CHAPTER 4
OPTIMAL COMPONENT SELECTION FOR FAULT TOLERANCE MODULAR SOFTWARE SYSTEM USING BUILD-OR-BUY POLICY UNDER FUZZY ENVIRONMENT

Software users demand faster deliveries, cheaper software and quality product whereas software developers aim at minimizing their development cost, maximizing the profit margins and meeting the competitive requirements. The resulting situation calls for trade-off between conflicting objectives prevailing between software user’s requirements with the developers in turn driving the management to determine the optimal reliability level of the software system. When the design of software architecture reaches a good level of maturity, software engineers have to undertake selection decision about software components.

In the previous two chapters we focused only on COTS based software development. Always using COTS components may not be profitable in terms of reliability, cost and availability. The organizations which are developing a software system may develop some of the components within the organization because they have the technology, resources and infrastructure. These components are known as in-house developed components. These components can be developed from the scratch or by reusing and modifying the existing components. Therefore, while developing a software, components can be both bought as

This chapter is based on the following papers
commercial–off-the-shelf (COTS) components, and probably adapted to work in the software system, or they can be developed in-house. This chapter focuses on build-or-buy strategy for component selection.

Cortellessa et al., (2008) introduced a framework that helps developers to decide whether to buy-or-build components for some software architecture. Once built software architecture, each component can be both purchased, and probably adapted to the new software system, and/or it can be developed in-house.

In developing a component based software system, different functions are required to be performed and different modules are available for those functions. To provide maximum reliable software, one should give emphasis on reliability of modules. In this chapter, the reliability of in-house and COTS based modular software is devised with possible redundancy at module level. We address the problem of (automatic) component selection. Informally, our problem is to select a set of components from the available component set which can satisfy a given set of requirements while minimizing the sum of the costs and maximizing the reliability of the selected components. The dependencies between the components must be taken into account. To achieve this goal, we should assign each component a set of requirements it satisfies. Each COTS component is assigned a cost which is the overall cost of acquisition, delivery time which is the duration of acquiring a component and also the reliability of the component. For each build component, we estimate the parameters of cost, delivery time and reliability of the component, whereas the parameters of the COTS components are given by its vendor. We assume that for all the alternatives available for a module, cost increases if higher reliability is desired. This is a realistic assumption, as COTS suppliers are ready to supply more reliable versions of the same component at a higher price. Purchase of high-quality COTS products can be justified by frequent use of the module. Hence, more than one version is available for alternatives of a module. Further, the best of testing efforts are required to improve the reliability of the in-house-built component. This also leads to an increase in cost. Therefore,
it is necessary to carry a trade-off between reliability and cost of the software system.

This chapter is divided into two sections. Optimization models for component selection are formulated for fault tolerant software system under consensus recovery block scheme using build-or-buy strategy. The structures of the software discussed in both the sections are different. In Section 4.1, optimization models are formulated for optimal component selection for a fault tolerant software system. A software system is developed to perform various functions and is composed of various modules. On an invocation of a function module is called which in turn calls an alternative. These alternatives can be either COTS software components or in-house built components. Therefore, this section aims at selection of components (alternatives) for modules in a software system. This approach is further broken down in Section 4.2, each alternative may have different versions and only one version can be selected. The objective of this section is to formulate optimization models to select a version for an alternative of a module in a software system. The issue of compatibility amongst the COTS components of the modules is also discussed in both the sections. Solution methodology is illustrated with numerical example in Section 4.1 and a case study in Section 4.2.

4.1 FUZZY MULTI-OPTIMIZATION “BUILD-OR-BUY” MODEL

In this section, we discuss the formulation and solution of fuzzy component selection problem for a fault tolerant software system incorporating “build-or-buy” strategy. A component based software system has a modular structure to perform a set of functions with different modules having different alternatives for each module. A schematic representation of the software system is given in figure 4.1.1. We are selecting the alternatives (in-house or COTS) for modules to maximize the system reliability by simultaneously minimizing the cost.
Figure 4.1.1: Structure of Software

- Structure of Software
- Function to be performed by software system
- System Module
- System Component
  - (COTS/ In-house)
- Set of Alternative Components

Figure 4.1.1: Structure of Software
4.1.1 Notations

- \( R \) System quality measure
- \( C \) Overall system cost
- \( f_i \) Frequency of use, of function \( l \)
- \( s_i \) 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 \( i \)
- \( t_1 \) Probability that next alternative is not invoked upon failure of 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 \( i \)
- \( C_{ij} \) Cost of alternative \( j \) for module \( i \) of a COTS component
- \( s_{ij} \) Reliability of alternative \( j \) of module \( i \) of a COTS component
- \( d_{ij} \) Delivery time of alternative \( j \) of module \( i \) of a COTS component
- \( c_{ij} \) Unitary development cost for alternative \( j \) of module \( i \) of in-house build component
- \( t_{ij} \) Estimated development time for alternative \( j \) of module \( i \) of in-house build component
- \( \tau_{ij} \) Average time required to perform a test case for alternative \( j \) of module \( i \) of in-house build component
- \( \pi_{ij} \) Probability that a single execution of software fails on a test case chosen from a certain input distribution of in-house build component
- \( R_0 \) Minimum reliability level (aspiration) required to attain for system reliability
- \( C_0 \) Maximum budget (aspiration level) required for component selection in CBSE
4.1.2 Assumptions

The proposed optimization models developed hold good for the following situations

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

2. 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.

3. Codes written for integration of modules don’t contain any bug.

4. 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 or In-house build components) are attached in the modules and work similar to the consensus recovery block scheme discussed in [Kumar, (1998); Berman and Kumar, (1999)].

5. The cost of an alternative is the development cost, if developed in house; otherwise it is the buying price for the COTS product. Reliability for all the COTS components are known and no separate testing is done.

6. Different COTS alternatives with respect to cost, reliability and delivery time of a module are available.

7. Different in-house alternatives with respect to unitary development cost, estimated development time, average time and testability of module are available.

8. Cost and reliability of an in-house component can be specified by using basic parameters of the development process, e.g. a component cost may depend on a measure of developer skills, or the component reliability depends on the amount of testing.
4.1.3 Model Formulation

Optimization models in this section aim at selecting the right mix of components for a fault tolerant software system. Let $S$ be a software architecture made of $n$ modules, with a maximum number of $m_i$ alternatives (COTS or in-house) available for each module.

4.1.3.1 Build versus Buy Decision

For each module $i$, if an alternative is bought (i.e. some $x_{ij} = 1$) then there is no in-house development (i.e. $y_{ij} = 0$) and vice versa.

$$y_{ij} + x_{ij} \leq 1 \quad ; \quad i=1,2,\ldots,n \quad ; \quad j=1,2,\ldots,m_i$$

4.1.3.2 Redundancy

The equation stated below guarantees that redundancy is allowed for both the build and buy components (i.e. in-house and COTS components).

$$y_{ij} + x_{ij} = z_{ij}$$

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

4.1.3.3 Probability of Failure Free In-house Developed Component

The possibility of reducing the probability that the $j^{th}$ alternative of $i^{th}$ module fails by means of a certain amount of test cases (represented by the variable $N_{ij}^{tot}$). Cortellessa et al, (2008) define probability of failure on demand of an in-house developed $j^{th}$ alternative of $i^{th}$ module, under the assumption that the on-field users’ operational profile is the same as the one adopted for testing [Bertolino and Strigini, (1996)]. Basing on the testability definition, we can assume that the number $N_{ij}^{ suc}$ of successful (i.e. failure free) tests performed on $j^{th}$ alternative of same module.

$$N_{ij}^{ suc} = (1 - \pi_{ij}) N_{ij}^{tot} \quad ; \quad i=1,2,\ldots,n \quad ; \quad j=1,2,\ldots,m_i$$
Let A be the event “$N_{ij}^{\text{succ}}$ failure-free test cases have been performed” and B be the event “the alternative is failure free during a single run”. If $\rho_{ij}$ is the probability that in-house developed alternative is failure free during a single run given that $N_{ij}^{\text{succ}}$ test cases have successfully performed, from the Bayes Theorem we get

$$
\rho_{ij} = P(B/A) = \frac{P(A/B)P(B)}{P(A/B)P(B) + P(A/B)P(B)}
$$

The following equalities come straight forwardly

- $P(A/B) = 1$
- $P(B) = 1 - \pi_{ij}$
- $P(A/B) = (1 - \pi_{ij})^{N_{ij}^{\text{succ}}}$
- $P(B) = \pi_{ij}$

Therefore, we have

$$
\rho_{ij} = \frac{1 - \pi_{ij}}{(1 - \pi_{ij}) + \pi_{ij}\left(1 - \pi_{ij}\right)^{N_{ij}^{\text{succ}}}} ; \quad i = 1,2,\ldots,n \quad ; \quad j = 1,2,\ldots,m_i
$$

### 4.1.3.4 Reliability Equation of Both In-house and COTS Components

The reliability ($r_{ij}$) of $j^{th}$ alternative of $i^{th}$ module of the software is given by the following equation

$$
r_{ij} = \rho_{ij}y_{ij} + x_{ij}s_{ij} \quad ; \quad i = 1,2,\ldots,n \quad ; \quad j = 1,2,\ldots,m_i
$$

### 4.1.3.5 Delivery Time Constraint

The delivery time ($d_{ik}$) of the COTS components is given by the vendor and the development time of in-house component ($t_{ij} + r_{ij}N_{ij}^{\text{succ}}$) is estimated by the software development team. To know their value precisely in a real world situation is a difficult task due to many factors involved in either developing or purchasing of the components.
The delivery time \( T_{DT} \) for acquiring all the components (COTS or in-house) for the development of modular software system can be estimated using the following equation

\[
\bar{T}_{DT} = \bar{T}_{SD} - \bar{T}_{IT} - \bar{T}_{ST}
\]

where, \( \bar{T}_{SD} \) is the system development time, which is a function of integration testing time denoted by \( \bar{T}_{IT} \), system testing time \( \bar{T}_{ST} \) and delivery time of acquiring the components \( \bar{T}_{DT} \). The development team estimates these values in the early stage of software development. \( \bar{T}_{SD} \) depend upon various factors such as testing strategies, testing environment, team constitution, market completion, vendors’ credentials, etc. The information and data needed to compute these either not available or partially available. This problem can be resolved by taking these values as fuzzy numbers.

