



UNIVERSITY OF
SOUTHERN CALIFORNIA

**COTS/NDI Software
Integration Cost Estimation**

&

**USC-CSE
COTS Integration Cost
Calculator V2.0
User Guide**

Revision 1.0
30 September 1997

**Christopher M. Abts
Barry W. Boehm**

University of Southern California
Center for Software Engineering

Table of Contents

I. Introduction.....	p.1
II. Tools Described.....	p.1
III. Scope.....	p.1
IV. Project Phases Covered.....	p.2
V. Estimating for NDI Components.....	p.2
VI. Estimating for COTS Components.....	p.5
VII. Definitions and Rating Scales for Drivers in the COTS Model.....	p.9
VIII. Definitions/Glossary.....	p.17

I. Introduction

This document explains how to use software cost modeling tools developed at the *USC Center for Software Engineering* in Los Angeles to estimate part of the initial cost of integrating commercial-off-the-shelf (COTS) and non-developmental-item (NDI) software into larger software systems. The COTS Calculator V2.0 was refined as part of a modeling effort sponsored by the Federal Aviation Administration.

II. Tools Described

For NDI Components:

USC COCOMO II.1997 (re-use model aspects)

For COTS Components:

USC-CSE COTS Integration Cost Calculator V2.0

III. Scope

The scope of the V2.0 cost estimation tools is deliberately narrow. While there are many factors¹ which should be examined when determining the true overall cost of integrating a COTS or NDI software component into a larger system, for the sake of simplicity and practicality, these V2.0 tools focus only on estimating the *coding effort* needed to integrate a COTS or NDI component for initial system delivery.

For NDI components, this effort may include changes to the existing source code as well as any additional code which must be written to get a component integrated (often referred to as “glue” code). For COTS components, which normally do not offer access to source code, the integration coding effort is usually restricted to the creation of “glue” code alone.

Glue code for NDI components, however, is often written as seamless extensions of the NDI source, allowing the embedding of the NDI source code directly within other source code of the larger system into which it is being integrated.

The glue code for COTS components, however, is usually clearly defined as connected to--but quite separate from--the COTS component itself, acting more as a “bridge” between the COTS component and the system into which it is being integrated, rather than permitting the COTS source to be embedded within the larger system.

¹ 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 the COTS/NDI 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.

Moreover, NDI and COTS components are subject to similar, though not identical, pressures which can best be described as “environmental.” In particular, by definition COTS components are much more at the mercy of commercial market forces and the speed with which those market forces change than is usually the case with NDI components. Unlike private sector COTS components which must compete on the open market and thus are continually being reshaped by that market, NDI components can come from either the private or the public sector, and especially if procured from the latter sector, do not often change as rapidly.

To put it another way, with NDI components, usually “what you see is what you get.” With COTS components, sometimes you’re not even sure of what you’re seeing.

These distinctions between NDI and COTS components in the way glue code functions and in the environmental pressures at play drives the difference in the way integration effort for each of these components is modeled, as will be described in sections V and VI.

IV. Project Phases Covered

The effort estimates provided by these tools are valid over and inclusive of the following project life-cycle phases *only*:

Preliminary Code Design	Detailed Code Design	Code and Unit Test	Integration and Test
------------------------------------	---------------------------------	-------------------------------	---------------------------------

Project phases specifically *not* covered by these tools are *system requirements definition, software requirements definition, COTS/NDI pre-integration assessment/evaluation, and maintenance.*

V. Estimating for NDI Components

Integrating open source NDI components is very similar in nature to any other standard reuse of software, with similar factors driving costs, particularly if an NDI component does not come from a commercial vendor. For this reason, when performing an integration coding effort estimate for NDI components, particularly for those in which some of the source code is to be modified, it is recommended that the Reuse Model of COCOMO II² be used, along with the existing set of COCOMO II cost drivers.

The appropriate COCOMO II reuse equations are presented on the following page, but for a detailed description, the reader is referred to the COCOMO manual cited in footnote #2.

² See the *COCOMO II Model Definition Manual*, Version 1.4, Section 2.3.3, p8
©1997 University of Southern California.

