International Journal of Computer and Communication Technology

Volume 5 | Issue 1 Article 12

January 2014

Goal Programming Approach for Selection of COTS Components in Designing a Fault Tolerant Modular Software System under Consensus Recovery Block Scheme

P. C. Jha
Department of Operational Research, University of Delhi, India, jhapc@yahoo.com

Vikram Bali
Galgotias College of Engineering and Technology , Greater Noida, U.P., India, vikramgcet@gmail.com

Recommended Citation
Jha, P. C. and Bali, Vikram (2014) "Goal Programming Approach for Selection of COTS Components in Designing a Fault Tolerant Modular Software System under Consensus Recovery Block Scheme," *International Journal of Computer and Communication Technology*: Vol. 5 : Iss. 1 , Article 12.
DOI: 10.47893/IJCCT.2014.1222
Available at: https://www.interscience.in/ijcct/vol5/iss1/12

Goal Programming Approach for Selection of COTS Components in Designing a Fault Tolerant Modular Software System under Consensus Recovery Block Scheme

P.C. Jha & Vikram Bali

Department of Operational Research, University of Delhi, India
Galgotias College of Engineering and Technology , Greater Noida, U.P., India
E-mail: jhapc@yahoo.com, vikramgcet@gmail.com

Abstract **-** The application of computer systems has now crossed many different fields. Systems are becoming more software intensive. The requirements of the customer for a more reliable software led to the fact that software reliability is now an important research area. One method to improve software reliability is by the application of redundancy. A careful use of redundancy may allow the system to tolerate faults generated during software design and coding thus improving software reliability. The fault tolerant software systems are usually developed by integrating COTS (commercial off-the-shelf) software components. This paper is designed to select optimal components for a fault tolerant modular software system so as to maximize the overall reliability of the system with simultaneously minimizing the overall cost. A chance constrained goal programming model has been designed after considering the parameters corresponding to reliability and cost of the components as random variable. The random variable in this case has been considered as value which has known mean and standard deviation. A chance constraint goal programming technique is used to solve the model. The issue of compatibility among different commercial off-the shelf alternatives is also considered in the paper. Numerical illustrations are provided to demonstrate the model.

Keywords- *Software Reliability, COTS Components, Fault Tolerance, Optimization, Chance Constraints, Goal Programming.*

I. INTRODUCTION

Today, almost everyone in the world is directly or indirectly affected by computer systems. Computers and computer systems have become a significant part of our modern society. As software systems have become more and more complex to design and develop, intensive studies are carried out to increase the chance that software systems will perform satisfactorily in operations. The application of computer systems has now crossed many different fields. Systems are becoming more software intensive. Financial systems include teller, automated teller and loan processing are software intensive. Everything from insurance rates to credit histories to hotel reservations to long distance telephone calls is performed by software. The requirements of the customer for a more reliable software led to the fact that software reliability is now an important research area. Reliability for a software system is defined as the probability that software operates without failure in a specified environment, during a specified exposure period [1]. One method to improve software reliability is by the application of redundancy. A careful use of redundancy may allow the system to tolerate faults generated during software design and coding thus improving software reliability. Fault tolerant techniques enable the system to tolerate software faults remaining in the system after its development. When fault occurs, one of the redundant software modules get executed and prevent system failure. The fault tolerant software systems are usually developed by integrating COTS (commercial off-the-shelf) software components. A COTS based software system is a system that has been built primarily by assembling a set of COTS software. These components that can be procedural or object libraries, stand-alone applications etc., are bought and integrated to form a complete system [2]. Respective developers of the components provide information about their quality normally in terms of reliability. Developers of the commercial product integrate new technologies and new standard into the product faster than an organization built software. In spite of many pros, these products

have cons too like developers may or may not change their source code, unavailability of complete and correct specifications, sometime the set of COTS components may be mismatched, etc. Since COTS products have many disadvantages but their use is increasing day by day due to their economic benefits [2]. Consequently, the component-based software development approach has great potential for reducing development time and cost.

