	PR¹
1 CPDM	1.30	1.15	1.00	0.85	0.70	1.86
2 CVEW	1.24	1.12	1.00	0.88	0.76	1.63
3 CIEP	1.28	1.14	1.00	0.86	0.72	1.78
4 CREL	1.22	1.11	1.00	0.89	0.78	1.56
5 CPAX	0.66	0.83	1.00	1.17	1.34	2.03
6 CIPC	1.48	1.24	1.00	0.76	0.52	2.85
7 CIAR	1.34	1.17	1.00	0.83	0.66	2.03
* 8 CCOS	<i>1.16</i>	<i>1.08</i>	<i>1.00</i>	<i>0.92</i>	<i>0.84</i>	<i>1.38</i>
9 CPER	1.20	1.10	1.00	0.90	0.80	1.50
10 CIXI	1.20	1.10	1.00	0.90	0.80	1.50
11 CVMS	1.26	1.13	1.00	0.87	0.74	1.70
12 CVPT	1.16	1.08	1.00	0.92	0.84	1.38
* 13 CPRT	<i>1.12</i>	<i>1.06</i>	<i>1.00</i>	<i>0.94</i>	<i>0.88</i>	<i>1.27</i>

* Initial provisional value restored as change in first revision was counter-intuitive.

Table VIII.17 - Second recalibration of parameter values (Case 3).

Industrial Data

Domain	CPDM	CVEW	CIEP	CREL	CPAX	CIPC	CIAR	CCOS	CPER	CIXI	CVMS	CVPT	CPRT
executive	N	VH	H	N	N	VH	VH	N	H	H	N	N	N
security	N	L	H	N	VH	VH	VH	N	H	H	L	N	N
reports	N	H	L	N	N	H	H	N	N	H	L	N	N
help/tutorial	N	H	H	N	N	H	H	N	N	H	N	N	N
data comm.	N	N	L	N	VH	VH	VH	N	H	H	L	N	N
devlp. env.	N	VL	H	N	N	H	H	N	N	H	N	N	N
sys. test tools	N	L	H	N	N	H	H	N	N	H	N	N	N

Table VIII.18 - COTS Integration model effort driver ratings as assumed for each industrial project by domain.

Assumed ratings were based upon a subjective consideration of four factors: COTS product functional domain, COTS product source, reputation of integrated system developer (which in all cases is the same technologically very highly regarded organization), and a logical allocation of the most valuable resources to the most critical functions. Even given the preceding, however, in some cases it was still impossible to provide reasonable discriminatory rating assessments (see CPDM, CREL, CCOS, CIXI, CVPT, CPRT).

Industrial Data

Project	Actual Effort	Estimated Effort (PM)		
		Initial Parm	1st Rev.	2nd Rev.
F1	24.00	37.17	25.51	32.71
F2	48.00	37.17	25.51	32.71
F3	24.00	24.78	17.01	21.80
F4	8.00	8.26	5.67	7.27
S	84.00	206.34	25.74	194.61
R	264.00	266.95	225.70	257.43
H	240.00	374.26	368.17	360.91
C	36.00	164.77	23.75	153.56
E1	24.00	83.17	49.43	84.76
E2	20.00	108.91	70.61	121.08
T1	48.00	76.24	52.84	76.56
T2	24.00	54.46	37.74	54.68

(A) - Estimated Effort

Project	Difference Between Estimated and Actual Effort (PM)		
	Initial Parm	1st Rev.	2nd Rev.
F1	13.40	1.51	8.71
F2	-10.83	-22.49	-15.29
F3	0.78	-6.99	-2.20
F4	0.26	-2.33	-0.73
S	122.34	-58.26	110.61
R	2.95	-38.30	-6.57
H	134.26	128.17	120.91
C	128.77	-12.25	117.56
E1	59.17	25.43	60.76
E2	88.91	50.61	101.08
T1	28.24	4.84	28.56
T2	30.46	13.74	30.68