It becomes arduous for the managers to determine the exact delivery time of acquiring the components for the development of modular software system. Therefore, the manager has to allow some level of tolerance to the delivery time constraint and the equation can be written as

\[
\bar{T}_{DT} \leq T_u
\]

The crisp form of the above delivery time constraint can then be written as

\[
y_{ij} \left( t_{ij} + \tau_{i}m^{tot}_{ij} \right) + d_{ij}x_{ij} \leq T_u \quad ; \quad i = 1,2,\ldots,n \quad ; \quad j = 1,2,\ldots,m_{l}
\]

where \( T_u \) is the tolerance level for the delivery time constraint and is decided by the manager.

### 4.1.4 Objective Function

Reliability and cost are the two objectives which are applicable to this model.

#### 4.1.4.1 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 (AHP) can be effectively used to calculate these weights.

\[
\text{Maximize } R = \sum_{i=1}^{I} f_i \prod_{j=1}^{R_i} R_i
\]

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

\[
R_i = \left[ \frac{1}{\prod_{j=1}^{m_i} (1 - \eta_j)^{s_j}} \right] \frac{1}{\prod_{j=1}^{m_i} (1 - \eta_j)^{s_j}} + \sum_{j=1}^{m_i} \left[ \prod_{k=1}^{l} \left( X_{ik} \right)^{z_{ik}} \right] \left( Y_{ik} \right)^{z_{ik}} - 1 \quad i=1,2, \ldots, n
\]

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

\[
P \left( Y_{ij} \right) = r_{ij} (1 - t_2)
\]

**4.1.4.2 Cost Objective Function**

Cost objective function minimizes the overall cost of the system. The sum of the cost of all the modules is selected from “build-or-buy” strategy. The in-house development cost of the alternative \( j \) of module \( i \) can be expressed as \( c_{ij} \left( t_{ij} + \tau y_{ij} \right) \):

\[
\text{Minimize } \tilde{C} = \sum_{i=1}^{n} \left( \sum_{j=1}^{m_i} c_{ij} \left( t_{ij} + \tau y_{ij} \right) y_{ij} + \sum_{j=1}^{m_i} C_{ij} x_{ij} \right)
\]

\( \sim \) on the objective functions represents that they are fuzzy numbers. The problem with reliability maximization and cost minimization objectives subject to delivery time and component selection constraints can be considered as a multiple objective problem of reliability and cost while solving with the fuzzy optimization.

**4.1.5 Problem Description**

Principle to multi-objective optimization is the concept of an “efficient solution”, where any improvement in one objective can only be achieved at the expense of another. Due to increase in multiple functionalities in software system development as a pre-requisite client constraint, component based modular software system has
gained prominence over other software structures. In real life due to number of reasons developers can only make ambiguous estimates on the available resources and their aspirations which bring uncertainty (fuzziness) in the problem formulation. 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. Crisp optimization technique does not provide mechanism to deal with such intense situation as it requires well defined and exact objective functions and constraints set. In such scenario, the fuzzy approach can be used as an effective tool for quickly obtaining a good compromised solution. As defining the problem under fuzzy environment provides a better platform to quantify such uncertainties. Therefore, in the following sections we have formulated fuzzy multi-objective optimization models for component selection using build-or-buy strategy. Two optimization models are formulated, one without compatibility constraint and other with compatibility constraint.

4.1.5.1 Fuzzy Multi-optimization “Build-or-Buy” Model without Compatibility Constraints

Multi-optimization “build-or-buy” model under fuzzy environment is formulated in Problem (4.1.P1). In the optimization model it is assumed that the alternatives of a module are in a Consensus Recovery Block [For details, refer Chapter 1, Section (1.1.7.3)].

Problem (4.1.P1)

\[
\begin{align*}
\text{Maximize} & \quad \bar{R} = \sum_{i=1}^{k} \prod_{j \in y} R_i \\
\text{Minimize} & \quad C = \sum_{i=1}^{n} \left( \sum_{j=1}^{m} c_{ij} \left( t_{ij} + \tau_{ij} N_{ij} \right) y_{ij} + \sum_{j=1}^{m} C_{ij} x_{ij} \right)
\end{align*}
\]

Subject to

\[
X \in S = \{ x_{ij} \text{ and } y_{ij} \text{ are binary variable/}
\]

\[
R_i = 1 + \left[ \sum_{j=1}^{m} \frac{1}{1 - y_j} \left( \prod_{k=1}^{m} \left[ 1 - (1-y_j)^{g} \right]^{z_k} \prod_{j=1}^{m} \left[ 1 - (1-y_j)^{g} \right]^{y_k} \right) \left[ \sum_{j=1}^{m} \left( \prod_{k=1}^{m} \left[ 1 - (1-y_j)^{g} \right]^{z_k} \right) \left( y_j^{y_k} \right)^{-1} \right] \right] i = 1, 2, \ldots, n \quad (4.1.1)
\]
Optimal Component Selection for Fault Tolerant Software Design under Consensus Recovery Block Scheme

\[ P(\mathbf{X}) = (1-t_1 \left[ (1-r_j)(1-t_3) + r_j t_2 \right] ) \]  \hspace{1cm} \text{(4.1.2)}

\[ P(\mathbf{Y}) = r_j (1-t_2) \]  \hspace{1cm} \text{(4.1.3)}

\[ N_{ij}^{\text{Suc}} = (1-\pi_{ij})^{X_{ij}^{\text{Tot}}} ; \; i=1,2,...,n \; \text{and} \; j=1,2,...,m_i \]  \hspace{1cm} \text{(4.1.4)}

\[ \rho_{ij} = \frac{1-\pi_{ij}}{(1-\pi_{ij})+\pi_{ij} \left( 1-\pi_{ij} \right) ^{N_{ij}^{\text{Suc}}}} ; \; i=1,2,...,n \; \text{and} \; j=1,2,...,m_i \]  \hspace{1cm} \text{(4.1.5)}

\[ r_{ij} = \rho_{ij} \cdot y_{ij} + s_{ij} \cdot x_{ij} ; \; i=1,2,...,n \; \text{and} \; j=1,2,...,m_i \]  \hspace{1cm} \text{(4.1.6)}

\[ y_{ij} + x_{ij} \leq 1 ; \; i=1,2,...,n \; \text{and} \; j=1,2,...,m_i \]  \hspace{1cm} \text{(4.1.7)}

\[ y_{ij} + x_{ij} = z_{ij} ; \; i=1,2,...,n \; \text{and} \; j=1,2,...,m_i \]  \hspace{1cm} \text{(4.1.8)}

\[ \sum_{j=1}^{m_i} z_{ij} \geq 1 ; \; i=1,2,...,n \]  \hspace{1cm} \text{(4.1.9)}

\[ y_{ij} \left( t_{ij} + r_{ij} N_{ij}^{\text{Tot}} \right) + d_{ij} x_{ij} \leq T_u ; \; i=1,2,...,n \; \text{and} \; j=1,2,...,m_i \]  \hspace{1cm} \text{(4.1.10)}

where \( \mathbf{X} \) is a vector of component \( x_{ijk} \), \( y_{ij} \) and \( z_{ij} \); \( i=1,2,...,n \; \text{and} \; j=1,2,...,m_i \; \text{and} \; k=1,2,...,V_{ij} \).

