Software Reliability Modeling with Impact of Beta Testing on Release Decision

Adarsh Anand, Navneet Bhatt, Deepti Aggrawal and Ljubisa Papic

Abstract Increased dependence of humans on technologies has made it necessary for developing the software with high reliability and quality. This has led to an increased interest of firms toward the development of software with high level of efficiency; which can be achieved by incorporating beta tests for improving and ensuring that the software is safe and completely free from errors. In a software release life cycle, beta testing is the last important step that software developers carry out before they launch new software. Beta testing is a unique testing process that helps software developers to test a software product in different environments before its final release in the market. In this chapter of the book, we develop a SRGM by inculcating the concept of beta testing in the fault removal process to account for situations that might occur when the software is used in diverse environments. This is done to evade the chances of system being failed in the field. Conducting beta tests results in enhancement of software reliability and has been widely acknowledged. Furthermore, we have developed an optimal scheduling model and showed the importance of beta test while determining the general availability time of the software and making the system more cost effective. For validating the accuracy and predictive capability of the proposed model, we analyzed it on real software data set.

Keywords Software reliability · Software release life cycle · Beta testing · Optimal release scheduling

A. Anand (✉) · N. Bhatt
Department of OR, University of Delhi, New Delhi 110007, India
e-mail: adarsh.anand86@gmail.com

N. Bhatt
e-mail: navneetbhatt@live.com

D. Aggrawal
Keshav Mahavidyalaya, University of Delhi, New Delhi 110034, India
e-mail: deepti.aggrawal@gmail.com

L. Papic
DQM Research Centre, Cacak, Serbia
e-mail: dqmcenter@open.telekom.rs

M. Ram and J.P. Davim (eds.), *Advances in Reliability and System Engineering*, Management and Industrial Engineering, DOI 10.1007/978-3-319-48875-2_5

1 Introduction

The growth of the internet—and the lucrative opportunities it presents—is bringing with it an explosion in software application development. Software has become an inherent part of every fabric of our lives. In today's connected economy, almost every government as well as private and nonprofit enterprise rely on software as a core business function. The growth in software advancement and rapid delivery of new features led to a major shift in the way to meet customer demands, and therefore an organized environment for development and testing becomes an integral part of the value chain. In 2015, according to Gartner, the worldwide size of the security software market was US$22.1 billion, an increase of 3.7 % over 2014 [1]. In India, the IT sector has increased its impact on India's GDP from 1.2 % in 1998 to 9.5 % in 2014, further aggregating a revenue of US$143 billion in FY2016, where export revenue raised to US$108 billion and domestic to US$ billion, rising by over 8.5 % [2].

As software application becomes ingrained in our day-to-day life, its failures result in disastrous situations which are becoming even more serious. Reports of tragic effects of software failure exist in large numbers. Some well-known failures such as programming errors in the radiation therapy machine result in the death of three persons due to the massive overdose of radium [3]. An on-board software program failure caused an explosion in the Ariane 5 heavy lift launch vehicle on June 4, 1996, which cost more than US$7.0 billion to the European Space Agency [4], and a software bug present in the engine control system of Royal Air Force helicopter caused its crash, killing more than 25 persons [5]. Also in the very last year some of the famous software glitch which resulted in severe disruption were entertained that includes the Amazon 1p price glitch which caused products on sale in marketplace for just one penny. This flaw resulted in a loss of $100,000 for the vendors. In September 2014, Apple had a major embarrassment when it had been forced to pull the iOS 8 update merely after its release due to the various malfunctions in the software [6].

For an increasing demand of delivering reliable software products to the users, software quality has become more vital recently due to the increased security concerns arising from software vulnerabilities. The quality of a software encountered by an end user is an association of the faults in a software product when it is released, plus the efforts that the developer makes to patch the imperfections after release. Once a software is up for general availability it has to be free from all the flaws. A software product has usually gone through a number of stages in its development before it is available for general availability. Software testing is one of the phases which is usually performed for various purposes: first, to improve quality; second, for verification and validation, and for reliability estimation [7]. In this regard, software reliability models can give significant level of reliability for a software during the development process. Over the past four decades, research activities in field of reliability engineering have been done, and various software reliability growth models (SRGMs) have been proposed in the literature [8].

SRGMs have been effective in assessing the software reliability and the potential number of faults in the software.

