Determination of the Optimal Allocation of Testing Resource for Modular Software Reliability Growth Using LINGO

N. Ahmad[1*], M. G. M. Khan[2]
[1] University Department of Statistics and Computer Applications, T. M. Bhagalpur University, Bhagalpur, India.
[2] School of Computing Information and Mathematical Sciences, the University of the South pacific, Suva, Fiji Island.

* Corresponding author. Tel.: +91-641-2501290; email: nesar_bgp@yahoo.co.in
Manuscript submitted January 15, 2016; accepted March 20, 2016.
doi: 10.17706/jsw.11.7.664-676

Abstract: It is quite natural to produce reliable software systems since the breakdown of the computer systems, which is caused by software faults, results in a tremendous loss and damage for social life. Hence, software reliability is a key factor in software development process. Testing phase of software begins with module testing whereby, modules are tested independently to remove substantial amount of faults within a specified testing resource. Therefore, in this paper four optimization problems are discussed for optimal allocation of testing resources for a modular software system. These optimization problems are formulated as nonlinear programming problems (NLPP), which incorporates log-logistic testing-effort function into inflection S-shaped software reliability growth model based on a non-homogeneous Poisson process. The NLPPs are solved using LINGO, a user-friendly software package for optimization. Finally, numerical example is given to demonstrate and to validate the performance of proposed strategies. In addition, the results of the strategies proposed in this paper are compared with [1]. It is revealed that the proposed strategies yield a gain in efficiency.

Key words: Software reliability growth model, resource allocation problem, nonlinear programming problem, dynamic programming technique, inflection S-shaped models.

1. Introduction

This is a century of computers in which our daily activities rely mostly on the use of the computer systems. Computers and software together has changed the way we live, trade, explore and enjoy life for the better. It plays a vital role in the modern life. Software is a functioning element embedded in computers. The computers are being almost used everywhere, like medical fields, businesses, chemical labs, air traffic control towers, ships, space ships, home appliances, communication, manufacture and many more. Hence, relying on machines and its reliability has been increased. Software reliability becomes a crucial feature of the computer systems. Therefore, the breakdown in the system could result in the fiscal, possessions, and human loss. A malfunctioning pacemaker in a heart patient or a Mars path finder that has lost contact due to a software bug or a hacker taking advantage of a bug in financial system to withdraw away cash electronically portrays the problem.

A computer system consists of two major components: hardware and software. Software is a created by man therefore, a high degree of reliability cannot be guaranteed. Thus, researches have been conducted

over the past decades to study the software reliability and many software reliability growth models (SRGM) have been proposed [2], [3]. Software reliability is defined as the probability of failure-free operation of a computer program for a specified time in a specified environment [3]. Software is considered to have performed a successful operation, when it functions completely as expected, without any failure. Hence, software reliability is a key factor in software development process.

A software development process consists of four phases: specification, design, coding and testing [4]. It is the final phase where the software is tested to detect and correct software faults in the following three consecutive stages: module testing, integration testing, and system testing. Software consists of several modules that are tested independently during the module testing, and the interfaces among these modules are tested in the integration testing. Lastly, the complete software system of which modules are interconnected is tested under the stimulated user environment in the system testing.

Moreover, all the testing activities of different modules should be completed approximately 40% - 50% of the total amount of software development resources [1]. Hence project managers should be able to effectively allocate the testing resources among all the modules to develop reliable software. Many researchers have addressed the optimal resource allocation problem over the years, including [1], [5-14]. They also discuss a management problem to achieve a reliable software system efficiently during module testing by applying a software reliability growth model. In order to develop a quality and reliable software system, it is very important for the manager to allocate the specified amount of testing-resource expenditures for module testing in terms of time-dependent behavior of the cumulative number of faults detected during the testing phase.

The reliability of software is determined upon the module testing, where significant number of faults from the modules is removed. Therefore, to maximize the number of faults to be removed is proportional to the maximization of the software reliability. Large software consists of modules that are developed independently and can be different from each other with respect to size and the number of faults. Hence, each of the modules can be tested through different SRGMs. In this paper, we propose four optimization problems to determine the optimum allocation of testing resource to the modules that maximize the total number of faults expected to be removed satisfying the limited resources. An inflection S- shaped SRGM with Log-logistic testing effort has been discussed to represent the removal of the fault for the four optimization problems. The optimization problem (P1) modified as (P2), which maximizes the number of faults expected to be removed from the modules with the available resource allocation. If the management seeks certain percentage of faults to be removed from each module during the testing, then a constraint has been added to the first optimization problem (P2) that gives the second optimization problem (P3). Sometimes, management seeks some reliability level of the customers to be achieved so that the results are more acceptable to the customers. Adopting this condition in (P2), the third optimization problem (P4) is obtained. However, if the management would like to satisfy both the additional requirements described in (P3) and (P4), the fourth optimization problem (P5) may be solved to achieve a certain percentage of fault to be removed as well as a desired level of reliability of the customer to be achieved during the module testing. All the four problems are treated as nonlinear programming problem (NLPP) and solved using the program coded for LINGO software. The methodology discussed in the paper has been illustrated through numerical examples.