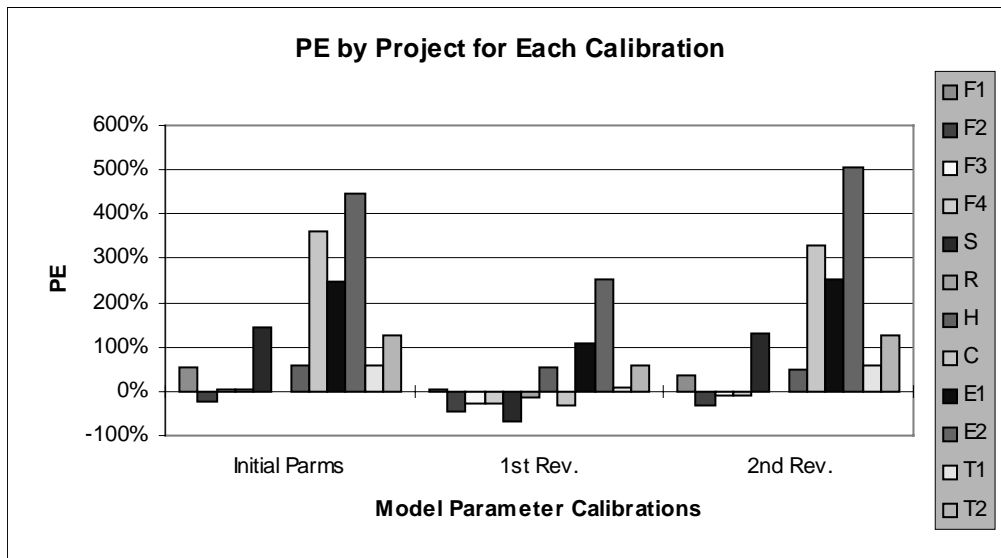
(B) - Difference = {Estimate - Actual}

Project	Percentage Error in Estimate Relative to Actual Effort		
	Initial Parm	1st Rev.	2nd Rev.
F1	55%	6%	36%
F2	-23%	-47%	-32%
F3	3%	-29%	-9%
F4	3%	-29%	-9%
S	146%	-69%	132%
R	1%	-15%	-2%
H	56%	53%	50%
C	360%	-34%	327%
E1	247%	106%	253%
E2	445%	253%	505%
T1	59%	10%	59%
T2	127%	57%	128%

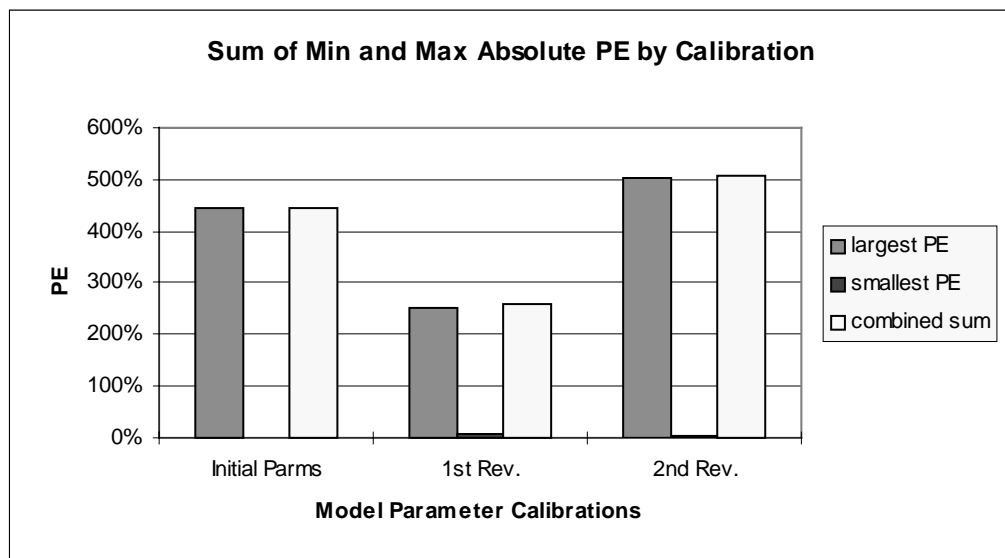
(C) - Percentage Error = $\{[(\text{Estimate} - \text{Actual})/\text{Actual}].100\}$

Table VIII.19 - Relative error in COTS integration effort estimates for each industrial project under each calibration of the model parameter values.

