Optimal Scheduling Policy for a Multi-upgraded Software System under Fuzzy Environment

Adarsh Anand[1], Subhrata Das[1*], Mohini Agarwal[2] & V.S.S. Yadavalli[3]

[1]Department of Operational Research, University of Delhi, Delhi-110007, India
[2]Amity School of Business, Amity University Uttar Pradesh, Noida, UP-201313, India
[3]Department of Industrial and Systems Engineering,
University of Pretoria, Republic of South Africa
*E-mail: shus.das@gmail.com

Abstract. Many reliability growth models related to the concept of multi up-gradation have recently been proposed. This concept has helped software developers to develop a competitive edge over competitors by regularly introducing their software upgrades in the market at the optimal moment. The software reliability literature offers many different release-time policies, both under crisp and fuzzy environment. This paper presents a generic model of a multi-upgraded software system. The optimal scheduling policy for software under a fuzzy environment was determined. The proposed model was examined on a real-life failure data set of four software releases. The results obtained are encouraging.

Keywords: *fault removal; multi up-gradation; testing phase; fuzzy environment; release time.*

1 Introduction

Software is an integral part of all activities performed on digital platforms, giving it an influence on each and every sector of modern society. The applicability of software has led to enormous growth in the day-to-day workings of business processes [1]. This makes it critical to have software that offers consistent performance and at the same provides highly reliable execution. Ample of cases exist where software was not able to deliver the task assigned, leading to serious damage [2]. Seeing the high dependence of mankind on software and its functionality, it is the foremost responsibility of software developers to create software that is optimally effective and efficient. Thus, testers are hired whose primary task it is to debug the software and make it bug-free before it is provided to end-users. The testing team will try to fix any flaws in order to enhance the quality of the software. Reliability, which is the probability of the failure-free operation of a product for a specific period of time under predefined conditions, is an important aspect of quality [1,3]. Software firms acknowledge the importance of testing their products in the operational phase and provide regular upgrades of their software products in order to

Received January 31st, 2019, Revised August 30th, 2019, Accepted for publication October 28th, 2019.
 ISSN: 2337-5760, DOI: 10.5614/j.math.fund.sci.2019.51.3.6

maintain their competitive position. Software up-gradation is a survival strategy for firms that helps in maintaining and increasing market share by providing advanced versions with enhanced functionality to users [4]. To a certain extent, publishing several versions is beneficial for firms but under certain circumstances increasing a piece of code may result in the introduction of new faults making the software prone to failure. Firms should focus on adapting techniques capable of identifying and removing a maximum number of faults. It is evident that all potential faults cannot be removed in one go, thus leftover bugs from previous versions can result in software failure. Therefore, getting a clear picture of error removal should be a main priority for software developers [5].

Multi up-gradation cannot be discussed without discussing the release-time problem. The optimal time to release software is an other important concern for software developers, as releasing prematurely will degrade the quality and releasing too late impacts market entry, the cost of software testing and poses many other challenges as well. In the literature, several attempts have been made to formulate and compute the optimal time for launching software based on goals set by management [1]. This can be minimization of costs and other resources, or maximization of reliability, failure intensity and market share subject to various sets of constraints. These sets of constraints have been formulated both under crisp and fuzzy environments. The present study focused on the fuzzy release time decision (FRTD) problem due to its capability of modeling qualitative information of decision-makers and to directly work on fuzzy information [6].

A generic model is proposed to determine bugs that have been removed in each release based on the impact of leftover bugs from the previous release and faults generated as a result of up-gradation. Moreover, a fuzzy release time decision based optimization analysis was performed to determine the release time of software under the influence of the attributes cost and reliability.

This paper is organized as follows. Section 2 presents the background and an overview of the literature on software reliability growth modeling and the fuzzy release time decision problem. The generalized modeling framework for determining faults in respective software releases is discussed in Section 3. Cost modelling is discussed in Section 4.The optimal launch time problem is discussed in Section 5. Validation and a numerical illustration are presented in Sections 6 and 7. Lastly, the conclusion is given in Section 8.

2 Background and Literature Review

A large amount of progress has been made in the study of multi up-gradation. The most basic approach is to compute the reliability of a software product

based on the effect of leftover faults from all of its preceding releases [7,8]. It was soon realized that this approach is not very effective and the influence on the reliability of the software is not very significant [9]. Thus, practitioners started taking into account only the impact of the previous version on the reliability of the present release. Another observation that researchers have made was that the testing team is not always able to remove all faults completely, which can lead to an imperfect debugging environment when only considering faults from the previous release, as modeled by Aggarwal, *et al*. [10]. Some researchers have studied the concept of stochasticity in multi up-gradation [5]. Several experts have analyzed the debugging environment and uncertainty to study the severity of faults and the distribution environment to simulate market scenarios [10,11]. The issues encountered in both the testing and the operational phase were given equal weight [3,12].

