A fuzzy approach for optimal selection of COTS components for modular software system under consensus recovery block scheme incorporating execution time

P.C. Jha*
Department of Operational Research,
University of Delhi, Delhi - 110007, India
E-mail: jhapc@yahoo.com
*Corresponding author

Shivani Bali
Lal Bahadur Shastri Institute of Management,
Dwarka, Delhi – 110075, India
E-mail: lbsshivani@gmail.com

U. Dinesh Kumar
Indian Institute of Management,
Banglore – 560076, India
E-mail: udkumar@gmail.com

Abstract

Today almost everyone in the world is directly or indirectly affected by computer systems. Computers are used in diverse areas for various applications including air traffic control, nuclear reactors, industrial process control, hospital health care etc. affecting millions of people. As the functionality of computer operations become more essential and yet more critical, there is a great need for the development of modular software system. Modular software systems developed by assembling COTS components have become a trend in integrated modern software systems. The number of functions to be included in a software system is decided during the software development. On an invocation of a function, the module is called which in turn calls alternatives. Execution times of all the alternatives are different. We have proposed a comprehensive approach that can be used to systematically evaluate component against selection criteria for functionality, execution time, fault tolerance and quality attributes. We have developed a model for enabling the selection of COTS components by maximizing reliability and minimizing the absolute deviational execution time which in turn minimizes the overall execution time of the software system. The model uses basic information on components reliability and execution time and allows the trade-off between two of them. Conventional optimization methods assume that all the parameters and goals of an optimization model are precisely known. However, in practice, it is not possible for a management to get precise value of reliability, execution
time and cost for a software system. When the precise values of parameter of the problem are not known, the problem can be formulated as a fuzzy optimization problem. A fuzzy multi objective optimization model for selecting COTS component based on system reliability and execution time of the software under budgetary constraint. Further an equivalent crisp mathematical programming problem is formulated using suitable membership function and solution is obtained for the same. Numerical illustrations are provided to demonstrate the model developed.

**Keywords:** Modular software, software reliability, COTS products, fault tolerance, fuzzy optimization

### 1. Introduction

Since 1970 many researches have been conducted to study the reliability of computer software. As software systems have become more complex to design and develop, intensive studies are carried out to increase the chance that software systems will perform satisfactorily in operation. Therefore, high degree of reliability of such systems is desired. In order to increase the performance of the software product and to improve the software development process, we must make thorough analysis of reliability. Reliability assessment methods and improvement techniques have a great value both to software managers and software practitioners. Methods such as structural programming, modular programming, and fault tolerance have been widely studied in order to produce more reliable software. Achievement of high reliability of operation through the use of redundant system elements is a fundamental principle in fault tolerance of hardware (physical) faults (Aviziens, 1971). The use of redundant software to recover from software malfunction, however, requires special caution due to the idiosyncratic characteristics of software. Software whose failure can have severe repercussions can be made fault tolerant through redundancy at module level (Belli & Jadrzejowich, 1993). Improving software reliability using redundancy however requires additional resources. Considering the concept of COTS in software development, the possibility of having redundant components within a specified budget can be explored. Performing a good COTS selection plays a critical role in the success of the final system (Maiden & Ncube, 1998). COTS selection involves many challenges such as high complexity of the process (Ruhe, 2003).

