CHAPTER 2
OPTIMAL COMPONENT SELECTION FOR COTS BASED MODULAR SOFTWARE SYSTEM

The importance of fault tolerance and reliability issues in real time computer control systems can easily be appreciated in the context of, the ever increasing use of computers in various application areas such as

- Control of hazardous chemical plants and nuclear reactors in process industry
- Battle management and weapon delivery in defense
- Intensive care and diagnostic systems in health care
- Control systems for air and high speed ground transportation

The use of computers in such systems for fault detection, diagnosis, and system reconfiguration, has the potential of dramatically improving the operational effectiveness of real time systems. The computer system being the principal component of monitoring and control equipment, its failure could result in disastrous consequences, and hence, such a system should be installed only after adequate demonstration of its required level of reliability.

There are two structural methodologies for fault tolerant system i.e. Recovery Block schemes and N-Version Scheme. The basic mechanism of both the schemes is to provide redundant software modules to tolerate software failures at system level. A careful use of redundancy may allow the system to tolerate faults generated during software design and coding thus improving software reliability. Improving software reliability, using redundancy, however, requires additional resources, such as

This chapter is based on the following papers
additional cost and additional hardware requirement. Therefore, the redundancy level to achieve fault tolerance must be carefully determined, and if possible, optimized. Belli and Jadrzejowicz in (1991) have studied two optimization models, one for recovery block and the other for N-version programming. They discussed the difficulties of identifying the dependencies between the resources and component reliabilities. Lack of necessary information on component reliability was also presented.

The growing availability of COTS (Commercial off-the Shelf) components in the software market has concretized the possibility of building whole systems based on components. In this multitude, a recurrent problem is the selection of the components that best fit the requirements. The development of COTS based systems largely depends on the success of selection process. COTS describe software or hardware products that are ready-made or available for sale, lease or license to the general public. If these COTS components are not available economically in the market then they can be developed within the organization. A COTS product is one that is used “as it is”. These products are designed to be easily installed and to interoperate with existing system components. Developers of the commercial product integrate new technologies and new standard into the product faster than an organization built software. In spite of many pros, these products have cons too like developers may or may not change their source code, unavailability of complete and correct specifications, sometime the set of COTS components may be mismatched, etc. Since COTS products have many disadvantages but their use is increasing day by day due to their economic benefits [Vigder, (1998)].

Many COTS components are available to perform substantial portions of the work that we used to have to develop ourselves. These range from smaller, widely-used components like date manipulation packages to larger, more application-specific components like general ledger packages, military map displays, etc. With advent of new technologies the complexity of the software has increased. Without spending time on technology changes and bringing the changes it is not possible to offer technologically advanced solutions to the customers. Because of this many organizations are using the COTS components to leverage the expertise available with the component manufacturer. The following parameters influence the system development
When we develop the applications with finite budget, it is necessary to utilize the resources in an optimized way without compromising on quality and reliability. In order to achieve this we need to have optimization models which will optimize the overall efforts of the system. When software system is developed, it is supposed to perform various functions. For each function multiple COTS products are available in the market. The vendor of COTS supplier gives the estimates of cost, execution time and reliability of the system to the software developer. The developer has to select the best set of components optimally for design of the fault tolerant software system. Since the alternatives are available in large number, therefore, it becomes difficult for the manager to evaluate each COTS product manually based on multiple conflicting objectives and come to a conclusion of optimal selection. The scientific technique of formulating mathematical optimization model will give the solution to the above mentioned problem. The crisp optimization models are formulated for the optimal selection of COTS products. By crisp optimization model we mean the values of the parameters (cost, execution time and reliability) of the model are assumed to be known and are precise in nature.

Large software system has modular structure to perform set of functions with different modules having different alternatives for each module and different versions (COTS components) for each alternative. Only one component has to be selected from each alternative, and there can be more than one alternative selected for a module. A schematic representation of the software system is given in figure 2.1. 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. Software whose failure can have bad effects afterwards can be made fault tolerant through redundancy at module level. Kapur et al, in (2003) determine budget as a constraint in which they put upper limit over the constraint.
The objective of this chapter is to discuss the problem of optimal COTS selection in design of a fault-tolerant modular software system under consensus recovery block scheme. It is a hybrid fault tolerant scheme which combines the features of both recovery block and N-version programming. The crisp optimization models based on...
the parameters of cost-reliability and execution time-reliability are formulated for the optimal selection of COTS products.

The chapter is divided into two sections. In both the sections bi-criteria optimization models are formulated by assembling COTS components for a fault tolerant modular software system. In Section 2.1 multi-optimization models are formulated for optimal component selection, where the two objectives are maximization of system reliability and minimization of overall system cost. Section 2.2 formulates the multi-optimization models for component selection, where the two objectives are maximization of software reliability and minimization of the overall execution time of the system, with the multiple constraints on budget, redundancy and component selection. The issue of compatibility amongst the COTS alternatives of the modules is also discussed in both the sections. Optimization models are formulated for the same.

Following are the generalized notations and assumptions which are required for model formulation of both the sections.

Notations

- \( R \): System quality measure
- \( C \): Overall system cost
- \( T \): Overall deviational execution time
- \( f_i \): Frequency of use, of function \( i \)
- \( s_l \): Set of modules required for function \( i \)
- \( r_i \): 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 \( i \)
- \( t_1 \): 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 \quad \text{Probability that an incorrect result is accepted as correct}

Y_{ij} \quad \text{Event that correct result of alternative } j \text{ of module } i \text{ is accepted}

X_{ij} \quad \text{Event that output of alternative } j \text{ of module } i \text{ is rejected}

r_j \quad \text{Reliability of alternative } j \text{ of module } i

r_{ijk} \quad \text{Reliability of version } k \text{ of alternative } j \text{ of module } i

\bar{t_i} \quad \text{Average execution time of module } i

\bar{t}_{ijk} \quad \text{Actual execution time of version } k \text{ of alternative } j \text{ of module } i

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

\textbf{Assumptions}

The following assumptions are applicable to both the sections.

