American Journal of Operations Research, **2011, 1, 249-258**
doi:10.4236/ajor.2011.14029 Published Online December 2011 (http://www.SciRP.org/journal/ajor)

A Fuzzy Approach for Component Selection amongst Different Versions of Alternatives for a Fault Tolerant Modular Software System under Recovery Block Scheme Incorporating Build-or-Buy Strategy

P. C. Jha[1], Ritu Arora[2], U. Dinesh Kumar[3]
[1]*Department of Operational Research, University of Delhi, Delhi, India*
[2]*Maharaja Agrasen Institute of Technology, GGSIP University, Delhi, India*
[3]*Indian Institute of Management, Bangalore, India*
E-mail: jhapc@yahoo.com, arora_ritu21@yahoo.co.in, dineshk@iimb.ernet.in
Received September 20, 2011; *revised October* 22, 2011; *accepted November* 8, 2011

Abstract

Software projects generally have to deal with producing and managing large and complex software products. As the functionality of computer operations become more essential and yet more critical, there is a great need for the development of modular software system. Component-Based Software Engineering concerned with composing, selecting and designing components to satisfy a set of requirements while minimizing cost and maximizing reliability of the software system. This paper discusses the fuzzy approach for component selection using "Build-or-Buy" strategy in designing a software structure. We introduce a framework that helps developers to decide whether to buy or build components. In case a commercial off-the-shelf (COTS) component is selected then different versions are available for each alternative of a module and only one version will be selected. If a component is an in-house built component, then the alternative of a module is selected. Numerical illustrations are provided to demonstrate the model developed.

Keywords: Modular Software, Software Reliability, Software Components (COTS and In-House), Fault Tolerance & Fuzzy Optimization

1. Introduction

Computer software is very important in today's world. In particular, science and technology demand high quality software for making improvements and breakthroughs. The software development companies are continuously developing/modifying/updating their software according to the changing needs and requirements. The concept of "software reliability" and its measurement is receiving much attention in the software development community. Software reliability is one of the important parameters of software quality and system dependability. Software reliability engineering balances customer needs in the major quality characteristics of reliability, availability, delivery time and cost more effectively. The reliability of software can be controlled during the development life cycle through the application of reliability improvement techniques. Two of the best-known fault tolerant software design methods are N-version programming and Recovery block scheme. The basic mechanism of both the schemes is to provide redundant software to tolerate software failures. Software whose failure can have bad effects afterwards can be made fault tolerant through redundancy at module level [1].

When the design of software architecture reaches a good level of maturity, software engineers have to take a decision on the selection of software components. Non functional aspects play a significant role in determining software quality. Given the fact that lack of proper handling of non functional aspects of a software application has led to a series of software failures [2], nonfunctional attributes such as reliability security and performance should be considered during the component selection phase of software development. This paper discusses a framework that helps developers to decide whether to buy or build components of software architecture on the basis of cost and non functional factors. While developing software, components can be both bought as commercial off-the shelf (COTS) products, and probably adapted to work in the software system, or they can be developed in-house. This decision is known as "build-or-buy decision". This decision affects the overall cost and reliability of the system. Most of the current software systems include one or more COTS products. COTS are pieces of software that can be reused by software projects to build new systems. Benefits of COTS based development include significant reduction in the development cost, time and improvement in the dependability requirement. The components, which are not available in the market or cannot be purchased economically, can be developed within the organization and are known as in-house built components. Reference [3] discussed issues related to reliability of systems, caused by integrating COTS components. The optimal selection is achieved through weighted maximization of system quality subject to budget as a constraint in which an upper limit is placed over the constraint.

This paper discusses the issues related to reliability of the software systems and cost caused by integrating COTS or in-house built components. Fault tolerance is achieved through redundancy in Recovery Block model and redundancy results in additional cost. We assume that for all the alternatives available for a module, cost increases if higher reliability is desired. Several alternatives of a software module may be available as COTS, almost equivalent from the functional viewpoint. Purchase of high quality COTS products can be justified by the frequent use of the module. Large software systems possess the modular structure to perform a set of functions. Each function is performed by different modules having different alternatives for each module. In case, a COTS component is selected then different versions are available for each alternative of a module and only one version will be selected. If a component is an in-house built component, then the alternative of a module is selected. A schematic representation of the software system is given in **Figure 1**.

