Proceedings of the First International Conference on Information Technology and Knowledge Management pp. 77–84 DOI: 10.15439/2018KM11 ISSN 2300-5963 ACSIS, Vol. 14

A study of Optimal Testing Resource Allocation Problem for Modular Software with Change Point

Gurjeet Kaur[1], Anu G. Aggarwal[2], Aman Kedia[3]

[1]Shaheed Sukhdev College of Business Studies, University of Delhi, India
[2]Department of Operational Research, University of Delhi, India
[3]Shaheed Sukhdev College of Business Studies, University of Delhi, India
[1]gurjeetkaur@sscbsdu.ac.in, [2]anuagg17@gmail.com ,[3]aman.kedia1111@gmail.com

***Abstract*—For evolving trustworthy software, engrossing on uncovering process of fault in software is central. Nevertheless, during testing, modifications in the testing routine, defect gravity or testing-skill maturity and working environment, there can be notable change in fault detection rate. When this sort of pattern is observed in testing time it is called change point. In this article, we inquire a resource distribution problem that optimally distributes software developing resources in such a way that the cost of development is curtailed to optimization. In this problem, for all modules the effect of chief circumstantial element of change-point is considered. The constraint of pulling off the desired reliability level for every individual module is also incorporated in the formulation of the problem. A framework based on Karush Kuhn Tucker (KKT) conditions is presented to work out the resulting non-linear optimization problem. A simulated numerical illustration has been analyzed to reflect the formulation of the case and its solution by the algorithm proposed.**

***Index Terms*—Modular Software, Change-point, Software Reliability Growth Model, Resource Allocation**

I. Introduction

'BREAKING a vast and complex assignment into small simple steps of objectives' is one of the deep-seated rules to success. This law prevails into software industry as well. The quality, convolution and size of software have risen, resulting in complete development of it as a hard process. So, today maximum soft coded sharewares are developed by amalgamating small ordered independent soft codes called modules. This methodology of developing Modular Software System helps soft code teams to proficiently build the large sized coded complex development process in a structured manner.

With the rising popularity of having efficient modular software, there arises the need of designing reliable structures. As per texts [13], Software reliability is kept as the chances that the software does not go wrong within an assigned period of time under given circumstances. Therefore, with shortened maturity cycles, raised complexity of software design, and great caustic penalty of software failures, a major responsibility lies in the area of software testing. During the advance in development of modular software, faults can seep in modules owing to individual flaw. These faults make themselves marked in expressions of malfunction when the codes are weathered autonomously during the module testing phase of software development life cycle. To evaluate modular software quantitatively and to find the total number of faults removed from each module mathematical tools namely software reliability growth models (SRGMs) are employed. Assorted noteworthy metrics, like primary number of faults, failure concentration, Trustworthiness within a specific period of time, number of faults residual, can be smoothly dogged through SRGM's.

Through the last three decades, a great number of SRGMs have been framed in literature [3, 13, 16, 19, 20, 22]. Early researches related SRGM with respect to testing time [3, 17]. However, incorporation of testing resources leads to development of more accurate SRGMs [4, 5, 9, 26]. The reason being, the detection of faults are more intimately linked to the amount of resources expended [18, 24] on testing as it includes-

(a) Manpower, that takes into account

- Testing group (malfunction recognition personnel).
- Debugging group (Programmers\Failure rectification personnel).

(b) CPU working moments

Huang et al. [5] worked with logistic resource function. They established that both exponential-type and S-type NHPP models can come under ideal and imperfect debugging situations. Later, Kapur et al.[9] deliberated on the testing resource reliant learning process and classified faults into two types on the source of amount of testing resources needed to remove them. This paper models fault exclusion rate in terms of testing resource.

For accomplishing the goal of developing reliable modular software, the consideration of fault detection process is vital as it aid in establishing the value of uncovering bugs that lie dormant in the software by test techniques and test cases. A lot of SRGMs in literature supposed that during the fault detection process each failure caused by a fault occurs independently and at random time according to the same distribution [3, 20]. However, practically as the testing evolves, the testing team gains insight and with the employment of new tools and techniques the fault detection rate (FDR) gets markedly changed. Further, the other factors that can affect the fault detection rate are operation environment, testing stratagem and shortcoming density. And the point of time where change in fault detection rate is observed is termed as ‘Change Point’. It was Zhao [28] who came up with the concept of Change point in software as well as hardware reliability modeling. He established that incorporation of change point in SRGMs is vital for effective fault detection modeling. Shyur [23], Wang and Wang [25] also made offerings in this area. In accumulation, some studies included change-point analysis in their models as the testing resource consumption may not be smooth over time [4, 10, 15].