1. There is a specified budget for the development of software system.

2. A software system consists of a finite number of modules.

3. A 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.

4. Codes written for integration of modules do not contain any bug.

5. 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 consensus recovery block scheme discussed in [Kumar, (1998); Berman and Kumar, (1999)].

6. The cost of an alternative is the buying price for the COTS product. Reliability for all the components are known and no separate testing is done.
7. Different versions with respect to cost and reliability of a module are available.

8. Other than available cost-reliability versions of an alternative, we assume the existence of a virtual version, which has a negligible reliability of 0.001 and zero cost. These components are denoted by index one in the third subscript of $x_{ijk}$, $c_{ijk}$ and $r_{ijk}$. For example $r_{j1}$ denotes the reliability of first version of alternative $j$ for module $i$, having the above property.

2.1 MULTI-OPTIMIZATION “COST-RELIABILITY” MODEL FOR COTS SELECTION

A component is chosen for a module from a number of alternatives available in the market so that the final system developed should have maximum reliability and minimum cost. Therefore, multi-optimization models for selection of COTS components are formulated in this section. The problem is demonstrated with a numerical illustration along with a diagrammatical representation for a better understanding of problem and its structure. The constraints and objectives of the models are discussed in the following sections.

2.1.1 Model Formulation

Let $S$ be a software architecture made of $n$ modules, with a maximum number of $m_i$ alternatives available for each module and each alternative has different versions (COTS components). Following are the constraints for the optimization models.

2.1.1.1 Reliability Equation of COTS Components

It is already mentioned in the introduction that the reliability of COTS component ($r_{ij}$) is given by the vendor. Therefore reliability ($r_{ij}$) of $j^{th}$ alternative of $i^{th}$ module of the software is given by

$$r_{ij} = \sum_{k=1}^{V_{ij}} x_{ijk} r_{ijk} ; j = 1, 2, \ldots, m_i ; i = 1, 2, \ldots, n$$
2.1.1.2 Version Selection for an Alternative

The following constraint ensures that exactly one version is chosen from each alternative of a module. It includes the possibility of choosing a dummy version also.

\[ \sum_{k=1}^{V_i} x_{ijk} = 1 \quad ; j = 1, 2, \ldots, m_i \quad ; i = 1, 2, \ldots, n \]

2.1.1.3 Redundancy Constraint

The equation stated below guarantees that redundancy is allowed for the components and also ensures that not all alternatives of the module are dummies.

\[ x_{ij1} + z_{ij} = 1 \quad ; j = 1, 2, \ldots, m_i \]

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

2.1.2 Objective Functions

Optimization models proposed in this section aims at optimal component selection so as to maximize system reliability and minimize system cost. Therefore, the two objectives for bi-criteria optimization models are given below

2.1.2.1 Reliability Objective Function

Reliability objective function maximizes the system quality (in terms of reliability) through a weighted function of module reliabilities. Reliability of modules that are invoked more frequently during use is given higher weights. Analytic Hierarchy Process can be effectively used to calculate these weights.

\[ \text{Maximize } R = \sum_{i=1}^{L} f_i \prod_{r \in x_i} R_i \]

Where \( R_i \) is the reliability of module \( i \) of the system under consensus recovery block scheme which is stated as
Chapter 2: Optimal Component Selection for COTS Based Modular Software System

2.1.2.2 Cost Objective Function

Cost objective function minimizes the overall cost of the system and can be written as

\[ \text{Minimize } C = \sum_{j=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{V_{ij}} c_{ijk} x_{ijk} \]

2.1.3 Problem Description

A software system is composed of different modules. Further each module is constructed by several components (COTS) available in the market. The basic problem is to select the COTS components in such a way that good quality and reliable software can be developed in a cost effective manner. So, it is necessary to carry a trade-off between the two. In this section, a bi-criteria optimization problem is formulated with the objectives of reliability maximization and cost minimization of a modular software system with respect to multiple constraints on component selection and redundancy. The problem is demonstrated with a numerical illustration along with a diagrammatical representation for better understanding of problem and its solution.

2.1.3.1 Multi-optimization “Cost-Reliability” Model without Compatibility Constraints

In the optimization model it is assumed that the alternatives of a module are in a Consensus Recovery Block. **Consensus Recovery Block** for achieving fault tolerance is used to run all the attached independent alternatives simultaneously and selecting the output by the voting mechanism. It requires independent development of independent alternatives of a module (which the components satisfy) and a voting procedure. Upon invocation of the consensus recovery block, all alternatives are
executed and their output is 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 alternative is designated as correct. Otherwise, the next stage is entered. At this stage, the best alternative is examined by the acceptance test. If the output is accepted, it is treated as the correct one. However, if the output is not accepted, the next best alternative is subjected to testing. This process continues until an acceptable output is found or all outputs are exhausted.

**Problem (2.1.P1)**

Maximize \[ R = \sum_{i=1}^{L} f_i \prod_{j} R_j \]

Minimize \[ C = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{V_k} c_{ijk} x_{ijk} \]

Subject to

\[ X \in S = \{ x_{ijk} \text{ is binary variable} \} \]

\[ R_i = 1 + \left[ \sum_{j=1}^{m} \frac{1}{1-r_{ij}} \prod_{k=1}^{V_k} (1-r_{ijk})^{z_{ijk}} \right] + \left[ \sum_{j=1}^{m} \prod_{k=1}^{V_k} (1-r_{ijk})^{z_{ijk}} \right] \]

\[ P(X_{ij}) = (1-t_i)(1-r_{ij}) + r_{ij} t_j \]

\[ P(Y_{ij}) = r_{ij} (1-t_j) \]

\[ r_{ij} = \sum_{k=1}^{V_k} x_{ijk} r_{ijk} ; j = 1, 2, ..., m_i \quad ; i = 1, 2, ..., n \]

\[ \sum_{k=1}^{V_k} x_{ijk} = 1 ; j = 1, 2, ..., m_i \quad ; i = 1, 2, ..., n \]

\[ x_{ijk} + z_{ijk} = 1 ; j = 1, 2, ..., m_i \]

\[ \sum_{j=1}^{m} z_{ij} \geq 1 \quad ; i = 1, 2, ..., n \]

\[ \{ \} \]
where X is a vector of components $x_{ijk}$ and $z_{ij} ; i=1,......,n ; j=1,......,m_{i} ; k=1,......,V_{ij}$