2. Software Reliability Growth Model for Modules

2.1. Notations

N: Number of modules in the Software (>1)

a (a_i): Expected number of faults in the software (ith module; i = 1, 2,..., N)

b (b_i): Proportionality constant (for the ith module)

r (r_i): Fraction of independent faults (for the ith module)

$x(t)$ ($x_i(t)$): Current testing effort expenditure at testing time t,

$$X(t) = \int_0^t x(w)dw, \ (X_i(t) = \int_0^t x_i(w)dw)$$

X_i, Z: The amount of testing resources to be allocated to the ith module and total testing resource available

$m(t)$ ($m_i(t)$): Number of faults removed in $(0, t]$ (for the $i = 1, 2,..., N$)

T: Total testing time

X_i^*: Optimal value of X_i ($i = 1, 2,..., N$)

l_i: desired level of reliability level (i.e. number of faults desired to be removed from the ith module)

2.2. Classification

NHPP: Non-homogeneous *Poisson* process

SRGM: Software reliability growth models

NLPP: Nonlinear *programming* problem

Testing-Resource: resource expenditure spent on software testing, e.g., man-power, CPU hours, executed test-cases

Fault: A *cause* of a failure which is unacceptable departure from nominal program operation

Module testing: *Verification* a single software module in a simulated software environment where the module is isolated from all other modules.

A software reliability growth model (SRGM) is a mathematical expression of the software fault occurrence and the removal process. A software reliability growth model explains the time dependent behavior of fault removal. In this paper a resource allocation problem is discussed using such a SRGM for the modules. Numerous SRGMs have been developed during the last three decades and they can provide very useful information *about* how to improve reliability [2], [3], [15].

Among these models, a non homogeneous Poisson process (NHPP) as the stochastic process has been used by Goel-Okumoto model to describe the fault process. Later, the concept of testing-effort was incorporated in an NHPP by [16-18] to the modified exponential Goel-Okumoto model to get a better description of the fault detection phenomenon.

2.3. Model Assumptions

Some of the assumptions that have been adopted for software reliability growth modeling are stated below [17]-[23]:

1) The software system is subject to failures at random times caused by faults remaining in the system.
2) Each time a failure occurs, the fault causing that failure is immediately removed and no new faults are introduced.
3) Fault removal phenomenon in software testing is modeled by non homogeneous Poisson process.
4) The expected number of faults detected in the time interval $(t, t + \Delta t)$ to the current testing-effort expenditures is proportional to the mean number of remaining faults in the system.

One of the SRGM, proposed by [24] is the inflection S-shaped software reliability growth model, which has been developed to analyze the software failure detection process and its underlying reasons by modifying the logistic curve model. The fault removal rate increases with time and assumed the presences of two types of faults in the software [12]. The distinctive assumption of the model can be summarized as follows [12]:

1) The fault detection rate is proportional to the current fault content in the software and the proportionality increases linearly with each additional fault removal.
2) Faults present in the software are of two types: mutually independent and mutually dependent.

The mutually independent faults lie on different execution paths, and mutually dependent faults lie on the same execution path. Thus, the second type of faults is detectable if and only if faults of the first type have been removed. The model can be summarized by the differential equation.

$$\frac{d}{dt}m(t) = \phi(t)(a - m(t)), \tag{1}$$

where

$$\phi(t) = b\left[r + (1-r)\frac{m(t)}{a}\right],$$

and r is called the inflection parameter and represents the proportion of independent faults present in the software.

Solving (1) with the initial condition that at $t=0,\ m(t)=0$, the following is obtained

$$m(t) = \frac{a\left(1-e^{-bt}\right)}{1+[(1-r)/r]e^{-bt}}. \tag{2}$$

The SRGM in (2) can describe inflection S-shaped growth curves, depending upon the value of r. The Ohba model describes the fault identification with respect to time. To incorporate the effect of the testing-effort, assumption (6) can be modified as:

1) The fault detection rate with respect to testing-effort ($x(t)$) expenditure is proportional linearly with each additional fault removal

Under the above assumptions the following differential equation may easily be written as:

$$\frac{(d/dt)m(t)}{x(t)} = \phi(t)(a - m(t)). \tag{3}$$

Solving (3) with the initial condition at $t=0,\ X(t)=0,\ m(t)=0$, the following is obtained:

$$m(t) = \frac{a\left[1 - e^{-bX(t)}\right]}{1 + [(1-r)/r]e^{-bX(t)}}. \tag{4}$$

As the reliability of software is a ratio of cumulative number of detected fault at time t to the expected number of initial faults and is given by [20]

$$R(t) = \frac{m(t)}{a} = \frac{\left[1 - e^{-bX(t)}\right]}{1 + [(1-r)/r]e^{-bX(t)}}. \tag{5}$$

To describe the behavior of testing effort, log-logistic function has been used [25]-[27].