Several optimization models have been proposed in the literature for the optimal selection of the COTS software components for the development of safe and reliable software systems. At the outset Scott and Gault [3] in 1987 examined three methods of creating fault tolerant software systems, Recovery Block Scheme, N-Version Programming, and Consensus Recovery Block. The authors presented reliability models for each technique. The models are used to show that one method, the Consensus Recovery Block, is more reliable than the other two. McAllister and Scott in [4] 1991 compared cost of a single version system with the three versions of fault tolerant software systems. The authors shown that in cases where failures are independent, Consensus Recovery Block followed by Recovery Block are the most cost justifiable fault tolerant techniques to be considered. Unless the voter is perfect, N-Version Programming does not compete with the other two methods. Ashrafi and Berman in [5] 1992 proposed two optimization models that address the tradeoff between reliability and cost. They applied to large software packages that consist of several programs. The authors used optimization models to determine the redundancy level of a software package consisting of several independent functions where each function is performed by program with known reliability and cost. Ashrafi and Berman in [6] 1993, however, breaks down this approach one step further and deals with software systems consisting of one or more programs where each program consists of series of modules, which upon sequential execution will perform a function. The optimal redundancy level of the modules is to be determined. Berman and Dinesh in [7] in 1999 presented optimization models for a fault tolerant software by selecting a set of versions for a given program. The objective is to maximize the reliability of the software satisfying the budget limitation. Berman and Dinesh in [8], 1999 developed reliability prediction techniques and optimization models for important fault tolerant software such as nested Recovery Block, Modified Recovery Block and Nested Consensus Recovery Block. The methods have significant use in selection of programs in COTS environment. Kapur et. al [9] in 2003 have chosen the recovery block reliability model for COTS based software system. Two optimization models, for optimal selection of components have been proposed. The objective functions in both the models perform the weighted maximization of system reliability, weights being decided with respect to access frequency of each module. Pankaj et al [10] in 2009 formulated fuzzy multi objective optimization models for selecting the optimal COTS software products in the development of software system based on COTS. Jha *et al* [11] formulated a fuzzy multi-objective optimization model for optimal selection of COTS components for a fault tolerant modular software system under consensus recovery block scheme.

Large software system has modular structure to perform set of functions with different modules having different alternatives for each module and different versions for each alternative. A schematic representation of the software system is given in figure 1 given in section IV. On the execution of a software system, the functions are invocated. The frequency with which the functions are used is not the same for all of them and not all the modules are called during the execution of the function, the software has in its menu. In this paper the problem of selecting optimum number of COTS components has been considered. The cost and reliability parameter have been given by the vendors and their values differ from vender to vendor. The problem has been formulated as a chance constraints goal programming problem after considering the reliability and cost parameters as the random variables. It has been assumed that the random variables corresponding to the reliability and cost are quantities with known mean and standard deviations. The parameter corresponding to reliability and cost can be obtained by finding the ideal solutions and the trend of which the parameter varies. Mathematical formulations for both the reliability and cost objectives have been discussed in section III-A. Chance constraint goal programming approach for multi-objective goal programming problem has been given in section III-B. To illustrate the solution methodology numerical example has been mentioned in section IV. Concluding remarks are made in section V.

II. NOTATIONS

R : System quality measure

C : Total budget available for all modules.

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

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

c_{ijk} : Cost of version k of alternative j of module i (COTS)

t_1 : Probability that next alternative is not invoked upon failure of the current alternative

t_2 : Probability that the correct result is judged wrong.

t_3 : Probability that an incorrect result is accepted as correct.

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

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

r_{ijk} : Reliability of version k of alternative j of module i (COTS)

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

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

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

III. MODEL FORMULATION

Before In this section, we formulate COTS software product selection problem as an optimization problem with multiple objectives. The optimization model holds good for the following situations.

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 components) are attached in the modules and work similar to the recovery block scheme discussed in [7, 8].
5. The cost of an alternative is the buying price for the COTS product. Reliability for all the components are known and no separate testing is done.
6. Different versions with respect to cost and reliability of a module are available.
7. Other than available cost-reliability versions of an alternative, we assume the existence of a virtual versions, which has a negligible reliability of 0.001 and zero cost. These components are denoted by index one in the third subscript of x_{ijk}, c_{ijk} and r_{ijk}. for example r_{ij1} denotes the reliability of first version of alternatives j for module i, having the above property.

A. Multi-objective Optimization Models

In the optimization model it is assumed that the alternatives of a module are in a Consensus Recovery Block. In Consensus Recovery Block, achieving fault tolerance is to run all the attached independent alternatives simultaneously and selecting the output by the voting mechanism. It requires independent development of independent alternatives of a program, which the COTS components satisfy and a voting procedure. Upon invocation of the consensus recovery block all alternatives are executed and their output is submitted by a voting procedure. Since it is assumed that there is no common fault, if two or more alternatives agree on one output then that alternative is designated as correct. Otherwise the next stage is entered. At this stage the best version is examined by the acceptance test. If the output is accepted, it is treated as the correct one. However, if the output is not accepted, the next best version is subjected to testing. This process continues until an acceptable output is found or all outputs are exhausted.