Here the first objective stated is the reliability objective and the second one is the cost objective. Constraint (2.1.1) estimates the reliability of module $i$ for a system under consensus recovery block scheme. Constraint (2.1.2) is the probability of event that output of alternative $j$ of module $i$ is rejected and Constraint (2.1.3) is the probability of event that correct result of alternative $j$ of module $i$ is accepted. Constraint (2.1.4) gives the reliability of alternative $j$ of module $i$. Constraint (2.1.5) ensures that exactly one version is chosen from each alternative of a module. It includes the possibility of choosing a dummy version. Constraints (2.1.6) and (2.1.7) guarantee that not all chosen alternatives of modules are dummies. The above optimization model is a 0-1 Bi-Criterion integer programming problem.

2.1.3.2 Multi-optimization “Cost-Reliability” Model with Compatibility Constraints

In optimization model discussed in previous section, we assumed that all COTS alternative of one module are compatible with the COTS alternative of other modules. However, sometimes it is observed that some alternatives of a module may not be compatible with alternatives of other modules due to issues such as implementation, interfaces, and licensing. Optimization model formulated in this section addresses the problem of compatibility. It is done by incorporating additional constraints in the optimization models [Jung and Choi, (1999)]. This constraint can be represented as $x_{gsq} \leq x_{huq}c$, which means that if alternative $s$ for module $g$ is chosen, then alternative $u_t, t=1,........z$ have to be chosen for module $h$. We also assume that if two alternatives are compatible, then their versions are also compatible.

$$x_{gsq} - x_{huqc} \leq M y_t ; q = 2,..........V_{gs} ; c = 2,..........V_{huq} ; s = 1,..........m_g \quad \ldots(2.1.8)$$

$$\sum y_t = z(V_{huq} - 2) \quad \ldots(2.1.9)$$

Constraints (2.1.8) and (2.1.9) make use of binary variable $y_t$ 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 (2.1.9) can be relaxed as

\[ \sum y_t \leq z\left(V_{hu_t} - 2\right) \]  

...(2.1.10)

Therefore, problem (2.1.P1) can be transformed to another optimization problem using compatibility constraint as

**Problem (2.1.P2)**

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

**Minimize** \[ C = \sum_{i=1}^{m} \sum_{j=1}^{V_i} \sum_{k=1}^{c(i)} c_{ijk} x_{ijk} \]

**Subject to**

\[ X \in S \]

\[ x_{gsq} - x_{hu_t, c} \leq M \sum y_t \quad ; \quad q = 2, ..., V_{gs} \quad ; \quad c = 2, ..., V_{hu_t} \quad ; \quad s = 1, ..., m_g \]

\[ \sum y_t \leq z(V_{hu_t} - 2) \]

### 2.1.4 Solution Procedure

This section presents the solution procedure to solve the optimization models discussed in the previous Section [For details, refer Section 2.1.3.1 and Section 2.1.3.2].

#### 2.1.4.1 Solution Procedure of Multi-optimization “Cost-Reliability” Model without Compatibility Constraints

**Step 1: Normalization**

The problem (2.1.P1) is Bi- criteria optimization problem in which on one hand system reliability is maximized and other hand cost of selected components to form/assemble the system is minimized. The reliability which is unit free is measured
between zero and one whereas cost has its unit. Two objectives can be converted to a single objective programming problem either if both objectives are of same unit or if both objectives can be made unit free. To make cost function unit free, the following transformation is used.

\[ c_{ijk} = \frac{c_{ijk}}{\sum_{j=1}^{m_j} \sum_{k=1}^{n_k} c_{ijk}} \]

The resulting problem then can be rewritten as

**Problem (2.1.P3)**

Maximize \( F_1(X) = \sum_{i=1}^{I} f_i \prod_{s \in s_j} R_i \)

Minimize \( F_2(X) = \sum_{i \in s_j} \sum_{j=1}^{m_j} \sum_{k=1}^{n_k} c_{ijk} x_{ijk} \)

Subject to

\( X \in S \)

**Step 2:** The problem (2.1.P3) can further be written as vector optimization problem as

**Problem (2.1.P4)**

Vector Max \( F(X) \)

Subject to

\( X \in S \)

where

\( F(X) = (F_1(X), F_2(X))^T \)

**Finding Properly Efficient Solution**

**Definition 1** [Steuer, (1986)]: A feasible solution \( X^* \in S \) is said to be an efficient solution for the (2.1.P3) problem if there exists no \( X \in S \) such that \( F(X) \geq F(X^*) \) and \( F(X) \neq F(X^*) \).
**Definition 2** [Steuer, (1986)]: An efficient solution $X^* \in S$ is said to be a properly efficient solution for the problem (2.1.P3) if there exist $\alpha > 0$ such that for each $r$

$$\left( F_r(X) - F_r(X^*) \right) / \left( F_j(X) - F_j(X^*) \right) < \alpha$$

for some $j$ with $F_j(X) \leq F_j(X^*)$ and $F_r(X) > F_r(X^*)$ for $X \in S$.

**Step 3:** Using Geoffrion’s scalarization, the problem (2.1.P3) is converted to

**Problem (2.1.P5)**

*Maximize* $Z = \lambda_1 F_1 + \lambda_2 F_2$

*Subject to*

$X \in S$

$\lambda_1 + \lambda_2 = 1$

$\lambda_1, \lambda_2 \geq 0$

**Lemma 1** [Geoffrion, (1968)]: The optimal solution of the problem (2.1.P5) for fixed $\lambda_1$ and $\lambda_2$ is a properly efficient solution for the problem (2.1.P4) and consequently (2.1.P2).