Another core apprehension in the software production is the software development outlay. Overspending on development cost can result in financial crisis for the company. On the other hand, spending less can result in low quality software product as in this case the software development firm will have to set low reliability aspiration level for each module. Thus, there arises the need of optimizing total development cost of modular software. In modular soft code testing stage, each modular code is tested autonomously. But this has to be carried out in finite time. However, completing this task involves on an average of 50% of the total budget of software development cost. Hence there is requirement of resource distribution decisions popularly known as “Testing Resource Allocation problems”. This research is adding contribution in Resource Allocation Area. We have developed change point incorporated non linear resource distribution case which is solved by Karush Kuhn Tucker (KKT) optimality conditions.

The work is organized as follows: Section 2 details the literature review on testing resource allocation problem Section 3 highlights on the Goel-Okumoto software reliability growth model with change point and testing resource, required for modeling the failure mechanism of the modules. Section 4 elaborates on formulation of our testing resource allocation problem. Further in this section we also discuss an optimization algorithm based on the KKT optimality conditions. Section 5 illustrates the optimization problem solution through a numerical example. Eventually, conclusions are derived and are given in section 6.

II. Literature Review on Testing Resource Allocation Problems

The value of SRGMs is not constrained to reliability assessment of software systems. There application is well researched and applied in resource distribution cases. These decision problems are critical for software development firms aligned to reliability, cost, time and resources parameters. But the goal goes specific with the company needs. There are software development firms that aim at optimization of failure numbers. There are some that goes with optimal cost decisions. Then there are others that go with constrained optimization problems. With these different interests consideration, software allocation has a rich repository of research papers. Ohetera and Yamada [21] proposed resource distribution problem for optimizing the remaining faults and optimizing resources respectively. Yamada et al. [27] also formulated a constrained resource distribution problem with constraint of reliability level. Hyper-geometric model for modeling software reliability growth was employed by Huo et al. [7] to make optimal distribution of resources amongst modules. Kapur et al. [8] discussed the concept of marginal testing effort function (MTEF) and related optimal resource distribution problem. Not only this, Kapur et al.[11,12, 29], using S-Shaped and exponential SRGMs [3,20] have provided different allocation cases studied various resource allocation problems maximizing the number of faults removed from each module. Khan et al. [14] Huang et al. [6] too formulated and solved constrained optimal resource distribution cases with one or more parameters of cost, fault and reliability. Sindhuja et al.[30] have investigated assorted software make public decision policies and resource distribution cases considering the dual restrictions of reliability and expenses.

However, the impact of change point has yet not been considered in allocation problems for modeling the reliability growth of modules. In this paper, we inquire a resource distribution problem that optimally distributes software developing resources in such a way that the cost of development is curtailed to optimization. In this problem, for all modules the effect of chief circumstantial element of change-point is considered. The constraint of pulling off the desired reliability level for every individual module is also incorporated in the formulation of the problem. (figure 2.1).

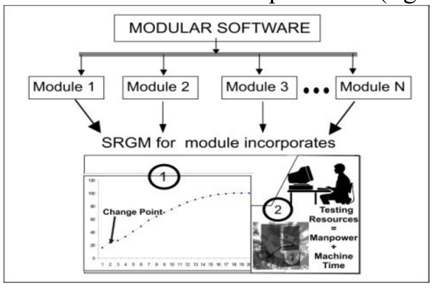
MODULAR SOFTWARE
Module 1
Module 2
Module 3
•••
Module N
SRGM for module incorporates
1
Change Point
2
Testing Resources = Manpower + Machine Time

Figure 2.1 Modular Software incorporating Change Point and Testing Resources

III. Goel-Okumoto(GO) Model Incorporating Testing Resource And Change Point

A. Model Notations

a	Initial number of faults
W(t)	Cumulative testing resource in the time interval (0,t]
w(t); $w(t) = \frac{d}{dt}W(t)$	Testing resource intensity
τ	Change Point
$m(t)$ or $m(W(t))$	Cumulative number of faults removed by time t
$b(t)$ or $b(W(t))$	Testing resource expenditure based fault detection rate per remaining fault

b_1	Fault detection rate per remaining fault before change point
b_2	Fault detection rate per remaining fault after change point

The GO software reliability growth model incorporating testing resource and change point [4] taken in this article has its foundation on Non Homogeneous Poisson Process (NHPP). The NHPP models are based on the supposition that the software system is constrained to failures at random times caused by manifestation of remaining faults in the system. So, for modeling the software fault detection phenomenon, counting process $\{N(t); t \geq 0\}$ is defined. This counting process exhibits the collective number of faults discovered by testing time t. And the SRGM formulated on this can be put mathematically as-

$$Pr\{N(t)=n\} = \frac{m(t)^n \cdot \exp(-m(t))}{n!}, \quad n = 0,1,2,... \tag{3.1}$$

B. Model Assumptions

- NHPP governs removal phenomenon.
- Software is prone to failures during the execution caused by faults remaining in the software.
- On a malfunction, the slip causing that failure is without delay distant and no fresh faults are introduced.
- The fault detection rate is with reverence to testing resource strength and is proportional to the existing fault content in the software.
- The fault recognition rate may transform at some time moment (called change point, denoted by τ).