The Cumulative testing-effort expenditure consumed in $(0, t]$ is depicted in the following:

$$X(t) = \alpha[1-\{1+(\beta t)^{\theta}\}^{-1}]$$
$$= \alpha[(\beta t)^{\theta}/(1+(\beta t)^{\theta})], \alpha>0,\ \beta>0, \theta>0, t>0. \tag{6}$$

and the current testing-effort consumed at testing time t is

$$x(t) = [\alpha\beta\theta(\beta t)^{\theta-1}]/[1+(\beta t)^{\theta}]^2, \quad t>0, \alpha>0, \beta>0, \theta>0, \tag{7}$$

where α is the total amount of testing-effort expenditures, β is the scale parameter, and θ are shape parameters.

2.4. Estimation of Parameters

Using the method of Least square estimation, the parameters α, β, and θ in Log-logistic testing-effort function can be estimated. These parameters are determined for *n* observed data pairs in the form (t_k, X_k) $(k=1,2,....,n; 0<t_1<t_2<....<t_n)$, where X_k is the cumulative testing-effort consumed in time $(0,t_k]$. The least square estimators α, β, and θ can be estimated by minimizing the following:

$$\text{Minimize} \quad \sum_{i=1}^{n}\left[X_i - \hat{X}\right],$$

Subject to $\hat{X} = X_n$ (i. e. the estimated value of testing effort is equal to the actual value).

Once the estimates of α, β, and θ are known, the parameters $a, b,$ and r of the SRGMs for the module can be estimated through maximum likelihood estimation method. The estimators of $a, b,$ and r are determined for the n observed data pairs in the form (t_k, y_k) $(k=1,2,...,n;\ 0<t_1<t_2<...<t_n)$ where y_k is the cumulative number of software faults detected up to time t_k or $(0,t_k]$.

The likelihood function for SRGM (4) is given by:

$$L(a,b,r/(y,X) = \prod_{k=1}^{n}\frac{[m(t_k)-m(t_{k-1})]^{(y_k-y_{k-1})}}{(y_k-y_{k-1})!}.e^{-[m(t_k)-m(t_{k-1})]},$$

where $t_0=0$ and $y_0=0$.

3. Testing-Resource Allocation Problem

In module testing two kinds of testing-resource allocation problems are considered to make the best use of a specified total testing-resource allocation. It has to be allocated appropriately to each software module, which is tested independently and simultaneously, so that the maximum reliability is achieved. In this section, we will consider a resource allocation problems based on an SRGM with log-logistic testing-effort function during software testing phase.

Assumptions [1], [28]:

1) The software system is composed of *N* independent modules, and the software modules are tested independently. The number of software faults remaining in each module can be estimated by the SRGM with log-logistic testing-effort.
2) The total amount of testing-resource expenditures for the module testing is specified.
3) The manager has to allocate the specified total testing-resource expenditures to each software module so that the software faults to be removed in the system may be maximized. The desired software reliability level is achieved.

The length of module testing is often fixed when scheduling is done for the whole testing phase. Therefore, limited resources are available, which need to be allocated wisely. The first optimization problem in view of the model from Section 2 with *N* modules can be formulated as an NLPP as given below:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{N} m_i(t) = \sum_{i=1}^{N} \frac{a_i\,(1 - e^{-b_i X_i(t)})}{1 + [(1-r_i)/r_i]e^{-b_i X_i(t)}}, \\ \text{Subject to} \quad & \sum_{i=1}^{N} X_i \le Z, \\ & X_i \ge 0, \quad i=1,\ldots,N, \end{aligned} \tag{P1}$$

where, m_i is the number of faults expected to be removed from the ith module with testing effort X_i and Z is the total resource available for the allocation to the modules. The NLPP (P1) considers the resource constraint only.

The total fault content in the software is calculated from the estimates of parameters of SRGMs for modules. With the availability of the resources, module testing targets at removing maximum numbers of faults. The important concern here is to how much to test. Therefore, the software testing time (T) is almost fixed. Let X_i be the testing effort that has to be spent on the ith module during testing time T, the mean value function can be then written as:

$$m_i(X_i) = \frac{a_i\,(1 - e^{-b_i X_i})}{1 + [(1-r_i)/r_i]e^{-b_i X_i}}, \qquad i = 1,\ldots,N\,.$$

where $X_i = \alpha_i(\beta_i t)^{\theta_i}/(1 + (\beta_i t)^{\theta_i})$

Hence, the problem (P1) can be restated as follows

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{N} m_i(X_i) = \sum_{i=1}^{N} \frac{a_i\,(1 - e^{-b_i X_i})}{1 + \delta_i\, e^{-b_i X_i}}, \\ \text{Subject to} \quad & \sum_{i=1}^{N} X_i \le Z, \\ & X_i \ge 0, \quad i=1,\ldots,N, \end{aligned} \tag{P2}$$

where $\delta_i = [(1-r_i)/r_i]$.