Here the first objective stated is the fuzzy reliability objective and the second one is the fuzzy cost objective. Constraint (4.1.1) estimates the reliability of module \( i \) for a system under consensus recovery block scheme. Constraint (4.1.2) is the probability of event that output of alternative \( j \) of module \( i \) is rejected and Constraint (4.1.3) is the probability of event that correct result of alternative \( j \) of module \( i \) is accepted. Constraint (4.1.4) ensures that failure free test cases have performed. Constraint (4.1.5) is the probability that in-house developed alternative is failure free during a single run given that test cases have successfully performed. Constraint (4.1.6) is the reliability of alternative \( j \) of module \( i \). Constraint (4.1.7) ensures that if an alternative is bought then there is no in-house development and vice-versa. Constraint (4.1.8)
and (4.1.9) guarantees that redundancy is allowed in the modules. Constraint (4.1.10) is the delivery time constraint.

### 4.1.5.2 Fuzzy Multi-optimization “Build-or-Buy” Model with Compatibility Constraints

The model discussed in this section is an extension of the problem (4.1.P1) [For details, refer (Section 4.1.5.1)]. It is observed that some COTS alternatives of a module may not be compatible with alternatives of another module. The optimization model discussed in this section addresses this problem. It is done, incorporating additional constraints in the optimization models. This constraint can be represented as \( x_{gs} \leq x_{hu} \), which means that if alternative \( S \) for module \( g \) is chosen, then alternative \( u_t, t = 1, \ldots, z \) have to be chosen for module \( h \).

\[
x_{gs} - x_{hu} \leq My_t \quad \ldots(4.1.11)
\]

\[
\sum_{i=1}^{z} y_i = z - 1 \quad \ldots(4.1.12)
\]

Constraint (4.1.11) and (4.1.12) 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 (4.1.12) can be relaxed as

\[
\sum_{i=1}^{z} y_i \leq z - 1 \quad \ldots(4.1.13)
\]

Therefore, the fuzzy multi-optimization “build-or-buy” model for component selection using compatibility constraints can be formulated as

**Problem (4.1.P2)**

Maximize \( \hat{R} = \sum_{j=1}^{L} f_j \prod_{i=x_j} R_i \)

Minimize \( \hat{C} = \sum_{i=1}^{n} \left( \sum_{j=1}^{m} c_{ij} \left( t_{ij} + \tau_{ij} N_{ij}^{tot} \right) y_{ij} + \sum_{j=1}^{m} C_{ij} x_{ij} \right) \)
Subject to

\[
X \in S \\
x_{gs} - x_{hu} \leq My \\
\sum_{i=1}^{z} y_i = z - 1
\]

Problems [(4.1.P1) and (4.1.P2)] cannot be solved using crisp optimization techniques as these methods do not provide well defined mechanism to handle ambiguities. Therefore, we adopt fuzzy optimization approach to solve these problems.

4.1.6 Solution Procedure

Following algorithm specifies the sequential steps required to solve the fuzzy mathematical programming problems discussed in Section 4.1.5

4.1.6.1 Solution Procedure of Fuzzy Multi-optimization “Build-or-Buy” Model without Compatibility Constraints

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

**Step 1:** Reliability \( R_0 \) and cost \( C_0 \) are considered to be Triangular Fuzzy Numbers (TFN). They are represented as \( A = (a_l, a_m, a_u) \). The crisp equivalents of the fuzzy parameters are computed using a defuzzification function (ranking of fuzzy numbers). We use the defuzzification function of the type \( F_2(A) = \frac{a_l + 2a_m + a_u}{4} \) to defuzzify the aspiration levels.

**Step 2:** Incorporate the objective function of the fuzzifier min (max) as a fuzzy constraint with a restriction (aspiration) level. The above problem (4.1.P1) can be rewritten as

**Problem (4.1.P3)**

Find \( X \)
Subject to

\[ R(X) = \sum_{j=1}^{n} f_j \prod_{i \in s_j} R_i \geq R_0 \]

\[ C(X) = \sum_{i=1}^{n} \sum_{j \in I_i} c_{ij} \left(t_{ij} + \tau_{ij} X_{ij} \right) y_{ij} + \sum_{i=1}^{n} C_{ij} x_{ij} \leq C_0 \]

\[ X \in S \]

where \( R_0 \) and \( C_0 \) are defuzzified aspiration levels of system reliability and cost.

**Step 3:** 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

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

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

\[
\mu_{C}(X) =\begin{cases} 
1 & ; C(X) \leq C_0 \\
\frac{C_0^* - C(X)}{C_0^* - C_0} & ; C_0 \leq C(X) < C_0^* \\
0 & ; C(X) > C_0^* 
\end{cases}
\]

where \( C_0 \) is the restriction and \( C_0^* \) is the tolerance levels to the fuzzy budget constraint.

**Step 4:** Employ extension principle to identify the fuzzy decision. While solving the problem its objective is 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 weight to measure the relative importance. On substituting the values for \( \mu_{R}(X) \) and \( \mu_{C}(X) \) the problem becomes
Problem (4.1.P4)

Maximize $\alpha$

Subject to

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

$\alpha \in [0, 1]$  
\(X \in S\)  
\(w_1, w_2 \geq 0\)  
\(w_1 + w_2 = 1\)

4.1.6.2 Solution Procedure of Fuzzy Multi-optimization “Build-or-Buy” Model with Compatibility Constraints

Similar algorithm (discussed in Section 4.1.6.1) for solving fuzzy optimization model can be followed for an optimization model with compatibility constraints. Hence the problem can be written as

Problem (4.1.P5)

Maximize $\alpha$

Subject to

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

$\alpha \in [0, 1]$  
\(X \in S\)  
\(x_{gs} - x_{hu} \leq My_t\)

\[
\sum_{i=1}^{z} y_t \leq z - 1
\]

\(w_1, w_2 \geq 0\)  
\(w_1 + w_2 = 1\)
4.1.7 Numerical Illustration

A numerical example is illustrated to describe the proposed methodology of fuzzy multi-optimization “build-or-buy” 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 available for module \((m_1)\), four alternatives are available for module \((m_2)\), and two alternatives for \((m_3)\).

A component-based software system is developed by integrating various COTS and in-house built components. For each function multiple COTS products are available in the market and the same components can also be developed in-house. The vendor provides information on cost and reliability of the COTS components. The cost and reliability values for in-house built components can be estimated by the software development team. The cost data set for COTS and in-house built alternatives are given in table 4.1.1 and 4.1.2 respectively.

Let the software is required to perform three functions, so \(L = 3\). The set of modules required for the three functions are given by \(S_1 = \{1,2,3\}, S_2 = \{1,3\}\) 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 4.1.1: Data Set of COTS Components for Fuzzy Multi-optimization “Build-or-Buy” Model

<table>
<thead>
<tr>
<th>Modules</th>
<th>COTS Alternative</th>
<th>Cost</th>
<th>Reliability</th>
<th>Delivery Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m_1)</td>
<td>(x_{11})</td>
<td>14.0</td>
<td>0.89</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(x_{12})</td>
<td>12.5</td>
<td>0.90</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>(x_{13})</td>
<td>15.0</td>
<td>0.86</td>
<td>2</td>
</tr>
<tr>
<td>(m_2)</td>
<td>(x_{21})</td>
<td>13.0</td>
<td>0.88</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>(x_{22})</td>
<td>11.0</td>
<td>0.98</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>(x_{23})</td>
<td>20.0</td>
<td>0.87</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(x_{24})</td>
<td>13.0</td>
<td>0.88</td>
<td>4</td>
</tr>
<tr>
<td>(m_3)</td>
<td>(x_{31})</td>
<td>16.0</td>
<td>0.87</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(x_{32})</td>
<td>16.0</td>
<td>0.91</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.1.1 shows the cost, reliability, and delivery time for the COTS components. The first column of table 4.1.1 lists the three modules of the software system. The
second column provides various alternatives for each module. The third, fourth and fifth columns provides the parameters of cost, reliability and delivery time for each alternative, respectively.

**Table 4.1.2: Data Set of In-house Built Components for Fuzzy Multi-optimization “Build-or-Buy” Model**

<table>
<thead>
<tr>
<th>Modules</th>
<th>Alternatives</th>
<th>$t_{ij}$</th>
<th>$\tau_{ij}$</th>
<th>$c_{ij}$</th>
<th>$\pi_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1$</td>
<td>$y_{11}$</td>
<td>8</td>
<td>0.05</td>
<td>5</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>$y_{12}$</td>
<td>6</td>
<td>0.05</td>
<td>4</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>$y_{13}$</td>
<td>7</td>
<td>0.05</td>
<td>4</td>
<td>0.002</td>
</tr>
<tr>
<td>$m_2$</td>
<td>$y_{21}$</td>
<td>9</td>
<td>0.05</td>
<td>5</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>$y_{22}$</td>
<td>5</td>
<td>0.05</td>
<td>2</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>$y_{23}$</td>
<td>6</td>
<td>0.05</td>
<td>4</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>$y_{24}$</td>
<td>5</td>
<td>0.05</td>
<td>3</td>
<td>0.002</td>
</tr>
<tr>
<td>$m_3$</td>
<td>$y_{31}$</td>
<td>6</td>
<td>0.05</td>
<td>4</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>$y_{32}$</td>
<td>5</td>
<td>0.05</td>
<td>3</td>
<td>0.002</td>
</tr>
</tbody>
</table>

Table 4.1.2 shows the parameters that we have collected for in-house development of components. For each component, the average development time $t_{ij}$ is given in the third column and the average time required to perform a single test $\tau_{ij}$ is given in the fourth column, the unitary development cost $c_{ij}$ is given in the fifth column, finally the component testability $\pi_{ij}$ is given in the last column.