Hence the final formulation of the problem is

**Problem (2.1.P6)**

*Maximize* $Z = \lambda_1 \left( \sum_{l=1}^{L} f_j \prod_{i \in s_j} R_i \right) - \lambda_2 \left( \sum_{i=s_j} m_j \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} x_{ijk} \right)$

*Subject to*

$X \in S$

$\lambda_1 + \lambda_2 = 1$

$\lambda_1, \lambda_2 \geq 0$

The original problem was a Bi-Criteria problem. Using Geoffrion’s scalarization the problem is converted to a single objective problem (2.1.P5) by attaching weights to both the objective functions.
2.1.4.2 Solution Procedure of Multi-optimization “Cost-Reliability” Model with Compatibility Constraints

The same steps can be applied to get the solution of optimization model with the compatibility constraint [For details, refer Section 2.1.4.1]. Therefore the problem (2.1.P2) can be restated as

**Problem (2.1.P7)**

Maximize $Z = \lambda_1 \left[ \sum_{l=1}^{L} f_l \prod_{i \in S_l} R_l \right] - \lambda_2 \left[ \sum_{i=s}^{m_1} \sum_{j=1}^{c} E_{ijk} x_{ijk} \right]$

Subject to

$X \in S$

$x_{gsq} - x_{huq} \leq M \sum y_t, q = 2, ..., V_{gs} ; c = 2, ..., V_{hu} ; s = 1, ..., m_g$

$\sum y_t \leq z \left( V_{hu} - 2 \right)$

$\lambda_1 + \lambda_2 = 1$

$\lambda_1, \lambda_2 \geq 0$

2.1.5 Numerical Illustrative

A numerical example is illustrated to describe the proposed methodology of multi-optimization cost reliability model for COTS selection. In the example the software system is decomposed into two modules ($m_1$) and ($m_2$). Three alternatives are available for module ($m_1$) and four alternatives are available for module ($m_2$). Each alternative has three versions $v_1, v_2$ and $v_3$.

A COTS based software system is developed by integrating various COTS products. For each function multiple COTS products are available in the market. The vendor provides information about cost and reliability of the COTS components. The cost reliability data for different versions are given in table 2.1.1. The cost of first version which is the virtual versions for all alternatives is zero and reliability is 0.001. This is done so, as if in the optimal solution for some module $x_{ij} = 1$ that implies corresponding alternative is not to be attached in the module.
Let the software is required to perform three functions, so \( L = 3 \). The set of modules required for the three functions are given by \( S_1 = \{1, 2\} \), \( S_2 = \{1\} \) and \( S_3 = \{2\} \). The frequency of use is given by \( f_1 = 0.5, f_2 = 0.3 \) and \( f_3 = 0.2 \). It is also assumed that \( t_1 = 0.01, t_2 = 0.05 \) and \( t_3 = 0.01 \).

**Table 2.1.1: Data Set of Cost and Reliability for Multi-optimization “Cost-Reliability” Model**

<table>
<thead>
<tr>
<th>Modules</th>
<th>Alternatives</th>
<th>Versions</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cost</td>
<td>Reliability</td>
<td>Cost</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.001</td>
<td>8.2</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0.001</td>
<td>7.5</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0.001</td>
<td>8.5</td>
<td>0.90</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0.001</td>
<td>3.2</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0.001</td>
<td>3.4</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0.001</td>
<td>5.0</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>0.001</td>
<td>4.8</td>
<td>0.86</td>
</tr>
</tbody>
</table>

**2.1.5.1 Solution of Multi-optimization “Cost-Reliability” Model without Compatibility Constraints**

The structure of software for optimal component selection is discussed in the introduction of this chapter [For details, refer Figure 2.1]. A component is to be selected for an alternative of a module. Different weights have been attached to reliability and cost objectives which are 0.6 and 0.4 respectively and can be represented as \((\lambda_1, \lambda_2) = (0.6, 0.4)\). The optimal solution set so obtained for problem (2.1.P6) is optimal for problem (2.1.P1) also. Solving problem (2.1.P6) for cost-reliability bi-criteria optimization model with the above data set, the solution for component selection as given in the table 2.1.2 is obtained. The problem is solved using software package LINGO (Version 11).
Table 2.1.2: Solution of Multi-optimization “Cost-Reliability” Model without Compatibility Constraints

<table>
<thead>
<tr>
<th>Module</th>
<th>Selected COTS Products</th>
<th>System Reliability</th>
<th>System Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(x_{113} = x_{123} = x_{133} = 1)</td>
<td>0.79</td>
<td>49.4</td>
</tr>
<tr>
<td>2</td>
<td>(x_{213} = x_{223} = x_{233} = x_{243} = 1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We can also show diagrammatically the representation of selected components for a component based software system as shown in figure 2.1.1.

Figure 2.1.1: Diagrammatic Representation for Solution of Multi-optimization “Cost-Reliability” Model without Compatibility Constraints

By table 2.1.2 and figure 2.1.1, we can easily conclude that components \(x_{113} = x_{123} = x_{133}\) got selected for module \((m_1)\). For the first alternative of module \((m_1)\), third version is selected \(x_{113}\). Similarly for module \((m_2)\) and \((m_3)\), third version is selected, \(x_{123}\) and \(x_{133}\) respectively. Therefore we can say that redundancy is allowed in module \((m_1)\),
because out of the three alternatives available with module \( (m_1) \) all three actual alternatives are chosen (viz. \( x_{113} = x_{123} = x_{133} \)).

Similarly we can conclude that components \( x_{213} = x_{223} = x_{233} = x_{243} \) got selected for the second module. It can be clearly seen from the table as well as from the figure, that no dummy version is selected for the alternatives. Therefore, redundancy is also allowed in module \( (m_2) \).