In the existing research related to the software decision, it is assumed that all the parameters of the problem are known precisely. Various objectives and restrictions are set by the management and cost coefficients involved in the cost function are determined based on past experience and the available data base. This makes it difficult for the management to provide precise values of the various cost coefficients and objectives to be met. Moreover due to changing customer specifications, lack of experience of testing team or novelty, changing testing environment, complexity in the project involved, unknown emerging factors at the start of the project adds

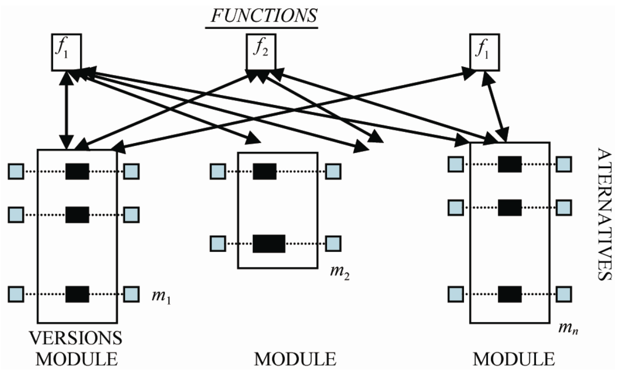

Figure 1. Structure of the software.

imprecision and ambiguity to the above-mentioned definitions. It may also be possible that the management itself does not set precise values in order to provide some tolerance on these parameters due to competitive considerations. All this leads to uncertainty (fuzziness) in the problem formulation. Crisp mathematical programming approaches provide no such mechanism to quantify these uncertainties. Fuzzy optimization is a flexible approach that permits more adequate solutions to real problems in the presence of vague information, providing well-defined mechanisms to quantify the uncertainties directly. The idea of fuzzy programming was first given by [4] and then developed by [5-7]. A number of researchers thereafter have contributed to the development of fuzzy optimization technique [8,9]. Today, similar to the developments in crisp optimization, different kinds of mathematical models have been proposed and many practical applications have been implemented by using the fuzzy set theory. Reference [10] formulated fuzzy multi objective optimization models for selecting the optimal COTS software products in the development of modular software system. Recently, Reference [11] de- velops a crisp multi-objective programming model from the fuzzy basic data. When a feasible solution to the problem exists, single and multiple objective fuzzy opti- mization procedure are used to solve the problem. How- ever, it is assumed that a crisp or a constant value of all the parameters is known. However, in practice, it is not possible for a management to obtain a precise value of reliability and cost for a software system. Or they may decide not to set precise levels due to the market considerations and are ready to have some tolerance of their objectives. When the precise values of parameter of the problem are not known, the problem becomes a fuzzy optimization problem and the solution so obtained is a fuzzy approximation.

This paper proposes two fuzzy multi-objective optimization models for selecting the best software product for each module. The first optimization model (optimization model-I) of this paper is a joint optimization problem that maximizes the system reliability with simultaneous minimization of the cost. The second optimization model (optimization model-II) considers the issue of compatibility between different alternatives of modules as it is observed that some COTS components cannot integrate with all the alternatives of another module. We apply fuzzy optimization procedure to solve the problem, when a feasible solution of the problem exists, but in case we reach the infeasibility case, then we apply fuzzy goal programming optimization technique to provide a compromised solution for the same. The rest of this paper is organized as follows. Section 2 consists of proposed notations. In Section 3, we discuss the assumptions of optimization models and we develop a crisp model for reliability and cost and in Section 4, we describe Fuzzy Multi-Objective Optimization Model for software products selection. In Section 5, Fuzzy optimization technique is discussed to solve the problem with numerical illustration. In Section 6, we furnish our concluding observations.

2. Notations

R : System quality measure

f_l : Frequency of use, of function l

s_l : Set of modules required for function l

R_i : Reliability of module i

L : Number of functions, the software is required to perform

n : Number of modules in the software

m_i : Number of alternatives available for module i

V_{ij} : Number of versions available for alternative j of module i

N_{ij}^{tot} : Total number of tests performed on the in-house developed instance (*i.e.* alternative of module i)

N_{ij}^{suc} : Number of successful (*i.e.* failure free) test performed on the in-house developed instance (*i.e.* 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.

X_{ij} : Event that output of alternative j of module i is rejected.