Several optimization models have been proposed in the literature for optimal selection of the COTS software components for the development of safe and reliable software systems. At the outset (Scott and Gault, 1987) examined three methods of creating fault tolerant software systems, Recovery Block Scheme, N-Version Programming, and Consensus Recovery Block. The authors presented reliability models for each technique. The models are used to show that one method, the Consensus Recovery Block, is more reliable than the other two. (Allister and Scott, 1991) compared cost of a single version system with the three versions of fault tolerant software systems. The authors showed that in cases where failures are independent, Consensus Recovery Block followed by Recovery Block is the most cost justifiable fault tolerant techniques to be considered. (Ashrafi and Berman, 1992) proposed two optimization models that address the tradeoff between reliability and cost. They applied their models to large software packages that consisted of several programs. The authors used these
optimization models to determine the redundancy level of a software package consisting of several independent functions where each function was performed by program with known reliability and cost. Berman and Ashrafi (1993) however, broke down this approach one step further and dealt with software systems consisting of one or more programs where each program consisted of series of modules, which upon sequential execution would perform a function. The optimal redundancy level of the modules was to be determined. Berman and Dinesh (1999) presented optimization models for a fault tolerant software by selecting a set of versions for a given program. The objective was to maximize the reliability of the software satisfying the budget limitation. (Kapur et. al, 2003) have chosen the recovery block reliability model for COTS based software system. Two optimization models, for optimal selection of components have been proposed. (Jha, 2003) chose the Recovery Block and Consensus Recovery Block model for COTS based software System. The objective was to maximize the system reliability with the budget constraint. Compatibility was considered for both the models. Pankaj et al. (2009) formulated fuzzy multi objective optimization models for selecting the optimal COTS software products in the development of software system based on COTS.

In this paper, we have chosen the Consensus Recovery Block reliability model for COTS based software system with some simplifying assumptions. Large software system has modular structure to perform set of functions with different modules having different alternatives and different versions for each alternative. A schematic representation of the software system is given in figure 1. On the execution of a software system, the functions are invoked. The frequency with which the functions are used is not the same for all of them and not all the modules are called during the execution of the function, the software has in its menu.

The execution times for all the functions called in the system are not necessarily identical. The execution time may vary because of the variations in the number of alternatives present in each module that are being called by the functions and also the nature of task to be performed by the components of the modules. On invocation of the function, the module is called. All the alternatives of that module get executed simultaneously. Execution times of all the alternatives are different. Alternatives whose execution complete early, will have to wait for their corresponding alternatives to complete their job. Therefore, our objective here is to minimize this waiting time. To resolve this problem, the absolute deviational execution time has been introduced. An average time for the execution of an alternative has been assumed, and to solve the problem the deviation from the average time has been taken. The deviation from the average time is used to address two issues. Firstly, because of those alternatives whose execution completes early before the assumed average time. And secondly, those alternatives whose execution takes longer time to complete, in other words the execution completes after the assumed average time. Therefore, in both the cases there is a deviation from average time. So this paper aims at minimizing this absolute deviational execution time by simultaneously maximizing the system reliability.

In the existing research in this area it is assumed that a crisp or a constant value of all the parameters is known. However, in practice, it is not possible for a management to get precise value of reliability, execution time and cost for a software system. Or it may
happen that they decide not to set precise levels due to the market considerations and are ready to have some tolerance of their objectives. When the precise values of parameter of the problem are not known, the problem becomes a fuzzy optimization problem and the solution so obtained is a fuzzy approximation.

This paper proposes two fuzzy multi-objective optimization models for selecting the best COTS software product for each module. The first optimization model (optimization model-I) of this paper is a joint optimization problem that maximizes the system reliability with simultaneously minimizing the absolute deviational execution time. The second optimization model (optimization model-II) considers the issue of compatibility between different alternatives of modules as it is observed that some COTS components cannot integrate with all the alternatives of another module. We assume the existence of virtual versions, apart from available versions, having negligible reliabilities and zero costs. Virtual versions are chosen only when we have insufficient budget. In a situation where this particular version is chosen, the corresponding alternative is not to be added to the system.

The rest of this paper is organized as follows. Section 2 consists of proposed notations. In section 3, we develop a crisp model for reliability and absolute deviational execution time and in section 4, we describe fuzzy membership functions in respect of both the chosen objectives, viz. the reliability and the absolute deviational execution time. In this section, we also present fuzzy multi-objective optimization models for selecting the best COTS product for each module. In Section 5 numerical example is illustrated. Section 6, we furnish our concluding observations.