### 2.1.5.2 Solution of Multi-optimization “Cost-Reliability” Model with Compatibility Constraints

To illustrate optimization model for compatibility, the result of previous section is used. It is assumed that second alternative of first module is compatible with second and third alternatives of second module. Solving problem (2.1.P7) for cost-reliability optimization model with compatibility constraints and with the above data set, the solution of component selection which is mentioned in table 2.1.3 is achieved.

#### Table 2.1.3: Solution of Multi-optimization “Cost-Reliability” Model with Compatibility Constraints

<table>
<thead>
<tr>
<th>Modules</th>
<th>Selected COTS Products</th>
<th>System Reliability</th>
<th>System Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( x_{113} = x_{123} = x_{133} = 1 )</td>
<td>( 0.78 )</td>
<td>( 48.5 )</td>
</tr>
<tr>
<td>2</td>
<td>( x_{213} = x_{222} = x_{233} = x_{243} = 1 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It is observed that due to the compatibility condition, second alternative of second module is chosen as it is compatible with second alternative of first module. Corresponding to the solution table 2.1.3, a diagrammatical representation of selected components is shown in figure 2.1.2.
2.2 MULTI-OPTIMIZATION “EXECUTION TIME-RELIABILITY” MODEL FOR COTS SELECTION

In this section a problem of component (COTS) selection for a fault tolerant modular software system incorporating execution time is formulated. 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 solve this problem, the concept of deviational 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 calculated. The deviation from the average time is used to address two issues. Firstly, to find those alternatives whose execution
completes early before the assumed average time. Secondly, those alternatives whose execution takes longer time to complete. In other words the execution completes after the assumed average time. In both the cases there is a deviation from average time.

Therefore, the optimization models in this section aim at optimal COTS selection for a software system by minimizing the deviational time by simultaneously maximizing the system reliability. The problem is demonstrated with a numerical illustration along with a diagrammatical representation for a better understanding of problem and its structure.

2.2.1 Additional Assumptions

In addition to the assumptions discussed in the introduction of this chapter, following assumptions are also applicable to this model.

9. Along with cost, reliability of different versions of a module, execution time is also available.

10. 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.

11. Deviational time is the difference between the average time and the actual execution time of the software.

12. The execution time of the virtual version is zero.
2.2.2 Model Formulation

Let $S$ be a software architecture made of $n$ modules, with a maximum number of $m_i$ alternatives available for each module and each COTS alternative has different versions. In addition to the constraints discussed in cost-reliability model [for details, refer to Section 2.1.1] one additional constraint is applicable to execution time-reliability model which is stated in the next section.

2.2.2.1 Budget Constraint

The equation below is a budget constraint for the overall software system. It states that if version $k$ of the alternative $j$ for module $i$ is selected then the corresponding cost is to be added to the overall cost. This overall cost must be less than some predetermined cost ($C$) which is decided by the management.

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} \sum_{k=1}^{V_{ij}} c_{ijk} x_{ijk} \leq C$$

2.2.3 Objective Functions

The model aims at optimal component selection for a software system with the two objectives of maximizing system reliability and minimizing deviational execution time under the constraints on component selection. These two objectives are discussed in details in the following sections.

2.2.3.1 Reliability Objective Function

Reliability objective function maximizes the system quality (in terms of reliability) through a weighted function of module reliabilities. Reliability of modules that are invoked more frequently during use is given higher weights. Analytic Hierarchy Process can be effectively used to calculate these weights.

Maximize $R = \sum_{i=1}^{l} f_i \prod_{k=i}^{R_i}$

where $R_i$ is the reliability of module $i$ of the system under consensus recovery block scheme which is stated as

$$R_i = 1 + \left[ \frac{1}{m_i} \prod_{j=1}^{m_i} (1-r_{ij})^{z_{ij}} \right] \left[ 1 - \left( 1 - r_{ij} \right)^{z_{ij}} \right] + \sum_{j=1}^{m_i} \left[ \prod_{k=1}^{m_i} (1-r_{ij})^{z_{ij}} \right] \left[ P(X_{ik})^{z_{ik}} \right] \left[ P(Y_{ij})^{\gamma} \right] - 1 \quad i=1,2,\ldots,n$$
Optimal Component Selection for Fault Tolerant Software Design under Consensus Recovery Block Scheme

\[ P(X_o) = (1 - t_o) \left[ (1 - r_o)(1 - t_o) + r_o t_o \right] \]

\[ P(Y_o) = r_o (1 - t_o) \]

2.2.3.2 Deviational Time Objective Function

Deviational time objective function minimizes the overall deviation time of the system which in turn minimizes the overall execution time and can be written as

\[
\text{Minimize} \sum_{i=1}^{L} f_i \sum_{s_i} \sum_{j=1}^{m_i} \sum_{k=1}^{v_i} \epsilon_{ijk} x_{ijk}
\]

where \( |t^*_{ijk} - t_i| = \epsilon_{ijk} \) gives the absolute deviational execution time of an individual component, i.e. average execution time subtracted from actual execution time.

2.2.4 Problem Description

The problem of choosing right mix of components becomes extremely difficult because of the number of COTS products available with different vendors in the market. Optimization models for COTS selection with the dual objectives of reliability maximization and execution deviational time minimization, under the multiple constraints on budget, redundancy and selection of components. Crisp optimization models are formulated for the same problem.

2.2.4.1 Multi-optimization “Execution Time-Reliability” Model without Compatibility Constraints

It is discussed that the alternatives of the modules are in consensus recovery block scheme [for details, refer Section 2.1.3].