The Reuse Model of COCOMO II adapted for NDI component integration effort estimation:

$$PM = A \times [Size']^B \times \prod_{i=1}^{17} EM_i$$

where

$$B = 1.01 + 0.01 \times \sum_{j=1}^5 SF_j$$

$$Size' = Size \times \left(1 + \frac{BRAK}{100} \right)$$

$$Size = KNSLOC + KASLOC \times (AAM)$$

$$AAM = \begin{cases} \frac{AA + AAF \times (1 + 0.02(SU)(UNFM))}{100}, & AAF \leq 0.5 \quad \text{or} \\ \frac{AA + AAF + (SU) \times (UNFM)}{100}, & AAF > 0.5 \end{cases}$$

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

The terms in the above equations are defined in Table 1 on the following page, however, the following points should be highlighted:

- 1) AA, the Assessment and Assimilation term in the Reuse model should be set to zero to ensure the effort estimate produced is consistent with the covered project phases described in section IV.
- 2) KNSLOC should be the estimated size of new glue code needed to integrate the NDI component.
- 3) KASLOC should be the size of the NDI component being integrated.
- 4) COCOMO II expects source code sized in terms of *logical* lines of source code as defined by the Software Engineering Institute (if not sized in Function Points).
- 5) COCOMO II is *not* calibrated for a total project SLOC of less than 2,000 lines.
- 6) COCOMO II defines a standard person-month to consist of 152 hours.

User Guide 1.0

Symbol	Description
A	Constant, provisionally set to 2.5
AA	Assessment and assimilation (set to zero)
BRAK	Breakage: Percentage of NDI integration code thrown away due to requirements volatility
CM	Percentage of NDI code modified
DM	Percentage of NDI design modified
EM	Effort Multipliers: RELY, DATA, CPLX, RUSE, DOCU, TIME, STOR, PVOL, ACAP, PCAP, PCON,
IM	Percentage of overall integration and test modified due to the presence of the NDI component
KASLOC	Size of the NDI component expressed in thousands of source lines of code
KNSLOC	Size of new NDI component glue code expressed in thousands of source lines of code
PM	Person Months of estimated effort
SF	Scale Factors: PREC, FLEX, RESL, TEAM, PMAT
SU	NDI Software understanding (zero if DM = 0 and CM = 0)
UNFM	Programmer Unfamiliarity with NDI Software

Table 1 - Definition of symbols appearing in the COCOMO II Reuse Model as adapted for NDI component integration effort estimation.

VI. Estimating for COTS Components

Estimating the cost of integrating a true COTS component in which no source code is modified should be done using the *USC-CSE COTS Integration Cost Calculator*. This is a **Microsoft Excel Version 7.0 (for Windows '95)** spreadsheet tool shown in Figure 1 and implementing the COTS integration cost model refined with the assistance of the FAA over the summer of 1997, building upon a COTS integration cost estimation model developed at USC during the previous year.

USC-CSE COTS Integration Cost Calculator V2.0														
Calibration Tables:														
Rating	Driver													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	ACIEP	ACIPC	AXICP	APCON	ACPMT	ACSEW	APCPX	ACPPS	ACPTD	APVOL	ACREL	AACPX	ACPER	ASPRT
VL	1.34	1.60		1.58	1.45				1.20	0.71				
L	1.16	1.27	1.12	1.26	1.20	1.07	0.82	1.14	1.09	0.84	0.88	0.84		
N	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
H	0.86	0.79	0.89	0.80	0.83	0.94	1.22	0.88	0.91	1.19	1.14	1.19	1.11	1.07
VH	0.75	0.62	0.79	0.63	0.69	0.88	1.48	0.77	0.84	1.33	1.30	1.42	1.22	1.14
Linear Scaling Factor														
A														
12.0														
Nonlinear Scaling Factor														
AAREN														
VL	L	N	H	VH										
4.00	3.00	2.00	1.00	0.00										
Inputs:														
Value	Driver													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	ACIEP	ACIPC	AXICP	APCON	ACPMT	ACSEW	APCPX	ACPPS	ACPTD	APVOL	ACREL	AACPX	ACPER	ASPRT
Value	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Scaling Factor														
AAREN														
Value	2.00													
Sizing														
KSLOC BRAK (%)														
Value	0.00 0													
Labor Cost														
Rate (\$/Pr-mt)														
Value	0.00													
Outputs:														
ESIZE	B	EM ₁₋₁₄	Estimated Effort (Prs-mts)										Estimated Cost (\$)	
0.00	1.08	1.00	0.00										\$0	

Figure 1- COTS integration cost estimation tool spreadsheet.