Selecting a suitable software reliability growth model for attaining a desired level of reliability; software engineer makes it sure that the testing has been performed till the time sufficient number of bugs or faults have been removed and the software offering is ready to be released. A crucial decision for the firms lies at time when the software has to be released in the market or to know the exact time till when the execution of testing activities should be done. Many researchers have established various software scheduling problems considering different aspects; to analyze the period of software testing phase in the literature [8]. Selecting an appropriate software reliability model uniquely identifies a time point suggesting the time at which the software is up for release in the market. But the problem aforesaid is not just associated with its general availability, it involves many factors and attributes that a firm have to take care while determining the optimum release time. If testing stops prematurely, pending errors in the software may leave the software developers with dissatisfied customers and can invite a high cost of fixing faults during its operational stage. On the other hand, if shipping a software is too late, it surely increases its reliability but the cost due to economic factors constitute may charge a firm with high testing cost, penalty cost due to late delivery. Hence, while deciding the optimum release time both factors have to be taken care judiciously.

In this chapter, we have focused our objective on testing the releases of a software during its lifetime and further implying that the reliability can be improved when the software undergoes phase transformation from alpha testing to beta testing phase. Furthermore, an optimal release scheduling model is provided which incorporates various costs and a preferred level of software reliability in determination of optimum release time of a software. The rest of this chapter is organized as follows. First, in the following section, a brief background of our study is provided. Section 3 provides the literature review of our study. In Sect. 4, we derive a SRGM based on Non-Homogenous Poisson Process (NHPP) to represent a fault detecting/removal process during the testing and further integrate the beta testing activities. Section 5 evaluates the proposed SRGM with a numerical example of a real software failure data. In Sect. 6 an optimal release time cost model is presented, finally conclusions and acknowledgement are given in Sects. 7 and 8.

2 Background and Motivation

2.1 Software Release Life Cycle

Prior to the testing, a software product has been put through the stages of development and maturity which include the activities: requirement overview, design making, coding, and system testing. The life cycle of a software is the collection of

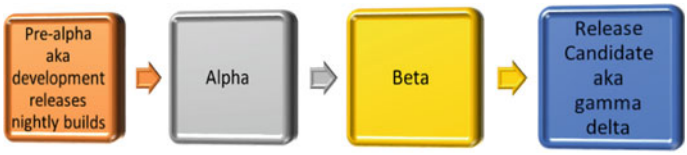

Fig. 1 Stages of development [9]

all the phases of development of an application: varying since its preliminary requirement to its ultimate release, along with the updated versions of the previous releases in order to fix loopholes and to improve the features present in the software. As shown in Fig. 1, a software release is characterized by different versions in its lifetime [9].

With the continuous improvement in the development of software in every stage; each version of the software is released either in private or public for testing. A final software release is preceded by the deployment of alpha and then beta versions. The final quality of a software mainly depends on the activities that are performed during the testing phase; that is, with the debugging of faults, reliability of the software improves. As mentioned earlier, the software testing process is very complex. Due to this reason there are many types of software testings available in the literature [10]. One of the types of software testing is alpha testing. Usually it is performed at the development site by a number of in-house testers or an independent test team. Alpha testing allows a developer to perform internal acceptance testing, which is normally employed prior to the beta testing [10, 11].

2.2 Beta Testing

As shown in Fig. 1, prior to release of beta version of software, after the completion of in-house testing activities for each pre-alpha and alpha versions of the software by the developer then the software ends with a feature freeze version, representing that, there will be no more design and feature changes in the software. Then, the software version is termed as feature complete. Beta phase usually commences after the transition of a software program to a feature complete application and the software is ready for the open and closed beta testing [9].

Every enterprise or organization follows its own software testing life cycle. We emphasize on a specific stage of software testing called beta testing. It is the first "user test" of a software that allows a developer to assess the usability and functionality feedback by the end users. During beta testing, a pre-release version "*Betaware*" is given to a group of users to test in "real world" environments; in order to validate the software offering. The motive of beta test is to improve the software prior to its release [12]. Contributing to the beta testing activities allows an enterprise to get benefits in several ways. First, the system environment of beta

testers can vary with respect to the both hardware and software point of view as it is often difficult to match in lab testing environment—since a software application is likely to have different performance related issues and can trigger bugs while running on different system environments, beta testers can help discover faults or compatibilities issues that cannot be detected in testing labs. Second, beta testers are independent and are likely to be less biased. Third, beta testing can save the costs that may result due to the reputation damage or warrant expenses if the offering has to be recalled from the market after launch [13].

Beta testing has been traditionally meant to a small number of selected and trained beta testers that check out a product and review the product based on their experiences. As in this competitive environment, an effective beta test can save valuable time and enable a firm to launch its offering earlier in this fast moving world. With the rapid growth of the Internet, most software firms release multiple beta versions of the software application for the open beta testing. Jiang et al. [13] and Fine [14] mentioned the term public beta testing, where the betaware is made available to websites and the interested users or testers who are comfortable living on the absolute bleeding edge may download the latest release of software before it hits the public and put a hand in volunteering the release with the goal of discovering and reporting bugs. Figure 2 shows a screenshot of an application Snap beta version; a free Google Play client for BlackBerry 10 operating system, mentioning the known issues and the changelog of the releases prior to the beta version3 reported by the beta testers [15].