Problem (2.2.P1)

\[
\text{Maximize} \quad R = \sum_{i=1}^{L} f_i \prod_{r_i} R_i
\]

\[
\text{Minimize} \quad T = \sum_{i=1}^{L} f_i \sum_{s_i} \sum_{j=1}^{m_i} \sum_{k=1}^{v_i} \epsilon_{ijk} x_{ijk}
\]
Subject to

\[ X \in S = \{ \ x_{ijk} \text{ is binary variable /} \]

\[ \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{V} c_{ijk} \ x_{ijk} \leq C \quad \ldots(2.2.1) \]

\[ R_i = 1 + \left( \sum_{j=1}^{m} \frac{1}{Z_j} \left( \prod_{m=1}^{M} (1 - r_{ik})^{y_{ik}} \right) \prod_{m=1}^{M} (1 - r_{ij})^{y_{ij}} \right) \left[ \sum_{j=1}^{m} z_{ij} \left( \prod_{m=1}^{M} p(X_{ik})^{y_{ik}} \right) p(y_{ij})^{y_{ij}} - 1 \right] ; \ i = 1, 2, \ldots, n \quad \ldots(2.2.2) \]

\[ p(Y_{ij}) = (1 - r_{ij}) \left( 1 - r_{ij} \right) \quad \ldots(2.2.3) \]

\[ p(Y_{ij}) = r_{ij} (1 - t_{ij}) \quad \ldots(2.2.4) \]

\[ r_{ij} = \sum_{k=1}^{V} x_{ijk} r_{ijk} ; \ j = 1, 2, \ldots, m_{j} ; \ i = 1, 2, \ldots, n \quad \ldots(2.2.5) \]

\[ \sum_{k=1}^{V} x_{ijk} = 1 ; \ j = 1, 2, \ldots, m_{j} ; \ i = 1, 2, \ldots, n \quad \ldots(2.2.6) \]

\[ x_{ij1} + z_{ij} = 1 ; \ j = 1, 2, \ldots, m_{j} \quad \ldots(2.2.7) \]

\[ \sum_{j=1}^{m} z_{ij} \geq 1 ; \ i = 1, 2, \ldots, n \quad \ldots(2.2.8) \]

where \( X \) is a vector of component \( x_{ijk} \) and \( z_{ij} \); \( i = 1, \ldots, n ; \ j = 1, \ldots, m_{i} ; k = 1, \ldots, V_{ij} \).

First objective function maximizes the system quality (in terms of reliability) through a weighted function of module reliabilities and second objective minimizes the execution deviational time. Constraint (2.2.1) is a budget constraint. Constraint (2.2.2) estimates the reliability of module \( i \). Constraint (2.2.3) is the probability of event that output of alternative \( j \) of module \( i \) is rejected and Constraint (2.2.4) is the probability of event that correct result of alternative \( j \) of module \( i \) is accepted. Constraint (2.2.5) gives the reliability of alternative \( j \) of module \( i \). Constraint (2.2.6) ensures that exactly one version is chosen from each alternative of a module. It includes the possibility of choosing a dummy version. Equations (2.2.7) and (2.2.8) guarantee that not all chosen alternatives of modules are dummies.


2.2.4.2 Multi-optimization “Execution Time-Reliability” Model with Compatibility Constraints

This section deals with the issue of compatibility of components among the modules and was discussed in detail in section 2.1.3. Therefore, execution time-reliability model with constraints on compatibility can be formulated as

**Problem (2.2.P2)**

\[
\text{Maximize } R = \sum_{i=1}^{l} f_i \prod_{m=j}^{R_i} \quad \text{Minimize } T = \sum_{i=1}^{l} f_i \sum_{j=1}^{m} \sum_{k=1}^{V_{ij}} \in_{ijk} x_{ijk}
\]

Subject to

\[
X \in S \\
x_{gsc} - x_{huc} \leq M \sum_{j=1}^{q} y_i \quad q = 2, \ldots, V_{gs} ; c = 2, \ldots, V_{hu} ; s = 1, \ldots, mg \\
\sum_{i=1}^{m} y_i \leq z(V_{hu} - 2) \\
\lambda_1 + \lambda_2 = 1 \\
\lambda_1, \lambda_2 \geq 0
\]

Similar constraints can be written for all pairs of compatible modules.

2.2.5 Solution Procedure

The following steps are required to solve the optimization models discussed in section 2.2.4.

2.2.5.1 Solution Procedure for Multi-optimization “Execution Time-Reliability” Model without Compatibility Constraints

**Step 1:** Normalization

The problem (2.2.P1) is multi-optimization problem in which on one hand system reliability is maximized and on other hand deviational time of selected components to form/assemble the system is minimized. The reliability which is unit free is measured between zero and one whereas deviational time has its unit. Two objectives can then be converted to single objective programming problem either if both objectives are of
same unit or if both objectives can be made unit free. To make cost function unit free, the following transformation is used.

\[
\bar{\epsilon}_{ijk} = \frac{\epsilon_{ijk}}{n \sum_{i=1}^{m_i} \sum_{j=1}^{m_j} \sum_{k=1}^{m_k} v_{ij}}
\]

The resulting problem then can be re-written as

**Problem (2.2.P3)**

\[
\begin{align*}
\text{Maximize } & F_1(X) = \sum_{l=1}^{L} f_l \prod_{i \in X_l} R_i \\
\text{Minimize } & F_2(X) = \sum_{l=1}^{L} f_l \sum_{i \in X_l} \sum_{j=1}^{m_i} \sum_{k=1}^{m_k} \bar{\epsilon}_{ijk} x_{ijk}
\end{align*}
\]

Subject to

\[X \in S\]

**Step 2:** The problem (2.2.P2) can further be written as vector optimization problem as

**Problem (2.2.P4)**

Vector Max \(F(X)\)

Subject to

\[X \in S\]

Where

\[
F(X) = (F_1(X), F_2(X))^T
\]

**Finding Properly Efficient Solution**

**Definition 1** [Steuer, (1986)]: A feasible solution \(X^* \in S\) is said to be an efficient solution for the problem (2.2.P2) if there exists no \(X \in S\) such that \(F(X) \geq F(X^*)\) and \(F(X) \neq F(X^*)\)