2. Notations

\[ R_i \] : System quality measure
\[ f_l \] : Frequency of use of function \( l \)
\[ s_l \] : Set of modules required for function \( l \)
\[ R_i \] : Reliability of module \( i \)
\[ L \] : Number of functions, the software is required to perform
\[ n \] : Number of modules in the software.
\[ m_i \] : Number of alternatives available for module \( i \)
\[ V_{ij} \] : Number of versions available for alternative \( j \) of module \( i \)
\[ c_{ijk} \] : Cost of version \( k \) of alternative \( j \) of module \( i \) (COTS)
\[ t_1 \] : Probability that next alternative is not invoked upon failure of the current alternative
\[ t_2 \] : Probability that the correct result is judged wrong
\[ t_3 \] : Probability that an incorrect result is accepted as correct
\[ Y_{ij} \] : Event that correct result of alternative \( j \) of module \( i \) is accepted
\[ X_{ij} \] : Event that output of alternative \( j \) of module \( i \) is rejected
\[ r_{ij} \] : Reliability of alternative \( j \) of module \( i \)
\[ r_{ijk} \] : Reliability of version \( k \) of alternative \( j \) of module \( i \)
\[ t_{li} \] : Average execution time of module \( i \) of function \( l \)
\( t_{ijk} \): Actual execution time of version \( k \) of alternative \( j \) of module \( i \) of function \( l \)

\( C \): Total budget available for all modules

\( x_{ijk} = \begin{cases} 
1, & \text{if version } k \text{ of alternative } j \text{ of module } i \text{ is selected} \\
0, & \text{otherwise}
\end{cases} \)

\( z_{ij} \): Binary variable taking value 0 or 1

\( \begin{cases} 
1, & \text{if alternative } j \text{ is present in module } i \\
0, & \text{otherwise}
\end{cases} \)

### 3. Multi-objective optimization model for COTS selection

In this section, we formulate COTS software products selection problem with multiple objectives of reliability maximization and absolute deviational execution time minimization. In the optimization models it is assumed that the alternatives of a module are in a consensus recovery block (Scott and Gault, 1987). Consensus recovery block requires independent development of independent alternatives of a program, which the COTS components satisfy and a voting procedure. Upon invocation of the consensus recovery block all alternatives are executed and their outputs are submitted to a voting procedure. Since it is assumed that there is no common fault, if two or more alternatives agree on one output then that output is designated as correct. Otherwise the next stage is entered. At this stage the best version is examined by an acceptance test. If the output is accepted, it is treated as the correct one. However if the output is not accepted, the next best version is subject to testing. This process continues until an acceptable output is found or all outputs are exhausted.

The first optimization model is developed for the following situations, which also holds good for the second model, but with additional assumptions related to compatibility among alternatives of a module. The following assumptions are common for the optimization models:

- \( a) \) Software system consists of a finite number of modules.
- \( b) \) Software system is required to perform a known number of functions. The program written for a function can call a series of modules \( (\leq n) \). A failure occurs if a module fails to carry out an intended operation.
- \( c) \) Codes written for integration of modules do not contain any bug.
- \( d) \) Several alternatives are available for each module. Fault tolerant architecture is desired in the modules (it has to be within the specified budget). Independently developed alternatives (primarily COTS components) are attached in the modules and work similar to the recovery block scheme discussed in (Berman and Dinesh, 1999, 1999)
- \( e) \) Execution times of all the alternatives, is different. Those alternatives whose execution complete early, will have to wait for their corresponding alternatives to finish up their job.
- \( f) \) Absolute deviational execution time is the absolute difference between the average time and the actual execution time of the software.
- \( g) \) The cost of an alternative is the buying price for the COTS product. Reliability for all the components is known.
Different versions with respect to time, reliability and cost of modules are available.

Other than available time-reliability-cost versions of an alternative, we assume the existence of a virtual version, which has a negligible reliability of 0.001 and zero execution time and cost. These components are denoted by index one in the third subscript of \( x_{ijk}, c_{ijk} \) and \( r_{ijk} \). For example, \( r_{ij} \) denotes the reliability of first version of alternatives \( j \) for module \( i \).