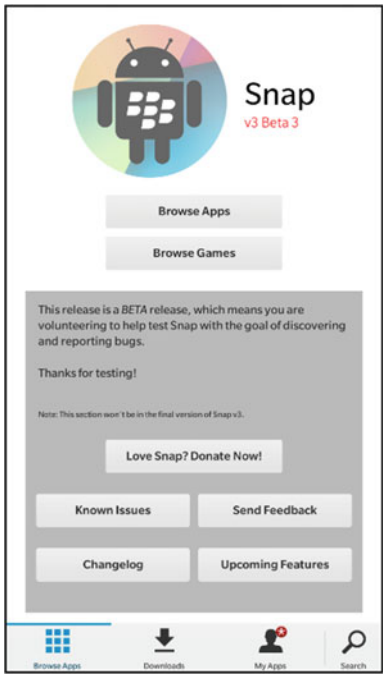

Fig. 2 Screenshot of snap beta version [15]

Like *snap*, most software releases various beta versions and after testing with each release publically, then arrives as a release candidate. A release candidate is generally considered as a final product and in this, all of its software features have been tested thoroughly in order to minimize the likelihood of fatal flaws while releasing a software. The final version of the software is generally termed as general availability (GA). As a brief review of some examples of well-known public betas, beta testing has been received a great response. In September 2000, a preview of Apple's next-generation operating system Mac® OS X was released for the beta testing [16]. On the same lines Microsoft released a community technology previews (CTPs), a beta testing program back in 2005 and 2006 for its operating system Windows Vista, and for its latest operating system Windows 10; Microsoft scheduled the operating system for the public beta testing through the Windows Insider program from October 2014. A total of 10 builds were released as of May 29, 2015 for the public beta testing before its general availability on July 29, 2015. Over 1.5 million users have already downloaded and installed the Windows 10 through the Microsoft insider program by the end of 2014. Other very popular betas are Google's Gmail, Google Calendar, and Google News [9, 17–21].

The public beta testing phenomenon has also influenced the mobile applications for smartphones and tablets. For example, services such as BlackBerry Beta Zone offered in BlackBerry 10 smartphones provide the beta versions of various mobile applications for the public beta testing. WhatsApp Messenger being available for the beta testing program in the BlackBerry Beta Zone has reported more than 450 bugs as of April, 2016 [22]. There are even websites being developed in bringing together developers and beta testers for Android and iPhone apps. For example, websites like TestFairy.com and iBetaTest.com allow iPhone and Android app developers to publish apps, and testers may download the apps and provide the general issues and feedbacks to the developers [13]. As of April, 2016, iBetaTest.com reportedly has around 16,600 beta testers around the world. This chapter is focused to understand the importance of beta testing before making a final call to release the software.

3 Literature Overview

In past decades, researchers have proposed a number of SRGMs under different set of assumptions and testing environment and most of them are based upon the Non-homogeneous Poisson Process (NHPP) in the literature [8, 23–25]. In 1983, Yamada et al. [26] considered the testing process as two stages by taking a time lag in between the fault detection and removal and called it as the delayed S-shaped model, similarly Ohba [27] also developed S-shaped models using different set of assumptions. Furthermore, many optimal release time models have been developed using different criterion, like cost and reliability, provided by the management team. The first unconstrained release time policy was derived by Okumoto and Goel [28] which was based on exponential SRGM. Later, Yamada and Osaki [29] developed

the constrained software release time problem with the objective as cost minimization and reliability maximization. Kapur et al. [30] discussed and studied the bi-criterion optimal release time problem. Jain et al. [31] considered warranty time and warranty cost in the development of software release time problem. Recently, Singh et al. [32] have formulated an optimal release time problem using the multi-attribute utility theory. Further, Singh et al. [33] used different set of attributes in the construction of release time problem. Moreover, Arora et al. [7] studied the impact of patching a software when the software product is available for purchase. Many authors investigated the importance of patching after the release of software in the market [34, 35]. It should be noted that the SRGMs proposed earlier [8] only considered the testing activities that are performed when the software has been tested under the lab environment. With consideration of faults detected under the beta testing environment and providing the significant measures to the reliability of software, this study tries to capture the behavior of reliability growth when both alpha and beta testing phases are studied consecutively.

