Ratio Mathematica, 21, 2011, pp. 91-105

An Optimization Framework for "Build-or-Buy" Strategy for component Selection in a Fault Tolerant Modular Software System under Recovery Block Scheme

P.C.Jha* Ritu Arora** U.Dinesh Kumar***

*Department of Operational Research , University of Delhi,India
**Maharaja Agrasen Institute of Technology, GGSIP University, Delhi,India.
***Indian Institute of Management, Bangalore, India

*jhapc@yahoo.com ** arora_ritu21@yahoo.co.in ***dineshk@iimb.ernet.in

Abstract

This paper discusses a framework that helps developers to decide whether to buy or build components of software architecture. Two optimization models have been proposed. First model is Bi-criteria optimization model based on decision variables in order to maximize the software reliability with simultaneous minimization of the overall cost of the system. The second optimization model deals with the issue of compatibility.

Keywords : Modular software, software reliability, software cost, fault tolerance, software components, recovery block scheme

1. Introduction

Science and technology demand high quality software for making improvement and breakthroughs. Today, computer hardware and software permeates our modern society. The newest cameras, VCRs, and automobiles cannot be controlled and operated without computers. When the requirement for and dependencies on computer increases, the possibility of crises from computer failures also increases. Software systems are developed as per the requirements given by the users. While developing the software, quality and reliability of the software are two key factors. Reliability of a software system is defined as the probability that software operates without failure in a specified environment, during a specified exposure period. Introduction of redundancy in the parts of the hardware and/or software components is one of the most followed ways to improve the reliability of the system under development. A careful use of redundancy may allow the system to tolerate faults. Despite that we still cannot guarantee error free software. A way of handling unknown and unpredictable

software failures is through fault tolerance. One way to reduce the risks of software design faults and thus enhance software dependability is to use software fault tolerance techniques. Software fault tolerance techniques are employed during the procurement, or development, of the software. They enable a system to tolerate software faults remaining in the system after its development. When a fault occurs, these techniques provide mechanisms to the software system to prevent system failure from occurring. There are two structural methodologies for Fault Tolerant System i.e. Recovery Block Scheme and N-Version Scheme. In this paper, we will discuss optimization model for recovery block. Non functional aspects play a significant role in determining software quality. Given the fact that lack of proper handling of non functional aspects (Cysneiros et al, [5]) of a software application has led to a series of software failures, 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 buying or building components of software architecture on the base of cost and non functional factors. While developing software, components can be both bought as COTS (Commercial Off-The Shelf) 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 today's 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. No changes are normally made to their source codes. COTS components are used without any code modification and inspection. 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. Kapur et al [8] discussed issues related to reliability of systems through weighted maximization of system quality subject to budgetary constraint.

This paper discusses the issues related with reliability of the software systems and cost produced by integrating COTS or in-house build components. Large software system has 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 and only one version will be selected for each alternative of a module. If a component is in-house build component, then the alternative of a module is selected. A schematic representation of the software system is given in Figure 1. We are selecting the components for modules to maximize the system reliability by simultaneously minimizing the cost. The frequency with which the functions are used is not same for all of them and not all the modules are called during the execution of a function, the software has in its menu. Software whose failure can have bad effects afterwards can be made fault tolerant through redundancy at module level (Belli and Jadrzejowicz, [1]). We assume that functionally equivalent and independently developed alternatives (i.e In-house or COTS) for each module are available with an estimated reliability and cost. The first optimization model (optimization model-I) of this paper maximizes the system reliability with simultaneously minimizing the cost. The model contains four problems (P1), (P2), (P3) and (P4). Problem (P1) is not in normalized form, therefore, it has been normalized and transformed into problem (P3) and (P4). 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. The models discussed are illustrated with numerical example.

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 j 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_{ijk} : Reliability of version k 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} : $\begin{cases} 1 \text{ if the } j\text{th alternative of } i\text{th module is in-house developed.} \\ 0 \text{ otherwise} \end{cases}$

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

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 hold good for the second model, but with additional assumptions related to compatibility among alternatives of a module.