As seen in Figure 1, the spreadsheet has tables divided among three sections:

- Calibration Tables
- Inputs
- Outputs

The Calibration Tables contain the calibrated parameter values for each cost driver in the model and should not be changed. These values have been derived from an analysis of actual project data provided by the FAA during this most recent modeling effort.

The Input tables are the locations in the spreadsheet in which the user inputs the data required by the tool to perform an effort estimate.

The Output tables are where the results of the model calculations are displayed for the user.

Cost Estimation Procedure:

- 1) Use the criteria provided in section VII for determining the appropriate rating (from Very Low to Very High) of each of the fourteen cost drivers and one nonlinear scale factor appearing in the model, based upon the development conditions which you expect will obtain during the integration of the COTS component.
- 2) Once you have ratings for all of the drivers, look at the appropriate calibration table to determine the numerical value for each parameter corresponding to the rating which you have assigned to each driver. For example, for the driver ACPMT, a rating of Low (L) corresponds to a parameter value of 1.20. For the scaling factor AAREN, a rating of High (H) corresponds to a parameter value of 1.00.
- 3) Manually type in the corresponding parameter value for each driver into the cell provided in the appropriate Input table. Again, continuing the example from above, 1.20 would be typed into the cell on the value line under ACPMT in the Driver Input table. For AAREN, 1.00 would be typed into the far left cell on the value line in the space provided in the Scaling Factor Input table. (The linear scale factor A is a constant and thus its actual parameter value does not need to be input like the others.)
- 4) In the Sizing Input table, type in your estimate in thousands of source-lines-of-code (KSLOC)³ for the amount of glue code required to integrate the COTS component. Then in the far left cell under BRAK, type in your estimate for the percentage glue for this component which will have to be reworked due to a change in project requirements. For example, if you estimate 12,500 lines of glue code with perhaps 7% breakage due to requirements change,

³ Unlike COCOMO II, which uses *logical* lines of code, the COTS model has been calibrated for “non commented/non blank” lines of code. This was necessitated by the project data provided. Also, the COTS model was calibrated with projects reporting glue code well under 100 lines to as great as several thousand lines. However, it is not recommended to use the model for integration efforts requiring much less than 1,000 lines of glue code due to the scale of the inputs and the fact that other factors other than the given drivers can potentially begin to have large impacts on the actual effort.

User Guide 1.0

you would type in 12.5 under KSLOC and 7 under BRAK.

- 5) In the Labor Cost Input table, type into the far left cell under Rate your actual labor rate per person-month⁴ in dollars. For example, if your rate is \$6,500 dollars per person-month, you would type 6500 into the space provided.
- 6) In the Output tables, you should now see values for the estimated person-hours and associated cost required to integrate the COTS component for initial delivery.

Figure 2 below shows a completed example of an integration effort estimation for a COTS component.