Sometimes some of the modules may not get any resources in the above allocation procedure, to which the management may not agree where one or more modules are not tested any further. However, during module testing it is expected that each module is tested satisfactorily so that a certain percentage of the fault content is desired to be removed from each module of the software. Consequently, the above NLPP (P2) needs to be modified to maximize the removal of the faults in the software under resource constraint and the desired percentage of faults to be removed from each of the modules in the software.

Let p_i be the minimum proportion of faults to be removed in ith module. Then

$$m_i = \frac{a_i\,(1 - e^{-b_i X_i})}{1 + \delta_i\, e^{-b_i X_i}} \ge p_i\, a_i\,,$$

Or

$$X_i \ge -\frac{1}{b_i} \ln\left(\frac{1 - p_i}{1 + p_i\, \delta_i}\right). \tag{8}$$

Therefore, adding (8) to NLPP (P2), the resulting resource allocation problem can be expressed as follows:

$$\text{Maximize} \quad \sum_{i=1}^{N} \frac{a_i (1 - e^{-b_i X_i})}{1 + \delta_i e^{-b_i X_i}}$$

$$\text{Subject to} \quad X_i \geq -\frac{1}{b_i} \ln\left(\frac{1 - p_i}{1 + p_i \delta_i}\right),$$

$$\sum_{i=1}^{N} X_i \leq Z,$$

$$X_i \geq 0, \quad i=1,\ldots,N. \tag{P3}$$

The management may wish to consider a certain level of reliability that a customer might desire and the resource must be allocated in such a way that the desired reliability level be achieved. In such cases, reliability condition should be added to the problem (P2).

Let R_0 be the minimum level of reliability to be achieved for *i*th module. Then

$$R_i = \frac{(1 - e^{-b_i X_i})}{1 + \delta_i e^{-b_i X_i}} \geq R_0,$$

Or

$$X_i \geq -\frac{1}{b_i} \ln\left(\frac{1 - R_0}{1 + R_0 \delta_i}\right). \tag{9}$$

Therefore, the resulting resource allocation problem can be expressed as follows:

$$\text{Maximize} \quad \sum_{i=1}^{N} \frac{a_i (1 - e^{-b_i X_i})}{1 + \delta_i e^{-b_i X_i}},$$

$$\text{Subject to} \quad X_i \geq -\frac{1}{b_i} \ln\left(\frac{1 - R_0}{1 + R_0 \delta_i}\right),$$

$$\sum_{i=1}^{N} X_i \leq Z,$$

$$X_i \geq 0, \quad i=1,\ldots,N. \tag{P4}$$

In another situation, the management would like to achieve a certain percentage of faults to be removed as well as a desired level of reliability of the customer to be achieved during the module testing within the available resources. Then, adding both the constraints (8) and (9) to (P2), the problem of optimum resource allocation can be expressed as the following NLPP:

$$\text{Maximize} \quad \sum_{i=1}^{N} \frac{a_i (1 - e^{-b_i X_i})}{1 + \delta_i e^{-b_i X_i}},$$

$$\text{Subject to} \quad X_i \geq -\frac{1}{b_i} \ln\left(\frac{1 - p_i}{1 + p_i \delta_i}\right),$$

$$X_i \geq -\frac{1}{b_i} \ln\left(\frac{1 - R_0}{1 + R_0 \delta_i}\right),$$

$$\sum_{i=1}^{N} X_i \leq Z,$$

$$X_i \geq 0, \quad i=1,\ldots,N. \qquad \text{(P5)}$$

For the known parameters a_i, b_i, r_i and the known values of p_i, R_0 and Z all the four problems (P2) to (P5) can be solved for optimum allocation $X_i; (i=1,2,\ldots,N)$ by using LINGO Software.

4. Numerical Example

A numerical example for the optimal testing-resource allocation problem is presented to validate and to demonstrate the performance of the proposed strategy. We consider a software failure data, which is use by [1], where the software system consisted of 10 modules. The model parameters a_i, b_i, and r_i $(i=1,2,\ldots,10)$ for each module were estimated by using the MLE method and are summarized in Table 1.