- ***Optimization Model-I***

Optimization Model-I aims at maximization of system reliability with the budget as one of the constraints.

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

Subject to

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\sum_{k=1}^{V_{ij}} C_{ijk}\ x_{ijk}\ \le C \qquad (2)$$

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

$$R_i = 1 + \left[\sum_{j=1}^{m_i} \frac{1}{(1-r_{ij})^{z_{ij}}} \left[\prod_{k=1}^{m_i}(1-r_{ik})^{z_{ik}}\right]\left[1-(1-r_{ij})^{z_{ij}}\right] + \prod_{j=1}^{m_i}(1-r_{ij})^{z_{ij}}\right] \qquad (3)$$

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

$$P(X_{ij}) = (1-t_1)\left[(1-r_{ij})(1-t_3) + r_{ij}t_2\right] \qquad (4)$$

$$P(Y_{ij}) = r_{ij}(1-t_2) \qquad (5)$$

$$r_{ij} = \sum_{k=1}^{V_{ij}} x_{ijk} r_{ijk} \quad \text{for } j = 1,.....,mi \ \&\ i = 1,....,n \qquad (6)$$

$$\sum_{k=1}^{V_{ij}} x_{ijk} = 1 \quad \text{for } j = 1,....,m_i \ \&\ i = 1,.....,n \qquad (7)$$

$$x_{ij1} + z_{ij} = 1;\ j = 1,2,....,m_i \qquad (8)$$

$$\sum_{j=1}^{m_i} z_{ij} \ge 1;\ i = 1,2,....,N\ \} \qquad (9)$$

Objective function (1) 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 and Constraint (2) is a budget constraint. Constraint (3) estimates the reliability of module i. As it has been assumed that the exception raising and control transfer programs work perfectly, a module fails if all attached alternatives fail.

Constraint (4) is the probability of event that output of alternative j of module i is rejected and Constraint (5) is the probability of event that correct result of alternative j of module i is accepted. Constraint (6) gives the reliability of alternative j of module i. Constraint (7) ensures that exactly one version is chosen from each alternative of a module. It includes the possibility of choosing a dummy version. Equation (8) and (9) guarantee that not all chosen alternatives of modules are dummies. Optimization model I is a 0-1 Bi-Criterion integer programming problem. An example is solved using software package LINGO.

- ***Optimization Model-II***

Optimization model II aims at minimization of system cost with reliability as one of the constraints.

Minimize $\sum_{i=1}^{n}\sum_{j=1}^{m_i}\sum_{k=1}^{V_{ij}} C_{ijk}\ x_{ijk}\ \le C$

Subject to

$$\sum_{l=1}^{L} f_l \prod_{i \in s_l} R_i\ <= \mathrm{R}$$

$$\mathrm{X} \in \mathrm{S}$$

Thus the multi-objective deterministic problem, by combining reliability and cost objectives can be written as follows:

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

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

Subject to

$$\mathrm{X} \in \mathrm{S}$$

B. Chance Constraint Goal Programming Approach

The goal programming (GP) model is one of the well-known multi-objective mathematical programming models. This model allows taking into account simultaneously several objectives in a problem for choosing the most satisfactory solution within a set of feasible solutions. More precisely, the GP designed to find a solution that minimizes the deviations between the achievement level of the objectives and the goals set for them. In the case where the goal is surpassed, the deviation will be positive and in the case of the underachievement of the goal, the deviation will be negative.

The general form of the GP model is

$$\textit{Minimize} \quad \bar{a} = \{g_1(\bar{n},\bar{p}), g_2(\bar{n},\bar{p}), ..., g_k(\bar{n},\bar{p})\}$$
$$\textit{Such that} \quad f_i(\bar{x}) + n_i - p_i = b_i, \qquad \bar{x},\bar{n},\bar{p} \ge 0,$$

where $g_k(\bar{n},\bar{p})$ is a linear function of the deviational variables. The dimension of $\bar{a}$ represents the number k of the preemptive priority levels. b_i represents the level of aspiration associated with the objective $f_i(\bar{x})$. The variables n_i and p_i indicate the negative and positive deviations respectively of the achievement level $f_i(\bar{x})$

from aspiration level. The goal or aspiration levels assigned to the various objectives can be probabilistic where the decision maker does not know its value with complete certainty. The first formulation of the stochastic GP model goes back to the late 1969s with Contini's works [12]. He considers the goal as uncertain variables with a normal distribution. Stancu-Minasian [13] and Stancu-Minasian and Giurgiutiu [14] present a synthesis of methodologies used in multiple objective programming in a stochastic contest. Several other techniques have been proposed to solve the SGP model. The most popular technique is a chance constrained programming developed by Charnes and Cooper [15, 16, 17].