**Aspiration & Tolerance Level of Reliability and Cost**

The TFN corresponding to the reliability and cost aspirations are tabulated in table 4.1.3. The values of these fuzzy parameters are assumed to be specified by the management based on past experiences and/or expert opinion. Defuzzified values of
these parameters are computed using the defuzzification function

\[ F_2(A) = \left( \frac{a_l + 2a_m + a_u}{4} \right) \]

and are given in table 4.1.3 along with their tolerance level.

Table 4.1.3: Defuzzified Values of Parameters (TFN)

<table>
<thead>
<tr>
<th>Delivery Time</th>
<th>TFN ((a_l, a_m, a_u))</th>
<th>Defuzzified Value ((F_2(A))) Aspiration Level</th>
<th>Tolerance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>(R=(0.79, 0.82, 0.85)) (C=(66, 71, 76))</td>
<td>(R_0=0.82) (C_0=71)</td>
<td>(R_0^* = 0.78) (C_0^* = 76)</td>
</tr>
<tr>
<td>10</td>
<td>(R=(0.990, 0.995, 0.999)) (C=(60, 64, 68))</td>
<td>(R_0=0.995) (C_0=64)</td>
<td>(R_0^* = 0.97) (C_0^* = 75)</td>
</tr>
<tr>
<td>15</td>
<td>(R=(0.990, 0.995, 0.999)) (C=(67, 71, 75))</td>
<td>(R_0=0.995) (C_0=71)</td>
<td>(R_0^* = 0.97) (C_0^* = 75)</td>
</tr>
</tbody>
</table>

Membership Functions

Membership functions for reliability and cost for different delivery time can be written in a similar way as discussed in step 3 of the algorithm discussed in Section 4.1.6.1.

Assignment of Weights

The assignment of weights is based on the expert’s judgment for the reliability and cost, i.e. \(w = (w_1, w_2) = (0.6, 0.4)\).

4.1.7.1 Solution of Fuzzy Multi-optimization “Build-or-Buy” Model without Compatibility Constraints

The optimal solution of problem (4.1.P4) is optimal for problem (4.1.P1). Solving problem (4.1.P4) for the data sets given in table 4.1.1, 4.1.2 and 4.1.3, we get the following solution. The problem is solved by software package LINGO (Version 11).
Table 4.1.4: Solution of Fuzzy Multi-optimization “Build-or-Buy” Model without Compatibility Constraints

<table>
<thead>
<tr>
<th>Delivery Time</th>
<th>COTS Components</th>
<th>In-House Components</th>
<th>Optimal Cost</th>
<th>Optimal Reliability</th>
</tr>
</thead>
</table>
| 5             | $x_{11} = x_{12} = 1$  
$y_{22} = 1$  
$x_{32} = 1$ | - | 73.5 | 0.80 |
| 10            | $x_{12} = x_{13} = 1$  
$x_{24} = 1$  
$x_{32} = 1$ | $y_{22} = 1$ | 66.5 | 0.992 |
| 15            | $x_{12} = x_{13} = 1$  
$x_{23} = 1$ | $y_{22} = 1$  
$y_{32} = 1$ | 72.5 | 0.994 |

Corresponding to output table 4.1.4, we can show diagrammatically the representation of outputs mentioned in below figures 4.1.2, 4.1.3 and 4.1.4.

Figure 4.1.2: Diagrammatic Representation for solution of Fuzzy Multi-optimization “Build-or-Buy” Model without Compatibility Constraints (Delivery Time = 5)
Chapter 4: Optimal Component Selection for Fault Tolerance Modular Software System using Build-or-Buy Policy under Fuzzy Environment

Figure 4.1.3: Diagrammatic Representation for solution of Fuzzy Multi-optimization “Build-or-Buy” Model without Compatibility Constraints (Delivery Time = 10)

Figure 4.1.4: Diagrammatic Representation for solution of Fuzzy Multi-optimization “Build-or-Buy” Model without Compatibility Constraints (Delivery Time = 15)
It can be seen from the table 4.1.4 and figures 4.1.2, 4.1.3 and 4.1.4 that when delivery time was low, then all COTS components are selected. As the delivery time increase from 5 to delivery time 10, there is a significant reduction in the cost and increase in the reliability. Therefore, solution at delivery time 10 can be considered as the best decision because if we increase the delivery time from 10 to 15 units, then cost has increased significantly and in reliability the change is negligible. Therefore the optimal decision is to buy four COTS components and build one component in-house. Redundancy is allowed in module (m_1) and (m_2) but not in module (m_3).

4.1.7.2 Solution of Fuzzy Multi-optimization “Build-or-Buy” Model with Compatibility Constraints

The aspiration and tolerance level of reliability, Cost, membership function and assignment of weights are exactly same as discussed in the above section [For details, refer table 4.1.3]. Since the solution at delivery time 10 is considered to be the best solution, therefore the compatibility condition is checked for the same. The optimal solution of problem (4.1.P5) is optimal for problem (4.1.P2). Solving problem (4.1.P5) for the data sets given in table 4.1.2, 4.1.3 and 4.1.4, we get the solution in table 4.1.5.

We assume third alternative of first module is compatible with first and third alternatives of second module.

**Table 4.1.5: Solution of Fuzzy Multi-optimization “Build-or-Buy” Model with Compatibility Constraints**

<table>
<thead>
<tr>
<th>Delivery Time</th>
<th>COTS Components</th>
<th>In-house Components</th>
<th>Optimal Cost</th>
<th>Optimal Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>( x_{12} = x_{13} = 1 ) ( x_{23} = 1 ) ( x_{32} = 1 )</td>
<td>( y_{22} = 1 )</td>
<td>73.5</td>
<td>0.992</td>
</tr>
</tbody>
</table>

It is observed by table 4.1.5 and figure 4.1.5 that due to the compatibility condition, third alternative of second module is selected as it is compatible with third alternative of first module.
4.2 FUZZY MULTI-OPTIMIZATION “BUILD-OR-BUY” MODEL FOR ENHANCED SOFTWARE STRUCTURE

In developing a software system, different functions are required to be performed and different modules are available for those functions. Each module is developed by integrating several alternatives which in turn have different versions. The schematic representation of the system is given in figure 4.2.1. Components are selected for modules using build-or-buy approach. If the selected component is a build component then the complete alternative is developed by the software development team. But if the component cannot be build in-house, then for each alternative different versions (COTS products) are available in the market. Cost, reliability and delivery time associated with different versions are obviously different. In this section, the joint optimization of reliability and cost is considered while incorporating the build-or-buy approach for selection of components in designing a fault-tolerant modular software system under the consensus recovery block scheme.

Figure 4.1.5: Diagrammatic Representation for solution of Fuzzy Multi-optimization “Build-or-Buy” Model with Compatibility Constraints (Delivery Time =10)
Figure 4.2.1: Enhanced Software structure
4.2.1 Notations

- \( R \) System quality measure
- \( C \) Overall system cost
- \( f_l \) 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 \( i \)
- \( t_1 \) 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 \( i \)
- \( C_{ijk} \) Cost of version \( k \) of alternative \( j \) of module \( i \) of a COTS component
- \( s_{ik} \) Reliability of version \( k \) of alternative \( j \) of module \( i \) of a COTS component
- \( d_{ijk} \) Delivery time of version \( k \) of alternative \( j \) of module \( i \) of a COTS component
- \( c_{ij} \) Unitary development cost for alternative \( j \) of module \( i \) of in-house build component
- \( t_{ij} \) Estimated development time for alternative \( j \) of module \( i \) of in-house build component
- \( \tau_{ij} \) Average time required to perform a test case for alternative \( j \) of module \( i \) of in-house build component
Optimal Component Selection for Fault Tolerant Software Design under Consensus Recovery Block Scheme

\( \pi_{ij} \) Probability that a single execution of software fails on a test case chosen from a certain input distribution of in-house build component

\( R_0 \) Minimum reliability level (aspiration) required to attain for system reliability

\( C_0 \) Maximum budget (aspiration level) required for component selection in CBSE

\( y_{ij} \) \( \begin{cases} 1, \text{ if the alternative } j \text{ of module } i \text{ is in-house developed.} \\ 0, \text{ otherwise} \end{cases} \)

\( x_{ijk} \) \( \begin{cases} 1, \text{ if the } k^{th} \text{ version of } j^{th} \text{ COTS alternative of the } i^{th} \text{ module is chosen} \\ 0, \text{ otherwise} \end{cases} \)

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

4.2.2 Assumptions

The following are the assumptions for the optimization models.

1. Software system consists of a finite number of modules.

2. 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.

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

4. 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/-in-House components) are attached in the modules and work similar to the consensus recovery block scheme is discussed. [Kumar, (1998); Berman and Kumar, (1999)]

5. The cost of an alternative is the development cost, if developed in-house; otherwise it is the buying price for the COTS product.