Table 1. Estimated Value of SRGM Parameters

Module	a_i	b_i	r_i	δ_i
1	63	0.00005332	1.002586957	-0.03545193
2	13	0.00025230	1.012080139	-0.01193595
3	6	0.00052620	1.038505577	-0.03707787
4	51	0.00005169	1.003830667	-0. 08999844
5	15	0.00017070	0.991108631	0.00897113
6	39	0.00005723	1.002393431	-0.00238772
7	21	0.00009938	1.001659538	-0.00165679
8	9	0.00017430	1.029696882	-0.02884041
9	23	0.00005057	1.003141885	-0.00313204
10	11	0.00008782	1.053658375	-0.05092578

Table 2. Optimal Resource Allocation

Module	X_i^*	m_i	% of fault removed	% of fault remained
1	25195.75	47.0	74.6	25.4
2	5276.95	9.6	73.8	26.2
3	2436.04	4.4	73.0	27.0
4	21009.54	34.8	68.3	31.7
5	6399.41	9.9	66.3	33.7
6	16627.52	24.0	61.4	38.6
7	8902.02	12.3	58.8	41.2
8	3454.27	4.1	46.0	54.0
9	5968.99	6.0	26.1	73.9
10	1729.52	1.6	14.7	85.3
Total	97000	153.8	61.3	38.7

Let the available total amount of testing resource Z be 97,000 man-hours. Then, the problem of determining the optimum resource allocation ($X_i^*; i=1,2,\ldots,10$) that maximizes the number of faults to be removed, which is stated in NLPP (P2), is given by

$$\text{Maximize}\left\{\begin{array}{l}\dfrac{63\left(1-e^{-0.00005332X_1}\right)}{1-0.03545193X_1}+\dfrac{13\left(1-e^{-0.00025230X_2}\right)}{1-0.01193595X_2}+\dfrac{6\left(1-e^{-0.00052620X_3}\right)}{1-0.03707787X_3}\\ +\dfrac{51\left(1-e^{-0.00005169X_4}\right)}{1-0.08999844X_4}+\dfrac{15\left(1-e^{-0.00017070X_5}\right)}{1+0.00897113X_5}+\dfrac{39\left(1-e^{-0.00005723X_6}\right)}{1-0.00238772X_6}\\ +\dfrac{21\left(1-e^{-0.00009938X_7}\right)}{1-0.00165679X_7}+\dfrac{9\left(1-e^{-0.00017430X_8}\right)}{1-0.02884041X_8}+\dfrac{23\left(1-e^{-0.00005057X_9}\right)}{1-0.00313204X_9}+\dfrac{11\left(1-e^{-0.00008782X_{10}}\right)}{1-0.05092578X_6}\end{array}\right\},$$

$$\begin{array}{ll}\text{Subject to} & X_1+X_2+X_3+X_4+X_5+X_6+X_7+X_8+X_9+X_{10}\le 97000,\\ \text{and} & X_1,X_2,X_3,X_4,X_5,X_6,X_7,X_8,X_9,X_{10}\ge 0.\end{array}$$

The above problem can be solved by using a program coded in LINGO and is given in Appendix. The optimal allocation of resources X_i^* for the modules are obtained and are summarized in the Table 2 along with the corresponding expected number of fault removed, percentages of fault removed, and the percentage of fault remained for each module.

The total number of faults removed from the software through this allocation is 153.8, that is, 61.3 % of the fault content being removed. It can be noticed that the percentage of the remaining faults in the last three modules (i.e. Module 8, 9 and 10) is still very high, which may not satisfy the management. Suppose the management desires to set the restriction that the percentage of initial faults from i th module with a certain reliability level R_0 = 0.5 should be removed. Thus, adding the restriction (9), the required problem of determining optimum resource allocation as stated in NLPP (P4) is given as:

$$\text{Maximize}\left\{\begin{array}{l}\dfrac{63\left(1-e^{-0.00005332X_1}\right)}{1-0.03545193X_1}+\dfrac{13\left(1-e^{-0.00025230X_2}\right)}{1-0.01193595X_2}+\dfrac{6\left(1-e^{-0.00052620X_3}\right)}{1-0.03707787X_3}\\ +\dfrac{51\left(1-e^{-0.00005169X_4}\right)}{1-0.08999844X_4}+\dfrac{15\left(1-e^{-0.00017070X_5}\right)}{1+0.00897113X_5}+\dfrac{39\left(1-e^{-0.00005723X_6}\right)}{1-0.00238772X_6}\\ +\dfrac{21\left(1-e^{-0.00009938X_7}\right)}{1-0.00165679X_7}+\dfrac{9\left(1-e^{-0.00017430X_8}\right)}{1-0.02884041X_8}+\dfrac{23\left(1-e^{-0.00005057X_9}\right)}{1-0.00313204X_9}+\dfrac{11\left(1-e^{-0.00008782X_{10}}\right)}{1-0.05092578X_6}\end{array}\right\},$$

$$\begin{array}{ll}\text{Subject to} & X_1\ge 12664.33,\\ & X_2\ge 2723.59,\\ & X_3\ge 1281.71,\\ & X_4\ge 12518.94,\\ & X_5\ge 4086.83;\\ & X_6\ge 12090.73,\\ & X_7\ge 6966.38,\\ & X_8\ge 3893.41,\\ & X_9\ge 13675.70,\\ & X_{10}\ge 7599.12,\\ & X_1+X_2+X_3+X_4+X_5+X_6+X_7+X_8+X_9+X_{10}\le 97000,\\ \text{and} & X_1,X_2,X_3,X_4,X_5,X_6,X_7,X_8,X_9,X_{10}\ge 0.\end{array}$$

Solving the above NLPP using LINGO, the optimal allocation of resources X_i^* for the modules are obtained and the results are summarized in Table 3.