In view of the above assumptions, the model can be summarized by the following differential equation.

$$\frac{\frac{d}{dt}m(t)}{w(t)} = b(t)(a - m(t)) \tag{3.2}$$

Since, fault detection rate (FDR) changes at time point τ , therefore, it is defined as:

$$b(t) = \begin{cases} b_1, & \text{when } 0 \leq t \leq \tau \\ b_2, & \text{when } t > \tau \end{cases} \tag{3.3}$$

Case 1: For $0 \leq t \leq \tau$

Using equation (3.2) and (3.3), we have the following differential equation

$$\frac{\frac{d}{dt}m(t)}{w(t)} = b_1(a - m(t)) \tag{3.4}$$

Solving (3.4) with initial condition m(t=0)=0, we get,

$$m(t) = a\left[1 - e^{-b_1 W(t)}\right] \tag{3.5}$$

Case 2: $t > \tau$

Again by equation (3.2) and (3.3), we have

$$\frac{\frac{d}{dt}m(t)}{w(t)} = b_2(a - m(t)) \tag{3.6}$$

The solution of above differential equation under initial condition $m(t=\tau) = m(\tau)$ is

$$m(t) = a\left[1 - e^{-b_1 W(\tau) - b_2(W(t) - W(\tau))}\right] \tag{3.7}$$

Combining Case 1 and Case 2, we have the following expression of m(t)

$$m(t) = \begin{cases} a\left[1 - e^{-b_1 W(t)}\right], & \text{when } 0 \leq t \leq \tau \\ m(t) = a\left[1 - e^{-b_1 W(\tau) - b_2(W(t) - W(\tau))}\right], & \text{when } t > \tau \end{cases}$$

where m(t) is the mean value function of the counting process N(t).

IV. Testing Resource Allocation Problem

Let us consider modular software with N modules. These soft coded structures vary in parameters like- (a) Complexity (b) Size (c) Performing functions. We also consider that the testing of these modular codes is done independently. We also take into account a reasonable assumption of not having infinite faults in the software modular structures. The optimization problem considered is to minimize the software development cost underneath a conjecture that there is change in fault detection rate in each soft code module and a set level of reliability is to be pulled-off.

A. Notations

i	Module number counter i=1,2,…N
a_i	Initial number of faults in module i
τ_i	Change point for ith module
W(τ_i)	Testing resource consumed by ith module till τ_i .
b_{1i}	Detection process pace in terms of rate before change point in each module
b_{2i}	Detection process pace in terms of rate after change point in each module
C_{1i}	Fault removal cost per fault before change point in ith module during testing phase
C_{2i}	Fault removal cost per fault after change point in ith module during testing phase
C_{3i}	Fault removal cost per fault from ith module in operational phase.
C_4	Testing cost
R_0	Reliability level
W	Sum testing resource

B. Modeling mean value and reliability function for the modules

It is not possible to test all the test cases of a large complex Software. The two major reasons for this infeasibility is (i) Deadline for the development, that is its release time (call it T). (ii) Resources of testing. Let W_i be the testing resource used up on the ith module during testing time T. Therefore, devoid of any loss of generality the numeral of faults distant by time T can be assumed to be a function of testing resource explicitly and using the model discussed in section 3, the mean value function for the faults removed from each module can be written as:

$$\text{i.e. } m_i(W_i) = \begin{cases} a_i\left(1 - e^{-b_{1i}W_i}\right) & \text{when } 0 \le t \le \tau_i \\ a_i\left(1 - e^{-b_{1i}W_i(\tau_i) - b_{2i}(W_i - W_i(\tau_i))}\right), & \text{when } t > \tau_i \end{cases}$$

$$\forall\ i = 1, 2, \cdots, N \qquad (4.1)$$

Modeling expected number of faults removed from each module using such an SRGM is advantageous because it takes into consideration the consequence of change point.

Mathematically, reliability of soft codes of modules can be described by [13]-

$$R_i(t) \equiv R_i(t + \Delta t \mid t) = \exp^{-\left(m_i(t+\Delta t) - m_i(t)\right)} \qquad (4.2)$$

There is one more way to define Eq. (4.2). This was kept by Huang et al. [5]. Eq. (4.3) states reliability as the fault removals to the proportion of initial number of faults. That is-
$R_i(t) = m_i(t)/a_i \quad \forall\ i = 1, 2, \cdots, N \qquad (4.3)$
Using Eq. (3.1) and (3.4) we have reliability of each module is given by:

$$\frac{m_i(W_i)}{a_i} \quad \forall\ i = 1, 2, \cdots, N \qquad (4.4)$$

When the reliability is defined by expression (4.3) it is assumed that all the faults of the software are of same type and are equally likely to be detected during testing. Thus assumption goes well with respect to the constant FDR for the model under consideration here.