### 3.1. Optimization model I

**Problem (P1)**

Maximize  
\[
R = \sum_{l=1}^{L} f_l \prod_{l \in s_l} R_i
\]  

(1)

Minimize  
\[
\sum_{l=1}^{L} f_l \sum_{i=1}^{m_i} \sum_{j=1}^{m_j} \sum_{k=1}^{m_k} \epsilon_{ijk} x_{ijk}
\]

(2)

Subject to

\[
\sum_{i=1}^{m_i} \sum_{j=1}^{m_j} \sum_{k=1}^{m_k} c_{ijk} x_{ijk} \leq C
\]

(3)

\( X \in S = \{ x_{ijk} \text{ is binary variable} \} \)

\[
R_i = 1 + \left[ \sum_{k=1}^{m_k} \frac{1}{n_i} \left( \prod_{l \in s_l} (1 - r_{ij})^{s_l} \right) \left( 1 - \left( 1 - r_{ij} \right)^{s_l} \right) \right] \left( \prod_{l \in s_l} P(X_{ik})^{s_l} \right) P(Y_j)^{\epsilon_{ijk}} - 1 ; \; i = 1, 2, \ldots, n
\]

(4)

\[
P(X_{ij}) = (1 - t_1) \left[ (1 - t_2) (1 - t_3) + t_2 t_3 \right]
\]

(5a)

\[
P(Y_j) = r_{ij} (1 - t_2)
\]

(5b)

\[
r_{ij} = \sum_{k=1}^{V_k} x_{ijk} r_{ijk} \quad j = 1, 2, \ldots, m_j \text{ and } i = 1, 2, \ldots, n
\]

(6)

\[
\sum_{k=1}^{V_j} x_{ijk} = 1, \text{ for } j = 1, 2, \ldots, m_j \text{ and } i = 1, 2, \ldots, n
\]

(7)

\[
x_{i1} + z_{ij} = 1; \; j = 1, 2, \ldots, m_i
\]

(8)

\[
\sum_{j=1}^{m_j} z_{ij} \geq 1; \; i = 1, 2, \ldots, n
\]

(9)

Where \( X \) is a vector of component \( x_{ijk} \) \( i=1, \ldots, n ; j=1, \ldots, m_i ; k=1, \ldots, V_{ij} \)
and $|t_{ijk} - t_i| = \epsilon_{ijk}$ gives the absolute deviational execution time of an individual component, i.e. average execution time subtracted from actual execution time.

Objective function (1) maximizes the system quality (in terms of reliability) through weighted sums of reliabilities of sequential modules required to perform different functions. Reliability of modules that are invoked more frequently during use is given higher weights. Analytic Hierarchy Process (AHP) can be effectively used to calculate these weights and (2) minimize the absolute deviational execution time. Constraint (3) is a budget constraint. Constraint (4) estimates the reliability of module $i$. As it has been assumed that the exception raising and control transfer programs work perfectly, a module fails if all attached alternatives fail. Constraint (5a) is the probability of event that output of alternative $j$ of module $i$ is rejected and Constraint (5b) is the probability of event that correct result of alternative $j$ of module $i$ is accepted. Constraint (6) gives the reliability of alternative $j$ of module $i$. Constraint (7) ensures that exactly one version is chosen from each alternative of a module. It includes the possibility of choosing a dummy version. Equation (8) and (9) guarantee that not all chosen alternatives of modules are dummies.