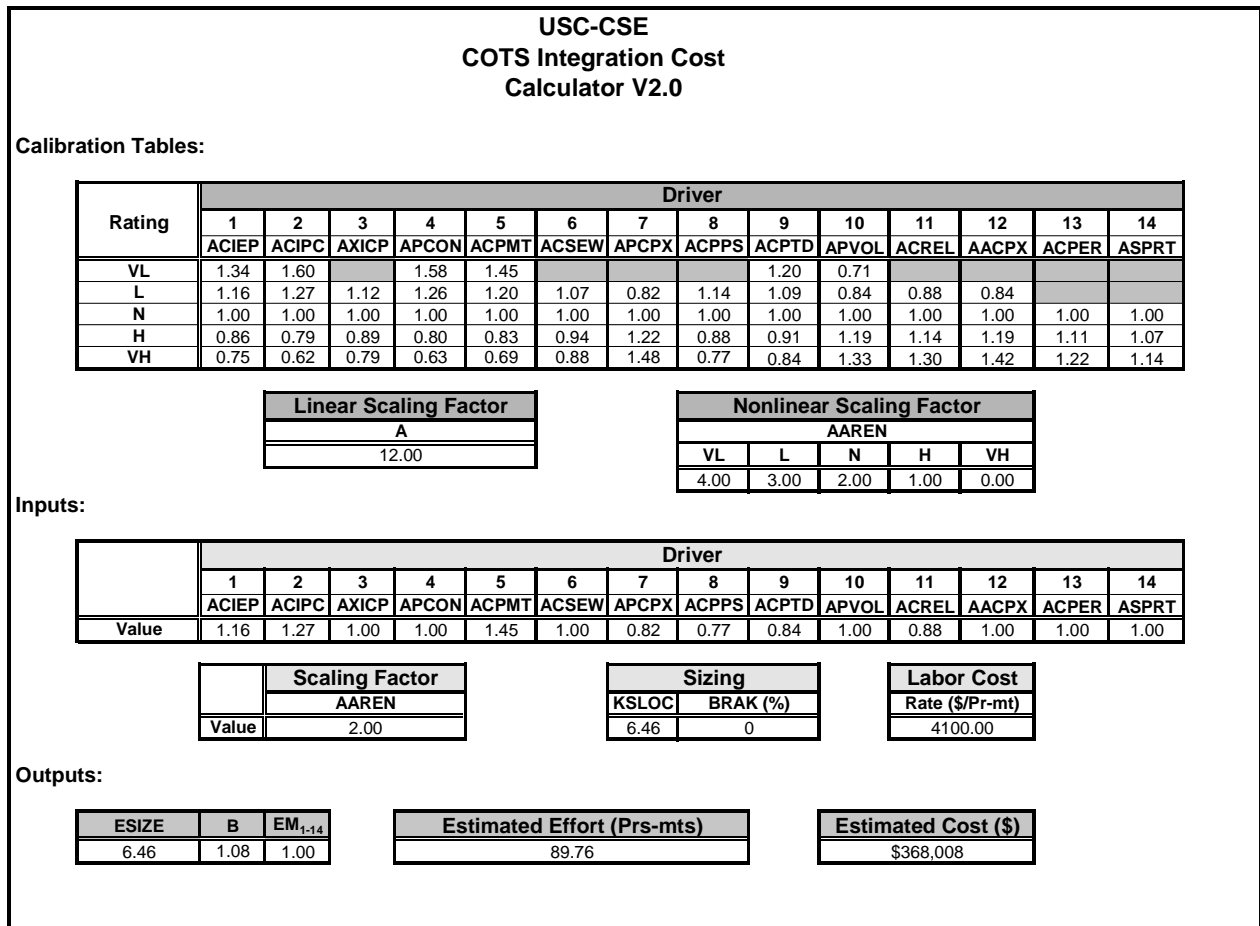


Figure 2 - COTS component integration cost estimation example.

⁴ Again, unlike COCOMO II, due to the data provided, the COTS model assumes 160 hours per standard person month as opposed to 152 hours.

The USC-CSE COTS Integration Cost Estimation Model Equations:

$$PM = A \times [Size']^B \times \prod_{i=1}^{14} EM_i$$

where

$$A = 12.0$$

$$B = 1.00 + (0.04 \times SF)$$

$$Size' = KSLOC \times \left(1 + \frac{BRAK}{100} \right)$$

Note: *Size'* is called **ESIZE** in the spreadsheet.

Symbol	Description
A	Constant, provisionally set to 12.0
BRAK	Breakage: Percentage of COTS glue code thrown away due to requirements volatility
KSLOC	Size of COTS component glue code expressed in thousands of source lines of code
PM	Person Months of estimated integration effort
EM	Effort Multipliers: ACIEP, ACIPC, AXICP, APCON, ACPMT, ACSEW, APCPX, ACPPS, ACPTD, APVOL, ACREL, AACPX, ACPER, ASPRT
SF	Scale Factor: AAREN

Table 2 - Definition of symbols appearing in the COTS integration cost estimation model.