C. Modeling Cost Function for Modular Software

For modeling cost function with respect to time, we have

$$C(T) = C_1 m(\tau) + C_2(m(T) - m(\tau)) + C_3(a - m(T)) + C_4 T \quad (4.5)$$

Eq. (4.5) is with respect to time. Cost modeling with respect to testing resources can be put as-

$$C(W) = C_1 m(W(\tau)) + C_2(m(W) - m(W(\tau))) + C_3(a - m(W)) + C_4 W \quad (4.6)$$

Further, since the software is modular and each module of software is designed independently; therefore, the outlay of fixing and testing of all components is the addition of individual modules testing cost. Mathematically, we have:

$$\begin{aligned} C(W) &= \sum_{i=1}^{N} C_i(W_i) \\ &= \sum_{i=1}^{N} C_{1i} m_i(W_i(\tau_i)) + \sum_{i=1}^{N} C_{2i}(m_i(W_i) - m_i(W_i(\tau_i))) + \\ &\quad \sum_{i=1}^{N} C_{3i}(a_i - m_i(W_i)) + C_4 \sum_{i=1}^{N} W_i \end{aligned} \qquad (4.7)$$

D. Problem Structure and Solution Algorithm

The resource distribution problem is structured with this scenario- (a) W is the sum resources that needs to be distributed in N independent modules. (b) The aim of the allocation problem is- Minimizing Software Development Cost given by Eq.(4.7) (c) The objective is constrained to aspired reliability of at least R_0

$$\begin{aligned} \text{Min} \quad C(W) &= \sum_{i=1}^{N} C_i(W_i) \\ &= \sum_{i=1}^{N} C_{1i} m_i(W_i(\tau_i)) + \sum_{i=1}^{N} C_{2i}(m_i(W_i) - m_i(W_i(\tau_i))) + \\ &\quad \sum_{i=1}^{N} C_{3i}(a_i - m_i(W_i)) + C_4 \sum_{i=1}^{N} W_i \end{aligned}$$

Constrained to $\sum_{i=1}^{N} W_i \le W,$

$$R_i \ge R_0 \qquad \forall\ i = 1, 2, \cdots, N$$

$$W_i \ge 0, \qquad \forall\ i = 1, 2, \cdots, N$$

(P1)

After substituting the value of $m_i(W_i)$ and R_i from (4.1) and (4.4) in (P1) we get problem as;

$$\begin{aligned} \text{Min} \quad C(W) &= \sum_{i=1}^{N} C_i(W_i) \\ &= \sum_{i=1}^{N} C_i m_i(W_i(\tau_i)) + \sum_{i=1}^{N} C_{2i}\left(\left(a_i\left(1 - e^{-b_{1i}W_i(\tau_i) - b_{2i}(W_i - W_i(\tau_i))}\right)\right) - m_i(W_i(\tau_i))\right) + \\ &\quad \sum_{i=1}^{N} C_{3i}\left(a_i - \left(a_i\left(1 - e^{-b_{1i}W_i(\tau_i) - b_{2i}(W_i - W_i(\tau_i))}\right)\right)\right) + C_4 \sum_{i=1}^{N} W_i \end{aligned}$$

Constrained to

$$\sum_{i=1}^{N} W_i \le W,$$

$$a_i\left(1 - e^{-b_{1i}W_i(\tau_i) - b_{2i}(W_i - W_i(\tau_i))}\right) \ge R_0 \qquad \forall\ i = 1, 2, \cdots, N$$

$$W_i \ge 0, \qquad \forall\ i = 1, 2, \cdots, N$$

Dropping the constant terms and re-writing the above problem in maximization form, we get the problem as:

$$\begin{aligned} \text{Max} \quad Z(W) &= -\sum_{i=1}^{N} C_i(W_i) = \\ &\sum_{i=1}^{N} (C_{3i} - C_{2i})\, a_i\left(\left(1 - e^{-b_{1i}W_i(\tau_i) - b_{2i}(W_i - W_i(\tau_i))}\right)\right) - C_4 \sum_{i=1}^{N} W_i \end{aligned}$$

Constrained to

$$\sum_{i=1}^{N} W_i \le W,$$

$$a_i\left(1 - e^{-b_{1i}W_i(\tau_i) - b_{2i}(W_i - W_i(\tau_i))}\right) \ge R_0 \quad \forall\ i = 1,2,\cdots,N$$

$$W_i \ge 0, \quad \forall\ i = 1,2,\cdots,N \qquad \text{(P2)}$$

Now, the reliability constraint of (P2) can be re-stated as:

$$W_i \ge \frac{-1}{b_{2i}}\left[\ln(1-R_0) + b_{1i}\tau_i - b_{2i}\tau_i\right] \cong (say\ Y_i)$$

Here, it may be noted that summation of Yi should always be less than equal to W, otherwise the constraints of the formulated problem will be inconsistent.