- ***Reliability and Cost Objective***

To determine the deterministic equivalent of the reliability and cost objectives the following procedure is adopted. Let R and C are the aspiration level of the first and the second objective functions. Those aspiration levels can be obtained by getting the ideal solutions of the first and second objectives separately. The maximization of the reliability objective and minimization of the cost objective can be written as chance constrained goal programming problem in which the probability that calculated value of reliability will be greater than the ideal solution (estimated target value of reliability) is greater than or equal to α or β (some acceptable probability range) or calculated value of cost is less than the ideal solution (estimated target value of cost).

$$\text{Prob}\left\{\sum_{l=1}^{L} f_l \prod_{i\in s_l} R_i \geq R\right\} \geq \alpha, \qquad (10)$$

$$\text{Prob}\left\{\sum_{i=1}^{n}\sum_{j=1}^{m_i}\sum_{k=1}^{V_{ij}} c_{ijk}x_{ijk} \geq C\right\} \geq \beta, \qquad (11)$$

where R and C are estimated value of the two objectives.

$$\text{Prob}\left\{\frac{\sum_{l=1}^{L} f_l \prod_{i\in s_l} R_i - E(R)}{\sqrt{Var(R)}} \geq \frac{R-E(R)}{\sqrt{Var(R)}}\right\} = \text{Prob}\left\{\frac{R-E(R)}{\sqrt{Var(R)}} \leq \frac{\sum_{l=1}^{L} f_l \prod_{i\in s_l} R_i - E(R)}{\sqrt{Var(R)}}\right\} \geq \alpha$$

Thus if e denotes the value of standard normal variable at which $\phi(e) = \alpha$

$$\phi\left\{\frac{\sum_{l=1}^{L} f_l \prod_{i\in s_l} R_i - E(R)}{\sqrt{Var(R)}} \geq \frac{R-E(R)}{\sqrt{Var(R)}}\right\} \geq \phi(e)$$

$$\Rightarrow \left\{\frac{\sum_{l=1}^{L} f_l \prod_{i\in s_l} R_i - E(R)}{\sqrt{Var(R)}} \geq \frac{R-E(R)}{\sqrt{Var(R)}}\right\} \geq (e)$$

$$\Rightarrow \sum_{l=1}^{L} f_l \prod_{i\in s_l} R_i - E(R) - e\sqrt{Var(R)} \geq 0 \qquad (12)$$

Similarly the chance constraint for cost goal is written as

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\sum_{k=1}^{V_{ij}} c_{ijk}x_{ijk} - E(C) + f\sqrt{Var(C)} \leq 0 \qquad (13)$$

Reformulating equations (12) and (13) as equality through the use of the deviational variables produces the following goal equations:

$$\sum_{l=1}^{L} f_l \prod_{i\in s_l} R_i + n_1 - p_1 = E(R) + e\sqrt{Var(R)} \qquad (14)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\sum_{k=1}^{V_{ij}} c_{ijk}x_{ijk} + n_2 - p_2 = E(C) - f\sqrt{Var(C)} \qquad (15)$$

The goal Programming formulation of the original multi-objective optimization models after combining the goals can be written as follows:

- ***Optimization Model-III***

Minimize $(n1 + p2)$

Subject to

$$\sum_{l=1}^{L} f_l \prod_{i\in s_l} R_i + n_1 - p_1 = E(R) + e\sqrt{Var(R)}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i}\sum_{k=1}^{V_{ij}} c_{ijk}x_{ijk} + n_2 - p_2 = E(C) - f\sqrt{Var(C)}$$

$$X \in S$$

n and p are the negative and positive deviational variables and are greater than equal to zero.