Among the family of studying this problem of optimal release time, the concept of beta testing has received less attention. As an aspect of software engineering, a few authors have considered the idea of beta testing in their study. Wiper and Wilson [36] described a fault detection process during the beta testing phases using the Bayesian statistical method and further developed a cost model for the optimal testing strategy. Kocbek and Heričko [11] showed the use of beta testing to identify the optimal number of testers for the mobile application. Another study by Jiang et al. [13] considered the impact of beta testing activities on the acceptance of final software product. Furthermore, Mäkinen et al. [37] examined the effect of open virtual forum on the adoption of beta products. However, no methodology has been proposed to measure the effectiveness of beta testing during the software testing.

The lack of research on beta testing activities concerning the development of software reliability modeling is a notable shortage that the critical role it plays in the software testing. This study tries to deviate the focus of software engineers to this vital area by studying and scheduling the release plan of software based on beta testing. To do that, emphasis is paid on the stages of the software release cycle with special attention to beta test phase. In the succeeding section, we have developed a mathematical model incorporating the process of beta testing. Furthermore, an optimal release time model is developed for the determination of software release time.

4 Model Formulation

In this section, we propose a new NHPP SRGM that incorporates the fault discovery process “beta testing” when the in-house alpha testing phase has been completed.

Research has been done to take some practical issues into the consideration of software reliability modeling. For example, post-release testing and software release policies can be found in [38]. In this chapter, we derive a new model incorporating the faults detected during the beta testing phases into software reliability

Fig. 3 Timeline of Testing

assessment. The framework has been modeled into three different phases considering the detection and correction, respectively. As illustrated in Fig. 3, a software being tested during its release life cycle undergoes testing to identify and remove defects before it is released to the customers. Following are the notations used in the proposed model.

$m_1(t)$ Expected no. of faults removed by time 't_α'
$m_2(t)$ Expected no. of faults detected by time 't_β'
$m_3(t)$ Expected no. of faults removed by time 't'
a Initial no. of faults
b_1 Fault detection/removal rate by the testers in time period $(0, t_\alpha)$
b_2 Fault detection/removal rate by the users in time period (t_α, t_β)
b_3 Fault detection/removal rate by the testers in time period (t_β, t).

4.1 Phase I: Alpha Testing Phase

In Phase I, the failure pattern is observed in the time period during which the SRGM undergoes alpha testing. Most of SRGMs proposed in the literature were based with the assumption that software testing has been performed only till the in-house testing phase and the software is up for release.

In the time interval $[0, t_\alpha)$, the failure intensity is proportional to the number of software faults remaining in the software till the time software undergoes alpha testing. The failure intensity during this period can be described using the following differential equation:

$$\frac{\mathrm{d}m_1(t)}{\mathrm{d}t} = b_1(a - m_1(t)) \quad 0 \leq t < t_\alpha \tag{1}$$

where

$$m_1(0) = 0$$

and a is the potential faults present in the software initially and b_1 is the fault detection rate in the alpha testing period. Solving the above equation, the mean value function for the first phase of testing can be obtained, as follows:

$$m_1(t) = a(1 - e^{-b_1 t}), \quad \text{when} \quad 0 \leq t < t_\alpha \tag{2}$$

4.2 Phase II: Beta Testing Phase

In Phase II, the testing process is considered as the fault discovering process called beta testing. After the completion of alpha testing phase at time t_α, a feature complete software version is introduced to the market which is likely to contain a number of known or unknown bugs as a betaware for beta testing. Here, the software is tested under user environment and the failure occurrences are reported back by the beta testers.

During the beta testing phase, the number of faults discovered can be modeled as follows:

$$\frac{dm_2(t)}{dt} = b_2[(a - m_1(t_\alpha)) - m_2(t)] \quad t_\alpha < t \leq t_\beta \tag{3}$$

where

$$m_2(t_\alpha) = m_0 \geq 0$$

and b_2 is the fault detection rate due to the beta testers. It represents that the fault intensity is proportional to the outstanding faults reported by time t_β. Solving Eq. (3), the mean value function for the faults discovered during the beta testing phase can be obtained as:

$$m_2(t) = ae^{-b_1 t_\alpha} + e^{-b_2(t - t_\alpha)}(m_0 - ae^{-b_1 t_\alpha}), \quad \text{when} \quad t_\alpha < t \leq t_\beta. \tag{4}$$

We refer to m_0 as the beta bugs, representing a finite number of bugs which are being detected instantly by the beta testers who use the software.