Y_{ij} : Event that correct result of alternative j of module i is accepted.

s_{ij} : Reliability of alternative j of module i

r_{ij} : Reliability of alternative j for module i

C_{ijk} : Cost of version k of alternative j for module i

r_{ijk} : Reliability of version k of alternative j for module i

d_{ijk} : Delivery time of version k of alternative j for module i

c_{ij} : Unitary development cost for alternative j of module i

t_{ij} : Estimated development time for alternative j of module i

τ_{ij} : Average time required to perform a test case for alternative j of module i

π_{ij} : Probability that a single execution of software fails on a test case chosen from a certain input distribution

$$y_t : \begin{cases} 0, & \text{if } t^{th} \text{ constraint is active} \\ 1, & \text{if } t^{th} \text{ constraint is inactive} \end{cases}$$

$$y_{ij} : \begin{cases} 1 & \text{if the } j^{th} \text{ alternative of } i^{th} \text{ module is} \\ & \text{in-house develoep.} \\ 0 & \text{otherwise} \end{cases}$$

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

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

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

3. Optimization Models

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

The following assumptions are common for 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 $(\le n)$. 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 recovery block scheme discussed in [12,13].

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 and testability of a module are available.

7) 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.

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

9) Other than available cost-reliability versions of an alternative, we assume the existence of virtual versions, which has a negligible reliability of 0.001, zero cost and zero delivery time. These components are denoted by index one in the third subscript of x_{ijk}, C_{ijk} and r_{ijk}. for example r_{ij1} denotes the reliability of first version of alternatives j for module i.

3.1. 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 COTS alternatives has different versions.

3.1.1. 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.* $y_{ij}=0$) and vice versa.

$$y_{ij}+\sum_{k=1}^{V_{ij}} x_{ijk}=1;\quad i=1,2,\cdots,n \quad\text{and}\quad j=1,2,\cdots,m_i$$

3.1.2. Redundancy Constraint

The equation stated below guarantees that redundancy is allowed for the components.

$$y_{ij}+\sum_{k=2}^{V_{ij}} x_{ijk}=z_{ij};\quad i=1,2,\cdots,n \quad\text{and}\quad j=1,2,\cdots,m_i$$

$$x_{ij1}+z_{ij}=1;\quad i=1,2,\cdots,n \quad\text{and}\quad j=1,2,\cdots,m_i$$

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

3.1.3. Probability of Failure Free In-House

Developed Components

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}). Reference [14] define 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 [15].

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.

$$N_{ij}^{suc}=\left(1-\pi_{ij}\right)N_{ij}^{tot};\quad i=1,2,\cdots,n \quad\text{and}\quad j=1,2,\cdots,m_i$$

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 ρ_{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 the following.

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

The following equalities come straightforwardly:

$$\bullet P\left(A/B\right)=1$$

$$\bullet P\left(B\right)=1-\pi_{ij}$$

$$\bullet P\left(A/\bar{B}\right)=\left(1-\pi_{ij}\right)^{N_{ij}^{suc}}$$

$$\bullet P\left(\bar{B}\right)=\pi_{ij}$$

therefore, we have

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

3.1.4. Reliability Equation of Both In-House and COTS Components

The reliability (s_{ij}) of j^{th} alternative of i^{th} module of the software.

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

where

$$r_{ij}=\sum_{k=1}^{V_{ij}} r_{ijk}x_{ijk};\quad i=1,2,\cdots,n \quad\text{and}\quad j=1,2,\cdots,m_i$$

3.1.5. Delivery Time Constraint

The maximum threshold T has been given on the delivery time of the whole system. In case of a COTS components the delivery time is simply given by d_{ijk}, whereas for an in- house developed alternative the delivery time shall be expressed as $\left(t_{ij}+\tau_{ij}N_{ij}^{tot}\right)$.

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\left(y_{ij}\left(t_{ij}+\tau_{ij}N_{ij}^{tot}\right)+\sum_{k=1}^{V_{ij}} d_{ijk}x_{ijk}\right)\le T$$

3.2. Objective Function

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

$$\text{Maximize } R(X) = \sum_{l=1}^{L} f_l \prod_{i \in s_l} R_i$$

where R_i is the reliability of module i of the system under Recovery Block stated as follows.

$$R_i = \sum_{j=1}^{m_i} z_{ij} \left[\prod_{k=1}^{j-1} P(X_{ik})^{z_{ij}} \right] P(Y_{ij})^{z_{ij}}; \quad i = 1,2,\cdots,n$$

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

$$P(Y_{ij}) = s_{ij}(1-t_2)$$

3.2.2. 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 $c_{ij}\left(t_{ij} + \tau_{ij} N_{ij}^{tot}\right)$

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

3.3. Optimization Model I

In the optimization model it is assumed that the alternatives of a module are in a Recovery Block. In a Recovery block, more than one alternative of a program exist. For COTS based software, multiple alternatives of a module can be purchased from different vendors. Each module works under a recovery block. Upon invocation of a module the first alternative is executed and the result is submitted for acceptance test. If it is rejected, the second alternative is executed with the original inputs. The same process continues through attached alternative until a result is accepted or the whole recovery block (module) fails. Fault tolerance in a recovery block is achieved by increasing the number of redundancies.