Industrial Data



Graph VIII.5 - Percentage error by project for each calibration (see Table VIII.19.C).



Graph VIII.6 - Combined sum of the minimum and maximum absolute percentage error for each calibration (see Table VIII.20).

Industrial Data

Spread in Percentage Error (PE) for Estimates Across All Projects			
	Initial Parm's	1st Rev.	2nd Rev.
most positive PE	445%	253%	505%
most negative PE	-23%	-69%	-32%
delta range in PE	468%	322%	537%
largest absolute PE	445%	253%	505%
smallest absolute PE	1%	6%	2%
combined sum	446%	259%	507%

Table VIII.20 - Range in error and combined sum of the largest and smallest absolute relative error in COTS integration effort estimates across all industrial projects for each calibration of the model parameters.

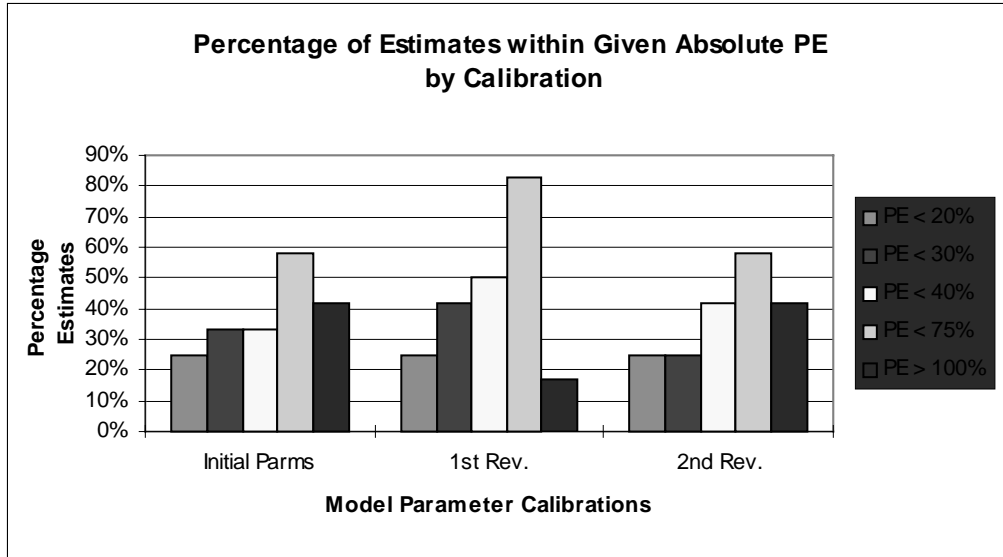
No. & Percentage of Estimates Over and Under Actual Reported Effort Across All Projects						
	Initial Parm's		1st Rev.		2nd Rev.	
over	11	92%	6	50%	8	67%
under	1	8%	6	50%	4	33%

Table VIII.21 - Simple tally of how many effort estimates under each calibration were over and under the actual effort reported for each project without regard to *how much* greater or smaller those estimates were relative to the actual effort.

No. & Percentage of Estimates within the given Absolute PE										
Calibr.	PE < 20%		PE < 30%		PE < 40%		PE < 75%		PE > 100%	
Initial Parm's	3	25%	4	33%	4	33%	7	58%	5	42%
1st Rev.	3	25%	5	42%	6	50%	10	83%	2	17%
2nd Rev.	3	25%	3	25%	5	42%	7	58%	5	42%

Table VIII.22 - Simple tally of how many estimates under each calibration of the model parameters had absolute PE's less than the given absolute PE's.

Industrial Data



Graph VIII.7 - Percentage of effort estimates under a given model parameter calibration that had absolute percentage errors within the given PE range (see Table VIII.22).