4.3 Phase III

Phase III corresponds to the testing process of SRGM as the fault removal phenomenon. After the completion of beta testing, the faults reported by the beta testers during the Phase II period are corrected after time t_β. In Phase III, the fault removal intensity is proportional to the residual faults that were being observed. The following system of differential equation describes the same:

$$\frac{\mathrm{d}m_3(t)}{\mathrm{d}t} = b_3\left[m_2(t_\beta) - m_3(t)\right] \quad t_\beta \le t \tag{5}$$

where

$$m_3(t_\beta) = m_1 \ge 0$$

and $m_2(t_\beta)$ is the cumulative faults at time t_β and b_3 is the fault removal rate. Using the initial condition provided, the mean value function is obtained as:

$$\begin{aligned} m_3(t) &= m_2(t) + (m_1 - m_2(t))\mathrm{e}^{-b_3(t-t_\beta)} \\ m_3(t) &= (a\mathrm{e}^{-b_1 t_\alpha} + (m_0 - a\mathrm{e}^{-b_1 t_\alpha})\mathrm{e}^{-b_2(t-t_\alpha)})(1 - \mathrm{e}^{-b_3(t-t_\beta)}) + m_1\mathrm{e}^{-b_3(t-t_\beta)}, \quad \text{when} \quad t_\beta \le t \end{aligned} \tag{6}$$

Here we mention $m_1 \ge 0$ as those beta bugs which are being removed as in when the fault removal phenomenon is started after beta testing. The faults captured by the beta tester were reported back during the beta phase allowing the software engineer to remove a certain number of beta bugs by the time Phase III has been started. Thus the total faults removed under overall testing of the software are the bugs from Phase I and Phase III which has the mathematical form as given below:

$$\begin{aligned} m^*(t) &= m_1(t) + m_3(t) \\ m^*(t) &= a\left(1 - \mathrm{e}^{-b_1 t}\right) + (a\mathrm{e}^{-b_1 t_\alpha} + (m_0 - a\mathrm{e}^{-b_1 t_\alpha})\mathrm{e}^{-b_2(t-t_\alpha)})(1 - \mathrm{e}^{-b_3(t-t_\beta)}) + m_1\mathrm{e}^{-b_3(t-t_\beta)} \end{aligned} \tag{7}$$

Above Eq. (7) denotes the total faults being removed during the total testing time which is divided into three phases as mentioned above.

5 Numerical Analysis and Model Validation

5.1 Data Description

For the validation of the proposed model, we have used data of a Brazilian switching software project. The data size of the software was about 300 KB and it was written in assembly language [39]. In the total execution period of 81 weeks, 461 faults have been removed. Also the entries of the data set comprises of different phases of testing, i.e., first 30 entries are from validation phase and next entries are from field trials and system operations. The data used in the estimation procedure gives a clear indication that the software project has undergone through its release life cycle with alpha testing and beta testing phases.

5.2 Performance Analysis

In this subsection, we examine the model given by Goel and Okumoto [23] with our proposed model. We have estimated the parameters of SRGM by using the methods of LSE. Nonlinear regression module of SPSS software package has been used for the estimation of parameters and the evaluation of goodness of fit criteria.

The parameter estimates and the goodness of fit criteria for the data set under consideration are given in Tables 1 and 2, respectively. Figure 4 represents the goodness of fit curve for the data set and the estimated and predicted values appear to be closely related.

From Table 2, we can see that the MSE, RMSPE, and Variation of the proposed model incorporating beta testing are less than the G-O model. We also see that the Biasness of our model attains smaller values. Moreover, the R^2 also conclude that the proposed model fit the data excellently. Altogether, it is sensible to conclude that the proposed model incorporating beta testing attains better goodness of fit criteria. The improvement achieved by our model is obvious due to the release of software for beta testing; which reflects a significant variation of the fault detection rates and instead of considering a constant fault detection rate throughout its release life cycle. On the basis of our results, we can conclude that software developing firms should practice pre-release testing activities for enhancing the product for general availability.

Table 1 Parameter estimates

Parameters	GO model	Proposed model
a	546.082	602.779
b_1	0.02421	0.023
b_2	–	0.115
b_3	–	0.026
m_0	–	16.410
m_1	–	47.850

Table 2 Goodness of fit

Parameters	GO model	Proposed model
MSE	99.86442	87.4553
Bias	−0.921008	0.000000101
Variation	10.01268	9.410026
RMSPE	10.05495	9.410026
R^2	0.994	0.995