The study by Singh, *et al*. [13] considered the effect of a Weibull type testing effort function (TEF) on the detection of faults in multiple versions of a software application. They further explain that TEF for each release follows an exponential and S-shaped curve. Kapur, *et al*. [14] developed an optimal cost model to find out the testing stop time for each release. They considered the accumulation of various costs incurred during the testing phase of new releases and the removal of leftover faults in the operational phase of the ongoing version and in the testing phase of the upcoming release. Singh, *et al*. [15] have proposed a unified approach to model successive releases of software under the influence of imperfect debugging. Numerous studies exist in the literature describing the fault removal phenomenon for multi up-gradation software versions and the associated cost analysis.

Several release-time policies have been proposed in the field of software reliability. The first ever release policy was given by Goel and Okumoto [16] to determine release time based on unconstrained cost minimization and reliability maximization as two separate sets of problems. Later, several release policies have been presented with different sets of optimization problems, for example: Yamada and Osaki [17] considered the joint impact of cost and reliability on the determination of optimal time to introduce the software on the market. Yun and Bai [18] focused on software with a random lifecycle and cost-based modeling for the determination of the optimal release time. Other researchers who have worked under same umbrella of cost analysis are: Huang [19]; Huang and Lyu [20]; Pham and Zhang [21]; Ramik [22]; Rommelfanger [23]; Tang and Wang [24];Ukimoto and Dohi [25] Xie and Yang [26]; and Yang, *et al*. [27].

All of the aforementioned scholars have worked on finding the optimal release time for a single software version. Very few attempts have been made to model and find the optimal software release time for multi up-graded software

systems. Something similar has been done by Kapur, *et al*. [14], who applied the concept of multi-attribute utility theory. However, these attempts were all done under a crisp environment. The field of cost optimization also deals with cases involving vagueness or fuzziness. This is an appealing topic in the field of software reliability, based on which a number of FRTD problems have been proposed. Models defined by utilizing fuzzy theory represent the conversion of subjective information and can help in incorporating vagueness in traditional crisp optimization problems. Kapur, *et al*. [28] formulated a fuzzy environment based on the cost minimization problem under constraint of failure intensity. Working from a similar starting-point, Jha, *et al*. [29] proposed a discrete SRGM-based FRTD problem by considering the cost and reliability objectives subject to budgetary constraints under a crisp scenario. Later, Jha, *et al*. [30] extended their work on the FRTD problem under the concept of imperfect debugging and an error generation software reliability growth model. Kumar, *et al*. [6] proposed an FRTD problem based on testing cost under the impact of warranty period. Recently, Kumar and Gupta [31] looked at an FRTD problem incorporating the effect of learning functions for fault detection and correction. However, none of these efforts were related to multi-upgraded software systems. As discussed above, and as per our available knowledge, this is the first attempt to model multi up-gradation under a fuzzy environment.

3 Modeling Methodology

The notation used in this paper is the following:

i	: number of versions$(i = 1,2,3,4)$
$m_i(t)$	: mean value function for fault removal
a_i	: total count of faults present in the software
b_i	: rate of fault debugging
β_i	: learning parameter

Supposing the case of multi-upgraded software and assuming that the first version of the software was initially launched at time point $'t_1 = 0'$ and the launch time of the succeeding i^{th} release will be t_i. The actual count of faults debugged in the i^{th} release of the software can be obtained as follows:

$$m_i(t) = Y_i(t)F_i(t) \tag{1}$$

Eq. (1) tries to elucidate the fraction $F_i(t)$of the total number of faults $Y_i(t)$ that can be debugged from the total fault count of any particular release. Also note that $F_i(t)$ represents the fraction of faults in the i^{th}version of the software that will be debugged in the i^{th}release of the software until the time period $'t'$, whereas $Y_i(t)$ represents the number of bugs in the i^{th} release, consisting of both the faults generated due to the addition of new features and bugs remaining

from the preceding release. The fraction of faults has the following mathematical form:

$$F_i(t)=\begin{cases}0 & t\le t_i\\ F_i(t-t_i) & t_i<t<t_{i+2}\\ 1 & t\ge t_{i+2}\end{cases} \tag{2}$$

In Eq. (2), it is important to note that if $t < t_i$ it means that the i^{th} release has not been tested yet, while if $t = t_i$ it means that the software has been given to testing team to begin debugging. Thus, in both cases the count of bugs discovered will be zero. If $t > t_{i+2}$ then the fraction of bugs discovered will be exactly one. Further, if $t_i < t < t_{i+2}$ then there is a positive fraction of faults being debugged for the i^{th} release at time point $'t'$.