***Problem* (*P*1)**

$$\text{Maximize } R(X) = \sum_{l=1}^{L} f_l \prod_{i \in s_l} R_i \tag{1}$$

$$\text{Minimize } C(X) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(c_{ij}\left(t_{ij} + \tau_{ij} N_{ij}^{tot}\right) y_{ij} + \sum_{k=1}^{V_{ij}} C_{ijk} x_{ijk} \right) \tag{2}$$

subject to

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

$$R_i = \sum_{j=1}^{m_i} z_{ij} \left[\prod_{k=1}^{j-1} P(X_{ik})^{z_{ij}} \right] P(Y_{ij})^{z_{ij}}; \quad i = 1,2,\cdots,n \tag{3}$$

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

$$P(Y_{ij}) = s_{ij}(1-t_2)$$

$$N_{ij}^{suc} = (1-\pi_{ij}) N_{ij}^{tot}; \quad i = 1,2,\cdots,n \text{ and } j = 1,2,\cdots,m_i \tag{4}$$

$$\rho_{ij} = \frac{1-\pi_{ij}}{(1-\pi_{ij}) + \pi_{ij}(1-\pi_{ij})^{N_{ij}^{suc}}}; \quad i = 1,2,\cdots,n \text{ and } j = 1,2,\cdots,m_i \tag{5}$$

$$s_{ij} = \rho_{ij} y_{ij} + r_{ij}; \quad i = 1,2,\cdots,n \text{ and } j = 1,2,\cdots,m_i \tag{6}$$

$$y_{ij} + \sum_{k=1}^{V_{ij}} x_{ijk} = 1; \quad i = 1,2,\cdots,n \text{ and } j = 1,2,\cdots,m_i \tag{7}$$

$$y_{ij} + \sum_{k=2}^{V_{ij}} x_{ijk} = z_{ij}; \quad i = 1,2,\cdots,n \text{ and } j = 1,2,\cdots,m_i \tag{8}$$

$$x_{ij1} + z_{ij} = 1; \quad i = 1,2,\cdots,n \text{ and } j = 1,2,\cdots,m_i \tag{9}$$

$$\sum_{j=1}^{m_i} z_{ij} \ge 1; \quad i = 1,2,\cdots,n \tag{10}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(y_{ij}\left(t_{ij} + \tau_{ij} N_{ij}^{tot}\right) + \sum_{k=1}^{V_{ij}} d_{ijk} x_{ijk} \right) \le T \tag{11}$$

$$\}$$

where X is a vector of component x_{ijk} and y_{ij}; $i = 1,2,\cdots,n$; $j = 1,2,\cdots,m_i$; $k = 1,2,\cdots,V_{ij}$.

3.4. Optimization Model II

As explained in the introduction, it is observed that some alternatives of a module may not be compatible with alternatives of another module [16]. The next optimization model II addresses this problem. It is done, incorporating additional constraints in the optimization models. This constraint can be represented as $x_{gsq} \le x_{hu_tc}$, which means that if alternative s for module g is chosen, then alternative u_t, $t = 1,\cdots,z$ have to be chosen for module h. We also assume that if two alternatives are compatible, then their versions are also compatible.

$$x_{gsq} - x_{hu_tc} \le My_t$$

$$q = 2,\cdots,V_{gs}, \quad c = 2,\cdots,V_{hu_t}, \quad s = 1,\cdots,m_g \tag{12}$$

$$\sum y_t = z\left(V_{hu_t} - 2\right) \tag{13}$$

Constraint (12) and (13) make use of binary variable y_t to choose one pair of alternatives from among different alternative pairs of modules.