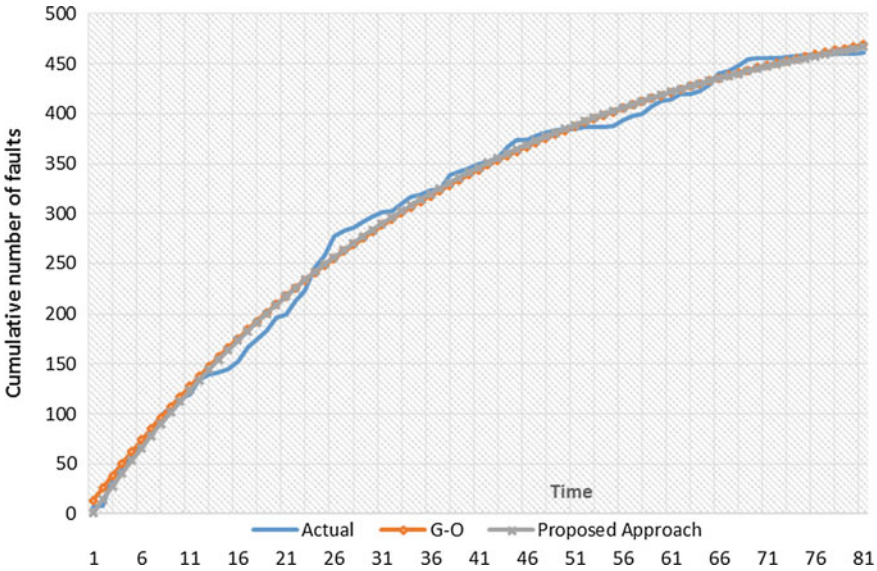

Fig. 4 Goodness of fit curve

6 A Cost Model for Optimal Release Time

In this section, we apply the NHPP model to establish the optimal scheduling policy for a software. Since, in practice every software developer needs to determine for how long software should be tested, such that the expected cost is optimally minimal and the reliability of the software product must satisfy customers' requirements as well. In the literature of software reliability engineering, these kinds of problems are termed as software release time decision problems. There may be a solution that no more testing should be done but generally speaking, the main goal is to achieve a balance between under-testing and over-testing. If a premature software is released in the market, the developer may suffer from loss of user confidence, and if there is a delay in release or over-testing it will impose the burden of penalty cost and overly time consuming. Therefore, optimal release policies are important and realistic issue. Research on software development cost estimation has been conducted in literature [8]. In this study, we have compared the well-known Okumoto and Goel [28] model with our proposed optimal release time problem which has taken the consideration of software cost when the software testing has been performed consecutively as alpha testing and beta testing.

6.1 Review of Cost Model (Okumoto and Goel [28])

In 1983, Okumoto and Goel [28] developed a mathematical model comprising various costs that occurred in the testing and maintenance of a software and is given as

$$K(t) = K_1 \times m(t) + K_2 \times (m(t_L) - m(t)) + K_3 \times t \tag{8}$$

Okumoto and Goel [28] used the mean value function given by Goel and Okumoto [23] in order to formulate the cost model. The cost function comprised of different cost components, i.e., the cost incurred due to the removal of a fault observed during testing and operational phases; as K_1 and K_2, respectively, and K_3 as per unit testing cost. The optimal release time of a software is determined by minimizing the following unconstrained function.

$$\text{Min}\, K(t) = K_1 \times m(t) + K_2 \times (m(t_L) - m(t)) + K_3 \times t \tag{9}$$

6.2 Cost Model Formulation

In contrast to policies given earlier, we here provide a scheduling policy. The proposition is based on certain basic assumptions which are as follows:

(a) Fault removal process is described by NHPP.
(b) During the alpha testing phase fault is removed perfectly.
(c) Faults detected during the field trial are removed after the completion of beta testing phase.
(d) Total fault content in the software is fixed.
(e) The cost incurred in testing is proportional to per unit testing time.
(f) The cost of fixing faults during the alpha and beta phase is proportional to the time required to isolate all the faults found by the end of testing.

In our case, the advantage of beta testing that we are claiming in the aforesaid sections provides a better fault count prediction for the software reliability if a developer releases its software for the field trials. As previously mentioned, software firms are keen to know the release time for their software product. We try to capture the optimal testing time, such that the firms expect a minimum cost with an achievable reliability.

The idea is to optimize the value t^* with respect to the cost involved. We have assumed the following costs in our study:

i. a cost C_1 for isolating a fault during the alpha testing. This reflects the cost incurred to the firm to remove a fault during the in-house testing activities. Typically, in the lab environment situation, this cost is likely to be less as fixing a detected fault incurs a small amount of CPU hours.

ii. a cost C_2 for isolating a fault after the beta testing. This reflects the cost of fixing the discovered bugs at the end of beta testing. Generally, in the beta testing situation, the cost of collecting the feedbacks and responding to the comments and queries of beta testers must be considered and this cost contributes a higher value to the firm.
iii. a cost C_3 per failure per unit time after the software is released. We would normally set this cost to be much higher than the previous costs as the damage caused by leaving high complex faults in the software would require many resources for the correction.
iv. a cost C_4 per unit time. This reflects the cost of testing incurred to the firm per unit time.