The $Y_i(t)$ represents the number of bugs generated in the i^{th} release of the software with addition of the leftover faults from the $(i-1)^{th}$ release. Thus (for $i > 1$), $Y_i(t)$ can be defined as:

$$Y_i(t)=Y_{i-1}\left[1-F_{i-1}(t_i)\right]+a_i \tag{3}$$

where a_1 represents the faults in the first release (for $i = 1$ we define $Y_1(t) = a_1$) and thereafter the faults of the succeeding releases are denoted by $a_1, i > 1$. The present proposal focuses on the case of four releases of the software.

Primary Release: Substituting $'i = 1'$ in Eqs. (1), (2) and (3) yields:

$$m_1(t)=Y_1(t)F_1(t) \tag{4}$$

where $Y_1(t)=a_1$ and $F_1(t)=F_1(t-t_1)$

First Upgraded Version: Substituting $'i = 2'$ in Eqs. (1), (2) and (3), the fault removal process for the first upgraded version can be given as follows:

$$m_2(t)=Y_2(t)F_2(t) \tag{5}$$

where $Y_2(t)=a_1\left[1-F_1(t_2-t_1)\right]+a_2$ and $F_2(t)=F_2(t-t_2)$

By substituting we get:

$$m_2(t)=\left[a_2+a_1.\left(1-F_1(t_2-t_1)\right)\right]F_2(t-t_2) \tag{6}$$

Second Upgraded Version: By considering $i = 3$ in Eqs. (1), (2) and (3), we have:

$$m_3(t) = Y_3(t) F_3(t) \tag{7}$$

where $Y_3(t) = a_2\left[1 - F_2(t_3)\right] + a_3$ and $F_3(t) = F_3(t - t_3)$

By substituting we get:

$$m_3(t) = \left[a_3 + a_2\left(1 - F_2(t_3 - t_2)\right)\right] F_3(t - t_3) \tag{8}$$

Similarly, the results for the third upgraded version can be obtained:

$$m_4(t) = \left[a_4 + a_3\left(1 - F_3(t_4)\right)\right] F_4(t - t_4) \tag{9}$$

4 Cost-Modeling for Multi Upgraded Software

The formulation of the optimal time of successive software releases determined by an optimization model is based on the objectives set by management. Firstly, management would move in a direction where there is minimum aggregate cost of debugging during the testing and the operational phase. Secondly, they may set the reliability level that is to be attained until the release time of the software on the market.

Release I: The general cost model distinguishes costs in three phases. The testing phase per unit cost of testing; the testing phase cost of debugging; and the operational phase cost of debugging can be given as:

$$\begin{aligned} C_1(t) &= C_{\text{per unit testing cost}} + C_{\text{debugging cost in testing phase}} \\ &\quad + C_{\text{debugging cost in operational phase}} \\ &= C_{10}.t^{\gamma} + C_{11}.a_1.F_1(t) + C_{12}.a_1.\left(1 - F_1(t)\right) \end{aligned} \tag{10}$$

Release II: The expenditure incurred for a new release comprises cost of testing of upcoming version, cost of debugging during the operational phase and cost of elimination of passed-on bugs from the prior version during both the testing and the operational phase of the present release.

$$\begin{aligned} C_2(t) &= C_{\text{per unit testing cost}} + C_{\text{debugging cost in testing phase}} \\ &\quad + C_{\text{debugging cost of remaining faults from previous release}} \\ &\quad + C_{\text{debugging cost in operational phase}} \end{aligned} \tag{11}$$

$$\begin{aligned} &= C_{20}.t^{\gamma} + C_{21}.a_2.F_2(t - t_1) + C_{22}.a_1.\left(1 - F_1(t_1)\right).F_2(t - t_1) \\ &\quad + C_{23}.\begin{pmatrix} \left[a_2 + a_1.\left(1 - F_1(t_1)\right)\right] \\ \left(1 - F_2(t - t_1)\right) \end{pmatrix}; \quad t_1 \le t \end{aligned}$$

Release III: It is assumed that the remaining faults in Release II will either be removed during the testing phase or during the operational phase of Release III. Thus, the total expected cost to debug the faults can be modeled as follows:

$$C_3(t) = C_{30}.t^{\gamma} + C_{31}.a_3.F_3(t-t_2) + C_{32}.a_2.\left(1-F_2(t_2)\right).F_3(t-t_2) + C_{33}.\begin{pmatrix}\left[a_3 + a_2.\left(1-F_2(t_2)\right)\right] \\ \left(1-F_3(t-t_2)\right)\end{pmatrix}; \qquad t_2 \le t \tag{12}$$

where C_{30}represents the testing cost per unit incurred in Release III; C_{31} denotes the cost due to debugging done in the testing phase; C_{32} represents the cost incurred in dealing with the leftover faults of the previous release; C_{33} is the cost incurred for the debugging done in the operational phase.