Therefore, using the constraint on reliability and letting $W_i = X_i + Y_i$, the problem can be transformed as:

Max Z(W)=

$$\sum_{i=1}^{N}(C_{3i} - C_{2i})\,a_i\left(\left(1 - e^{-b_{1i}W_i(\tau_i) - b_{2i}(X_i+Y_i-W_i(\tau_i))}\right)\right) - C_4\sum_{i=1}^{N}(X_i + Y_i)$$

Subject to

$$\sum_{i=1}^{N} X_i \le W - \sum_{i=1}^{N} Y_i,$$

$$X_i \ge 0, \quad \forall\ i = 1,2,\cdots,N \qquad \text{(P3)}$$

The first term of the objective function of (P3) is concave and second term $X_i + Y_i$ is linear, therefore $-(X_i + Y_i)$ can be treated as concave. Hence the objective function of (P2) is concave function given to linear constraint. Therefore the case is of convex programming problem.

Thus, the essential most favorable situations pertaining to Karush Kuhn-Tucker for convex programming problem is also sufficient [1]. For the problem (P3) we can affirm the follow saddle value decision making statement:

$$\underset{X_i;i=1,2,..N}{Max}\ \underset{\phi}{Min}\ F(X_1,X_2,\ldots X_N,\phi) = \sum_{i=1}^{N}(C_{3i} - C_{2i})\,a_i\left(\left(1 - e^{-b_{1i}W_i(\tau_i) - b_{2i}(X_i+Y_i-W_i(\tau_i))}\right)\right) - C_4\sum_{i=1}^{N}(X_i + Y_i) + \phi\left(\sum_{i=1}^{N}X_i - W + \sum_{i=1}^{N}Y_i\right) \qquad \text{(P4)}$$

The necessary and sufficient situations for (X^0, ϕ^0) where $X^0 = \{X_i^0: i = 1,\ldots, N\}$ to be a saddle spot for the saddle value cases are based on the KKT conditions and are specified by the next theorem.

Theorem 1. A feasible decision X_i (i =1,..,N) of (P4) is best possible if and only if

$$(1)\ \phi \le C_4 - \left[(C_{3i} - C_{2i})\,a_i b_{2i} e^{-b_{1i}W_i(\tau_i) - b_{2i}(X_i+Y_i-W_i(\tau_i))}\right]$$

$$(2)\ X_i \times \left[\phi - C_4 + \left[(C_{3i} - C_{2i})\,a_i b_{2i} e^{-b_{1i}W_i(\tau_i) - b_{2i}(X_i+Y_i-W_i(\tau_i))}\right]\right] = 0$$

This theorem can be derived straight from KKT conditions.

Ruling a feasible key at optimality condition

Concerning KKT conditions to the (P3) we get-

$$\frac{\partial F(X_1, X_2,\ldots,X_N,\phi)}{\partial X_i} = 0$$

implies

$$X_i^0 = \frac{1}{b_{2_i}}\left[\ln\left[(C_{3i} - C_{2i})\,a_i b_{2i}\right] - \ln(C_4 - \phi) - b_{1i}\tau_i - b_{2i}(Y_i - \tau_i)\right] \qquad (4.8)$$

and $\frac{\partial F(X_1, X_2,\ldots,X_N,\phi)}{\partial \phi} = \sum_{i=1}^{N} X_i - W + \sum_{i=1}^{N} Y_i = 0$ implies

$$\phi^0 = C_4 - \exp\left[\frac{\sum_{i=1}^{N}(1/b_{2_i})\left[\ln\left[(C_{3i} - C_{2i})\,a_i b_{2i}\right] - b_{1i}\tau_i - b_{2i}(Y_i - \tau_i)\right] - W + \sum_{i=1}^{N} Y_i}{\sum_{i=1}^{N}(1/b_{2_i})}\right] \qquad (4.9)$$

The solution algorithm of above problem, using KKT is as follows:

Algorithm 1:

```
Step 0: Calculate $\sum_{i=1}^{N} Y_i$
        If $\sum_{i=1}^{N} Y_i$ >W then
    Available resource W is insufficient to meet reliability aspiration level of all the modules.
            Stop
        Else
                Goto Step 1.
        End if
Step 1: Set S = 0.
Step 2: Calculate $X_i$, $i = 1,\ldots,N-S$; $\phi$ using equation (3.9) and (3.10)
    $X_i^0 = \frac{1}{b_{2_i}}\left[\ln\left[(C_{3i} - C_{2i})a_i b_{2i}\right] - \ln(C_4 - \phi) - b_{1i}\tau_i - b_{2i}(Y_i - \tau_i)\right]$
    $\phi^0 = C_4 - \exp\left[\frac{\sum_{i=1}^{N}(1/b_{2_i})\left[\ln\left[(C_{3i} - C_{2i})a_i b_{2i}\right] - b_{1i}\tau_i - b_{2i}(Y_i - \tau_i)\right] - W + \sum_{i=1}^{N} Y_i}{\sum_{i=1}^{N}(1/b_{2_i})}\right]$
Step 3: Rearrange index i such that:
        $X_1 \ge X_2 \ge X_3 \ldots\ldots \ge X_{N-S}$
Step 4: If $X_{N-S} > 0$ then Stop (the solution is optimal)
        Else $X_{N-S} = 0$; set S = S+1,
        End if
```