Table 3. Optimal Resource Allocation with $R_0 = 0.5$

Module	R_0	$-\frac{1}{b_i}\ln\left(\frac{1-R_0}{1+R_0\delta_i}\right)$	X_i^*	m_i	% of fault removed	% of fault remained
1	0.5	12664.33	21791.29	43.8	69.5	30.5
2	0.5	2723.59	4547.78	8.9	68.5	31.5
3	0.5	1281.71	2091.85	4.1	67.6	32.4
4	0.5	12518.94	17655.90	31.7	62.1	37.9
5	0.5	4086.83	5306.98	8.9	59.4	40.6
6	0.5	12090.73	13397.18	20.9	53.6	46.4
7	0.5	6966.38	7040.79	10.6	50.4	49.6
8	0.5	3893.41	3893.41	4.5	50.0	50.0
9	0.5	13675.70	13675.70	11.5	50.0	50.0
10	0.5	7599.12	7599.12	5.5	50.0	50.0
Total			97000	150.3	59.9	40.1

The results in Table 3 shows that a total of the number of faults removed from the software through this allocation is 150.3 and will ensure the management's desire to remove a percentage of initial faults from each module with the reliability level $R_0 = 0.5$.

Further allocation of testing resource can also be obtained whereby the developer desires to remove a certain proportion of faults form each module as stated in problem (P3), and also wishes simultaneously to remove a certain proportion of faults and to achieve a level of reliability stated in NLPP problem (P5).

5. Comparison and Discussion

In this section, an investigation is carried out to compare the strategies of optimum resource allocation proposed in this article to that of Yamada *et al*. [1].

Reference [1] also considered two problems of determining the optimum resource allocation, which minimizes the number of remaining faults given a fixed amount of testing-effort and a reliability goal to be achieved. Since the minimization of the number of remaining faults is the same as the maximization of the number of faults removed, the formulation of Yamada *et al*. [1] for the problem of resource allocation based on the exponential type SRGM is an equivalent to the proposed formulation stated in (P2) for $r_i = 1;\ (i = 1,2,...,10)$.

To compare the proposed allocation with that of Yamada et al. [1], the results obtained for optimum, X_i^* are presented in Table 4.

Table 4 shows that the total number of faults removed by the Yamada et al. [1] and the proposed models are 152.4 (60.7%) and 153.8 (61.3%), respectively. The results reveal that the overall relative gain (RG) due to use of the proposed strategy over the Yamada el al. is 40.25%, where relative gain (RG) is define as $(m_i^*\text{proposed} - m_i^*\text{Yamada})/m_i^*\text{Yamada}$.

Table 4. Optimum Allocation of Resources X_i^* and Number of Faults m_i^* Removed Using Yamada *et al*. and Proposed

Module	Yamada *et al*.[1]			Proposed			RG in %
	X_i^*	m_i^*	% of fault removed	X_i^*	m_i^*	% of fault removed	
1	25435.2	46.8	74.3	21791.3	47.0	74.6	0.42
2	5280.7	9.6	73.8	4547.8	9.6	73.8	-0.03
3	2459.5	4.4	73.3	2091.8	4.4	73.0	-0.46

4	21548.6	34.3	67.3	17655.9	34.8	68.3	1.58
5	6354.5	9.9	66.0	5307.0	9.9	66.3	0.39
6	16554.3	23.9	61.3	13397.2	24.0	61.4	0.26
7	8857.2	12.3	58.6	7040.8	12.3	58.8	0.31
8	3412.3	4.1	45.6	3893.4	4.1	46.0	0.89
9	5845.6	5.9	25.7	13675.7	6.0	26.1	1.81
10	1251.9	1.2	10.9	7599.1	1.6	14.7	35.08
Total	97000	152.4	60.7	97000	153.8	61.3	40.25

6. Conclusion

This paper developed the strategies on the optimal testing resource allocation problems for modular software system, which are modeled by inflection S-shaped software reliability growth model based on a non-homogeneous Poisson process which incorporates log-logistic testing-effort function. The proposed strategies maximized the total number of faults removed with a fixed amount of testing effort, a certain percentage of faults to be removed, and a desired reliability is to be achieved. The problem is formulated as a NLPP and is solved using a user friendly software LINGO. We discussed numerical examples to show the applications and impacts of proposed strategies. Finally, we compared the experimental results with the results of Yamada *et al*. [1]. Based on the experimental results, we conclude that the overall gain in percentage of the proposed strategy is more than that of Yamada *et al.* [1] and the proposed strategies may be helpful to software project managers for making the best decisions in developing quality and reliable software.