Similarly, the total cost linked with Release IV can be given as follows:

$$C_4(t) = C_{40}.t^{\gamma} + C_{41}.a_4.F_4(t-t_3) + C_{42}.a_3.\left(1-F_3(t_3)\right).F_4(t-t_3) + C_{43}.\begin{pmatrix}\left[a_4 + a_3.\left(1-F_3(t_3)\right)\right] \\ \left(1-F_4(t-t_3)\right)\end{pmatrix}; \qquad t_3 \le t \tag{13}$$

where C_{40} is the testing cost per unit incurred in Release IV; C_{41} is the cost due to debugging done during the testing phase of the release; C_{42} denotes the cost of dealing with the leftover faults of Release III; C_{43} is the cost incurred for the debugging done in the operational phase of the fourth release.

5 Optimal Release-Time Problem

All the above mentioned sets of costs, Eq. (10) to (13), are used in order to determine the release time of the respective releases of the software, for which the decision-making problem can be defined as follows:

$$\begin{aligned} &Min\ C_i(t) \\ &Subject\ to\ R_i(x \mid t) \tilde{\geq} R_{0i}; \qquad i = 1,2,3,4 \end{aligned} \tag{14}$$

The solution to Eq. (14) can be obtained by using the fuzzy optimization approach given by Zimmermann [32]. The first step is to convert the problem by adding a fuzzifier in the objective function as a limit to the constraint [33]. Thus, the restated problem may have the following structure:

$$\begin{aligned} &\textit{Find} \quad t \\ &\textit{Subject to} \quad C_i(t) \tilde{\leq} C_{0i} \\ &\qquad R_i(x|t) \tilde{\geq} R_{0i}; \quad t \geq 0; \quad i = 1,2,3,4 \end{aligned} \tag{15}$$

Further, the membership functions $\mu_{ij}(t); i = 1,2,3,4$ and $j = 1,2$ for each of the fuzzy inequalities can be defined as:

$$\mu_{i1}(t) = \begin{cases} 1 & ; C_i(t) \leq C_{0i} \\ \dfrac{C_i^* - C_i(t)}{C_i^* - C_{0i}} & ; C_{0i} < C_i(t) \leq C_i^* \\ 0 & ; C_i(t) > C_i^* \end{cases} \tag{16}$$

$$\mu_{i2}(t) = \begin{cases} 1 & ; R_i(x|t) \geq R_{0i} \\ \dfrac{R_i(x|t) - R_i^*}{R_{0i} - R_i^*} & ; R_i^* \leq R_i(x|t) < R_{0i} \\ 0 & ; R_i(x|t) < R_i^* \end{cases} \tag{17}$$

In Eq. (16), C_{0i} and C_i^* represent the available budget and the maximum tolerance value for the budget. In Eq. (17), R_{0i} and R_i^* represent the maximum level of reliability to be maintained and the minimum tolerance value of reliability. Subsequently, the principle of Bellman and Zadeh [34] is used to recognize the fuzzy decision by solving the fuzzy set of inequalities for the corresponding problem. The resultant crisp optimization problem can be given as follows:

$$\begin{aligned} &\textit{Maximize} \quad \alpha_i \\ &\textit{Subject to} \ \mu_{ij}(t) \geq \alpha_i \quad i = 1,2,3,4; \ j = 1,2; \\ &\qquad \alpha_i \geq 0, \quad t \geq 0 \end{aligned} \tag{18}$$

After solving Eq. (18) and incorporating the parameter values, we can calculate the optimal time to release for each particular version of the software on the market.

6 Data Analysis

For the purpose of estimation, fault removal process $F_i(t)$ was assumed to follow Yamada logistic pattern [1], which can be given as follows:

$$F_i(t) = \frac{1-\left(1+b_i\left(t-t_i\right)\right).e^{-b_i\left(t-t_i\right)}}{1+\beta_i.e^{-b_i\left(t-t_i\right)}};\ i=1,2,3,4 \tag{19}$$

The proposed model was analyzed on four different releases of the data sets given by Sun [35] using the SAS software package [36]. The estimated values of the parameters of the proposed models were computed using the non-linear least square method, as shown in Table 1 while the comparison criteria are shown in Table 2. The estimated values of the parameters were quite close to the actual values, which indicates the prediction capacity of the model. Goodness of fit curves are represented in Figure 1.

Table 1 Parameter estimates for four releases.

Parameters	Release 1	Release 2	Release 3	Release 4
a	604.5	443.449	362.221	440.489
b	0.434	0.449	0.845	0.591
β	5.133	0.541	21.319	5.928

Table 2 Comparison criteria for four releases.