The absolute deviational execution time can be calculated as follows: $(t_{ijk} - t_i) = \epsilon_{ijk}$ if $t_{ijk} > t_i$; $(t_i - t_{ijk}) = \epsilon_{ijk}$ if $t_{ijk} \leq t_i$

### 3.2. Optimization model II

As explained in the introduction, it is observed that some alternatives of a module may not be compatible with alternatives of another module. The next optimization model II addresses this problem. It is done, incorporating additional constraints in the optimization model I. This constraint can be represented as $x_{gsq} \leq x_{huc}$, which means that if alternative $s$ for module $g$ is chosen, then alternative $u$, $t = 1, \ldots, z$ have to be chosen for module $h$. We also assume that if two alternatives are compatible, then their versions are also compatible.

$$x_{gsq} - x_{huc} \leq My_i, \quad q = 2, \ldots, V_{gs}, \quad c = 2, \ldots, V_{hu}, \quad s = 1, \ldots, m_g$$

$$\sum y_i = z(V_{hu} - 2)$$

Constraints (3) to (9) of optimization model I are also the constraints of optimization model II in addition to the compatible constraints. Constraint (10) and (11) make use of binary variable $y_i$ to choose one pair of alternatives from among different alternative pairs of modules. If more than one alternative compatible component is to be chosen for redundancy, constraint (11) can be relaxed as follows.

$$\sum y_i \leq z(V_{hu} - 2)$$
Optimization Model II can be re-written as:

**Problem (P2)**

Maximize  
\[ R = \sum_{l=1}^{L} f_l \prod_{i \in R_l} R_i \]

Minimize  
\[ \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{V_0} \epsilon_{ijk} x_{ijk} \]

Subject to  
\[ \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{V_0} c_{ijk} x_{ijk} \leq C \]
\[ \sum_{i=1}^{n} y_i = z(V_{hu} - 2) \]
\[ X \in S \]

**4. Fuzzy multi-objective optimization model for COTS selection**

Conventional optimization methods assume that all parameters and goals of an optimization model are precisely known. But for many practical problems there are incompleteness and unreliability of input information. This is caused to use fuzzy multi-objective optimization method with fuzzy parameters. In many practical design situations, reliability apportionment is complicated because of the presence of several conflicting objecting objectives and imprecise statement regarding budget and maximum allowable time to execute any function of the software. For instance, a designer is required to minimize the system cost and execution time for each function while simultaneously maximizing the system reliability. Therefore multiple objective functions become an important aspect in the reliability design of the engineering systems. Several techniques have been developed for solving constrained optimal-reliability allocation problems.

In general reliability optimization problem is solved with the assumption that the coefficients or cost of components is specified in a precise way. In real life, there are many diverse situations due to uncertainty in judgments, lack of evidence, etc. Sometimes it is not possible to get relevant precise data for the reliability system and execution time. This type of imprecise data is not always well represented by random variable selected from a probability distribution. Fuzzy number may represent this data, so fuzzy optimization method with fuzzy parameters is needed for a fuzzy reliability optimization model.

Therefore, we formulate fuzzy multi-objective optimization model for COTS software products selection based on vague aspiration levels, the decision maker may decide his aspiration levels on the basis of past experience and knowledge possessed by him. To express vague aspiration levels of the decision, various membership functions have been proposed (Zimmermann, 1976 & 1978).
4.1 Optimization model III

Following algorithm specifies the sequential steps to solve the fuzzy mathematical programming problems.

**Step I:** Compute the crisp equivalent of the fuzzy parameters using a defuzzification function. Same defuzzification function is to be used for each of the parameters. Here we use the defuzzification function of the type

\[ F_2(A) = \left( a^1 + 2a^2 + a^3 \right) / 4 \]

where \( a^1, a^2, a^3 \) are the triangular fuzzy numbers.

**Step II:** Incorporate the objective function of the fuzzifier min (max) as a fuzzy constraint with a restriction (aspiration) level. The above problem (P1) can be rewritten as

**Problem (P3)**

Find \( X \)

Subject to

\[ R(X) = \sum_{i=1}^{l} f_i \prod_{\alpha_k} R_{i} \geq R_0 \]

\[ T(X) = \sum_{i=1}^{l} \sum_{j=1}^{n} \sum_{k=1}^{m} c_{ijk} x_{ijk} \leq T_0 \]

\[ C(X) = \sum_{i=1}^{l} \sum_{j=1}^{n} \sum_{k=1}^{m} c_{ijk} x_{ijk} \leq C_0 \]

\( X \in S \)