Future research may include extending the present study to delay S-shaped software reliability growth model with testing effort functions.

Appendix

```
max=63*(1-@exp(-0.00005332*x1))/(1-0.03545193*@exp(-0.00005332*x1)) +
13*(1-@exp(-0.00025230*x2))/(1-0.01193595*@exp(-0.00025230*x2)) +
6*(1-@exp(-0.00052620*x3))/(1-0.03707787*@exp(-0.00052620*x3)) +
51*(1-@exp(-0.00005169*x4))/(1-0.08999844*@exp(-0.00005169*x4)) +
15*(1-@exp(-0.00017070*x5))/(1+0.008971134*@exp(-0.00017070*x5)) +
39*(1-@exp(-0.00005723*x6))/(1-0.00238772*@exp(-0.00005723*x6)) +
21*(1-@exp(-0.00009938*x7))/(1-0.00165679*@exp(-0.00009938*x7)) +
9*(1-@exp(-0.00017430*x8))/(1-0.02884041*@exp(-0.00017430*x8)) +
23*(1-@exp(-0.00005057*x9))/(1-0.00313204*@exp(-0.00005057*x9)) +
11*(1-@exp(-0.00008782*x10))/(1-0.05092578*@exp(-0.00008782*x10)) ;
X1+x2+x3+x4+x5+x6+x7+x8+x9+x10<=97000;
```

Acknowledgment

The authors would like to thank the Editor in Chief and the referee for their valuable comments and suggestions which led to improve the draft manuscript.

References

[1] Yamada, S., Ichimori, T., & Nishiwaki, M. (1995). Optimal allocation policies for testing-resource based on a software reliability growth model. *International Journal of Mathematical and Computer Modeling, 22 (10-12)*, 295-301.

[2] Lyu, M. R. (1996). *Handbook of Software Reliability Engineering*. New York: McGraw Hill.

[3] Musa, J. D., Iannino, A., & Okumoto, K. (1987). *Software Reliability: Measurement, Prediction and Application*. New York: McGraw-Hill.

[4] Myers, G. J. (1976). *Software Reliability: Principles and Practices*. New York: John Wiley & Sons.

[5] Ahmad, N., Khan, M. G. M., & Islam, S. F. (2012). Optimal allocation of testing resource for modular software based on testing-effort dependent software reliability growth. *Proceedings of the third International Conference on Computing Communication and Networking Technologies (ICCCNT-2012)* (pp. 1 - 7). Coimbatore, Tamilnadu, India: New York: IEEE Computer Society.

[6] Khan, M. G. M., Ahmad, N., and Rafi, L. S. (2008). 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 (CSSE-2008)* (pp. 759-762). Wuhan, China: *I*EEE Computer Society.

[7] Khan, M. G. M., Ahmad, N., & Rafi, L. S. (2016). Determining the optimal allocation of testing resource for modular software system using dynamic programming. *Communications in Statistics – Theory and Methods*, *45(3)*, 670-694.

[8] Huang, C. Y., Lo, J. H., Kuo, S. Y., & Lyu, M. R. (2002). Optimal resource allocation and reliability analysis for component-based software applications. *Proceedings of the 26th IEEE Annual International Computer Software and Applications Conference (COMPSAC'2002)* (pp. 7-12). Los Alamito, CA: IEEE Computer Society.

[9] Huang, C. Y., Lo, J. H., Kuo, S. Y., & Lyu, M. R. (2002). Optimal allocation of testing resources for modular software systems. *Proceedings of the Thirteenth IEEE International Symposium on Software Reliability Engineering* (pp. 129-138). Annapolis, Maryland: IEEE Computer Society.

[10] Huang, C. Y., Lo, J. H., Kuo, S. Y., & Lyu, M. R. (2004). Optimal allocation of testing resource considering cost, reliability, and testing-effort. *Tenth IEEE Pacific Rim International Symposium on Dependable Computing* (pp. 103-112). Papeete, Tahiti, French Polynesia: IEEE Computer Society.

[11] Huang, C. Y., & Lyu, M. R. (2005). Optimal testing resource allocation and sensitivity analysis in software development. *IEEE Transaction on Reliability, 54(4*), 592-603.

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

[13] Lyu, M. R., Rangarajan, S., & Van Moorsel, A. P. A. (2002). Optimal allocation of test resources for software reliability growth modeling in Software Development. *IEEE Transactions on Reliability, 51(2)*, 183-192.

[14] Yamada, S., & Ohtera, H. (1990). Optimal allocation and control problems for software-testing resources. *IEEE Transactions on Reliability, 39(2)*, 171-176.

[15] Xie, M. (1991). *Software Reliability Modeling*. Singapore: World Scientific.

[16] Bokhari, M. U., & Ahmad, N. (2014). Incorporating burr type XII testing-efforts into software reliability growth modeling and actual data analysis with applications. *Journal of Software, 9(6)*, 1389-1400.