Criterion	Release 1	Release 2	Release 3	Release 4
SSE	1220.2	1846.8	1565.5	2075.6
MSE	93.865	142.1	195.7	259.5
Root MSE	9.688	11.919	13.989	16.107
R^2	0.998	0.993	0.991	0.992
AIC	119.708	118.818	79.559	90.018

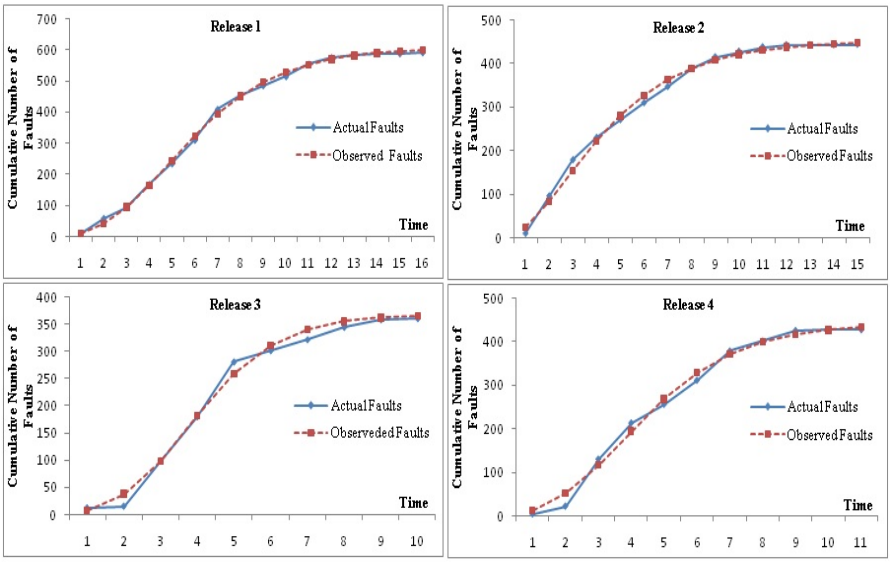

Figure 1 Goodness of fit curves for four releases.

7 Numerical Illustration

For application of the proposed release-time policy, the parameters as obtained in Table 1 were used.

Release I: Considering the parameters as $a_1 = 604.5, b_1 = 0.434$ and $\beta_1 = 5.133$. Let us assume that the cost parameters are $C_{10} = \$18$, $C_{11} = \$21$, and $C_{12} = \$48$. Also, the operational mission time is assumed as $x = 1$ CPU hour and learning parameter $\gamma = 0.85$. Further, it was assumed that the total budget available for testing purpose $C_0 = \$12{,}110$ and the reliability requirement of $R_{01} = 0.95$, with tolerance on cost and reliability $C^* = \$15{,}000$ and $R_1^* = 0.75$ (these assumed values may vary as they are set by management based on past experience). The MVF for failure and the reliability function can be given as follows:

$$m(t) = \frac{604.5\left(1-\left(1+0.434*t\right)e^{-0.434*t}\right)}{\left(1+5.133e^{-0.434*t}\right)} \text{ and } \quad R\left(x|t\right) = e^{-\left(m(t+1)-m(t)\right)}.$$

Correspondingly the membership function pertaining to the fuzzy cost and the reliability constraint can be defined as:

$$\mu_1(t) = \frac{\left(15000 - \left(\begin{array}{l} 18*t^{0.85} \\ +\left(\begin{array}{l} 21*604.5*\left(1-\left(1+0.434*t\right)e^{-0.434*t}\right) \\ /\left(1+5.133e^{-0.434*t}\right) \end{array}\right) \\ +48*\left(\begin{array}{l} 604.5 \\ -604.5*\left(1-\left(1+0.434*t\right)e^{-0.434*t}\right) \\ /\left(1+5.133e^{-0.434*t}\right) \end{array}\right) \end{array}\right)\right)}{\left(15000-12110\right)}$$

where $12110 < C(t) \le 15000$

$$\mu_2(t) = \frac{e^{-\left(m(t+1)-m(t)\right)} - 0.75}{0.95 - 0.75}$$

where $0.75 \le e^{-\left(m(t+1)-m(t)\right)} < 0.95$

On plotting both membership functions, an intersection point is obtained that describes the optimal introduction time of software release, which can also be

computed by solving the crisp optimization problem using an optimization solver such as LINGO [37].