6. Different in-house alternatives with respect to unitary development cost, estimated development time, average time to perform a test case and testability of a module are available.
7. Cost, reliability and development time of an in-house component can be estimated by using basic parameters of the development process, e.g., a component cost may depend on a measure of developer skills, or the component reliability depends on the amount of testing.

8. Different versions (COTS products) with respect to cost, reliability and delivery time of alternatives of a module are available.

9. Other than the available cost-reliability-delivery time versions of an alternative, we assume the existence of a virtual version, which has a negligible reliability of 0.001 and zero cost and delivery time. These COTS components are denoted by index one in the third subscript of \( x_{ijk} \), \( c_{ijk} \), \( d_{ijk} \) and \( s_{ijk} \), for example, \( s_{i1j} \) denotes the reliability of the first version of alternative \( j \) for module \( i \).

4.2.3 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. The following are the constraints for optimization models.

4.2.3.1 Build versus Buy Decision

For each module \( i \), if an alternative is bought (i.e., some \( x_{ijk} = 1 \)), then there is no in-house development (i.e., \( y_j = 0 \)) and vice versa.

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

4.2.3.2 Redundancy Constraint

The equation stated below guarantees that redundancy is allowed for both the build-and-buy components (i.e., in-house and COTS components).

\[
y_j + \sum_{k=2}^{V_i} x_{ijk} = z_{ij}
\]

\[
x_{ij1} + z_{ij} = 1 \quad ; \quad j = 1, 2, \ldots, m_i
\]
\[ \sum_{j=1}^{m_i} z_{ij} \geq 1 \quad ; \quad i = 1, 2, ..., n \]

### 4.2.3.3 Probability of Failure-free In-house Developed Component

The possibility of reducing the probability that the \( j^{th} \) alternative of \( i^{th} \) module fails by means of a certain amount of test cases (represented by the variable \( N_{ij}^{tot} \)). Cortellessa et al. (2008) defined the probability of failure on demand of an in-house developed \( j^{th} \) alternative of \( i^{th} \) module, under the assumption that the on-field users’ operational profile is the same as the one adopted for testing [Bertolino and Strigini, (1996)].

Basing on the testability definition, we can assume that the number \( N_{ij}^{suc} \) of successful (i.e. failure-free) tests performed on \( j^{th} \) alternative of same module.

\[
N_{ij}^{suc} = \left( 1 - \pi_{ij} \right) N_{ij}^{tot} \quad ; \quad i = 1, 2, ..., n \quad ; \quad j = 1, 2, ..., m_i
\]

Let \( A \) be the event “\( N_{ij}^{suc} \) failure-free test cases have been performed” and \( B \) be the event “the alternative is failure-free during a single run”. If \( \rho_{ij} \) is the probability that the in-house developed alternative is failure-free during a single run given that \( N_{ij}^{suc} \) test cases have been successfully performed, from the Bayes theorem, we get

\[
\rho_{ij} = P(B / A) = \frac{P(A / B)P(B)}{P(A / B)P(B) + P(A / \overline{B})P(\overline{B})}.
\]

The following equalities come straight forwardly

- \( P(A / B) = 1 \)
- \( P(B) = 1 - \pi_{ij} \)
- \( P(A / \overline{B}) = (1 - \pi_{ij})^{N_{ij}^{suc}} \)
- \( P(\overline{B}) = \pi_{ij} \)

Therefore, we have

\[
\rho_{ij} = \frac{1 - \pi_{ij}}{\left( 1 - \pi_{ij} \right) + \pi_{ij} \left( 1 - \pi_{ij} \right)^{N_{ij}^{suc}}} \quad ; \quad i = 1, 2, ..., n \quad ; \quad j = 1, 2, ..., m_i
\]
### 4.2.3.4 Reliability Equation of Both In-house and COTS Components

As already mentioned, 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

\[
 r_{ij} = \rho_{ij} y_{ij} + s_{ij} \quad ; \quad i = 1, 2, \ldots, n \quad ; \quad j = 1, 2, \ldots, m_i
\]

where

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

### 4.2.3.5 Delivery Time Constraint

The delivery time \( d_{ijk} \) of the COTS components is given by the vendor and the development time of in-house component \( t_{ij} + \tau_{ij} N_{\text{tot}} \) is estimated by the software development team. To know their value precisely in a real world situation is a difficult task due to many factors involved in either developing or purchasing of the components.

The delivery time \( T_{DT} \) for acquiring all the components (COTS or in-house) for the development of modular software system can be estimated using the following equation

\[
 T_{DT} = T_{SD} - T_{IT} - T_{ST}
\]

where, \( T_{SD} \) is the system development time, which is a function of integration testing time denoted by \( T_{IT} \), system testing time \( T_{ST} \) and delivery time of acquiring the components \( T_{DT} \). The development team estimates these values in the early stage of software development. \( T_{SD} \) depend upon various factors such as testing strategies, testing environment, team constitution, market competition, vendors’ credentials, etc. The information and data needed to compute these either not available or partially available. This problem can be resolved by taking these values as fuzzy numbers.
It becomes arduous for the managers to determine the exact delivery time of acquiring the components for the development of modular software system. Therefore, the manager has to allow some level of tolerance to the delivery time constraint and the equation can be written as

\[ \tilde{T}_{DR} \leq T_{u} \]

The crisp form of the above delivery time constraint can then be written as

\[ y_{ij} \left( t_{ij} + \tau_{ij} N_{ij}^{tot} \right) + \sum_{k=1}^{l_{ij}} d_{ijk} x_{ijk} \leq T_{u} \quad \forall i = 1, 2, ..., n \quad \forall j = 1, 2, ..., m_{ij} \]

where \( T_{u} \) is the tolerance level for the delivery time constraint and is decided by the manager.

### 4.2.4 Objective Function

The Reliability and Cost objective functions are discussed in the following sections.

#### 4.2.4.1 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.

\[ \text{Maximize} \quad \tilde{R} = \sum_{i=1}^{j} f_{i} \prod_{i \in \mathcal{S}_{i}} R_{i} \]

where \( R_{i} \) is the reliability of module \( i \) of the system under the consensus recovery block scheme which is stated

\[ R_{i} = 1 + \left[ \sum_{j=1}^{m_{ij}} \frac{1}{1-(1-r_{ij})^{y_{ij}}} \left[ \prod_{k=1}^{m_{ij}} (1-r_{ijk})^{z_{ijk}} \right] \right] \prod_{j=1}^{m_{ij}} (1-t_{ij})^{r_{ij}} \left[ \prod_{k=1}^{m_{ij}} P(X_{ijk})^{z_{ijk}} \right] P(Y_{ij}) - 1 \quad \forall i = 1, 2, ..., n \]

\[ P(X_{ij}) = (1-t_{ij}) \left[ (1-r_{ij})(1-t_{ij}) + r_{ij} t_{ij} \right] \]

\[ P(Y_{ij}) = r_{ij} (1-t_{ij}) \]
4.2.4.2 Cost Objective Function

Cost objective function minimizes the overall cost of the system. The sum of the cost of all the modules is selected from the “build-or-buy” strategy. The in-house development cost of the alternative \( j \) of module \( i \) can be expressed as \( c_{ij} \left( t_{ij} + \tau_{ij} N_{ij} \right) \).

\[
\text{Minimize } C = \sum_{i=1}^{n} \sum_{j=1}^{m} \left( c_{ij} \left( t_{ij} + \tau_{ij} N_{ij} \right) \right) y_{ij} + \sum_{k=1}^{p} C_{jk} x_{jk}
\]

\( \sim \) on the objective functions represents that they are fuzzy numbers. Here we assume them to be Triangular Fuzzy Numbers (TFN) [Bector and Chandra, (2005)]. We have adopted fuzzy optimization technique to solve the component selection problem discussed in next section.

4.2.5 Problem Description

Selection of components using build-or-buy strategy for the development of modular software system is a complex task. The functions that software has to perform are identified during the initial phase of the software development. For most of the functionalities, various COTS components are available in the market. These components can also be developed in-house. It is difficult to know the exact cost and reliability values of the components because of many factors involved in the development process which are vague and imprecise such as testing efficiency, testing strategies, market competition, vendors’ credentials etc. Fuzzy optimization is a flexible approach that permits more adequate solutions of real problems in the presence of vague information, providing the well-defined mechanisms to quantify the uncertainties directly. Therefore, we formulate fuzzy multi-objective optimization model for software components selection using build-or-buy strategy.