Step 5: For re-allocating testing resources to remaining N-S modules go to **Step 2.**

The optimal solution is given by:

$$\begin{cases} X_i^0 = \dfrac{1}{b_{2_i}}\left[\ln\left\{(C_{3i} - C_{2i})a_i b_{2i}\right\} - \ln(C_4 - \phi) - b_{1i}\tau_i - b_{2i}(Y_i - \tau_i)\right], i = 1, \dots, N - S \\ \text{where} \\ \phi^0 = C_4 - \exp\left[\dfrac{\sum_{i=1}^{N}(1/b_{2_i})\left[\ln\left\{(C_{3i} - C_{2i})a_i b_{2i}\right\} - b_{1i}\tau_i - b_{2i}(Y_i - \tau_i)\right] - W + \sum_{i=1}^{N} Y_i}{\sum_{i=1}^{N}(1/b_{2_i})}\right] \\ X_i^0 = 0, \text{ otherwise} \end{cases}$$

V. Numerical Example

Presume that the sum total of testing resource accessible for modules coded software is 60,000. This modular software has six such modules. Further, it is supposed that there is change in fault detection rate in each module. Further, because of change point the cost of testing before and after change point differs. These costs are taken as 1 and 2 units respectively for each module. The price of debugging a fault for any module in operational segment is 8 units and charge of testing up to release time is 0.5 units. The assumed estimated values of all the parameters for modules M1, M2, M3, M4, M5 and M6 are tabulated in table 5.1. Also, it is aspired that the reliability level of each module should be at least 0.8.

Table 5.1 Parameter Estimates for six modules

Module	a	b_1	b_2	$W(\tau)$
M1	1321	0.000213	0.000211	642.85
M2	950	0.000181	0.000129	505.02
M3	1639	0.000112	0.000156	759.18
M4	1450	0.000198	0.000213	580.02
M5	1350	0.000218	0.000229	462.69
M6	987	0.000125	0.000321	315.11
Total	**7697**			

Using algorithm 1 of section 3 the optimal allocation of the resource for the six modules is shown in table 5.2.

Table 5.2 Optimal Allocation of resources for six modules

Module	a	W	m	Reliability	Cost
M1	1321	9438.22	1141	0.864	3553.429
M2	950	12272.7	760	0.8	2957.012
M3	1639	12434.9	1396	0.851	4605.807
M4	1450	9878.21	1272	0.877	3813.009
M5	1350	9176.5	1185	0.877	3566.017
M6	987	6799.44	869	0.88	2646.096
Total	7697	60000	6623		51141.37(including constant cost)

From table 5.2 we have that within the available budget of 6000, the total expected cost of testing all the modules such that the reliability of each module is at least 0.8 is 51141.37. The number of faults removed from the software is 6623.

The impact of increasing aspired reliability level for all modules-

For studying the impact of increasing aspiration level for all modules on the allocation of testing resources, we solved the above numerical illustration by taking reliability level as 0.85 and 0.9 for each module.

Case 1: When aspired reliability for each module is increased from 0.8 to 0.85

In this case we found, that by raising the aspiration of reliability to 0.85, there was allocation of resources to modules (given in table 5.3) but with increase software development cost as compared to the cost of development when reliability level of each module is 0.8(refer table 5.2).

Table 5.3 Optimal Allocation of resources for six modules

Module	a	W	m	Reliability	Cost
M1	1321	8985	1123	0.85	3661.857
M2	950	14502.8	808	0.85	2672.012
M3	1639	12375.2	1394	0.85	4619.499
M4	1450	9216.19	1245	0.858	3975.098
M5	1350	8560.73	1159	0.858	3716.78
M6	987	6360.15	851	0.861	2753.65
Total	7697	60000	6580		51398.90(including constant cost)

Figure 5.1 and 5.2 shows the comparison of Reliabilities and Cost of Modules respectively when Aspiration level is increased from 0.8 to 0.85.