$$\textit{Maximize } \alpha$$

$$\textit{Subject to}$$

$$\mu_1(T) = \frac{\left(15000 - \left(\begin{array}{l} 18*t^{0.85} \\ +\left(\begin{array}{l} 21*604.5*\left(1-\left(1+0.434*t\right)e^{-0.434*t}\right) \\ /\left(1+5.133e^{-0.434*t}\right) \end{array}\right) \\ +48*\left(\begin{array}{l} 604.5-604.5*\left(1-\left(1+0.434*t\right)e^{-0.434*t}\right) \\ /\left(1+5.133e^{-0.434*t}\right) \end{array}\right) \end{array}\right)\right)}{(15000-12110)} \geq \alpha$$

$$\mu_2(T) = \frac{e^{-\left(m(t+1)-m(t)\right)} - 0.75}{0.95-0.75} \geq \alpha$$

$$\alpha \geq 0,\ \alpha \leq 1,\ t \geq 0$$

Solving the above problem, we obtained optimal release time $t^* = 23.64$ and $\alpha^* = 0.7028$.

Release II: Proceeding in the same manner as for Release I, the parameter values for the second release were: $a_2 = 443.44$, $b_2 = 0.434$ and $\beta_2 = 0.541$. Assuming the cost parameters to be $C_{20} = \$25$, $C_{21} = \$30$, $C_{22} = \$45$, and $C_{23} = \$48$. The operational mission time and value for the learning parameter were kept the same. Further, it was assumed that the total budget available for the purpose of testing $C_0 = \$10{,}000$ and the reliability requirement $R_{02} = 0.95$, with tolerance on cost and reliability $C^* = \$15{,}000$ and $R_2^* = 0.75$. Solving the problem, the optimal time to release the software $t^* = 19.56$ with $\alpha^* = 0.2058$.

Release III: Proceeding in the same manner as for Release I, the parameter values for the second upgraded version (the third release) were: $a_3 = 362.221$, $b_3 = 0.845$ and $\beta_3 = 21.319$. Assuming the cost parameters to be $C_{30} = \$15$, $C_{31} = \$19$, $C_{32} = \$30$, and $C_{33} = \$65$. The operational mission time and value for learning parameter were kept the same. Further, it was assumed that the total budget available for the purpose of testing $C_0 = \$1{,}500$ and the reliability requirement $R_{03} = 0.95$, with tolerance on cost and reliability $C^* = \$10{,}000$

and $R_3^* = 0.75$. Solving the problem, the optimal time to release the software $t^* = 12.86$ with $\alpha^* = 0.3346$.

Release IV: Following the same procedure as for Release I, the parameter values for the third upgraded version (the fourth release) were: $a_4 = 440.48$, $b_4 = 0.591$ and $\beta_4 = 5.928$. Assuming the cost parameters to be $C_{40} = \$15$, $C_{41} = \$26$, $C_{42} = \$38$, and $C_{43} = \$48$. The operational mission time and value for the learning parameter were kept the same. Further, it was assumed that the total budget available for purpose of testing $C^* = \$11{,}000$ and the reliability requirement $R_{04} = 0.95$, with tolerance on cost and reliability $C^* = \$15{,}000$ and $R_4^* = 0.75$. Solving the problem, the optimal time to release the software $t^* = 17.77$ with $\alpha^* = 0.8198$.

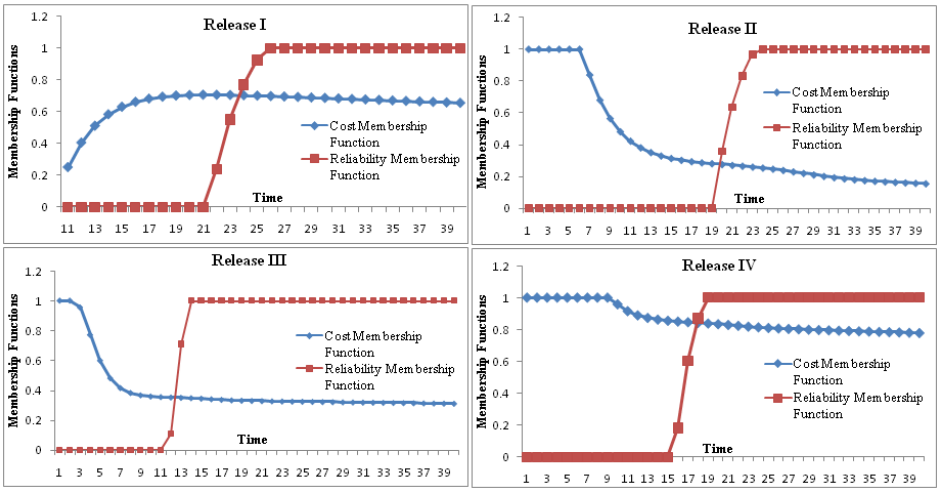

Figure 2 Membership function of cost and reliability for four releases.