4.2.5.1 Fuzzy Multi-optimization “Build-or-Buy” Model for Enhanced Software Structure without Compatibility Constraints

Multi-optimization “build-or-buy” model under fuzzy environment is formulated in Problem (4.2.P1). Also in the optimization model it is assumed that the alternatives of a module are in a Consensus Recovery Block [For details, refer Chapter 1, Section (1.1.7.3)].
Problem (4.2.P1)

Maximize $R = \sum_{i=1}^{n} R_i$

Minimize $C = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left( c_{ij} \left( t_{ij} + \tau_{ij} N_{ij}^{tot} \right) y_{ij} + \sum_{k=1}^{v_i} C_{ijk} x_{ijk} \right)$

Subject to

$X \in S = \{ x_{ijk} \}$ and $y_{ij}$ are binary variable/

$R_i = 1 + \left[ \sum_{j=1}^{m_i} \frac{1}{\pi_{ij}} \left( \pi_{ij} \right)^{R_i} \right] + \prod_{k=1}^{v_i} P(X_{ij}) = \left( R_i \right)_{ij}$

$... (4.2.1)$

$P(Y_{ij}) = r_{ij} \left( 1 - t_2 \right)$

$... (4.2.2)$

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

$... (4.2.3)$

$N_{ij}^{SUC} = \left( 1 - \pi_{ij} \right) N_{ij}^{tot}$

$... (4.2.4)$

$\rho_{ij} = \frac{1 - \pi_{ij}}{\left( 1 - \pi_{ij} \right) + \pi_{ij} N_{ij}^{SUC}}$

$... (4.2.5)$

$r_{ij} = \rho_{ij} y_{ij} + s_{ij}$

$... (4.2.6)$

$y_{ij} + \sum_{k=1}^{v_i} x_{ijk} = 1$

$... (4.2.7)$

$y_{ij} + \sum_{k=2}^{v_i} x_{ijk} = z_{ij}$

$... (4.2.8)$

$x_{ij1} + z_{ij} = 1$

$... (4.2.9)$

$\sum_{j=1}^{n_i} z_{ij} \geq 1$

$... (4.2.10)$

$y_{ij} \left( t_{ij} + \tau_{ij} N_{ij}^{tot} \right) + \sum_{k=1}^{v_i} x_{ijk} \leq T_u$

$... (4.2.11)$
where $X$ is a vector of components $x_{ijk}$, $y_{ij}$, and $z_{ij}$; $i = 1, \ldots, n$; $j = 1, \ldots, m_i$; $k = 1, \ldots, V_{ij}$

Here the first objective stated is the fuzzy reliability objective and the second one is the fuzzy cost objective. Constraint (4.2.1) estimates the reliability of module $i$ for a system under consensus recovery block scheme. Constraint (4.2.2) is the probability of event that output of alternative $j$ of module $i$ is rejected and Constraint (4.2.3) is the probability of event that correct result of alternative $j$ of module $i$ is accepted. Constraint (4.2.4) ensures that failure free test cases have performed. Constraint (4.2.5) is the probability that in-house developed alternative is failure free during a single run given that test cases have successfully performed. Constraint (4.2.6) is the reliability of alternative $j$ of module $i$. Constraint (4.2.7) ensures that if an alternative is bought then there is no in-house development and vice-versa. Constraint (4.2.9) and (4.2.10) guarantee that not all chosen alternatives of modules are dummies. Constraint (4.2.11) is the delivery time constraint.

4.2.5.2 Fuzzy Multi-optimization “Build-or-Buy” Model for Enhanced Software Structure with Compatibility Constraints

In a structured software design, functionality and data are arranged in software modules. Each module has a set of procedures, or methods, for accessing the encapsulated data. Modules can be treated as components, for example, taken from libraries, or implemented by different vendors. This raises the question of when two modules are compatible. Optimization model discussed in this section is an extension of optimization model discussed in Section 4.2.5.1. In optimization model discussed in Section 4.2.5.2, we assumed that all alternative COTS products of one module are compatible with the alternative COTS products 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 problems such as implementation, interfaces, and licensing [Jung and Choi, (1999)]. Optimization model in this section addresses this problem. It is done by incorporating additional constraints in the optimization models [For details, refer Chapter 2 (Section 2.1.3.2)].
Problem (4.2.P2)

\[
\tilde{\text{Maximize}} \ R = \sum_{i=1}^{l} f_{i} \prod_{j=1}^{x} R_{i} \\
\tilde{\text{Minimize}} \ C = \sum_{i=1}^{n} \sum_{j=1}^{m_{i}} \left( c_{y} \left( t_{y} + r_{j} N_{j}^{\text{tot}} \right) y_{j} + \sum_{k=1}^{y_{j}} C_{y,k} x_{y,k} \right)
\]

Subject to

\[X \in S \]
\[x_{gs} - x_{mu} \leq M_{yt}, \; q = 2, \ldots, V_{gs} ; \; c = 2, \ldots, V_{mu} ; \; s = 1, \ldots, m_{s} \] (4.2.12)
\[\sum y_{i} \leq z \left( V_{mu} - 2 \right) \] (4.2.13)

Problems [4.2.P1 and 4.2.P2] cannot be solved using crisp optimization techniques as these methods do not provide well defined mechanism to handle ambiguities. Therefore, fuzzy optimization approach is used to solve these problems.

4.2.6 Solution Procedure

This section presents the solution procedure to solve the optimization problems discussed in Section 4.2.5.

4.2.6.1 Solution procedure of Fuzzy Multi-optimization “Build-or-Buy” Model for Enhanced Software Structure without Compatibility Constraints

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

**Step 1:** Reliability $R_{0}$ and cost $C_{0}$ are considered to be Triangular Fuzzy Numbers (TFN). They are represented as $A=\left(a_{l}, a_{m}, a_{u}\right)$. The crisp equivalents of the fuzzy parameters are computed using a defuzzification function (ranking of fuzzy numbers).

We use the defuzzification function of the type $F_{2}(A)=\frac{a_{l}+2a_{m}+a_{u}}{4}$ to defuzzify the aspiration levels.

**Step 2:** Incorporate the objective function of the fuzzifier min (max) as a fuzzy constraint with a restriction (aspiration) level. The above problem (4.2.P1) can be rewritten as
Chapter 4: Optimal Component Selection for Fault Tolerance Modular Software System using Build-or-Buy Policy under Fuzzy Environment

Problem (4.2.P3)

Find $X$

Subject to

$$R(X) = \sum_{i=1}^{l} f_i \prod_{s \in _i} R_s \geq R_0$$
$$C(X) = \sum_{i=1}^{m} \sum_{j=1}^{m_i} \left( c_{ij} \left( t_{ij} + \tau_{ij} N_{ij}^{out} \right) \right) y_{ij} + \sum_{k=1}^{k_i} C_{ijk} x_{ijk} \leq C_0$$

$X \in S$

where $R_0$ and $C_0$ are defuzzified aspiration levels of system reliability and cost.

Step 3: Define appropriate membership functions for each fuzzy inequality as well as constraint corresponding to the objective function. The membership function for the fuzzy parameters less than or equal to and greater than or equal to type are given as

$$\mu_R(X) = \begin{cases} 1 & ; R(X) \geq R_0 \\ \frac{R(X) - R_0'}{R_0 - R_0'} & ; R_0' \leq R(X) < R_0 \\ 0 & ; R(X) < R_0' \end{cases}$$

where $R_0$ is the aspiration level and $R_0'$ is the tolerance levels to the fuzzy reliability objective constraint.

$$\mu_C(X) = \begin{cases} 1 & ; C(X) \leq C_0 \\ \frac{C_0 - C(X)}{C_0 - C_0'} & ; C_0 \leq C(X) < C_0' \\ 0 & ; C(X) > C_0' \end{cases}$$

where $C_0$ is the restriction and $C_0'$ is the tolerance level to the fuzzy budget constraint.

Step 4: Employ extension principle to identify the fuzzy decision, which results in a crisp mathematical programming problem given by

Problem (4.2.P4)

Maximize $\alpha$
Subject to
\[ \mu_R(x) \geq \alpha \]
\[ \mu_C(x) \geq \alpha \]
\[ X \in S \]
where \( \alpha \) represents the degree up to which the aspiration of the decision-maker is met. The above problem can be solved by the standard crisp mathematical programming algorithms.

**Step 5**: While solving the problem following steps 1-4, the 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 a different level of importance and can be assigned weight to measure the relative importance. The resulting problem can be solved by the weighted min (max) approach. The crisp formulation of the weighted problem is given as

**Problem (4.2.P5)**

Maximize \( \alpha \)
Subject to
\[ \mu_R(x) \geq w_1 \alpha \]
\[ \mu_C(x) \geq w_2 \alpha \]
\[ X \in S \]
\[ w_1, w_2 \geq 0 \]
\[ w_1 + w_2 = 1 \]

If the constraints are fuzzy as well as crisp, then in the equivalent crisp mathematical programming problem, the original crisp constraints will not show any change as their tolerances are zero. The problem (4.2.P5) can be solved using the standard mathematical programming approach.

**Step 6**: On substituting the values for \( \mu_R(x) \) and \( \mu_C(x) \) the problem becomes

**Problem (4.2.P6)**

Maximize \( \alpha \)
Subject to

\[ R(x) \geq R_0 - (1 - w_1 \alpha)(R_0 - R_0^*) \]
\[ C(x) \leq C_0 + (1 - w_2 \alpha)(C_0^* - C_0) \]
\[ X \in S \]
\[ \alpha \in [0,1] \]
\[ w_1, w_2 \geq 0 \]
\[ w_1 + w_2 = 1 \]

Step 7: If a feasible solution is not obtained for the problem \(4.2.P5\) or \(4.2.P6\), then we can use the fuzzy goal programming approach to obtain a compromised solution given by Mohamed, (1997). The method is discussed in detail in the case study [For details, refer Section 4.2.7].