Calibration Results

An examination of the data in the preceding tables and graphs reveals that the industrial data resulted in calibrations that were quite weak, with percentage errors all over the map. As a result, it was decided to base the final calibration of the USC COTS integration cost model version 1.0 on that of case 5 under the student data, subject to the recommendation of future recalibration after separating out the COTS assessment effort as noted on page 56. (Using the student data as the basis of the final calibration also is what necessitated the change in units of the model response variable from person-months to person-hours as noted in section VII.C on page 33, due to the way the student effort data was initially reported.)

D. Final Version 1.0 Multiplier Values

Below are the final parameter values chosen for version 1.0 of the COTS integration cost model. They are the values derived in case 5 of the student projects calibration efforts.

Final Preliminary Calibration Result (Values used in USC COTS Integration Cost Calculator V1.0)

Parameter	VL	L	N	H	VH	PR¹
1 CPDM	1.30	1.15	1.00	0.85	0.70	1.86
2 CVEW	1.24	1.12	1.00	0.88	0.76	1.63
3 CIEP	1.28	1.14	1.00	0.86	0.72	1.78
4 CREL	1.22	1.11	1.00	0.89	0.78	1.56
5 CPAX	0.66	0.83	1.00	1.17	1.34	2.03
6 CIPC	1.48	1.24	1.00	0.76	0.52	2.85
7 CIAR	1.34	1.17	1.00	0.83	0.66	2.03
8 CCOS	1.16	1.08	1.00	0.92	0.84	1.38
9 CPER	1.20	1.10	1.00	0.90	0.80	1.50
10 CIXI	1.20	1.10	1.00	0.90	0.80	1.50
11 CVMS	1.26	1.13	1.00	0.87	0.74	1.70
12 CVPT	1.16	1.08	1.00	0.92	0.84	1.38
13 CPRT	1.12	1.06	1.00	0.94	0.88	1.27

Table VIII.23 - Final model calibration values.

IX. Cost Estimation Procedure Using the Model

A. Estimation Procedure

The following procedure is suggested when attempting to use our model for a COTS integration cost estimation exercise (refer back to the formula shown under section VII.C on page 33):

- 1) Estimate the amount of glue code that is expected to be needed to integrate a set of COTS products into a software application, in terms of Unadjusted Function Points. These are derived from the full set of Function Point types (External Inputs, External Outputs, External Inquiries, Internal Logical Files, External Interface Files), but only as they apply to the details of the integration or glue code linking the COTS component to the larger application, *not* the code internal to the COTS component itself.
- 2) Estimate the percentage of glue code which will be lost due to breakage during the integration effort, again in terms of UFP; this will be a function of the number of COTS packages being integrated into the new system overall, the average number of updated product releases expected per COTS package over the life of the system development, and the average interface breakage per product release.
- 3) Determine the effective size of the glue code development effort by feeding the estimates derived in steps 1 and 2 into the formula for ESIZE.
- 4) Assign each effort multiplier a rating on the given scale from very low to very high, which best characterizes the unique conditions obtaining during this COTS integration effort.
- 5) Determine the overall estimated effort for this integration task by feeding the estimate for ESIZE and the rated effort multipliers into the formula for Person-months (PM).
- 6) Determine the estimated cost by multiplying estimated PM by the estimated average labor rate (\$\$/PM).

This completes the cost estimation procedure.

B. How to Size Glue Code via Function Points

The question of how to size glue code is not straight forward, as there is often controversy over where glue code ends and new development code begins. One way to bound the problem is to consider glue code to consist of any of the following:

- Any code required to facilitate information or data exchange between the COTS component and the application.
- Any code needed to “hook” the COTS component into the application, even though it may not necessarily facilitate data exchange.
- Any code needed to provide functionality that was originally intended to be provided by the COTS component, AND which must interact with that COTS component.

Once the glue code has been bounded according to the above criteria, a standard function point count can now be attempted on that code using IFPUG counting rules. The last bounding statement in particular requires that the individual doing the sizing in function points apply any and all of the standard functional types¹⁹ as needed:

- External Inputs
- External Outputs
- External Inquiries
- Internal Logical Files
- External Interface Files

C. How to Estimate Percentage Breakage

Breakage refers to COTS integration code that must be reworked as a result of either a change in system requirements, or a new release by the vendor of a COTS product which necessitates that the newer version of the product be installed before system delivery. It does NOT refer to code that must be reworked due to bugs introduced by the programmer, or due to defects in design.