Both membership functions corresponding to cost and reliability were plotted for all releases, as shown in Figure 2. There is a clear intersection point of both curves, indicating the optimal launch time for the software. The obtained results corresponding to each release of the software can be summarized in tabular form as given in Table 3.

Table 3 Optimal release time for all releases.

Releases	t^*	α^*	Actual release time
Release-I	23.64	0.7028	16
Release-II	19.56	0.2058	15
Release-III	12.86	0.3346	10
Release-IV	17.77	0.8198	11

From Table 3 it can be clearly seen that each software release was subjected to under-testing. Thus it is wise to assume that for achieving the threshold value of reliability, the software should be tested for a longer duration.

8 Conclusion

To make the software bug free and enhance its operational capability, firms keep testing and trying to fix them by issuing consecutive upgrades. With this in mind, an alternative mathematical framework was presented in this paper that can capture the faults generated due to addition of certain novel features and leftover faults from its immediately preceding release. The set of proposed models was analyzed on real-life data sets of four different releases. Further, fuzzy release time problems were constructed for the four versions of the software and the optimal time to launch of each version was computed.

Acknowledgments

The work reported in this paper was supported by grants to the first author from the Department of Science and Technology, India through DST PURSE PHASE II scheme, India; and to the second author from the Rajiv Gandhi National Fellowship from University Grants Commission, New Delhi, India and fourth author would like to be thankful to National Research Foundation (South Africa) for the financial support.

References

[1] Kapur, P.K., Pham, H., Gupta, A. & Jha, P.C., *Software Reliability Assessment with OR Applications*, Springer, London, 2011.

[2] Charette, R.N., *Why Software Fails*, IEEE Spectrum, http://spectrum.ieee.org/computing/software/why-software-fails, (21 June 2015).

[3] Malaiya, Y.K., *Software Reliability: A Quantitative Approach*, System Reliability Management: Solutions and Technologies, **205**, 2018.

[4] Anand, A., Singh, O. & Das, S., *Fault Severity Based Multi Up-Gradation Modeling Considering Testing and Operational Profile*, International Journal of Computer Applications, **124**(4), 2015.

[5] Anand, A., Singh, A., Kapur, P.K. & Das, S., *Modeling Conjoint Effect of Faults Testified from Operational Phase for Successive Software Releases*, Proceedings of the 5thInternational Conference on Life Cycle Engineering and Management (ICDQM), pp. 83-94, 2014.

[6] Kumar, A., Anand, A., Garg, P.K. & Agarwal, M., *Optimal Release Time Decision from Fuzzy Mathematical Programming Perspective*, arXiv:1509.08086, 2015.

[7] Kapur, P.K., Tandon, A. & Kaur, G., *Multi Up-Gradation Software Reliability Model*, In Reliability, Safety and Hazard (ICRESH) 2010, 2nd International Conference on IEEE, pp. 468-474, 2010.

[8] Das, S., Aggrawal, D. & Anand, A., *An Alternative Approach to Model Multi Up-gradations for Software Systems*, Recent Advancements in Software Reliability Assurance, CRC Press (Taylor & Francis Group), pp. 93-105, 2019.

[9] Singh, O., Kapur, P.K., Khatri, S.K. & Singh, J.N.P., *Software Reliability Growth Modeling for Successive Releases*, Proceeding of 4th International Conference on Quality, Reliability and Infocom Technology (ICQRIT), pp. 77-87, 2012.

[10] Aggarwal, A.G., Kapur, P.K. & Garmabaki, A.H.S., *Imperfect Debugging Software Reliability Growth Model for Multiple Releases,* Proceedings of the 5th National Conference on Computing for Nation Development-INDIACOM, New Delhi, India, pp. 337-344, 2011.

[11] Singh, O., Kapur, P.K. & Anand, A., *A Stochastic Formulation of Successive Software Releases with Faults Severity,* Industrial Engineering and Engineering Management (IEEM), 2011 IEEE International Conference on IEEE, pp. 136-140, 2011.

[12] Garmabaki, A.H.S., Kapur, P.K., Aggarwal, A.G. & Yadavali, V.S.S., *The Impact of Bugs Reported from Operational Phase on Successive Software Releases*, International Journal of Productivity and Quality Management, **14**(4), pp. 423-440, 2014.

[13] Singh, O., Kapur, P.K. & Singh, J.N.P., *Testing-Effort Based Multi Upgradation Software Reliability Growth Model*, Communications in Dependability and Quality Management – An International Journal (CDQM), **15**(1), pp. 88-100, 2012.

[14] Kapur, P. K., Sachdeva, N. & Singh, J.N., *Optimal Cost: A Criterion to Release Multiple Versions of Software*, International Journal of System Assurance Engineering and Management, **5**(2), pp. 174-180, 2014.