4.2.6.2 Solution Procedure of Fuzzy Multi-optimization “Build-or-Buy” Model for Enhanced Software Structure with Compatibility Constraints

Following the similar algorithm of fuzzy optimization [for details, refer (Section 4.2.6.1)] problem \(4.2.P2\) can be transformed to

Problem (4.2.P7)

Maximize \(\alpha\)

Subject to

\[ R(x) \geq R_0 - (1 - w_1 \alpha)(R_0 - R_0^*) \]
\[ C(x) \leq C_0 + (1 - w_2 \alpha)(C_0^* - C_0) \]
\[ X \in S \]
\[ x_{gsq} - x_{huqc} \leq M y_t, q = 2, \ldots, V_{gs}; c = 2, \ldots, V_{huq}; s = 1, \ldots, m_g \]
\[ \sum y_t \leq z \left( V_{huq} - 2 \right) \]
\[ \alpha \in [0,1] \]
\[ w_1, w_2 \geq 0 \]
\[ w_1 + w_2 = 1 \]
4.2.7 Case Study

In this section, a case study of Component-Based Development (CBD) is presented to illustrate the proposed methodology of optimizing the selection of software components for a modular software system. A local software system supplier planned to develop a software system for small and medium size retail organizers. Nine functional requirements of the system were identified, namely, sales, payment collection and authorization, shift-wise reporting and statistics, inventory control and movements, e-commerce, automatic updates, security and administration, business rules, financials and reporting. The software system development team of the company has defined three software modules, front office ($m_1$), back office/store ($m_2$) and finance/accounts ($m_3$), that the retail software system needs to contain.

The front office ($m_1$) module mainly provides the functions of sales, payment collection and authorization, shift-wise reporting and statistics. The back office/store ($m_2$) module mainly provides the functions of inventory control and movements, e-commerce, automatic updates, security and administration, while the finance/accounts ($m_3$) module provides business rules, financials and reporting.

<table>
<thead>
<tr>
<th>Modules</th>
<th>Functional Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front Office</td>
<td>Sales</td>
</tr>
<tr>
<td></td>
<td>Payment Collection and Authorization</td>
</tr>
<tr>
<td></td>
<td>Shift-wise Reporting and Statistics</td>
</tr>
<tr>
<td>Back Office/Stores</td>
<td>Inventory Control and Movements</td>
</tr>
<tr>
<td></td>
<td>E-commerce</td>
</tr>
<tr>
<td></td>
<td>Automatic Updates</td>
</tr>
<tr>
<td></td>
<td>Security and Administration</td>
</tr>
<tr>
<td>Finance/Accounts</td>
<td>Business Rules/ Protocols</td>
</tr>
<tr>
<td></td>
<td>Financials and Reporting</td>
</tr>
</tbody>
</table>

A total of 18 COTS components available in market are considered. Further the cost of building these components are also estimated as the software supplier can also build these components. The decision is to choose the right mix of components for
each software module so as to get a reliable software system at a minimum cost in the
desired delivery time.

The system consists of three modules; front office \( (m_1) \), back office/store \( (m_2) \) and
finance/ account \( (m_3) \). Each module provides different functional requirements
mentioned in table 4.2.1. A system is to be developed by integrating components/
alternatives which can be either COTS or the in-house-built components. The
objective of this study is to select the optimal set of alternatives for each module so as
to get a highly reliable retail software system. For each module, various alternatives
are available, various COTS versions are available for each alternative of a module.
An in-house alternative for each module can be built. The data sets for COTS and in-
house-developed components are given in tables 4.2.2 and 4.2.3, respectively.

Let the software is required to perform three functions, so \( L = 3 \). The set of modules
required for the three functions are given by \( S_1 = \{1,2,3\} \), \( S_2 = \{1,3\} \), \( S_3 = \{2\} \). The
frequency of use is given by \( f_1 = 0.50 \), \( f_2 = 0.30 \) and \( f_3 = 0.20 \). It is also assumed that
\( t_1 = 0.01 \), \( t_2 = 0.05 \) and \( t_3 = 0.01 \).

**Table 4.2.2: Data Set of COTS Components for Fuzzy Multi-optimization
“Build-or-Buy” Model for Enhanced Software Structure**

<table>
<thead>
<tr>
<th>Modules</th>
<th>Alternatives</th>
<th></th>
<th>Versions</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cost</td>
<td>Reliability</td>
<td>Delivery Time</td>
<td>Cost</td>
<td>Reliability</td>
<td>Delivery Time</td>
</tr>
<tr>
<td>Front Office</td>
<td>1</td>
<td>0</td>
<td>0.001</td>
<td>0</td>
<td>19.0</td>
<td>0.77</td>
<td>4</td>
<td>16.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0.001</td>
<td>0</td>
<td>17.5</td>
<td>0.79</td>
<td>5</td>
<td>23.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0.001</td>
<td>0</td>
<td>22.0</td>
<td>0.80</td>
<td>3</td>
<td>20.0</td>
</tr>
<tr>
<td>Back Office/</td>
<td>1</td>
<td>0</td>
<td>0.001</td>
<td>0</td>
<td>18.0</td>
<td>0.81</td>
<td>5</td>
<td>22.5</td>
</tr>
<tr>
<td>Store</td>
<td>2</td>
<td>0</td>
<td>0.001</td>
<td>0</td>
<td>16.0</td>
<td>0.83</td>
<td>6</td>
<td>17.0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0.001</td>
<td>0</td>
<td>23.0</td>
<td>0.89</td>
<td>3</td>
<td>20.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>0.001</td>
<td>0</td>
<td>18.0</td>
<td>0.88</td>
<td>5</td>
<td>19.0</td>
</tr>
<tr>
<td>Finance/</td>
<td>1</td>
<td>0</td>
<td>0.001</td>
<td>0</td>
<td>21.0</td>
<td>0.92</td>
<td>4</td>
<td>23.0</td>
</tr>
<tr>
<td>Accounts</td>
<td>2</td>
<td>0</td>
<td>0.001</td>
<td>0</td>
<td>21.0</td>
<td>0.97</td>
<td>4</td>
<td>22.0</td>
</tr>
</tbody>
</table>
Table 4.2.2 gives cost, reliability, and delivery time for the COTS components. The first column of table 4.2.2 lists the three modules of the software system. The second column provides various alternatives for each module. Each alternative of a module has three versions. The third column provides the parameters of cost (in Kilo Euros, KE), reliability and delivery time (in weeks) for each version. Note that the cost of first version, i.e., the virtual versions for all COTS alternatives is zero and reliability is 0.001. This is done because “if in the optimal solution, for some module $x_{ij} = 1$, it implies corresponding alternative is not to be attached in the module”.

Table 4.2.3: Data Set for In-house Components for Fuzzy Multi-Optimization “Build-or-Buy” Model for Enhanced Software Structure

<table>
<thead>
<tr>
<th>Modules</th>
<th>Alternatives</th>
<th>Development Time, $t_{ij}$</th>
<th>Testing Time, $\tau_{ij}$</th>
<th>Unitary Development Cost, $c_{ij}$</th>
<th>Probability of Testability, $\pi_{ij}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front Office</td>
<td>1</td>
<td>9</td>
<td>0.005</td>
<td>4</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7</td>
<td>0.005</td>
<td>3</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>8</td>
<td>0.005</td>
<td>3</td>
<td>0.002</td>
</tr>
<tr>
<td>Back Office/Store</td>
<td>1</td>
<td>8</td>
<td>0.005</td>
<td>4</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>0.005</td>
<td>1</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>6</td>
<td>0.005</td>
<td>3</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6</td>
<td>0.005</td>
<td>2</td>
<td>0.002</td>
</tr>
<tr>
<td>Finance/Accounts</td>
<td>1</td>
<td>6</td>
<td>0.005</td>
<td>3</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>0.005</td>
<td>2</td>
<td>0.002</td>
</tr>
</tbody>
</table>

Table 4.2.3 shows the parameters that we have collected for in-house development of components. For each component, the average development time $t_{ij}$ (in weeks) is given in the third column and the average time required to perform a single test $\tau_{ij}$ (in weeks) is given in the fourth column, the unitary development cost $c_{ij}$ (KE per week) is given in the fifth column, finally the component testability $\pi_{ij}$ is given in the last column.
Aspiration & Tolerance Level of Reliability and Cost

The TFN corresponding to the reliability and cost aspirations are tabulated in table 4.2.4. The values of these fuzzy parameters are assumed to be specified by the management based on past experiences and/or expert opinion. Defuzzified values of these parameters are computed using the defuzzification function \( F_2(A) = \frac{a_l + 2a_m + a_u}{4} \) and are given in table 4.2.4 along with their tolerance level.