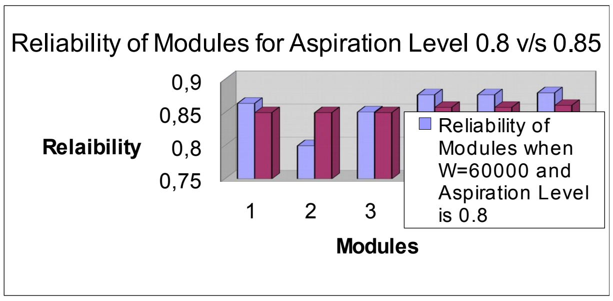

Figure 5.1 Reliability of Modules for Aspiration Level 0.8 v/s 0.85

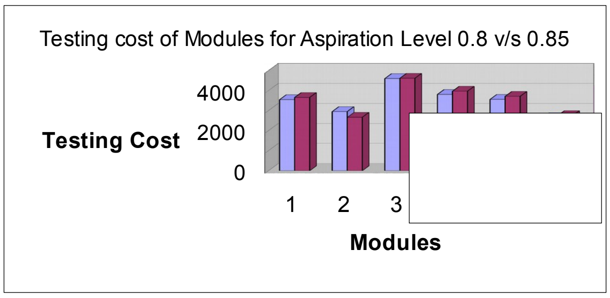

Figure 5.2 Testing Cost of Modules for Aspiration Level 0.8 v/s 0.85

Case 2: When aspired reliability for each module is increased from 0.8 to 0.9

In this case we observed that $\sum_{i=1}^{6} Y_i$ is strictly greater than W. So using the proposed algorithm we stop with the conclusion that the given set of constraint inequalities is inconsistent.

The impact of increasing/decreasing total testing resource budget on the optimal allocation among modules

For studying the impact of change in total testing resource budget on the optimal allocation among modules, we increase and decrease the total testing resource W by 20%.

Case 1: W is increased by 20%
In this case we got the allocation as given in table 5.4.

Table 5.4 Distribution of resources for six modules

Module	a	W	m	Reliability	Cost
M1	1321	11786.4	1212	0.917	3131.269
M2	950	12715	771	0.811	2893.786
M3	1639	15611.1	1491	0.909	4034.809
M4	1450	12204.4	1342	0.925	3394.813
M5	1350	11340.2	1249	0.925	3177.04
M6	987	8342.98	916	0.927	2368.601
Total	7697	72000	6981		55000.32(including constant cost)

From table 5.4 we observe that by increasing the budget there is increase in reliability level of modules but with raise in total cost. Figure 5.3 depicts the comparison of Relaibility of Modules when W=60000 and when it is increased by 20%. The total cost comparison with W=60000 and W=72000 is shown in Figure 5.4.

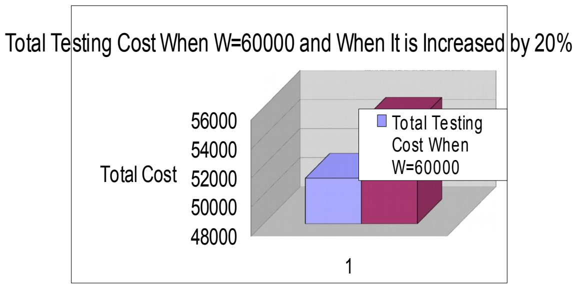

Figure 5.3 Reliability of Modules when W=60000 and When It is increased by 20%

Case 2: W is decreased by 20%

In this case we get that $\sum_{i=1}^{6} Y_i$ is strictly greater than W. So using the proposed algorithm we stop with the conclusion that the given set of constraint inequalities is inconsistent.

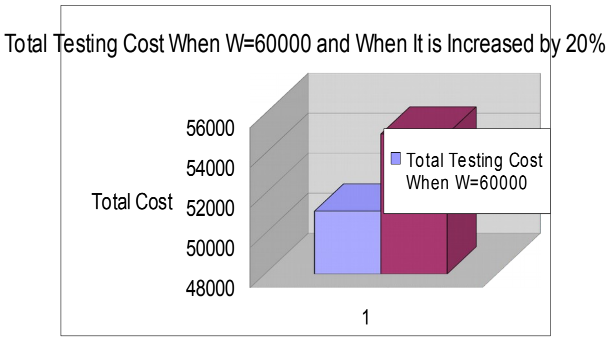

Figure 5.4 Total Testing Cost when W=60000 and When It is increased by 20%

VI. Conclusion

Allocation of testing resources during module testing phase is an important issue for the project managers in developing a reliable and economical modular software system. This research takes into account change point in modular software development and its associated resource distribution problem. For modeling the failure process of modules an exponential SRGM with testing resource and change point is used. The allocation problem is convex programming problem and is solved using Karush Kuhn Tucker (KKT) optimality conditions. Using the numerical example some important observations related to allocation problem is also presented in the paper.

The present study is done under the assumption of independence of the failures of different modules. We can also explore the possibility of including multi dimensional software reliability growth modeling so as to take care the effect of not only testing resource but also other testing factors like testing coverage, testing time/number of test cases on the fault removal process simultaneously. This paper takes into account independent behavior of faults in modules soft codes. In future interactions among modules and dependence of the failures can also be incorporated in the model building.