VII. Definitions and Rating Scales for Drivers in the COTS Model

Integration Personnel Drivers

1) ACIEP - COTS Integrator Experience with Product

How much experience did/does the development staff have with running, integrating, and maintaining the COTS product?

Metric: months/years of experience with product.

Very Low	Low	Nominal	High	Very High
Staff on average has no experience with the product.	Staff on average has less than 6 month's experience with the product.	Staff on average has between 6 month's and 1 year's experience with the product.	Staff on average has between 1 and 2 years' experience with the product.	Staff on average has more than 2 years' experience with the product.

2) ACIPC - COTS Integrator Personnel Capability

What were/are the overall software development skills and abilities which your team as a whole on average brought/bring to the product integration task AS WELL AS experience with the specific tools, languages, platforms, and operating systems used/being used in the integration task?

Metric: months/years of experience.

Very Low	Low	Nominal	High	Very High
Staff on average has no development experience or with the specific environmental items listed.	Staff on average has less than 6 month's development experience or with the specific environmental items listed.	Staff on average has between 6 month's and 1 year's development experience or with the specific environmental items listed.	Staff on average has between 1 and 2 years' development experience or with the specific environmental items listed..	Staff on average has more than 2 years' development experience or with the specific environmental items listed..

3) AXCIP - Integrator Experience with COTS Integration Processes

Does a formal and validated COTS integration process exist within your organization and how experienced was/is the development staff in that formal process?

Metric: a mix of conditions including SEI CMM level, ISO 9001 certification, and number of times integration team as a whole on average has used the defined COTS integration process.

Low	Nominal	High	Very High
CMM level =1 OR there is no formally defined COTS integration process.	[CMM level =2 OR ISO 9001 certified] AND there is a formally defined COTS integration process AND the integration team has never used the process before.	[CMM level =3 OR ISO 9001 certified] AND there is a formally defined COTS integration process AND the integration team has used the process 1 or 2 times before.	[CMM level > 3 OR ISO 9001 certified] AND there is a formally defined COTS integration process AND the integration team has used the process 3 or more times before.

4) APCON - Integrator Personnel Continuity

How stable was/is your integration team? Are the same people staying around for the duration of the task, or must you keep bringing in new people and familiarizing them with the particulars of the project because experienced personnel leave?

Metric: annual integration personnel turnover rate (a high personnel turnover rate implies a low personnel continuity).

Very Low	Low	Nominal	High	Very High
48% or more per year.	Between 24% and 47% per year.	Between 12% and 23% per year.	Between 6% and 11% per year.	5% or less per year.

COTS Component Drivers

5) ACPMT - COTS Product Maturity

How many copies have been sold or used previously of the major version (as opposed to release of that version) of the COTS component you integrated or intend to integrate? How long has the version been on the market or available for use? How large is the version’s market share or installed user base? How thoroughly has the version been used by others in the manner you used or intend to use it?

Metric: time on market.

Very Low	Low	Nominal	High	Very High
Version in pre-release beta test.	Version on market/available less than 6 months.	Version on market/available between 6 months and 1 year.	Version on market/available between 1 and 2 years.	Version on market/available more than 2 years.

6) ACSEW - COTS Supplier Product Extension Willingness

How willing was/is the supplier 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? In the case of COTS components, this refers to changes that would appear in market releases of the product. In the case of NDI components, this refers to changes that would appear in copies being distributed to all users of the component. This does NOT include specialty changes in the COTS component that would appear in your copy only.

Metric: number and nature of changes supplier will make.

Low	Nominal	High	Very High
Supplier will not make any changes.	Supplier will make a few minor changes.	Supplier will make one major change and several minor ones.	Supplier will make two or more major changes along with any minor changes needed.

7) APCPX - COTS Product Interface Complexity

What are the nature of the interfaces between the COTS component and the glue code connecting it to the main application? Are there difficult synchronization issues? Must the interface balance conflicting criteria (e.g., security, safety, accuracy, ease of use, speed)?