This figure is best estimated based upon acquiring knowledge of two things: 1) the vendor’s past history regarding releases of the COTS product in question or of similar products that the vendor markets, and 2) the customer’s past history regarding demanding changes in requirements after development and COTS product integration has begun.

¹⁹ D. Garmus and D. Herron, Measuring the Software Process: A Practical Guide to Functional Measurements, Yourdon Press, Prentice-Hall PTR, NJ, 1996.

X. Relating/Using the COTS Estimate with a COCOMO II Project Estimate

At the moment, the COTS integration cost model functions independently of COCOMO II. Use COCOMO as you would normally when estimating the new development or reuse aspects of a project. Use the COTS integration model when determining effort associated with developing the integration or glue code required for a COTS product. Then add the resultant COTS effort to the COCOMO derived effort to get the total project development effort.

XI. Overall COTS Integration Cost Estimation/Mitigation Guidelines

The four key COTS integration characteristics below make COTS integration significantly different from other forms of software development (including maintenance). They require traditional approaches to software development to be significantly revised. The characteristics are:

1. You have no control over a COTS product's functionality or performance.
2. Most COTS products are not designed to interoperate with each other.
3. You have no control over a COTS product's evolution.
4. COTS vendor behavior varies widely.

1. You have no control over a COTS product's functionality or performance.

If you can modify the source code, it's not really COTS--and its future becomes your responsibility. Even as black boxes, big COTS products have formidable complexity: Microsoft people have indicated that Windows 95 has 25,000 entry points.

Resulting Pitfalls

- *Using the waterfall model on a COTS integration project.* With the waterfall model, you specify requirements, and these determine the capabilities. With COTS products, it's the other way around: the capabilities determine the "requirements" or the delivered system features.
- *Using evolutionary development with the assumption that every undesired feature can be changed to fit your needs.* COTS vendors do change features, but they respond to the overall marketplace and not to individual users.
- *Believing that advertised COTS capabilities are real.* COTS vendors may have had the best of intentions when they wrote the marketing literature, but that doesn't help you when the advertised feature isn't there.

Resulting Recommendations

- *Use risk management and risk-driven spiral-type process models.* Assess risks via prototyping, bench-marking, reference checking, and related techniques. Focus each spiral cycle on resolving the most critical risks. The Raytheon "Pathfinder" approach²⁰ is a particularly effective way to address these and other risks. (This is an architecture based approach with an investment in COTS qualifications, and needs comparable performance measures.)
- *Perform the equivalent of a "receiving inspection" upon initial COTS receipt,* to ensure that the COTS product really does what it is expected to do.

²⁰ A. Sardi, "COTS Risk Mitigation: Strategies and Goals," *Proceedings: Focused Workshop on System Integration with COTS Software*, USC Center for Software Engineering, November 6-8, 1996, p.257.

- *Keep requirements negotiable until the system's architecture and COTS choices stabilize.*
- *Involve all key stakeholders in critical COTS decisions.* These can include users, customers, developers, testers, maintainers, operators, or others as appropriate.

2. Most COTS products are not designed to interoperate with each other.

Garlan, et al,²¹ provides a good case study and explanation for why interoperability problems can cause COTS integration cost and schedule overruns by factors of four to five.

Resulting Pitfalls

Lack of COTS interoperability exacerbates each of the previously cited pitfalls. Some additional direct pitfalls are:

- *Premature commitment to incompatible combinations of COTS products.* This can happen in many ways: haste, desire to show progress, politics, or uncritical enthusiasm with features or performance. Short-term emphasis on rapid application development is another source of this pitfall.
- *Trying to integrate too many incompatible COTS products.* Four can be too many²². In general, trying to integrate more than a half-dozen COTS products from different sources should place this item on your high-risk assessment list.
- *Deferring COTS integration till the end of the development cycle.* This puts your most uncontrollable problem on your critical path as you approach delivery.
- *Committing to a tightly-coupled subset of COTS products with closed, proprietary interfaces.* These restrict your downstream options; once you're committed, it's hard to back yourself out.