References

[1] Bazaraa, S. M. and C. M. Setty, “Nonlinear programming: theory and algorithm”, John Wiley and Sons,1979, New York.
[2] Camuffo M., Maiocchi M., Morselli M. (1990) “Automatic Software Test Generation Inform Software Technology”, 32(5), 337-346.
[3] Goel A. L. and Okumoto, K., “Time dependent error detection rate model for software reliability and other performance measures”, IEEE Transactions on Reliability, 1979, R-28, 206–211.
[4] Huang C. Y., “Performance analysis of software reliability growth models with testing-effort and change-point”, Journal of Systems and Software, 2005, 76, 181–194.
[5] Huang C. Y., Kuo S. K., Lyu M. R., “An assessment of testing-effort dependent software reliability growth models”, IEEE Transactions on Reliability, 2007, 56, 198–211.
[6] Huang C. Y., Lo J. H., Kuo S. K. and Lyu M. R., “Optimal allocation of testing resources considering cost, reliability, and testing –effort”, Proceedings of the 10th IEEE Pacific International Symposium on dependable Computing, 2004.
[7] Huo R. H., Kuo S. K., Chang Y. P., “Needed resources for software module test, using the hyper-geometric software reliability growth model”, IEEE Trans. on Reliability, 1996, 45(4), 541-549.

[8] Kapur P. K., Bardhan A., Yadavalli V., “On allocation of resources during testing phase of a modular software”, International Journal of Systems Science, 2007, 38(6), 493-499.

[9] Kapur P. K., Goswami D. N., Bardhan A., Singh O., “Flexible software reliability growth model with testing effort dependent learning process”, Appl. Math. Model, 2008, 32, 298–307.

[10] Kapur P. K., Gupta Anu, Shatnawi Omar, Yadavalli V. S. S., “Testing Effort Control Using Flexible Software Reliability Growth Model With Change Point” International Journal of Performability Engineering- Special issue on Dependability of Software/ Computing Systems, 2006, 2(3,), 245-262.

[11] Kapur P. K., Jha P., Bardhan A., “Dynamic programming approach to testing resource allocation problem for modular software”, Ratio Mathematica, Journal of Applied Mathematics, 2003, 14, 27-40.

[12] Kapur P. K., Jha P., Bardhan A., “Optimal allocation of testing resource for a modular software”, Asia Pacific Journal of Operational Research, 2004, 21(3), 333-354

[13] Kapur, P. K., Garg, R. B. and Kumar, S., Contributions to Hardware and Software Reliability, World Scientific: Singapore 1999.

[14] Khan M., Ahmad N., and Rafi L.,” Optimal Testing Resource Allocation for Modular Software Based on a Software Reliability Growth Model: A Dynamic Programming Approach”, Proceedings of the International Conference on Computer Science and Software Engineering, 2008.

[15] Lin C. T., Huang C. Y., “Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models, Journal of Systems and Software, 2008, 81, 1025–1038.

[16] Lyu M. R., Handbook of Software Reliability Engineering, McGraw-Hill, New York, 1996.

[17] Musa J. D. “A Theory of Software Reliability and its Application”, IEEE Transaction Software Engineering, 1975, SE-1, 312-327.

[18] Musa J. D. and Okumoto K. “A Logarithmic Poisson Execution Time Model for Software Reliability Measurement”, in Proceedings of 7th International Conference on Software Engineering”, 1984, 230-238.

[19] Musa J. D., Iannino A., Okumoto K., Software Reliability: Measurement, Prediction, Application, McGraw-Hill, New York, 1987.

[20] Obha, M., “Software reliability analysis models”,IBM Journal of Research and Development, 1984, 28, 428–443.

[21] Ohetera H. and Yamada S., “Optimal allocation and control problems for software testing resources”, IEEE Transactions on Reliability, 1990, 39 (2), 171-176.

[22] Pham H., Software Reliability, Springer-Verlag, Singapore, 2000.

[23] Shyur H. J., “A stochastic software reliability model with imperfect-debugging and change-point, Journal of Systems and Software, 2003, 66, 135–141.

[24] [Trachtenberg M., “A General Theory of Software-Reliability Modeling”, IEEE Transaction on Reliability, 1990, 39, 92-96.

[25] Wang Z. and Wang J. “Parameter Estimation Of Some NHPP Software Reliability Models With Change-Point” Communications in Statistics- Simulation and Computation, 2005, 34, 121-134.

[26] Yamada S., Ohtera H., and Narihisa H., “Software Reliability Growth Models with Testing-Effort”, IEEE Transactions on Reliability, 1986, R-35(1), 19-23.

[27] Yamada S. Ichimori T. Nishiwaki M., “Optimal allocation policies for testing-resource based on a software reliability growth model”, Mathematical and Computer Modelling, 1995, 22(10-12), 295-301.

[28] Zhao, M., “Change-point problems in software and hardware reliability” Communications in Statistics—Theory and Methods, 1993, 2 (3), 757–768.

[29] Kapur, P K, Anu G Aggarwal and Gurjeet Kaur, “Optimal Testing Resource Allocation for Modular Software Considering Cost, Testing Effort and Reliability using Genetic Algorithm”. International Journal of Reliability, Quality and Safety Engineering. 2010, 16(6): 495-508..

[30] Sai Sindhuja K., Yuva Krishna A., Four Problem Minimization in Software Testing Resource Allocation, IOSR Journal of Computer Engineering, 2016, 18(6), 109-116.