[15] Singh, O., Kapur, P.K., Shrivastava, A.K. & Das, L., *A Unified Approach for Successive Release of a Software Under Two Types of Imperfect Debugging*, Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2014 3rd International Conference on IEEE, pp. 1-6, 2014.

[16] Goel, A.L. & Okumoto, K., *Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures*, IEEE Transactions on Reliability, **28**(3), pp. 206-211, 1979.

[17] Yamada, S. & Osaki, S., *Optimal Software Release Policies with Simultaneous Cost and Reliability Requirements*, European Journal of Operational Research, **31**(1), pp. 46-51, 1987.

[18] Yun, W.Y. & Bai, D.S., *Optimum Software Release Policy with Random Life Cycle*, IEEE Transactions on Reliability, **39**(2), pp. 167-170, 1990.

[19] Huang, C.Y., *Cost-Reliability-Optimal Release Policy for Software Reliability Models Incorporating Improvements in Testing Efficiency*, Journal of Systems and Software, 77(2), pp. 139-155, 2005.

[20] Huang, C.Y. & Lyu, M.R., *Optimal Release Time for Software Systems Considering Cost, Testing-Effort, and Test Efficiency*, IEEE Transactions on Reliability, **54**(4), pp. 583-591, 2005.

[21] Pham, H. & Zhang, X., *A Software Cost Model with Warranty and Risk Costs*, IEEE Transactions on Computers, **48**(1), pp. 71-75, 1999.

[22] Ramík, J., *Soft Computing: Overview and Recent Developments in Fuzzy Optimization*, Ostravska Univerzita, Listopad, pp. 33-42, 2001.

[23] Rommelfanger, H., *The Advantages of Fuzzy Optimization Models in Practical Use*, Fuzzy Optimization and Decision Making, **3**(4), pp. 295-309, 2004.

[24] Tang, J. & Wang, D., *Modelling and Optimization for A Type of Fuzzy Nonlinear Programming Problems in Manufacturing Systems*, In Decision and Control, 1996, Proceedings of the 35th IEEE Conference on IEEE, (4), pp. 4401-4405, 1996.

[25] Ukimoto, S. & Dohi, T., *A Software Cost Model with Reliability Constraint Under Two Operational Scenarios*, International Journal of Software Engineering and Its Applications, **7**(1), pp. 415-426, 2003.

[26] Xie, M. & Yang, B., *A Study of the Effect of Imperfect Debugging on Software Development Cost*, IEEE Transactions on Software Engineering, **29**(5), pp. 471-473, 2003.

[27] Yang, B., Hu, H. & Jia, L., *A Study of Uncertainty in Software Cost and Its Impact on Optimal Software Release Time*, IEEE Transactions on Software Engineering, **34**(6), pp. 813-825, 2008.

[28] Kapur, P.K., Pham, H., Gupta, A. & Jha, P.C., *Optimal Release Policy Under Fuzzy Environment*, International Journal of Systems Assurance Engineering and Management, **2**(1), pp. 48-58, 2011.

[29] Jha, P.C., Singh, O., Indumati & Kapur, P.K., *Bi-criterion Release Time Problem Incorporating Effect of Two types of Imperfect Debugging under Fuzzy Environment*, Parkash, O., (Ed.), Advances in Information Theory and Operations Research: Interdisciplinary Trends, 2010.

[30] Jha, P.C., Indumati, Singh, O. & Gupta, D., *Bi-Criterion Release Time Problem for A Discrete SRGM Under Fuzzy Environment*, International Journal of Mathematics in Operational Research, **3**(6), pp. 680-696, 2011.

[31] Kumar, D. & Gupta, P., *Fuzzy Software Release Problem with Learning Functions for Fault Detection and Correction Processes*, Software Engineering, Springer, Singapore, pp. 655-661, 2019.

[32] Zimmermann, H.J., *Applications of Fuzzy Set Theory to Mathematical Programming, Information Sciences*, **36**(1-2), pp. 29-58, 1985.

[33] Lee, K.H., *First Course on Fuzzy Theory and Applications*, Vol. 27, 1st Ed., Springer-Verlag Berlin Heidelberg, 2006.

[34] Bellman, R.E. & Zadeh, L.A., *Decision-Making in A Fuzzy Environment*, Management Science, **17**(4), pp. B-141, 1979.
[35] Sun, H.W., *Analysis of Costs and Delivery Intervals for Multiple-release Software*, PhD Dissertation, Department of Industrial and Manufacturing Engineering, New Jersey Institute of Technology, New Jersey, 2002.
[36] SAS, *SAS/ETS User's Guide version. 9.1*, Ed. Cary, SAS Institute Inc., North Carolina, 2004.
[37] Thiriez, H., *OR Software Lingo*, European Journal of Operational Research, **12**, pp. 655-656, 2000.