Resulting Recommendations

The previously cited recommendations on risk-driven processes and co-evolving your requirements and architecture are also appropriate here. In addition:

- *Use the Life Cycle Architecture milestone as an anchor point for your development process²³.* In particular include demonstrations of COTS interoperability and scalability as risks to be resolved and documented in the Architecture Rationale.
- *Use the Architecture Review Board (ARB) best commercial practice at the Life Cycle Architecture milestone.²⁴* AT&T has documented at least 10% savings in using it over a period of 8 years.

²¹ D. Garlan, R. Allen and J. Ockerbloom, "Architectural Mismatch: Why Reuse is So Hard," *IEEE Software*, November 1995, pp.17-26.

²² See Garlan, et al, footnote 21.

²³ Boehm, B.W., "Anchoring the Software Process," *IEEE Software*, July 1996, pp. 73-82.

²⁴ Best Current Practices: Software Architecture Validation, Lucent/AT&T, 1991.

- *Go for open architectures and COTS substitutability.* In the extremely fast-moving software field, the ability to adapt rapidly to new best-of-breed COTS products is competitively critical.

3. You have no control over a COTS product's evolution.

Again, COTS vendors respond to the overall marketplace and not to individual users. Upgrades are frequently not upward compatible. And old releases become obsolete and unsupported by the vendor. If COTS architectural mismatch doesn't get you initially, COTS architectural drift can easily get you later. Current COTS-intensive systems often have higher software maintenance costs than traditional systems²⁵, but that good practices can make them lower. (This does not imply that it's cheaper for you to maintain you own home-brew relational DBMS than to pay the COTS maintenance fees for a clean SQL-compliant product.)

Resulting Pitfalls

Lack of evolution controllability exacerbates each of the previously cited pitfalls. Some additional direct pitfalls are:

- *"Snapshot" requirements specs and corresponding point-solution architectures.* These are not good practices for traditional systems; with uncontrollable COTS evolution, the maintenance headaches become even worse.
- *Under-staffing for software maintenance,* and lack of COTS adaptation training for maintenance personnel.
- *Tightly coupled, independently evolving COTS products.* Just two of these will make maintenance difficult; more than two is much worse.
- *Assuming that uncontrollable COTS evolution is just a maintenance problem.* It can attack your development schedules and budgets as well.

Resulting Recommendations

The previously-cited risk-driven and architecture-driven recommendations are also appropriate here. In addition:

- *Stick with dominant commercial standards.* These make COTS product evaluation and substitutability more manageable.
- *Use likely future system and product line needs as well as current needs as COTS selection criteria.* These can include portability, scalability, distributed processing, user interface media, and various kinds of functionality growth.

²⁵ Lockheed-Martin, "COTS Integration: Application of Lessons Learned," *Proceedings: Focused Workshop on System Integration with COTS Software*, USC Center for Software Engineering, November 6-8, 1996, p.435.

- *Use flexible architectures facilitating adaptation to change.* These can include message/event-based software bus, encapsulation, and layering.
- *Carefully evaluate COTS vendors' track records with respect to predictability of product evolution.*
- *Establish a pro-active system release strategy, synchronizing COTS upgrades with system releases.*

4. COTS vendor behavior varies widely.

Vendor behavior varies widely with respect to support, cooperation, and predictability. Sometimes a COTS vendor is not even the developer, just a value-added re-seller. Given the three major sources of COTS integration difficulty above, an accurate assessment of a COTS vendor's ability and willingness to help out with the difficulties is tremendously important. The workshop identified a few assessment heuristics, such as the experience that the value of a COTS vendor's support follows a convex curve with respect to the vendor's size and maturity (see figure 7 below). Small vendors often lack the capability to support you; very large vendors have the capability, but not the motivation (the classic example is Microsoft).