Metric: the scale for this driver uses a subjective average of the three equally weighted facets of interface complexity described in table VII.A on the following page. To rate this driver, first rate the three items (interface conventions, control aspects, data) in table VII.A individually according to the criteria given in the table. Next, sum the total point score as described in the table for the combination of ratings you selected, and determine which gross category that score corresponds to on the scale below. Finally, using your best engineering judgment, adjust your final rating for this driver above or below the center mark of the gross category as needed.

Example: individual ratings of Low for Interface , Low for Control , and Very High for Data would result in a point total of 9, indicating a gross combined rating of Nominal. To recognize the existence of at least one individual rating of Very High, however, it might be reasonable to circle the tic mark immediately to the right of the center tic mark in the Nominal category on the scale below when assigning a final rating for this driver.

Low	Nominal	High	Very High
Point total is between 5 and 7.	Point total is between 8 and 10.	Point total is between 11 and 13.	Point total is between 14 and 15.

Complexity Elements	Very Low (point value = 1)	Low (point value = 2)	Nominal (point value = 3)	High (point value = 4)	Very High (point value = 5)	Corresponding Points
Interface Conventions (e.g., naming, relevant usage scenarios, service signature, service order)	N/A	Nearly all API conventions are clear and consistent.	Most API conventions are clear and consistent.	Few API conventions are clear and consistent.	API conventions are non-existent.	-----
Control Aspects (e.g., consistent and clear error handling/recovery)	N/A	Nearly all control aspects are well defined and consistently applied.	Most control aspects are well defined and consistently applied.	Few control aspects are well defined and consistently applied.	No control aspects are well defined and consistently applied.	-----
Data (e.g., conversion, number/range typing)	No data conversion required.	Little data conversion required and standard data types used.	Some data conversion required and standard data types used.	Significant data conversion required and/or use of non-standard data types.	Extensive data conversion required and/or use of non-standard data types.	-----

Total Point Score = _____

Table VII.A - Facets of Complexity

Use this table in evaluating complexity drivers 7 (APCPX) and 12 (AAPX). Use it once for APCPX, then repeat its use for AAPX. Rate each complexity element described in the table individually, recording the point value associated with your rating in the far right column. Then sum all three point values to arrive at a total point score (minimum score possible is 5, maximum score is 15). Then apply that total point score to the scales provided for each of the two cost drivers as indicated under their descriptions.

8) ACPPS - COTS Supplier Product Support

What is the nature of the technical support for the COTS component that was/is available AND PROCURED for the integration team during the development, either directly from the component supplier or through third parties?

Metric: the level of support available and procured.

Low	Nominal	High	Very High
Product is unsupported.	Help desk support.	Trained technical support.	Formal consulting help.

9) ACPTD - COTS Supplier Provided Training and Documentation

How much training and/or documentation for the COTS component was/is available AND PROCURED for the integration team during the development, either directly from the component supplier or through third parties?

Metric: the amount of training and/or documentation available and procured.

Very Low	Low	Nominal	High	Very High
No training and very little documentation procured.	Roughly ¼ of the needed training and/or documentation procured.	Roughly ½ of the needed training and documentation procured.	Roughly ¾ of the needed training and/or documentation procured.	As much training and/or documentation procured as needed.

10) APVOL - COTS Product Volatility

How many releases of the COTS component (or patches to releases) were/can be expected to be issued by the component supplier during the development?

Metric: number of expected releases or patches during development.

Very Low	Low	Nominal	High	Very High
Zero.	One.	Two.	Three.	More than three.

APPLICATION/SYSTEM Drivers

11) ACREL - Constraints on System/Subsystem Reliability

How severe are the overall reliability constraints on the system or subsystem into which the COTS component was/is being integrated? What are the potential consequences if the component fails to perform as required in any given time frame? (Note that *availability* is considered an issue different than reliability and is NOT addressed in this cost driver.)

Metric: the potential threat if the component fails to perform as expected.

Low	Nominal	High	Very High
Threat is low; if a failure occurs losses are easily recoverable (e.g., document publishing).	Threat is moderate; if a failure occurs losses are fairly easily recoverable (e.g., support systems).	Threat is high; if a failure occurs the risk is to mission critical requirements.	Threat is very high; if a failure occurs the risk is to safety critical requirements.