**Definition 2** [Steuer, (1986)]: An efficient solution \(X^* \in S\) is said to be a properly efficient solution for the problem (2.2.P2) if there exist \(\alpha > 0\) such that for each \(r\)
\[(F_r(X) - F_j(X^*) - F_j(X)) < \alpha \text{ for some } j \text{ with } F_j(X) \leq F_j(X^*) \text{ and } F_r(X) > F_r(X^*) \text{ for } X \in S.\]

**Step 3:** Using Geoffrion’s scalarization the problem (2.2.P2) reduces to

**Problem (2.2.P5)**

Maximize \(Z = \lambda_1 F_1 + \lambda_2 F_2\)

**Subject to**

\[X \in S\]
\[\lambda_1 + \lambda_2 = 1\]
\[\lambda_1, \lambda_2 \geq 0\]

**Lemma 1** (Geoffrion, (1968)): The optimal solution of the problem (2.2.P4) for fixed \(\lambda_1\) and \(\lambda_2\) is a properly efficient solution for the problem (2.2.P3) and consequently (2.2.P1).

Hence the final formulation of the problem is

**Problem (2.2.P6)**

Maximize \(Z = \lambda_1 \sum_{i=1}^{L} f_j \prod_{i \in S_j} R_i - \lambda_2 \sum_{i=1}^{L} f_j \sum_{i \in S_j} m_i \sum_{j=1}^{L} \sum_{k=1}^{m_i} \gamma_{ijk} x_{ijk}\)

**Subject to**

\[X \in S\]
\[\lambda_1 + \lambda_2 = 1\]
\[\lambda_1, \lambda_2 \geq 0\]

The original problem was a multi-optimization problem, using Geoffrion’s scalarization the problem gets converted to a single objective problem (2.2.P5) by attaching the weights to both the objective functions.

**2.2.5.2 Solution Procedure for Multi-optimization “Execution Time-Reliability” Model with Compatibility Constraints**

The final execution time-reliability model with constraints on compatibility can be formulated in a similar way as discussed in (Section 2.2.5.1) and can be stated as
Problem (2.2.P7)

Maximize \( Z = \lambda_1 \left[ \sum_{l=1}^{L} \sum_{i \in S_l} R_i \right] - \lambda_2 \left[ \sum_{l=1}^{L} \sum_{i \in S_l} \sum_{j=1}^{m_l} \sum_{k=1}^{V_{ij}^k} \right] \)

Subject to

\[
X \in S \\
x_{gq} - x_{hc} \leq M \sum y_i \quad ; q = 2, \ldots, V_{gs} \quad ; c = 2, \ldots, V_{hu} \quad ; s = 1, \ldots, mg \\
\sum y_i \leq z(V_hu - 2) \\
\lambda_1 + \lambda_2 = 1 \\
\lambda_1, \lambda_2 \geq 0
\]

Similar constraints can be written for all pairs of compatible modules. The model is illustrated with a numerical example.

2.2.6 Numerical Illustration

A numerical example is illustrated to describe the proposed methodology of multi-optimization “execution time-reliability” model for COTS selection. In the example the software system is decomposed into three modules \((m_1), (m_2)\) and \((m_3)\). Three alternatives are given for module \((m_1)\), four alternatives are given for module \((m_2)\) and two alternatives are given for module \((m_3)\). Each alternative has three versions \(v_1, v_2\) and \(v_3\).

A COTS based software system is developed by integrating various COTS products. For each function multiple COTS products are available in the market. The vendor provides information on cost and reliability of the COTS components. The cost reliability data for different versions are given in table 2.2.1. The cost of first version which is the virtual versions for all alternatives is zero and reliability is 0.001. This is done so, as if in the optimal solution for some module \(x_{ij1} = 1\) that implies corresponding alternative is not to be attached in the module.

Let the software is required to perform three functions, so \(L = 3\). The set of modules required for the three functions are given by \(S_1 = \{1, 2\}, S_2 = \{1\} \) and \(S_3 = \{2\}\). The frequency of use is given by \(f_1 = 0.5, f_2 = 0.3 \) and \(f_3 = 0.2\). It is also assumed that \(t_1 = 0.01, t_2 = 0.05 \) and \(t_3 = 0.01\). The execution time, in seconds taken by each component
when called by different functions are given in table 2.2.2, 2.2.3 and 2.2.4 respectively.

**Table 2.2.1: Data Set of Cost and Reliability for Multi-optimization “Execution Time-Reliability” Model**

<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></td>
<td></td>
<td>Cost</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8.2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7.5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>8.5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3.4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4.8</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.2</td>
</tr>
</tbody>
</table>

**Table 2.2.2: Execution Time Data Set for First Function (Multi-optimization “Execution Time-Reliability” Model)**

<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></td>
<td></td>
<td>Cost</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 2.2.3: Execution Time Data Set for Second Function (Multi-optimization “Execution Time-Reliability” Model)

<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 2.2.4: Execution Time Data Set for Third Function (Multi-optimization “Execution Time-Reliability” Model)

<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>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>
</tbody>
</table>

It is mentioned that the actual execution time of a dummy version is zero. Data Set for actual execution time is given in tables 2.2.2, 2.2.3 and 2.2.4. The optimization model aims at finding out deviational time of those components that are attached to the module of the system. Deviational time is defined as:

Deviational time = Actual Execution Time – Average Time

Here we are taking the transformed tables where the actual execution of the dummy version is the average time so that in the final table, the deviational time of the dummy version becomes zero.
Table 2.2.5: Transformed Execution Time Data Set for First Function (Multi-optimization “Execution Time-Reliability” Model)

<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 2.2.6: Transformed Execution Time Data Set for Second Function (Multi-optimization “Execution Time-Reliability” Model)

<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>
</tbody>
</table>
Table 2.2.7: Transformed Execution Time Data Set for Third Function (Multi-optimization “Execution Time-Reliability” Model)

<table>
<thead>
<tr>
<th>Module</th>
<th>Alternatives</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<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>

2.2.6.1 Solution of Multi-optimization “Execution Time-Reliability” Model without compatibility Constraints

The problem is solved using software package LINGO (Version 11). The optimal solution set so obtained for problem (2.2.P5) is optimal for problem (2.2.P1). Solving problem (2.2.P5) for execution time-reliability optimization model with the above data sets on cost, reliability and execution time, the following solution is obtained which is discussed in relation to case 1 and case 2 of this section. In both the cases different weights have been attached to the two objective functions, which are reliability and execution deviational time.