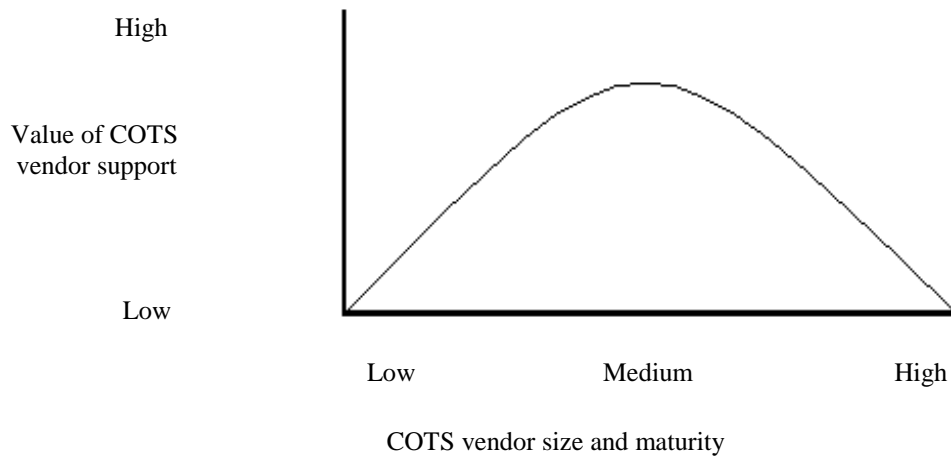


Figure 7 - Variation of COTS Vendor Support with Size

Resulting Pitfalls

Poor COTS vendor support exacerbates each of the previously-cited pitfalls. Some additional direct pitfalls are:

- *Uncritically accepting COTS vendors' statements about product capabilities and support.*
- *Lack of fallbacks or contingency plans, for such contingencies as product substitution or escrow of a failed vendor's product.*
- *Assuming that an initial vendor support honeymoon will last forever.*

Resulting Recommendations

The previously cited recommendations are also appropriate here. In addition:

- *Perform extensive evaluation and reference-checking of a COTS vendor's advertised capabilities and support track record.*
- *Establish strategic partnerships or other incentives for COTS vendors to provide support.* Incentives can include financial incentives, early experimentation with and adoption of new COTS vendor capabilities, sponsored COTS product extensions or technology upgrades.
- *Negotiate and document critical vendor support agreements.* Establish a "no surprises" relationship with vendors.

The preceding is summarized in table XI.1 on the following page.

Pitfalls to Avoid	Recommended Practices to Adopt
<i>1. You have no control over a COTS product's functionality or performance.</i>	
<ul style="list-style-type: none"> Using the waterfall model on a COTS integration project. Using evolutionary development with the assumption that every undesired feature can be changed to fit your needs. Believing that advertised COTS capabilities are real. 	<ul style="list-style-type: none"> Use risk management and risk-driven spiral-type process models. Perform the equivalent of a “receiving inspection” upon initial COTS receipt. Keep requirements negotiable until the system’s architecture and COTS choices stabilize. Involve all key stakeholders in critical COTS decisions.
<i>2. Most COTS products are not assigned to interoperate with each other.</i>	
<ul style="list-style-type: none"> Premature commitment to incompatible combinations of COTS products. Trying to integrate too many incompatible COTS products. Deferring COTS integration till the end of the development cycle. Committing to a tightly-coupled subset of COTS products with closed, proprietary interfaces. 	<ul style="list-style-type: none"> Use the Life Cycle Architecture milestone as a process anchor point. Use the Architecture Review Board (ARB) best commercial practice at the Life Cycle Architecture milestone. Go for open architectures and COTS substitutability.
<i>3. You have no control over a COTS product's evolution.</i>	
<ul style="list-style-type: none"> “Snapshot” requirements specs and corresponding point-solution architectures. Understaffing for software maintenance, Tightly coupled, independently evolving COTS products. Assuming that uncontrollable COTS evolution is just a maintenance problem. 	<ul style="list-style-type: none"> Stick with dominant commercial standards. Use likely future system and product line needs as well as current needs as COTS selection criteria. Use flexible architectures facilitating adaptation to change. Carefully evaluate COTS vendors’ track records with respect to predictability of product evolution. Establish a pro-active system release strategy, synchronizing COTS upgrades with system releases.
<i>4. COTS vendor behavior varies widely.</i>	
<ul style="list-style-type: none"> Uncritically accepting COTS vendors’ statements about product capabilities and support. Lack of fallbacks or contingency plans. Assuming that an initial vendor support honeymoon will last forever. 	<ul style="list-style-type: none"> Perform extensive evaluation and reference-checking of a COTS vendor’s advertised capabilities and support track record. Establish strategic partnerships or other incentives for COTS vendors to provide support. Negotiate and document critical vendor support agreements.