Finally, model can be re-written as

***Problem* (*P*2)**

$$\text{Maximize}\, R(X) = \sum_{l=1}^{L} f_l \prod_{i \in s_l} R_i$$

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

subject to

$$X \in S$$

$$x_{gsq} - x_{hu_t c} \le My_t$$

$$q = 2,\cdots,V_{gs},\ \ c = 2,\cdots,V_{hu_t},\ \ s = 1,\cdots,m_g,$$

$$\sum y_t = z\left(V_{hu_t} - 2\right)$$

If more than one alternative compatible component is to be chosen for redundancy, constraint (13) can be relaxed as follows.

$$\sum y_t \le z\left(V_{hu_t} - 2\right)$$

4. Fuzzy Multi-Objective Optimization Model for Software Products

Principle to multi-objective optimization is the concept of an "efficient solution", where any improvement of one objective can only be achieved at the expense of another. The fuzzy approach can be used as an effective tool for quickly obtaining a good compromise solution. Conventional optimization methods assume that all parameters and goals of an optimization model are precisely known. However, for many practical problems there are incompleteness and unreliability of input information. This has resulted in use of fuzzy multi-objective optimization method with fuzzy parameters. For instance, a designer is required to minimize the system cost while simultaneously maximizing the system reliability. Therefore multiple objective functions become an important aspect in the reliability design of the engineering systems.

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

Therefore, we formulate fuzzy multi-objective optimization model for software products selection based on vague aspiration levels, the decision maker may decide his aspiration levels on the basis of past experience and knowledge possessed by him. To express vague aspiration levels of the decision, various membership functions have been proposed [6,7]. A fuzzy mathematical programming problem with non linear membership function results in a non linear programming problem. Usually, a linear membership function is employed to avoid nonlinearrity. Further, if membership function is interpreted as the fuzzy utility of the decision maker, which describes the behavior of indifference, preference or aversion towards uncertainty, a non linear membership function is a better representation than a linear membership function.

4.1. Problem Formulation

Fuzzy multi-objective optimization model for software products selection based on maximizing the fuzzifier reliability function and minimizing the fuzzifier cost function subject to crisp constraints are stated as follows.

***Problem* (*P*3)**

$$\text{Maximize}\, \tilde{R}(X) = \sum_{l=1}^{L} f_l \prod_{i \in s_l} R_i$$

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

subject to $X \in S$

Here, we have defined the two objective functions, the reliability and cost that are considered to be vague and uncertain (*i.e.* fuzzy in nature) and the constraints are of crisp nature. Cut-throat competition in the existing market, system complexity, and intended flexibility makes it difficult for the management to precisely define their goals and constraints. Moreover a slight shift on bounds can provide a more efficient solution. Hence, we have used fuzzy optimization technique (fuzzy mathematical programming) to solve the fuzzy multi-objective optimization problem.

4.2. Problem Solution

The following steps are used to solve the fuzzy mathematical programming problem.

Step 1. Compute the crisp equivalent of the fuzzy parameters using a defuzzification function. Same defuzzification function is to be used for each of the parameters.

Here we use the defuzzification function of the type

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

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

Step 2. Incorporate the objective function of the fuzzifier min (max) as a fuzzy constraint with a restriction (aspiration) level. The above problem (*P*3) can be rewritten as

***Problem* (*P*4)**

Find X subject to $R(X) = \sum_{l=1}^{L} f_l \prod_{i \in s_l} R_i \gtrsim R_0$

$$C(X) = \sum_{i=1}^{n}\sum_{j=1}^{m_i}\left(c_{ij}\left(t_{ij} + \tau_{ij} N_{ij}^{tot}\right) y_{ij} + \sum_{k=1}^{V_{ij}} C_{ijk} x_{ijk} \right) \lesssim C_0$$

$$X \in S$$

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

$$\mu_R(X) = \begin{cases} 1; & R(X) \ge R_0 \\ \dfrac{R(X) - R_0^*}{R_0 - R_0^*}; & R_0^* \le R(X) < R_0 \\ 0; & R(X) < R_0^* \end{cases}$$

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

$$\mu_C(X) = \begin{cases} 1; & C(X) \le C_0 \\ \dfrac{C_0^* - C(X)}{C_0^* - C_0}; & C_0 \le C(X) < C_0^* \\ 0; & C(X) > C_0^* \end{cases}$$

where C_0 is the restriction level and C_0^* is the tolerance level to the fuzzy budget constraint.

Step 4. Extension principle is used to identify the fuzzy decision, which results in a crisp mathematical programming problem given by

***Problem* (*P*5)**

Maximize α subject to

$$\mu_R(X) \ge \alpha, \ \mu_C(X) \ge \alpha, \ X \in S$$

can be solved by the standard crisp mathematical programming algorithms.

