Cost Framework for COTS Evaluation

Norman F. Schneidewind
Naval Postgraduate School
Email: nschneid@nps.navy.mil

Cost as the Universal COTS Metric

We focus on factors that the user should consider when deciding whether to use COTS software. We take the approach of using the common denominator cost. This is done for two reasons: First, cost is obviously of interest in making such decisions and second a single metric – cost in dollars – can be used for evaluating the pros and cons of using COTS. The reason is that various software system attributes, like acquisition cost and availability (i.e., the percentage of scheduled operating time that the system is available for use), are non-commensurate quantities. That is, we cannot relate quantitatively “a low acquisition cost” with “high availability”. These units are neither additive nor multiplicative. However, if it were possible to translate availability into either a cost gain or loss for COTS software, we could operate on these metrics mathematically. Naturally, in addition to cost, the user application is key in making the decision. Thus one could develop a matrix where one dimension is application and the other dimension is the various cost elements. We show how cost elements can be identified and how cost comparisons can be made over the life of the software. Obviously, identifying the costs would not be easy. The user would have to do a lot of work to set up the decision matrix but once it was constructed, it would be a significant tool in the evaluation of COTS. Furthermore, even if all the required data cannot be collected, having a framework that defines software system attributes would serve as a user guide for factors to consider when making the decision about whether to use COTS software or in-house developed software.

Certainly, different applications would have varying degrees of relationships with the cost elements. For example, flight control software would have a stronger relationship with the cost of unavailability than a spreadsheet application. Conversely, the latter would have a stronger relationship with the cost of inadequacy of tool features than the former. Due to the difficulty of identifying specific COTS-related costs, our initial approach is to identify cost elements on the ordinal scale. Thus, the first version of the decision matrix would involve ordinal scale metrics (i.e., the cost of unreliability is more important for flight control software than for spreadsheet applications). As the field of COTS analysis matures and as additional data is collected about the cost of using COTS, we will be able to refine our metrics to the ratio scale (e.g., the cost of unreliability in COTS systems is two times that in custom systems).

The cost elements for comparing COTS software with in-house software are identified below. This list is not exhaustive; its purpose is to illustrate the approach. These elements apply whether we are comparing a system comprised of all COTS components with all in-house components or comparing only a subset of COTS components with corresponding in-house components. Explanatory comments are made where necessary. Mean values are used for some quantities in the initial framework. This is the case because it will be a challenge to collect any data for some applications. Therefore, the initial framework should not be overly complex. Variance and statistical distribution information could be included as enhancements if the initial framework proves successful.

Cost Elements

\[ C_i(j) = \text{Cost of acquiring COTS software in year } j. \]
\[ C_i(j) = \text{Cost of developing in-house software in year } j. \]
\[ U_i(j) = \text{Cost of upgrading COTS software in year } j. \]
\[ U_i(j) = \text{Cost of upgrading in-house software in year } j. \]
\[ P(j) = \text{Cost of personnel who use the software system in year } j. \] This quantity represents the value to the customer of using the software system.
\[ M_i(j) = \text{Cost per unit time of repairing a fault in COTS software in year } j. \] This is the cost of customer time involved in resolving a problem with the vendor.
\[ M_i(j) = \text{Cost per unit time of repairing a fault in in-house software in year } j. \]
\( R_c(j) = \text{Mean time of repairing a fault that causes a failure in COTS software in year } j. \) This is the average time that the user spends in resolving a problem with the vendor.

\( R_i(j) = \text{Mean time of repairing a fault that causes a failure in in-house software in year } j. \)

\( T(j) = \text{Scheduled operating time for the software system in year } j. \)

\( A_c(j) = \text{Availability of software system that uses COTS software in year } j. \)

\( A_i(j) = \text{Availability of software system that uses software developed in-house in year } j. \)

These quantities are the fractions of \( T(j) \) that the software system is available for use.

\( F_c(j) = \text{Failure rate of COTS software in year } j. \)

\( F_i(j) = \text{Failure rate of COTS software in year } j. \)

These quantities are the number of failures per year that cause loss of productivity and availability of the software system.

In some applications, some or all of the above quantities may be known or assumed to be constant over the life of the software system. Using the above cost elements, we derive the equations for the annual costs of the two systems and the difference in these costs. In the cost difference calculations that follow, a positive quantity is favorable to in-house development and a negative quantity is favorable to COTS.

### Cost of Acquiring Software

\[
\text{Difference in annual cost} = C_c(j) - C_i(j) \quad (1)
\]

### Cost of Upgrading Software

\[
\text{Difference in annual cost} = U_c(j) - U_i(j) \quad (2)
\]

### Cost of Software being Unavailable for Use

Annual cost of COTS software being unavailable for use

\[
= (1 - A_c(j)) \times P(j).
\]

Annual cost of the in-house software being unavailable for use

\[
= (1 - A_i(j)) \times P(j).
\]

\[
\text{Difference in annual cost} = P(j) \times (A_i(j) - A_c(j)) \quad (3)
\]

### Cost of Repairing Software

Average annual cost of repairing failed COTS software

\[
= F_c(j) \times T(j) \times R_c(j) \times M_c(j).
\]

Average annual cost of repairing failed in-house software

\[
= F_i(j) \times T(j) \times R_i(j) \times M_i(j).
\]

\[
\text{Difference in annual cost} = T(j) \times (F_c(j) \times R_c(j) \times M_c(j)) - (F_i(j) \times R_i(j) \times M_i(j)) \quad (4)
\]

Then, \( T_C \), total difference in cost in year \( j \), is the sum of (1), (2), (3), and (4). Because there is the opportunity to invest funds in alternate projects, costs in different years are not equivalent (i.e., funds available today have more value than an equal amount in the future because they could be invested today and earn a future return). Therefore, a stream of costs over the life of the software for \( n \) years must be discounted by \( k \), the rate of return on alternate use of funds. Thus the total discounted cost differential between COTS software and in-house software is:

\[
\sum T_C/(1+k)^n
\]

In this initial formulation, we have not included possible differences in functionality between the two approaches. However, a reasonable assumption is that COTS software would not be considered unless it could provide minimum functionality to satisfy user requirements. Thus, a typical decision for the user is whether it is worth the additional life cycle costs to develop an in-house software system with all the desirable attributes.