Table XI.1 - Key COTS Integration Characteristics and Their Implication

XII. Conclusion/Future Directions

This study has been a first step in developing a comprehensive methodology for estimating the costs of using COTS software in any system development. But the scope of the model developed under this study is deliberately narrow, focusing only on the initial integration coding effort. Many other factors should be examined when determining the true overall cost of integrating COTS software into larger systems. These factors include not only the traditional costs associated with new software development, such as the cost of requirements definition, design, code, test, and software maintenance, but also the cost of licensing and redistribution rights, royalties, effort needed to understand COTS software, pre-integration assessment and evaluation, post-integration certification of compliance with mission critical or safety critical requirements, indemnification against faults or damage caused by vendor supplied components, and costs incurred due to incompatibilities with other needed software and/or hardware.

The preceding indicates the myriad of directions in which the COTS integration cost model just developed can be expanded and improved, and suggests goals for future phases of this research.

XIII. References

1. B.W. Boehm, Technical Proposal: Added Tasks for Contract F30602-94-C-1095, “Next Generation Software Processes and Their Environment Support,” USC Center for Software Engineering, January 4, 1996.
2. B.W. Boehm, Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1981.
3. W. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy and R. Selby, “Cost Models for Future Software Life Cycle Process: COCOMO 2.0,” *Annals of Software Engineering Special Volume on Software Process and Product Management*, J.D. Arthur and S.M. Henry, Eds., J.C. Balter AG, Science Publishers, Amsterdam, The Netherlands, 1995, Vol. 1, pp.45-60.
4. T. Ellis, “COTS Integration in Software Solutions—A Cost Model,” in *Systems Engineering in the Global Marketplace*, NCOSE International Symposium, St. Louis, MO, July 24-26, 1995.
5. O.Helmer, *Social Technology*, Basic Books, NY, 1966.
6. R. Stutzke, “Costs Impact of COTS Volatility,” *Knowledge Summary: Focused Workshop on COCOMO 2.0*, USC Center for Software Engineering, May 16-18, 1995.
7. M. Karpowich, T. Sanders and R. Verge, “An Economic Analysis Model for Determining the Custom vs. Commercial Software Tradeoff,” in T. Gullledge and W. Hutzler, Analytical Methods in Software Engineering Economics, Springer-Verlag, 1993.
8. Garmus and D. Herron, Measuring the Software Process: A Practical Guide to Functional Measurements, Yourdon Press, Prentice-Hall PTR, NJ, 1996.
9. Sardi, “COTS Risk Mitigation: Strategies and Goals,” *Proceedings: Focused Workshop on System Integration with COTS Software*, USC Center for Software Engineering, November 6-8, 1996, p.257.
10. Garlan, R. Allen and J. Ockerbloom, “Architectural Mismatch: Why Reuse is So Hard,” *IEEE Software*, November 1995, pp.17-26.
11. Boehm, B.W., “Anchoring the Software Process,” *IEEE Software*, July 1996, pp. 73-82.
12. Best Current Practices: Software Architecture Validation, Lucent/AT&T, 1991.

13. Lockheed-Martin, "COTS Integration: Application of Lessons Learned,"
Proceedings: Focused Workshop on System Integration with COTS Software, USC
Center for Software Engineering, November 6-8, 1996, p.435.

Appendices

Appendix A
COTS Integration Cost
Calculator V1.0

Appendix B

Calibration Case Data

B1. Student Projects

B2. Industrial Projects

Appendix C
SEI Risk Repository Data

C1. COTS Related SERR Statements

C2. COTS Related Lexical Maps

Appendix D

Data Collection Instruments

D1. Delphi Exercise

D2. Data Request Letter & Round 1 Survey

D3. Round 2 Survey