This implies that the overall expected cost function of testing is

$$C(t) = C_1 \times m_1(t) + C_2 \times m_3(t) + C_3 \times (a - m^*(t)) + C_4 \times t \tag{10}$$

In the above Eq. (10), the first component corresponds to the cost involved in testing the software during the alpha phase [where $m_1(t)$ is as given in Eq. (2)]. Second component represents the cost incurred while removing the faults detected in the beta phase and $m_3(t)$ is taken as given in Eq. (6). Third component talks about the cost required to correct the faults during the operational phase and $m*(t)$ incorporates the fault correction phenomenon during the overall testing phase; as is given in Eq. (7). The last component of the cost function denotes the per unit testing cost. Hence, we find the optimal release time of the software that minimizes the total testing cost subject to achieving a reliability level, R_0. Then the optimization problem can be expressed as

$$\begin{aligned} &\text{Min. } C(t) = C_1 \times m_1(t) + C_2 \times m_3(t) + C_3 \times (a - m^*(t)) + C_4 \times t \\ &\text{s.t} \\ &R(x/t) \geq R_0 \end{aligned} \tag{11}$$

Solving the aforesaid problem under the minimization criteria provides the optimal time to release the software in the market.

6.3 Numerical Illustration

The estimates of G-O model and the proposed model are used in the cost optimization problem (Table 1). Making use of OPTMODEL procedure of SAS software [41] to determine the optimal release time and the associated testing cost, and for checking the appropriateness of the proposed approach, we consider two cases corresponding to the optimization problem as below.

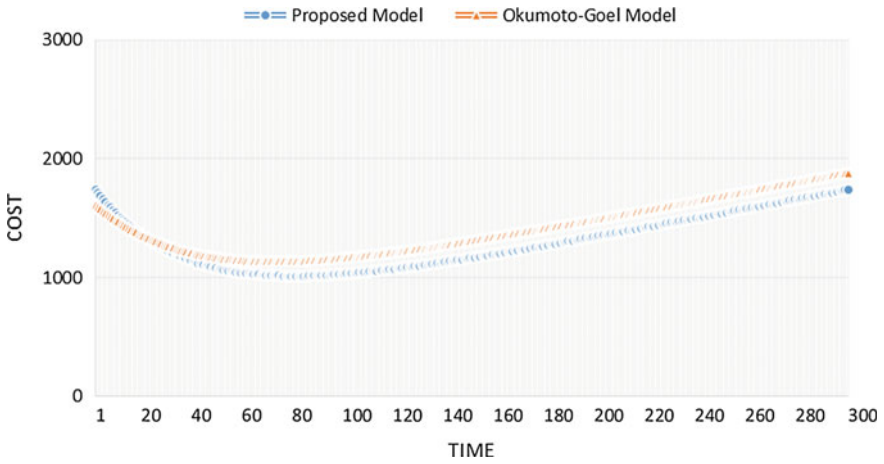

Fig. 5 Expected cost function

Case 1: Okumoto and Goel optimal release time model

We have assumed the cost coefficients to be $K_1 = 1.25$, $K_2 = 3$ and $K_3 = 4$ (the cost in thousands). After solving Eq. (9), we get the optimal release time $t^* = 73.438$ and expected cost $K(t^*) = 1137.84$.

Case 2: Anand optimal release time model (Proposed)

The cost coefficients in the cost model have been assumed to be $C_1 = 0.5$, $C_2 = 0.75$, $C_3 = 3$, $C_4 = 4$ and the desired reliability is assumed to be 0.8. On solving Eq. (11), we get the optimal release time $t^* = 79.77$ and expected cost $C(t^*) = 1011.75$.

After solving both the cases, it is obvious that the optimal release time of a software incorporating beta testing coincide with the actual release time than the optimal time obtained on solving traditional releases policy. Also, when considering best testing activity the optimal time is much larger than the optimal time without beta testing. Moreover, while following the proposed strategy in determination of optimal release time, we observe that the total cost incurred is minimum when compared with the traditional cost model. The detailed view of the cost function versus testing time is illustrated in Fig. 5 (both traditional and proposed methodology).

7 Conclusion

With the availability and easy accessibility of web, beta testing has attained enormous growth in the software market. However, the advantages of beta testing have not been examined deeply. For a software developer, testing a software before

its launch is not only vital but also a critical task. The acceptability of a software by the end users can be only judged by exposing the software to real life usage. Beta testing can be considered an important task as any bugs, errors, or problems noticed during the beta tests will make the product viable, stable, and robust before its general availability. The developed SRGM is unique as it considers the faults discovered during the beta testing phase. Moreover, in this study we have emphasized on providing beta releases for software testing and determination of optimal release time. The modeling framework presented in this chapter aims to fill the void by releasing the software after examining the faults detected by the beta testers rather than releasing the product after the completion of in-house testing activities as considered in the previous literature. The outcomes from this study to develop model incorporating beta testing are very promising and showing an improvement in reliability and optimal release time. The accuracy of the proposed model has been verified by the numerical example.