- ***Optimization Model-IV***

Optimization model-IV is an extension of optimization model-III. In optimization model-III, we assumed that all alternative COTS products of one module are compatible with the alternative COTS products for 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. Optimization model-IV addresses this problem. It is done by incorporating additional constraints in the optimization models. This constraint can be represented as $x_{gsq} \le x_{hu_t c}$, which means that if alternative s for module g is chosen, then alternative $u_t, t = 1,.....z$ have to be chosen for module h. We also assume that if two alternatives are compatible, then their versions are also compatible.

$$x_{gsq} - x_{hu_t c} \le My_t,$$
$$q = 2,.....,V_{gs},\ c = 2,.....,V_{hu_t},\ s = 1,.....,m_g \quad (16)$$

$$\sum y_t = z\left(V_{hu_t} - 2\right) \quad (17)$$

Constraints (16) and (17) make use of binary variable y_t to choose one pair of alternatives from among different alternative pairs of modules. If more than one alternative compatible component is to be chosen for redundancy, constraint (17) can be relaxed as follows.

$$\sum y_t \le z\left(V_{hu_t} - 2\right) \quad (18)$$

Optimization model-III can be transformed to another optimization problem using compatibility constraint as follows. Therefore, optimization model-IV can be written as follows:

Minimize $(n1 + p2)$

Subject to

$$\sum_{l=1}^{L} f_l \prod_{i \in s_l} R_i + n_1 - p_1 = E(R) + e\sqrt{Var(R)}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} \sum_{k=1}^{V_{ij}} c_{ijk} x_{ijk} + n_2 - p_2 = E(C) - f\sqrt{Var(C)}$$

$$x_{gsq} - x_{hu_t c} \le My_t,$$
$$q = 2,.....,V_{gs},\ c = 2,.....,V_{hu_t},\ s = 1,.....,m_g$$

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

IV. ILLUSTRATIVE EXAMPLE

Consider a software system (Fig. 1) having five modules with more than one alternative for each module. The data sets for COTS components are given in Table 1. It should be noted that the cost of the first version, i.e., the virtual versions for all COTS alternatives is 0 and reliability is 0.001. This is done for the following reason: If in the optimal solution, for some module $x_{ij1} = 1$, that implies corresponding alternative is not to be attached in the module.Let $L = 4,\ s_1 = \{1,2,4,5\},\ s_2 = \{1,3,4\},\ s_3 = \{2,4,5\},$ $s_4 = \{1,2\}\ f_1 = 0.30,\ f_2 = 0.25,\ f_3 = 0.25$ and $f_4 = 0.20$

It is also assumed that $t_1 = 0.01,\ t_2 = 0.05$ and $t_3 = 0.01$.

$E(C) = 270$ and $\sqrt{Var(C)} = 10.5$

$E(R) = 0.80$ and $\sqrt{Var(R)} = 0.020$

$e = f = 1.96$

Structure of Software

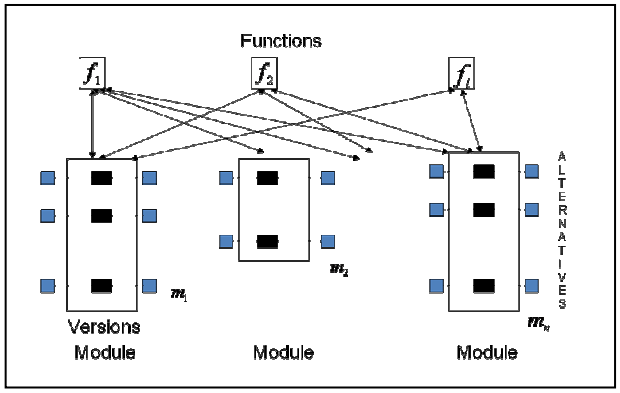

Figure 1. Structure of Software

A. *Data Set*

Table 1 gives cost, reliability, and delivery time for the COTS components.

Modules	Alternatives	Versions 1		2		3	
		Cost	Reliability	Cost	Reliability	Cost	Reliability
1	1	0	0.001	20	0.84	25	0.88
	2	0	0.001	25	0.90	22	0.86
	3	0	0.001	27	0.89	24	0.87
2	1	0	0.001	30	0.92	32	0.91
	2	0	0.001	28	0.87	22	0.85
	3	0	0.001	35	0.95	30	0.93
	4	0	0.001	37	0.96	35	0.97
3	1	0	0.001	23	0.92	25	0.94
	2	0	0.001	26	0.95	23	0.90
4	1	0	0.001	17	0.87	21	0.90
	2	0	0.001	15	0.82	12	0.80
	3	0	0.001	20	0.90	18	0.88
5	1	0	0.001	25	0.95	29	0.96
	2	0	0.001	30	0.95	32	0.92

Table 1. Data Sets

C. Optrimization model III

Optimization model-III is a chance constraint goal programming model, where the two goals are reliability and cost of the overall system. The problem is solved using software called LINGO [18].