The following assumptions are common for the optimization models are:

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 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/ In-House components) are attached in the modules and work similar to the recovery block scheme discussed in (Berman et al., [2] and Kumar, [9]).
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, reliability and development time 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 having 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.

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

3.1.2 Redundancy Constraint

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

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

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

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

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}). Cortellessa et al [4] 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 (Bertolino and Strigini, [3]). *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.*

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

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 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

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

The following equalities come straightforwardly:

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

therefore, we have

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

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.

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

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

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 $(t_{ij} + \tau_{ij} N_{ij}^{tot})$.

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

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.

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

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

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

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 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 (P1)

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

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

Subject to $X \in S = \left\{ x_{ijk} \text{ and } y_{ij} \text{ are binary variable/}\right.$

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

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

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

$$N_{ij}^{suc} = \left(1-\pi_{ij}\right)N_{ij}^{tot} \quad ,i = 1,2,....,n \text{ and } j = 1,2,....,m_i \qquad (4)$$

$$\rho_{ij} = \frac{1-\pi_{ij}}{\left(1-\pi_{ij}\right)+\pi_{ij}\left(1-\pi_{ij}\right)^{N_{ij}^{suc}}};\ \ i=1,2,....,n \text{ and } j=1,2,....,m_i \qquad (5)$$

$$s_{ij} = \rho_{ij} y_{ij} + r_{ij};\ i=1,2,....,n \text{ and } j=1,2,....,m_i \qquad (6)$$

$$y_{ij} + \sum_{k=1}^{V_{ij}} x_{ijk} = 1;\ \ i=1,2,....,n \text{ and } j=1,2,....,m_i \qquad (7)$$

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

$$x_{ij1} + z_{ij} = 1;\ i=1,2,....,n \text{ and } j=1,2,....,m_i \qquad (9)$$

$$\sum_{j=1}^{m_i} z_{ij} \geq 1;\ i=1,2,....,n \qquad (10)$$

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

Where X is a vector of elements : x_{ijk} and y_{ij}; $i=1,2,.....n$; $j=1,2,....,m_i$; k=1,2,....V_{ij}

3.3.1 Normalization

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

$$c = \sum_{i=1}^{n}\sum_{j=1}^{m_i}\sum_{k=1}^{V_{ij}} C_{ijk}\ , \qquad \bar{c} = \sum_{i=1}^{n}\sum_{j=1}^{m_i} c_{ij}\left(t_{ij}+\tau_{ij}N_{ij}^{tot}\right)$$

Now $\overline{C_{ijk}} = \frac{C_{ijk}}{c+\bar{c}}$, $\overline{c_{ij}} = \frac{c_{ij}\left(t_{ij}+\tau_{ij}N_{ij}^{tot}\right)}{c+\bar{c}}$ and $\overline{C}_{ijk} + \bar{c}_{ij} = 1$

The resulting problem then can be rewritten as follows.

Problem (P2) *Maximize* $\mathrm{F}_1(X) = \sum_{l=1}^{L} f_l \prod_{i\in s_l} R_i$

$$\textit{Minimize} \quad F_2(X) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(\overline{c_{ij}} y_{ij} + \sum_{k=1}^{V_{ij}} \overline{C_{ijk}} x_{ijk} \right)$$

Subject to $X \in S$

The problem (P2) can further be written as vector optimization problem as.

Problem (P3) *Vector Max* $F(X)$

Subject to $X \in S$

where $F(X) = (F_1(X), F_2(X))^T$

3.3.2 Finding Properly Efficient Solution

Definition 1 (Steuer, [10]): A feasible solution $X^* \in S$ is said to be an efficient solution for the below problem if there exists no $X \in S$ such that $F(X) \geq F(X^*)$ and $F(X) \neq F(X^*)$

Definition 2 (Steuer, [10]): An efficient solution $X^* \in S$ is said to be an properly efficient solution for the problem (P2) if there exist $\alpha > 0$ such that for each r

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

Using Geoffrion's scalarization the problem (P2) reduces to

Problem (P4)

Maxize $Z = \lambda_1 F_1 - \lambda_2 F_2$

Subject to $X \in S$

$\lambda_1 + \lambda_2 = 1 \quad \lambda_1, \lambda_2 \geq 0$

Lemma(Geoffrion,[6]): The optimal solution of the problem (P4) for fixed λ_1 and λ_2 is a properly efficient solution for the problem (P3) as well as (P1).