Step 5. While solving the problem, 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 weights according to their relative importance. The resulting problem can be solved by the weighted minmax approach. The crisp formulation of the weighted problem is given as

***Problem* (*P*6)**

Maximize α subject to

$$\mu_R(X) \ge w_1\alpha, \ \mu_C(X) \ge w_2\alpha, \ X \in S$$

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

where, α represents the degree up to which the aspiration of the decision maker is met. If the constraints are fuzzy as well as crisp, then in the equivalent crisp mathematical programming problem, there will be no change in the original crisp constraints since their tolerances are zero except for those constraints which are fuzzy in nature. The problem (*P*6) can be solved using standard mathematical programming approach.

Step 6. On substituting the values for $\mu_R(X)$ and $\mu_C(X)$ the problem becomes

***Problem* (*P*7)**

Maximize α subject to

$$R(X) \ge R_0 - (1 - w_1\alpha)\left(R_0 - R_0^*\right)$$

$$C(X) \le C_0 + (1 - w_2\alpha)\left(C_0^* - C_0\right)$$

$$X \in S \quad \alpha \in [0,1]$$

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

Step 7. If a feasible solution is not obtainable for the problem (*P*7) then we can use fuzzy goal programming approach to obtain a compromised solution [9]. The method is discussed in detail in the numerical illustration.

5. Illustrative Examples

Consider a software system having two modules with more than one alternative for each module. The data sets for COTS and in-house developed components are given in **Table 1** and **Table 2**, respectively.

Let $L = 3$, $s_1 = \{1,2,3\}$, $s_2 = \{1,3\}$, $s_3 = \{2\}$, $f_1 = 0.5$, $f_2 = 0.3$, $f_3 = 0.2$. It is also assumed that $t_1 = 0.01$, $t_2 = 0.05$ and $t_3 = 0.01$.

The assignment of weights is based on the expert's judgment for the reliability and the cost criteria. Weights assigned for reliability and cost are 0.7 and 0.3 respectively.

5.1. Minimum and Maximum Level of Reliability and Cost

Firstly, the triangular fuzzy reliability and cost values are computed using fuzzied values of these parameters and

Table 1. Data set for COTS components.

Modules	Alternatives	Version 1		
		Cost	Reliability	Delivery Time
	1	0	0.001	0
1	2	0	0.001	0
	3	0	0.001	0
	1	0	0.001	0
2	2	0	0.001	0
	3	0	0.001	0
	1	0	0.001	0
3	2	0	0.001	0
	3	0	0.001	0
Modules	**Alternatives**	**Version 2**		
		Cost	**Reliability**	**Delivery Time**
	1	14	0.90	3
1	2	12.5	0.86	4
	3	17	0.90	2
	1	13	0.87	4
2	2	11	0.91	5
	3	18	0.89	2
	1	13	0.86	4
3	2	16	0.85	3
	3	16	0.89	3
Modules	**Alternatives**	**Version 3**		
		Cost	**Reliability**	**Delivery Time**
	1	11	0.88	4
1	2	18	0.92	2
	3	15	0.88	3
	1	17.5	0.86	2
2	2	12	0.89	4
	3	15	0.86	3
	1	14	0.88	3
3	2	18	0.90	2
	3	17	0.87	2

Table 2. Data set for in-house components.

Module i	Alternatives	t_{ij}	τ_{ij}	c_{ij}	π_{ij}
	1	8	0.005	5	0.002
1	2	6	0.005	4	0.002
	3	7	0.005	4	0.002
	1	9	0.005	5	0.002
2	2	5	0.005	2	0.002
	3	6	0.005	4	0.002
	4	5	0.005	3	0.002
3	1	6	0.005	4	0.002
	2	5	0.005	3	0.002

then defuzzied using Heilpern' s defuzzier. If the available reliability and cost are specified as TFN (triangular fuzzy number) given as follows

$$\tilde{R}_0 = \{0.93, 0.95, 0.97\};\ \tilde{C}_0 = \{70, 75, 80\}.$$

The aspiration level of reliability is $R_0 = 0.95$ and the restriction on cost is $C_0 = 75$. The tolerance level for reliability and cost is $R_0^* = 0.85$ and $C_0^* = 85$.