[17] Yamada, S., Ohtera, H., & Narihisa, H. (1986). Software reliability growth models with testing-effort. *IEEE* Transactions *on Reliability, 35(1)*, 19-23.

[18] Yamada, S., Hishitani, J., & Osaki, S. (1993). Software reliability growth with a weibull test-effort: A model and application. *IEEE Transactions on Reliability, 42(1)*, 100-105.

[19] Ahmad, N., Khan, M. G. M., & Rafi, L. S. (2010). A study of testing-effort dependent inflection s-shaped software reliability growth models with imperfect debugging. *International Journal of Quality and Reliability Management, 27(1)*, 89-110.

[20] Huang, C. Y., & Lo, J. H. (2006). Optimal resource allocation for cost and reliability of modular software systems in the testing phase. *Journal of Systems and Software, 79(5)*, 653-664.

[21] Kapur P. K., Garg R. B., & Kumar S. (1999). *Contributions to Hardware and Software Reliability*. Singapore: World Scientific.

[22] Kuo, S. Y., Huang, C. Y., & Lyu, M. R. (2001). Framework for modeling software reliability, using various testing-efforts and fault-detection Rates. *IEEE Transitions on Reliability, 501(3)*, 310-320.

[23] Yamada, S., & Osaki, S. (1985). Software reliability growth modeling: Models and applications. *IEEE* Transactions *on Software Engineering, 11(12)*, 1431-1437

[24] Obha, M. (984). Software reliability analysis model. *IBM Journal Res. Development,28 (4),* 428-443.

[25] Ahmad, N., Khan, M. G. M., & Rafi, L. S. (2011). Analysis of an inflection s-shaped software reliability model considering log-logistic testing-effort and imperfect debugging. *International Journal of Computer Science and Network Security, 11 (1)*, 161 – 171.

[26] Ahmad, N., Khan, M. G. M., & Rafi, L. S. (2010). Software reliability modeling incorporating log-logistic testing-effort with imperfect debugging. *Proceedings of the International Conference on Modeling, Optimization and Computing (ICMOC-2010)* (pp. 651-657). Durgapur, India: American Institute of Physics.

[27] Bokhari, M. U., & Ahmad, N. (2006). Analysis of a software reliability growth models: the case of log-logistic test-effort function. *Proceedings of the 17th IASTED International Conference on Modeling and Simulation (MS'2006)* (pp. 540-545). Montreal, Canada: Octa Press.

[28] Xie, M., & Yang, B. (2001). Optimal testing-time allocation for modular systems. *International Journal of Quality and Reliability Management, 18(8)*, 854-863.

Nesar Ahmad is an associate professor and the head of the Department in University Department of Statistics and Computer Applications at Tilka Manjhi Bhagalpur University, Bhagalpur, India. He received the B. Sc. degree in mathematics from Bihar University, India, in 1984 and the M.Sc., M.Phil., and Ph.D. degrees in statistics from Aligarh Muslim University, Aligarh, India, in 1987, 1990, and 1993, respectively.

He joined the University Department of Statistics and Computer Applications at Tilka Manjhi Bhagalpur University, Bhagalpur, India in 1996. From 1995 to 1996 he was a research associate of UGC/CSIR at Aligarh Muslim University, Aligarh. He has been a lecturer at the University of the South Pacific, Suva, Fiji Islands from January 2006 to December 2009. He has about 21 years of experience in teaching and research. He has published more than 75 papers in journals, and conferences in these areas. His research interests include life testing, statistical modeling, reliability analysis, software testing, software reliability engineering and optimization Techniques.

Dr. Ahmad is an executive member of Indian Society of Information Theory and Its Applications (ISITA) and life member of Indian Science Congress. He is in the editorial board of International Journal of Scientific and Statistical Computing (IJSSC) and Journal of Convergence Information Technology (JCIT).

M. G. M. Khan is an associate professor in statistics at the University of the South Pacific, Suva, Fiji. He has been in the teaching profession since 1996 when he first joined as a lecturer in statistics at Poona College, India. Dr. Khan graduated from the University of Calcutta in 1985. He obtained his M.Sc. from the Aligarh Muslim University in 1989. In 1992 he awarded M.Phil. degree in statistics followed by Ph.D. degree in 1996 from the Aligarh Muslim University.

Since 1998, he has been a lecturer at Mohamed Sathak College, Chennai, India. In 2001 he joined at the University of the South Pacific (USP) as a Lecturer in Statistics. Presently, he is also the acting head of the School of Computing, Information and Mathematical Sciences at USP. His research interest is in the fields of mathematical programming, sampling, statistical modeling, reliability analysis and software reliability analysis. He has published numerous articles in statistical journals. He has also presented several research papers in international conferences.

Dr. Khan is an elected member of the International Statistical Institute and also active member of International Association of Survey Statistician and American Statistical Association.