**Step III:** Define appropriate membership functions for each fuzzy inequalities as well as constraint corresponding to the objective function. The membership function for the fuzzy less than or equal to and greater than or equal to type are given as

\[ \mu_{\delta}(X) = \begin{cases} 
1 & ; R(X) \geq R_0 \\
\frac{R(X) - R_0}{R_0 - R_0^*} & ; R_0^* \leq R(X) < R_0 \\
0 & ; R(X) < R_0^*
\end{cases} \]

Where \( R_0 \) is the aspiration level and \( R_0^* \) is the tolerance levels to the fuzzy reliability objective function constraint.

\[ \mu_{\gamma}(X) = \begin{cases} 
1 & ; T(X) \leq T_0 \\
\frac{T_0^* - T(X)}{T_0^* - T_0} & ; T_0 \leq T(X) < T_0^* \\
0 & ; T(X) > T_0^*
\end{cases} \]

Where \( T_0 \) is the restriction and \( T_0^* \) is the tolerance levels to the fuzzy absolute deviational execution time objective function constraint.
Where \( C_0 \) is the restriction and \( C_0^* \) is the tolerance levels to the fuzzy budget constraint.

**Step IV:** Employ extension principle to identify the fuzzy decision, which results in a crisp mathematical programming problem given by

\[
\begin{align*}
\text{Maximize} & \quad \alpha \\
\text{Subject to} & \quad \mu_R(X) \geq \alpha, \\
& \quad \mu_T(X) \geq \alpha, \\
& \quad \mu_C(X) \geq \alpha, \\
& \quad X \in S
\end{align*}
\]

This problem can be solved by the standard crisp mathematical programming algorithms.

**Step V:** While solving the problem, objective of the problem is also treated as a constraint. Each constraint is considered to be an objective for the decision maker and the problem can be looked as a fuzzy multiple objective mathematical programming problem. Further each objective can have different level of importance and can be assigned weights according to their relative importance. The resulting problem can be solved by the weighted minmax approach. The crisp formulation of the weighted problem is given as

\[
\begin{align*}
\text{Maximize} & \quad \alpha \\
\text{Subject to} & \quad \mu_R(X) \geq w_1 \alpha, \\
& \quad \mu_T(X) \geq w_2 \alpha, \\
& \quad \mu_C(X) \geq \alpha, \\
& \quad X \in S \\
& \quad w_1, w_2 \geq 0, \quad w_1 + w_2 = 1
\end{align*}
\]

where, \( \alpha \) represents the degree up to which the aspiration of the decision maker is met.

If the constraints are fuzzy as well as crisp, then in the equivalent crisp mathematical programming problem, the original crisp constraints will not have any change as for them the tolerances are zero except for those constraints which are fuzzy in nature. The problem (P5) can be solved using standard mathematical programming approach.
**Step VI:** On substituting the values for $\mu_R(X), \mu_T(X)$ and $\mu_C(X)$ the problem becomes

\[
\text{Problem (P6)}
\]

Maximize $\alpha$

Subject to

\[
\begin{align*}
R(X) & \geq R_0 - (1 - w_1 \alpha)(R_0 - R^*_0) \\
T(X) & \leq T_0 + (1 - w_2 \alpha)(T_0 - T^*_0) \\
C(X) & \leq C_0 + (1 - \alpha)(C^*_0 - C_0)
\end{align*}
\]

$X \in S$

$\alpha \in [0,1]$

$X \geq 0$

$w_1, w_2 \geq 0$, $w_1 + w_2 = 1$

**Step VII:** If a feasible solution is not obtainable for the problem (P5) or (P6) then we can use fuzzy goal programming approach to obtain a compromised solution [Mohamed, (1997)]. The method is discussed in detail in the numerical illustration.