5.2. Fuzzy Goal Programming Approach

On solving the problem, we found that the problem (*P*7) is not feasible; hence the management goal cannot be achieved for a feasible value of $\alpha \in [0,1]$. Now we use fuzzy goal programming technique to obtain a compromised solution. The approach is based on the goal programming technique for solving crisp goal programming problem (Mohamed, 1997). The maximum value of any membership function can be 1; maximization of $\alpha \in [0,1]$ is equivalent to making it as close to 1 as best as possible. This can be achieved by minimizing the negative deviational variables of goal programming (*i.e.* η) from 1. The fuzzy goal programming formulation for the given problem (*P*7) introducing the negative and positive deviational variables η_j and ρ_j is given as

Minimize *u* subject to

$$\mu_R(X) + \eta_1 - \rho_1 = 1 \quad \mu_C(X) + \eta_2 - \rho_2 = 1$$

$$u \ge w_j \times \eta_j;\ \eta_j \times \rho_j = 0;\ \eta_j, \rho_j \ge 0$$

$$j = 1,2 \quad X \in S;\ \alpha \in [0,1];$$

$$w_1, w_2 \ge 0;\ w_1 + w_2 = 1;\ \alpha = 1 - u$$

5.3. Optimization Model I

Table 3 presents the solution for optimization model I. The problem is solved using software package LINGO [17]. The solution to the model gives the optimal component selection for the software system along with the corresponding cost and reliability of the overall system. The sensitivity analysis to the delivery time constraint has been performed. It is clearly seen from the table that in case 1, when the delivery time was 15 units then one in-house and other COTS components were selected while in all other cases when the delivery time increases along with in-house components there will be a corresponding change in reliability and cost. In case 2, when the delivery time was 18 units, our system reliability and cost also increases and in case 3 as compared to case 2, when delivery time was 20 units, system reliability increases and cost decreases. Therefore, if the customer is ready to wait then case 3 is an optimal solution. Redundancy is also present in all the three cases.

5.4. Optimization Model II

To illustrate optimization model for compatibility, we use previous results.

Case 1. *Delivery time is assumed to be* 15 *units.*

We assume that second alternative of first module is compatible with second and third alternative of second module. Following solution was obtained using LINGO.

Table 3. Solution for optimization model I.

Case No.	Delivery Time	In-House	COTS	System Reliability	Overall System Cost	α Value
1	15	$y_{32} = 1$	$x_{111} = x_{123} = x_{132} = 1$ $x_{211} = x_{223} = x_{232} = x_{241} = 1$ $x_{311} = 1$	0.92	75	0.84
2	18	$y_{24} = y_{32} = 1$	$x_{111} = x_{123} = x_{132} = 1$ $x_{211} = x_{223} = x_{232} = 1$ $x_{311} = 1$	0.93	77.5	0.86
3	20	$y_{24} = y_{32} = 1$	$x_{111} = x_{123} = x_{132} = 1$ $x_{211} = x_{223} = x_{232} = 1$ $x_{311} = 1$	0.94	76	0.90

$$y_{32} = 1 \quad x_{111} = x_{123} = x_{132} = 1$$
$$x_{211} = x_{223} = x_{233} = x_{241} = 1 \quad x_{311} = 1$$

It is observed that due to the compatibility condition, third alternative of second module is chosen as it is compatible with second alternative of first module. The system reliability for the above solution is 0.86 and cost is 85 units. The achievement level of the membership function is $\alpha = 0.40$.

Case 2. *Delivery time is assumed to be* 20 *units.*

We assume that second alternative of second module is compatible with second and third alternative of first module. Following solution was obtained using LINGO.

$$y_{24} = y_{32} = 1 \quad x_{111} = x_{123} = x_{133} = 1$$
$$x_{211} = x_{222} = x_{231} = 1 \quad x_{311} = 1$$

It is observed that due to the compatibility condition, third alternative of first module is chosen as it is compatible with second alternative of second module. The system reliability for the above solution is 0.93 and cost is 77 units. The achievement level of the membership function is $\alpha = 0.90$.

6. Conclusions

We have presented optimization models that supports the decision whether to buy software components for software architecture or to build them in-house. We have formulated bi-criteria optimization models based on decision variables indicating the set of structural components "to buy" and "to build" in order to minimize the software cost with simultaneous maximization of system reliability. The problem is formulated for Recovery Block fault-tolerant software system. It may be appreciated that when different alternatives of the same module are available with variations in the attributes of reliability and cost, then it involves multi-objective decision making environment that befits more of fuzzy approximation than deterministic formulation. Therefore, we have drawn on fuzzy methodology for the estimation of reliability and cost. This developed approach can effectively deal with the vagueness and subjectivity of expert's information. Fuzzy predictions of the triangular fuzzy statistical data have been defuzzified using Heipern's defuzzifier and a crisp multi-objective model has been developed using the defuzzified values. The component selection problem is formulated as a multi-objective programming problem and fuzzy goal programming technique is used to provide a feasible solution.