<table>
<thead>
<tr>
<th>Delivery Time</th>
<th>Triangular Fuzzy Numbers ((a_l, a_m, a_u))</th>
<th>Defuzzified Value (F_2(A))</th>
<th>Tolerance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>(R = (0.72,0.74,0.80)) (C = (105,108,115))</td>
<td>(R_0 = 0.75) (C_0 = 109)</td>
<td>(R_0^* = 0.71) (C_0^* = 117)</td>
</tr>
<tr>
<td>5</td>
<td>(R = (0.992,0.995,0.999)) (C = (83,91,95))</td>
<td>(R_0 = 0.995) (C_0 = 90)</td>
<td>(R_0^* = 0.94) (C_0^* = 95)</td>
</tr>
<tr>
<td>10</td>
<td>(R = (0.992,0.995,0.999)) (C = (83,91,95))</td>
<td>(R_0 = 0.995) (C_0 = 90)</td>
<td>(R_0^* = 0.94) (C_0^* = 95)</td>
</tr>
</tbody>
</table>

Membership Functions

Membership functions for reliability and cost for different delivery time can be written in a similar way as discussed in step 3 of the algorithm discussed in Section (4.2.6.1).

Assignment of Weights

The assignment of weights is based on the expert’s judgment for the reliability and cost, i.e. \(w = (w_1, w_2) = (0.6, 0.4)\).

Fuzzy Goal Programming Approach

On solving the problem, we found that the problem (4.2.P6) is not feasible; hence the management goal cannot be achieved for a feasible value of \(\alpha \in [0,1]\). Then, we use
the fuzzy goal programming technique to obtain a compromised solution. The approach is based on the goal programming technique for solving the crisp goal programming problem given by 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 (4.2.P6) introducing the negative and positive deviational variables $\eta_j$ and $\rho_j$ is given as

**Problem (4.2.P8)**

**Minimize** $u$

**Subject to**

$$
\mu_R (X) + \eta_1 - \rho_1 = 1
$$

$$
\mu_C (X) + \eta_2 - \rho_2 = 1
$$

$$
u \geq w_j \times \eta_j
$$

$$
\eta_j \times \rho_j = 0
$$

$$
\eta_j, \rho_j \geq 0 \quad ; j=1,2
$$

$$X \in S$$

$$\alpha \in [0,1]$$

$$w_1, w_2 \geq 0$$

$$w_1 + w_2 = 1$$

$$\alpha = 1 - u$$

### 4.2.7.1 Solution of Fuzzy Multi-optimization “Build-or-Buy” Model for Enhanced Software Structure without Compatibility Constraints

The solution to multi-optimization “build-or-buy” model for enhanced software structure without compatibility constraints for different delivery time are given in figure 4.2.2, figure 4.2.3 and figure 4.2.4. The same results is summarized in table 4.2.5. The model is solved using a software package called LINGO (Version 11).
The solution to the model gives the optimal components selection for the software system along with the corresponding cost and reliability of the overall system under fuzzy environment. The optimal solution of problem (4.2.P8) is optimal for problem (4.2.P1). Solving problem (4.1.P8) for the data sets given in table 4.2.2, 4.2.3 and 4.2.4.

CASE 1: Delivery time is assumed to be 3 weeks

At delivery time of 3 weeks, all COTS components are selected. For the front office module, third version of the second alternative and second version of the third alternative are selected. For the back office module, third version of the first alternative and second version of the third alternative are selected. For the finance/accounts module, third version of the second alternative is selected. Since more than one alternative is selected for front office \( (m_1) \) and back office \( (m_2) \).
modules, redundancy is allowed in these two modules. The overall system cost is 112.5 and system reliability is 0.74.

CASE 2: Delivery time is assumed to be 5 weeks

As delivery time increases to 5 weeks along with COTS components, in-house-built component is also selected. Redundancy is allowed for the first and second modules. For the back office module, two COTS components and an in-house-built component are selected. The overall system cost is reduced to 93 and system reliability is also improved and is 0.97.
CASE 3: Delivery time is assumed to be 10 weeks

At delivery time of 10 weeks, more than one in-house-built components are selected. For front office and back office modules, both COTS and in-house components are selected and hence redundancy is allowed. The overall system cost is 92 and system reliability is 0.990.

As we can see from the solution given in figure 4.2.2, figure 4.2.3, figure 4.2.4 and table 4.2.5, when delivery time increases from 3 to 5 weeks, there is a significant reduction in the overall cost and also there is a significant improvement in the reliability of the system. Also, the selected components are a combination of both COTS and in-house-built components. When we are increasing delivery time from 5 to 10 weeks, we are not achieving much reliability level and are not able to reduce the cost. So, it is advisable to keep delivery time at 5 weeks and its corresponding solutions.
The results discussed above in the three cases are summarized in the table 4.2.5

Table 4.2.5: Solution Summary of Cases

<table>
<thead>
<tr>
<th>Delivery Time</th>
<th>Modules</th>
<th>COTS Components</th>
<th>In-house Components</th>
<th>Optimal Cost</th>
<th>Optimal Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Front Office</td>
<td>( x_{111} = x_{123} = x_{132} = 1 )</td>
<td>-</td>
<td>112.5</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>Back Office/Store</td>
<td>( x_{213} = x_{221} = x_{232} = x_{241} = 1 )</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finance/Accounts</td>
<td>( x_{311} = x_{323} = 1 )</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Front Office</td>
<td>( x_{111} = x_{123} = x_{133} = 1 )</td>
<td>-</td>
<td>93.0</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>Back Office/Store</td>
<td>( x_{213} = x_{232} = x_{241} = 1 )</td>
<td>( y_{22} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finance/Accounts</td>
<td>( x_{311} = x_{322} = 1 )</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Front Office</td>
<td>( x_{111} = x_{123} = 1 )</td>
<td>( y_{13} )</td>
<td>92.0</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>Back Office/Store</td>
<td>( x_{211} = x_{241} = 1 )</td>
<td>( y_{22} = y_{23} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finance/Accounts</td>
<td>( x_{311} = x_{323} = 1 )</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2.7.2 Solution of Fuzzy Multi-optimization “Build-or-Buy” Model for Enhanced Software Structure with Compatibility Constraints

The aspiration and tolerance level of reliability, cost, membership functions and assignment of weights are exactly the same as discussed in the above section 4.1.7.1. The problem is solved by goal programming approach. By adding constraints of compatibility to problem (4.2.P8), a new problem can be written as (4.2.P9).
Chapter 4: Optimal Component Selection for Fault Tolerance Modular Software System using Build-or-Buy Policy under Fuzzy Environment

Problem (4.2.P9)

Minimize $u$

Subject to

\[
\mu_R (X) + \eta_1 - \rho_1 = 1 \\
\mu_C (X) + \eta_2 - \rho_2 = 1 \\
u \geq w_j * \eta_j \\
\eta_j * \rho_j = 0 \\
\eta_j, \rho_j \geq 0 ; j=1,2 \\
X \in S \\
x_{gsq} - x_{htc_{s}} \leq My_{t}, q = 2, \ldots, V_{gs} ; c = 2, \ldots, V_{htc_{s}} ; s=1, \ldots, mg \\
\sum y_{t} \leq z(V_{htc} - 2) \\
\alpha \in [0,1] \\
w_1, w_2 \geq 0 \\
w_1 + w_2 = 1 \\
\alpha = 1-u
\]

To check compatibility amongst the alternatives of the modules, we have considered case no. 2 of the above solution. The optimal solution of problem (4.2.P9) with compatibility constraints is optimal for problem (4.2.P2). Solving problem (4.1.P9) for the data sets given in table 4.1.2, 4.1.3 and 4.1.4, we get the solution in table 4.1.6

CASE 2: Delivery time is assumed to be 5 weeks

We assume that the second alternative of third module is compatible with second and third alternatives of the first module. The manager is interested in keeping the solution when delivery time is 5 weeks which according to him is the best decision.
Figure 4.2.5: Diagrammatic Representation for Solution of Fuzzy Multi-optimization “Build-or-Buy” Model for Enhanced Software Structure with Compatibility Constraints (Case 2)

Table 4.2.6: Solution of Fuzzy Multi-optimization “Build-or-Buy” Model for Enhanced Software Structure with Compatibility Constraints (Case 2)

<table>
<thead>
<tr>
<th>Delivery Time</th>
<th>Modules</th>
<th>COTS Components</th>
<th>In-house Components</th>
<th>Optimal Cost</th>
<th>Optimal Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Front Office</td>
<td>$x_{i11} = x_{i12} = x_{i13}$</td>
<td>-</td>
<td>94</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>Back Office/Store</td>
<td>$x_{i21} = x_{i23} = x_{i24}$</td>
<td>$y_{22}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finance/Accounts</td>
<td>$x_{i31} = x_{i32}$</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It is observed that due to the compatibility condition, third alternative of first module is chosen as it is compatible with second alternative of third module. The overall system cost is 94 and system reliability is 0.96.