### 5. Numerical illustration

Consider a software system having three modules with more than one alternative for each module. The cost reliability data set is given in Table-I. Note that the cost of first version i.e. the virtual versions for all alternatives is zero and reliability is 0.001. This is done for the following reason: If in the optimal solution, for some module, $x_{ij} = 1$, that implies corresponding alternative is not to be attached in the module. Let $L = 3$, $s_1 = \{1,2,3\}$, $s_2 = \{1,3\}$, $s_3 = \{2\}$, $f_1 = 0.5$, $f_2 = 0.3$ and $f_3 = 0.2$. It is also assumed that $t_1 = .01, t_2 = .05$ and $t_3 = .01$. The data set for execution time is further given in Table II, III & IV for respective functional requirements.

**Structure of Software**

![Figure 1. Structure of the software](image-url)
### Data Set Execution Time

#### Table I. Data Set Cost and Reliability

<table>
<thead>
<tr>
<th>Module</th>
<th>Alternatives</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Table II. Function 1

<table>
<thead>
<tr>
<th>Modules</th>
<th>Alternatives</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.00</td>
</tr>
</tbody>
</table>

#### Table III. Function 2

<table>
<thead>
<tr>
<th>Modules</th>
<th>Alternatives</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Table IV. Function 3

<table>
<thead>
<tr>
<th>Modules</th>
<th>Alternatives</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>0.15</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>0.03</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>0.52</td>
</tr>
<tr>
<td>4</td>
<td>0.00</td>
<td>0.14</td>
</tr>
</tbody>
</table>

Assignment of Weights

The assignment of weights is based on the expert’s judgement for the reliability and the absolute deviational execution time criteria. Weights assigned for reliability and deviational execution time are 0.6 and 0.4 respectively.

Minimum and Maximum level of Reliability, Deviational Execution Time and Cost.

Firstly, the triangular fuzzy reliability, absolute deviational execution time and cost values are computed using fuzzied values of these parameters and then defuzzied using Heilpern’s defuzzier. If the available reliability, absolute deviational time and cost are specified as TFN given as follows:

\[
\hat{R} = (0.88, 0.92, 0.96)
\]

\[
\hat{E} = (0.56, 0.60, 0.64)
\]

\[
\hat{C} = (42, 46, 50)
\]

The aspiration level for reliability is \( R_0 = 0.92 \) and the restriction on execution time and cost are \( T_0 = 0.60 \) and \( C_0 = 46 \) respectively.

The tolerance level for reliability, absolute deviational execution time and cost are \( R_0^* = 0.87 \), \( T_0^* = 0.67 \) and \( C_0^* = 52 \).

5.1 Fuzzy Goal Programming Approach

On solving the problem, we found that the problem (P6) is not feasible; hence the management goal can’t be achieved for a feasible value of \( \alpha \in [0,1] \). Now we use fuzzy goal programming technique to obtain a compromised solution. The approach is based on the goal programming technique for solving crisp goal programming problem [Mohamed, (1997)]. The maximum value of any membership function can be 1; maximization of \( \alpha \in [0,1] \) is equivalent to making it as close to 1 as best as possible.
This can be achieved by minimizing the negative deviational variables of goal programming (i.e. $\eta$) from 1. The fuzzy goal programming formulation for the given problem (P6) introducing the negative and positive deviational variables $\eta_j$ and $\rho_j$ is given as

$$\text{Problem (P7)}$$

**Minimize** $u$

**Subject to**

$$\begin{align*}
\mu_K(X) + \eta_1 - \rho_1 &= 1 \\
\mu_T(X) + \eta_2 - \rho_2 &= 1 \\
\mu_C(X) + \eta - \rho &= 1 \\
u &\geq w_j * \eta_j ; \\
\eta_j * \rho_j &= 0 ; \quad \eta_j, \rho_j \geq 0 \\
X &\in S ; \quad \alpha \in [0,1] ; \quad w_1, w_2 \geq 0 ; \quad w_1 + w_2 = 1 ; \quad \alpha = 1 - u
\end{align*}$$

**Optimization Model I**

The problem is solved using software package LINGO. Reliability is considered to be an important attribute in software reliability; therefore higher weights have been attached to reliability objective in comparison to the deviational time objective. Following solution is obtained. On solving the model, following COTS components were selected.