3.4 Optimization Model II

Optimization model II is an extension of optimization model I. As explained in the introduction, it is observed that some alternatives of a module may not be compatible with alternatives of another module (Jung and Choi, [7]). The next optimization model II addresses this problem. It is done, incorporating additional constraints in the optimization models. This constraint can be represented as $x_{gsq} \leq x_{hu_ic}$, 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.

$$x_{gsq} - x_{hu_t c} \le My_t,\ q = 2,.....,V_{gs},\ \mathrm{c} = 2,.....,V_{hu_t},\ s = 1,.....,m_g \qquad (12)$$

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

Constraint (12) and (13) make use of binary variable y_t to choose one pair of alternatives from among different alternative pairs of modules. Problem (P3) can be transformed to another optimization problem using compatibility constraints and if more than one alternative compatible component is to be chosen for redundancy, constraint (13) can be relaxed as follows.

$$\sum y_t \le z\left(V_{hu_t} - 2\right) \qquad (14)$$

4. 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 II, respectively. Let $L = 3,\ s_1 = \{1,2,3\},\ s_2 = \{1,3\},\ s_3 = \{2\},\ f_1 = 0.5,\ f_2 = 0.3$ and $f_3 = 0.2$. It is also assumed that $t_1 = .01, t_2 = .05$ and $t_3 = .01$

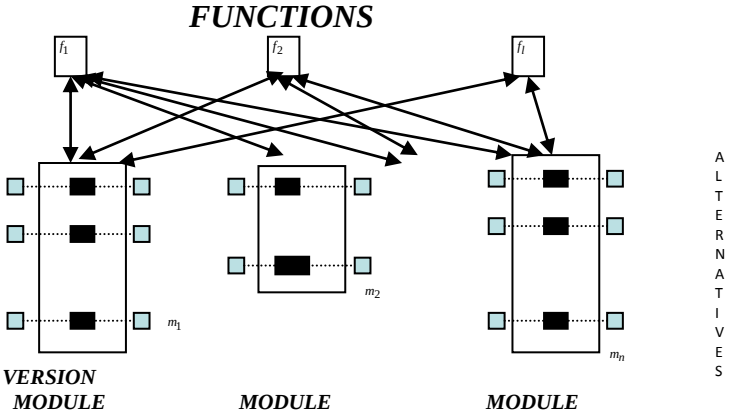

Figure 1 Structure of the software

Table 1: Data set for COTS components

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

Table 2Data set for In-House conponents

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

4.1 Optimization Model – I

Table 3 presents the solution for optimization model I. The problem is solved using software package LINGO (Thiriez, [11]). 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 when the delivery time was 10 units, then only COTS components were selected. When the delivery time increases along with the COTS components, in house build components were also selected. When the delivery time was 12 units, only one in-house component was developed with the minimum cost 79 units attained at reliability level 0.85.Our system cost decreases while the corresponding reliability increases because the components developed in-house decreases the cost initially but later if the level of reliability has to be kept at 0.90 then by increasing delivery time by 5 and 9 units respectively, more in-house build components were selected which in turn increases the cost and reliability of the overall system. Redundancy is also there in all the four cases.