$x_{111} = x_{122} = x_{132}$

$x_{212} = x_{221} = x_{232} = x_{242}$

$x_{311} = x_{322}$

$x_{411} = x_{423} = x_{432}$

$x_{511} = x_{522}$

It is observed that two or more alternatives are chosen for first, second and fourth module. Redundancy is allowed for these modules. The overall cost of the above system is 242 units and reliability is 0.84.

D. Optrimization model IV

To check compatibility among the alternatives of the modules we have considered optimization model-III, We assume that the second alternative of first module is compatible with the third and fourth alternatives of the second module.

$x_{112} = x_{123} = x_{133}$

$x_{213} = x_{221} = x_{233} = x_{242}$

$x_{311} = x_{322}$

$x_{411} = x_{422} = x_{432}$

$x_{511} = x_{522}$

It is observed that owing to the compatibility condition, the third alternative of the second module is chosen as it is compatible with the second alternative of the first module.

The overall cost of the above system is 256 units and reliability is 0.81.

V. CONCLUSION

In this paper the problem of optimal selection of COTS components for a fault tolerant modular software system under consensus recovery block scheme is considered. Components are selected in such a way so as to achieve dual objective of maximizing system reliability and minimizing the overall system cost. The parameters corresponding to reliability and cost have been considered as a random variable with known mean and standard deviations and modelled as a chance constraint goal programming problem. The aspiration levels of the objectives have been taken as their ideal solutions. The issue of compatibility among different COTS alternatives was also considered by adding constraints on compatibility to optimization model-III. The problem is then solved using software called LINGO.

REFERENCES

[1] R.S. Pressman, “Software Engineering a Beginners Guide”, McGraw Hill, 1986.

[2] M. Vigder Architecture for COTS based software systems. NRC report 41603, National Research Council, Canada, 1998.

[3] R.K. Scott, J.W. Gautt and D.F. Mc Alliter “Fault tolerant software reliability modelling” IEEE transactions on Software Engineering 1987; 582-592

[4] D.F. McAllister and R.K. Scott, “Cost Modelling of Fault Tolerant Software”. Information and Software Technology, vol. 33, no. 8, 1991.

[5] N. Ashrafi and O. Berman, “Optimal design of large software systems considering reliability and cost”. IEEE Trans. Reliability, vol. 41, no. 2, 1992, pp.281-287

[6] O. Berman N. Ashrafi “ Optimization Models for Reliability of Modular Software Systems”, IEEE Transactions on Software Engineering, vol. 19, no. 11, 1993.

[7] U. D. Kumar, “Reliability analysis of fault tolerant recovery block”. OPSEARCH, 35(2), 1998, 281-294.

[8] O. Berman and U. Dinesh Kumar, " Optimization models for recovery block scheme". European Journal of Operations Research.,vol. 115, no. 2, 1999, pp. 368-379.

[9] 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.

[10] P.Gupta, M.K. Mehlawat, Garima Mittal, Shilpi Verma, Reliability Modelling. IEEE Transactions on Software Engineering, SE – 13, no. 5, 2009.

[11] P.C. Jha, Shivani Bali, P.K. Kapur, Fuzzy Approach for Selecting Optimal COTS based Software Products under Consensus Recovery Block Scheme, Paper accepted for publication in "BVICAM's International Journal of Information Technology (BIJIT)" 3(1) (2011).

[12] B. Contini. A stochastic approach to goal programming. Operations Research 16 (3), 576–586 1968.

[13] I.M. Stancu-Minasian. Stochastic Programming with Multiple Objective Functions. D. Reidel Publishing Company, Dordrecht, 1984.

[14] I.M. Stancu-Minasian, V . Giurgiutiu, Stochastic Programming with Multiple Objective Functions (Mathematics and its Applications). Kluwer Academic Publishers. 1985.

[15] A. Charnes, W.W. Cooper, Chance constraints and normal deviates. Journal of American Statistics Association 57, 134–148. 1952.

[16] A. Charnes, W.W. Cooper, Chance constrained programming. Management Sciences 6, 73–80.1959

[17] A. Charnes, W.W. Cooper, Deterministic equivalents for optimizing and satisfying under chance constraints. Operations Research 11, 18–39. 1963

[18] H. Thiriez, "OR software LINGO", Eur. J. Opl. Res. **124** (2000) 655-656.

◈◈◈