CASE 1: Weights attached to reliability is 0.6 and deviational time is 0.4. Budget is assumed to be 60 units.

Table 2.2.8: Solution of Multi-optimization “Execution Time-Reliability” Model without Compatibility Constraints (Case 1)

<table>
<thead>
<tr>
<th>Module</th>
<th>Selected COTS Components</th>
<th>System Reliability</th>
<th>Deviational Time</th>
<th>System Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( x_{111} = x_{122} = x_{131} = 1 )</td>
<td>0.90</td>
<td>0.21</td>
<td>25.8</td>
</tr>
<tr>
<td>2</td>
<td>( x_{211} = x_{221} = x_{233} = x_{243} = 1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( x_{311} = x_{323} = 1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The diagrammatic representation of the solution mentioned in table 2.2.8 can be seen as

By table 2.2.8 and figure 2.2.2, we can easily conclude that components $x_{111} = x_{122} = x_{131} = 1$ got selected for module $(m_1)$. For the first and third alternatives of module $(m_1)$, virtual versions are selected, viz., $x_{111}$ and $x_{131} = 1$. Only for second alternative, actual version is selected. Therefore it can be concluded that redundancy is not allowed in module $(m_1)$, as out of the three alternatives only one is selected and the remaining two are dummy versions.

Similarly we can conclude that components $x_{211} = x_{221} = x_{233} = x_{243} = 1$ got selected for the second module. It can be clearly seen from the table as well as from the figure, that two dummy versions are selected for first and second alternatives. Redundancy is allowed in module $(m_2)$ as out of four alternatives selected two are the actual versions which are selected for third and fourth module. (i.e. $x_{233}$ and $x_{243}$).
Since only one actual version is selected for module \((m_3)\), therefore, redundancy is not allowed in third module.

**CASE 2:** Weights attached to reliability are \((0.7 & 0.3)\) and deviational times are \((0.8 & 0.2)\) respectively. Budget is assumed to be 60 units.

**Table 2.2.9:** Solution of Multi-optimization “Execution Time-Reliability” Model without Compatibility Constraints (Case 2)

<table>
<thead>
<tr>
<th>Modules</th>
<th>Selected COTS Components</th>
<th>System Reliability</th>
<th>Deviational Time</th>
<th>System Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(x_{111} = x_{122} = x_{133} = 1)</td>
<td>0.97</td>
<td>0.37</td>
<td>35.3</td>
</tr>
<tr>
<td>2</td>
<td>(x_{211} = x_{221} = x_{233} = x_{243} = 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(x_{311} = x_{323} = 1)</td>
<td>[x_{312} = x_{322} = x_{333} = 1]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Redundancy is allowed for the first two modules. The diagrammatic representation of the solution above can be seen as

![Diagrammatic Representation](image)

**Figure 2.2.3:** Diagrammatic Representation for Solution of Multi-optimization “Execution Time-Reliability” Model (Case 2)
2.2.6.2 Solution of Multi-optimization “Execution Time-Reliability” Model with Compatibility Constraints

To illustrate the optimization model for compatibility, the results of section (2.2.6.1) are used.

CASE 1: Weight attached to reliability is 0.6 and deviational time is 0.4. Budget is assumed to be 60 units.

It is assumed that the second alternative \((x_{122})\) of first module \((m_1)\) is compatible with second \((x_{223})\) and fourth \((x_{342})\) alternatives of second module \((m_2)\). Following solution was obtained.

Table 2.2.10: Solution of Multi-optimization “Execution Time-Reliability” Model with Compatibility Constraints (Case 1)

<table>
<thead>
<tr>
<th>Modules</th>
<th>Selected COTS Components</th>
<th>System Reliability</th>
<th>Deviational Time</th>
<th>System Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(x_{111} = x_{122} = x_{131} = 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(x_{211} = x_{223} = x_{231} = x_{243} = 1)</td>
<td>0.90</td>
<td>0.55</td>
<td>23.3</td>
</tr>
<tr>
<td>3</td>
<td>(x_{311} = x_{323} = 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It is observed that due to the compatibility condition, second \((x_{223})\) alternative of second module \((m_2)\) is chosen as it is compatible with second \((x_{122})\) alternative of first module \((m_1)\). Corresponding to the solution table 2.2.10, we can show diagrammatically the representation of selected components for a component based software system.
CASE 2: Weights attached to reliability are (0.7 & 0.3) and deviational times (0.8 & 0.2) respectively. Budget is assumed to be 60 units.

Table 2.2.11: Solution of Multi-optimization “Execution Time-Reliability” Model with Compatibility Constraints (Case 2)

<table>
<thead>
<tr>
<th>Modules</th>
<th>Selected COTS Components</th>
<th>System Reliability</th>
<th>Deviational Time</th>
<th>System Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x_{111} = x_{122} = x_{133} = 1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$x_{211} = x_{223} = x_{231} = x_{243} = 1$</td>
<td>0.98</td>
<td>0.73</td>
<td>32.8</td>
</tr>
<tr>
<td>3</td>
<td>$x_{311} = x_{322} = 1$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It is observed that due to the compatibility condition, second ($x_{223}$) alternative of second module ($m_2$) is chosen as it is compatible with second ($x_{122}$) alternative of first module.
Corresponding to the solution table 2.2.11, we can show diagrammatically the representation of selected components for a component based software system.

Figure 2.2.5: Solution of Multi-optimization “Execution Time-Reliability” Model with Compatibility Constraints (Case 2)