Table 3: Solution of Optimization Model I

Case No.	Delivery Time	COTS	IN-House	System Reliability	Overall system Cost	Joint Objective Value
1	10	$x_{111} = x_{123} = x_{132} = 1$ $x_{211} = x_{221} = x_{232} = x_{242} = 1$ $x_{311} = x_{322} = 1$	Nil	0.84	82	0.66
2	12	$x_{111} = x_{123} = x_{132} = 1$ $x_{211} = x_{232} = x_{241} = 1$ $x_{311} = x_{322} = 1$	$y_{22} = 1$	0.85	79	0.68
3	17	$x_{111} = x_{123} = x_{132} = 1$ $x_{211} = x_{221} = x_{232} = 1$ $x_{311} = 1$	$y_{24} = y_{32} = 1$	0.93	86	0.74
4	21	$x_{111} = x_{132} = 1$ $x_{211} = x_{221} = x_{232} = 1$ $x_{311} = 1$	$y_{12} = y_{24} = y_{32} = 1$	0.94	92	0.75

4.2 Optimization Model-II

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

Case 1. Delivery Time is assumed to be 10 units.

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

$x_{111} = x_{123} = x_{133} = 1$

$x_{211} = x_{221} = x_{232} = x_{242} = 1$

$x_{311} = x_{322} = 1$

It is observed that due to the compatibility condition, third alternative of first module is chosen as it is compatible with third alternative of second module. The system reliability for the above solution is 0.84 and cost is 81 units.

Case 2. Delivery Time is assumed to be 12 units.

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

$y_{22} = 1$;

$x_{111} = x_{123} = x_{133} = 1$

$x_{211} = x_{232} = x_{241} = 1$

$x_{311} = x_{322} = 1$

It is observed that due to the compatibility condition, third alternative of first module is chosen as it is compatible with second alternative of third module. The system reliability for the above solution is 0.85 and cost is 77 units.

Case 3. Delivery Time is assumed to be 10 units.

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

$y_{24} = y_{32} = 1$

$x_{111} = x_{123} = x_{133} = 1$

$x_{211} = x_{221} = x_{232} = 1$

$x_{311} = 1$

It is observed that due to the compatibility condition, third alternative of first module is chosen as it is compatible with third alternative of second module. The system reliability for the above solution is 0.94 and cost is 84 units.

5. Conclusions

We have presented optimization models that supports the decision whether to buy software components or to build them in-house upon designing structure. A fault tolerant software structure for Recovery block scheme is discussed. A numerical example is presented to support these models. When delivery time is small then all the COTS components were selected and redundancy is allowed. But as the delivery time increases along with the COTS components in-house components

were also selected and different impacts on cost and reliability were considered. Redundancy was also there in all the cases.

References

[1] F. Belli and P. Jadrzejowicz, An approach to reliability optimization software with redundancy, *IEEE Transaction of Soft. Engineering,* vol.17/3(1991), pp. 310-312.

[2] O. Berman and U. D.Kumar, Optimization models for reliability of modular software system, *IEEE Transactions of Software Engineering, vol.* 19/11(1993), pp.1119-1123.

[3] A. Bertolino, and L. Strigini, On the use of testability measures for dependability assessment, *IEEE Transactions on Software Engineering,* 22/2(1996), pp.97-108.

[4] V. Cortellessa, F. Marinelli, and P. Potena, An optimization framework for "build-or-buy" decisions in software architecture, *Computers and Operations Research*, vol.35(2008), pp. 3090-3106.

[5] L. M. Cysneiros and J.C.S. Leite, Nonfunctional requirements: from elicitation to conceptual models, *IEEE transactions on Software engineering*, vol.30 (2004), pp. 328-350.

[6] A. M. Geoffrion, Proper efficiency and theory of vector maximization, *Journal of Mathematical Analysis and Application*, vol.22 (1968), pp. 613-630.

[7] H. W.Jung and B. Choi, Optimization models for quality and cost of modular software system, *European Journal of Operations Research,* vol.112(1998), pp. 613-619.

[8] P. K. Kapur, A.K. Bardhan and P.C. Jha, Optimal reliability allocation problem for a modular software system, *OPSEARCH*, vol.40/2(2003).

[9] U. D. Kumar, Reliability analysis of fault tolerant recovery block, *OPSEARCH*, vol.35(1998),pp. 281-294.

[10] R. E. Steuer, Multiple Criteria optimization: theory, computation and application, Wiley, New York ,1986.

[11] H. Thiriez, OR software LINGO, *European Journal of Operational Research,* vol.12(2000), pp.655-656.