7. Acknowledgements

One of the authors, Ritu Arora, gratefully acknowledges the Director and Management of Maharaja Agrasen Institute of Technology for their permission to publish this research work.

8. References

[1] F. Belli and P. Jadrzejowicz, "An Approach to Reliability Optimization Software with Redundancy," *IEEE Transaction of Software Engineering*, Vol. 17, No. 3, 1991, pp. 310-312.

[2] L. M. Cysneiros and J. C. S. Leite, "Nonfunctional ReQuirements: From Elicitation to Conceptual Models," *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, 2004, pp. 328-350. doi:10.1109/TSE.2004.10

[3] P. K. Kapur, A. K. Bardhan and P. C. Jha, "Optimal Reliability Allocation Problem for a Modular Software System," *OPSEARCH*, Vol. 40, No. 2, 2003, pp. 138-148.

[4] R. E. Bellman and L. A. Zadeh, "Decision-Making in a Fuzzy Environment," *Management Science*, Vol. 17, No. 4, 1970, pp. B141-B164. doi:10.1287/mnsc.17.4.B141

[5] H. Tanaka, T. Okuda and K. Asai, "On Fuzzy Mathematical Programming," *Journal of Cybernetics*, Vol. 3, No. 4, 1974, pp. 37-46. doi:10.1080/01969727308545912

[6] H. J. Zimmermann, "Description and Optimization of Fuzzy Systems," *International Journal of General Systems*, Vol. 2, No. 4, 1976, pp. 209-215. doi:10.1080/03081077608547470

[7] H. J. Zimmermann, "Fuzzy Set Theory and Its Applica-

tions," Kluwer Academic Publishers, Boston, 1985.

[8] H. J. Zimmermann, "Fuzzy Set Theory and Its Applications," Academic Publisher, Dordrecht, 1991.

[9] R. H. Mohamed, "The Relationship between Goal Programming and Fuzzy Programming," *Fuzzy Sets and Systems*, Vol. 89, No. 2, 1997. pp. 215-222. doi:10.1016/S0165-0114(96)00100-5

[10] P. Gupta, M. K. Mehlawat, G. Mittal and S. Verma, "A Hybrid Approach for Selecting Optimal COTS Products," *Computational Science and Its Application-ICCSA*, Springer Publication, Berlin, 2009, pp. 949-962.

[11] B. P. Gladish, I. Gonzalez; A. B. Terol and M. A. Parra, "Planning a TV Advertising Campaign: A Crisp Multi objective Programming Model from Fuzzy Basic Data," *Omega*, Vol. 38, No. 1-2, 2010, pp. 84-94. doi:10.1016/j.omega.2009.04.004

[12] O. Berman and U. D. Kumar, "Optimization Models for Reliability of Modular Software System," *IEEE Transaction of Software Engineering*, Vol. 19, No. 11, 1993, pp. 1119-1123.

[13] U. D. Kumar, "Reliability Analysis of Fault Tolerant Recovery Block," *OPSEARCH*, Vol. 35, No. 4, 1998, pp. 281-294.

[14] V. Cortellessa, F. Marinelli and P. Potena, "An Optimization Framework for 'Build-Or-Buy' Decisions in Software Architecture," *Computers and Operations Research*, Vol. 35, No. 10, 2008, pp. 3090-3106. doi:10.1016/j.cor.2007.01.011

[15] A. Bertolino and L. Strigini, "On the Use of Testability Measures for Dependability Assessment," *IEEE Transactions on Software Engineering*, Vol. 22, No. 2, 1996, pp. 97-108. doi:10.1109/32.485220

[16] H. W. Jung and B. Choi, "Optimization Models for Quality and Cost of Modular Software System," *European Journal of Operations Research*, Vol. 112, No. 3, 1999, pp. 613-619. doi:10.1016/S0377-2217(98)00169-6

[17] H. Thiriez "OR Software LINGO," *European Journal of Operational Research*, Vol. 12, 2000, pp. 655-656.