$$\begin{align*}
x_{112} = x_{123} &= x_{132} = 1 \\
x_{211} = x_{222} = x_{232} &= x_{243} = 1 \\
x_{311} = x_{322} &= 1
\end{align*}$$

It is observed that two or more alternatives are chosen for each module. Redundancy is allowed for first and second modules. The values of the fuzzy objectives and constraints are given in Table V

**TABLE V. Optimization Model I Solution**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Weights to Reliability = 0.6 and Absolute Deviational Time = 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Reliability</td>
<td>0.91</td>
</tr>
<tr>
<td>Execution Time</td>
<td>1.41</td>
</tr>
<tr>
<td>Deviational Time</td>
<td>0.61</td>
</tr>
<tr>
<td>System Cost</td>
<td>47</td>
</tr>
</tbody>
</table>

The achievement level of the membership function is $\alpha = 0.91$. 

58
Optimization Model II

To illustrate optimization model for compatibility, we use the results of optimization model I. We assume that second alternative of first module is compatible with second and third alternatives of second module. Following solution was obtained using LINGO.

\[
\begin{align*}
    x_{112} &= x_{123} = x_{132} = 1 \\
    x_{211} &= x_{222} = x_{233} = x_{243} = 1 \\
    x_{311} &= x_{332} = 1
\end{align*}
\]

It is observed that due to compatibility condition third alternative of second module is compatible with second alternative of first module.

**TABLE VI. Optimization Model II Solution**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Weights to Reliability = 0.6 and Absolute Deviational Time = 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Reliability</td>
<td>0.91</td>
</tr>
<tr>
<td>Execution Time</td>
<td>1.58</td>
</tr>
<tr>
<td>Deviational Time</td>
<td>0.50</td>
</tr>
<tr>
<td>System Cost</td>
<td>49</td>
</tr>
</tbody>
</table>

The achievement level of the membership function is \( \alpha = 0.91 \).

6. Conclusions

In this paper we have presented fuzzy multi objective optimization models for selecting the optimal COTS software products in the development of software system based on COTS. The problem is formulated for Consensus Recovery Block fault tolerant software system. It may be appreciated that when different alternatives of the same module are available with variations in the attributes of reliability and absolute deviational execution time, then it involves multi objective decision making environment that befits more of fuzzy approximation than deterministic formulation. Therefore, we have drawn fuzzy methodology for the estimation of reliability and absolute deviational execution time. This developed approach can effectively deal with the vagueness and subjectivity of experts’ information. Optimization model I deals with the optimal selection of COTS components for a modular software system. Fuzzy predictions of the triangular fuzzy statistical data have been defuzzified using Heipern’s defuzzifier and a crisp multi-objective component selection model has been developed using the defuzzified values. The component selection problem is formulated as a multi-objective programming problem and fuzzy goal programming technique is used to provide a feasible solution. In Optimization model II issue of compatibility amongst the alternatives of the modules is discussed.
References


Appendix

It is mentioned in the paper that the actual execution time of a dummy version is zero. Data set for actual execution time is mentioned above. This paper attempts to find out the absolute deviational execution time of those components that get executed. Deviational time is defined as:

\[ \text{Deviational Executional time} = \text{Actual Execution Time} - \text{Average Time} \]

Here we are taking the transformed tables (Table VII, VIII and IX) where the actual execution of the dummy version is the average time so that in the final table, the deviational execution time of the dummy version becomes zero.

**Table VII.** Function 1

<table>
<thead>
<tr>
<th>Modules</th>
<th>Alternatives</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.16</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.16</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.11</td>
</tr>
</tbody>
</table>

**Table VIII.** Function 2

<table>
<thead>
<tr>
<th>Modules</th>
<th>Alternatives</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.16</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.11</td>
</tr>
<tr>
<td>Modules</td>
<td>Alternatives</td>
<td>Versions</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.16</td>
</tr>
</tbody>
</table>