12) AACPX - Application Interface Complexity

What are the nature of the interfaces between the main application system or subsystem and the glue code used to connect the system to the COTS component? Are there difficult synchronization issues? Must the interface balance conflicting criteria (e.g., security, safety, accuracy, ease of use, speed)?

Metric: the same subjective averaging of the items in table VII.A as used for the driver APCPX. See the explanation provided for rating that driver under item 7, and then repeat the use of table VII.A to evaluate this current driver 12.

Low	Nominal	High	Very High
Point total is between 5 and 7.	Point total is between 8 and 10.	Point total is between 11 and 13.	Point total is between 14 and 15.

13) ACPER - Constraints on System/subsystem Technical Performance

How severe were/are the technical performance constraints (e.g., storage, memory, reserve, flow through capacity, etc.) on the application system or subsystem that the COTS component needed to/must meet?

Metric: the presence or absence of constraints.

Nominal	High	Very High
There are no technical constraints or real time processing needs.	Real time processing must be performed OR other technical constraints exist.	Real time processing must be performed AND other technical constraints exist.

14) ASPR - System Portability

What were/are the overall system or subsystem portability requirements that the COTS component needed to/must meet?

Metric: the nature of portability requirements.

Nominal	High	Very High
There are no portability requirements at the system/subsystem level.	System must be portable across platforms within the same family (e.g., across different versions of UNIX).	System must be portable across divergent platforms (e.g., from UNIX to VMS).

Nonlinear Scale Factor

1) AAREN - Application Architectural Engineering

How adequate/sophisticated were the techniques used to define and validate the overall systems architecture?

Metric: architecture validation techniques.

Very Low	Low	Nominal	High	Very High
No architecture validation done.	Paper analysis performed.	Peer reviews of architectural design (including interface definitions).	Prototyping/demos of the architecture performed.	Simulations of the architecture created.

VIII. Definitions/Glossary

COTS software - “commercial-off-the-shelf” software commercially available as stand-alone products and which offer specific functionality needed by a larger system into which they might be incorporated. Generally there is no access to source code for COTS products, which are treated as black boxes with application program interfaces. (In some cases, however, some access to COTS source code is available, in which case these products have been described as “gray” or “white” box COTS.)

NDI software - “non-developmental item” software available from some source other than the organization developing the system into which the NDI component is to be integrated. The source can be commercial, private, or public sector, just so long as the procuring organization expended no resources on the NDI component’s initial development. Source code is usually available for an NDI component, which may or may not be able to function as a stand-alone item.

Integration or “glue” code - software developed in-house and composed of 1) code needed to facilitate data or information exchange between the COTS/NDI component and the system or other COTS/NDI component into which it is being integrated, 2) coded needed to connect or “hook” the COTS/NDI component into the system or other COTS/NDI component but does not necessarily enable data exchange, and 3) code needed to provide required functionality missing in the COTS/NDI component AND which depends upon or must interact with the COTS/NDI component.

User Guide 1.0

AA	Percentage of reuse effort due to assessment and assimilation
AAF	Adaptation Adjustment Factor
AAM	Adaptation Adjustment Multiplier
ASLOC	Adapted Source Lines of Code
BRAK	Breakage. The amount of controlled change allowed in a software development before requirements are "frozen."
CASE	Computer Aided Software Engineering
CM	Percentage of code modified during reuse
COCOMO	Constructive Cost Model
Cost Driver	A particular characteristic of the software development that has the effect of increasing or decreasing the amount of development effort, e.g. required product reliability, execution time constraints, project team application experience.
COTS	Commercial Off The Shelf
DI	Degree of Influence
DM	Percentage of design modified during reuse
ESLOC	Equivalent Source Lines of Code
FP	Function Points
GFS	Government Furnished Software
IM	Percentage of integration redone during reuse
KASLOC	Thousands of Adapted Source Lines of Code
KESLOC	Thousands of Equivalent Source Lines of Code
KSLOC	Thousands of Source Lines of Code
PM	Person Months. A person month is the amount of time one person spends working on the software development project for one month.
SEI	Software Engineering Institute
SLOC	Source Lines of Code
SU	Percentage of reuse effort due to software understanding
UNFM	Programmer Unfamiliarity

End of User Guide.