Acknowledgments The research work presented in this chapter is supported by grants to the first author from University of Delhi, R&D Grant No-RC/2015/9677, and Delhi, India.

References

1. Press Release (2016) Gartner Says Worldwide Security Software Market Grew 3.7 Percent in 2015. http://www.gartner.com/newsroom/id/3377618. Accessed 12 August 2016
2. India IT-BPM Overview (2016) http://www.nasscom.in/indian-itbpo-industry. Accessed 15 June 2016
3. Levenson NG, Turner CS (1993) An investigation of the Therac-25 accidents. Computer 26 (7): 18-41
4. Dowson M (1997) The Ariane 5 software failure. Software Engineering Notes 22 (2): 84.
5. Rogerson S (2002) The Chinook helicopter disaster. IMIS Journal 12(2).
6. Charlotte Jee (2015) Top 10 software failures of 2014. http://www.computerworlduk.com/galleries/infrastructure/top-10-software-failures-2014-3599618/. Accessed 15 June 2016
7. Arora A, Caulkins JP, Telang R (2006) Research Note: Sell First, Fix Later: Impact of Patching on Software Quality. Management Science 52(3): 465-471
8. Kapur PK, Pham H, Gupta A, Jha PC (2011) Software Reliability assessment with OR application. Springer, Berlin
9. Software release life cycle (2016) https://en.wikipedia.org/wiki/Software_release_life_cycle. Accessed 15 June 2016
10. Pradhan T (2012) All Types of Software Testing. http://www.softwaretestingsoftware.com/all-types-of-software-testing/. Accessed 15 June 2016
11. Kocbek M, Hericko M (2013) Beta Testing of a Mobile Application: A Case Study. SQAMIA 29-34
12. Buskey CD (2005) A Software Metrics Based Approach to Enterprise Software Beta Testing Design. Dissertation, Pace University
13. Jiang Z, Scheibe KP, Nilakanta S (2011) The Economic Impact of Public Beta Testing: The Power of Word-of-Mouth. Supply Chain and Information Management Conference Papers: Posters and Proceedings Paper 10
14. Fine MR (2002) Beta testing for better software. John Wiley & Sons, New York
15. Bla1ze (2016) Snap v3 Beta sees over 11,000 downloads in the first 24 hours. http://crackberry.com/snap-v3-sees-over-11000-downloads-first-24-hours. Accessed 15 June 2016

16. Apple Press Info (2000) Apple Releases Mac OS X Public Beta. http://www.apple.com/pr/library/2000/09/13Apple-Releases-Mac-OS-X-Public-Beta.html. Accessed 15 June 2016
17. Bogdan Popa (2014) Windows 10 Preview Was Installed by 1.5 Million Users. http://news.softpedia.com/news/Windows-10-Preview-Was-Installed-by-1-5-Million-Users-467767.shtml. Accessed 15 April 2016
18. Posts by Gabe Aul (2016) http://blogs.windows.com/bloggingwindows/author/gabeaul/. Accessed 15 June 2016
19. Press Release (2005) Microsoft Windows Vista October Community Technology Preview Fact Sheet. http://www.microsoft.com/presspass/newsroom/winxp/WinVistaCTPFS.mspx. Accessed 15 June 2016
20. Terry Myerson (2015) Hello World: Windows 10 Available on July 29. http://blogs.windows.com/bloggingwindows/2015/06/01/hello-world-windows-10-available-on-july-29/. Accessed 15 June 2016
21. Fester P (2005) A long winding road out of beta. ZDNet. http://www.zdnet.com/article/a-long-winding-road-out-of-beta/. Accessed 15 June 2016
22. BlackBerry Beta Zone (2015) WhatsApp Messenger Bug Reports. https://appworld.blackberry.com/webstore/content/35142896/?lang=en. Accessed 15 June 2016
23. Goel AL, Okumoto K (1979) Time dependent error detection rate model for software reliability and other performance measures. IEEE Trans Reliability 28(3): 206–211
24. Kapur PK, Garg RB, Kumar S (1999) Contributions to hardware and software reliability. World Scientific Publishing Co. Ltd, Singapore
25. Musa JD, Iannino A, Okumoto K (1987) Software reliability: measurement, prediction, applications. McGraw Hill, New York
26. Yamada S, Ohba M, Osaki S (1984) S-shaped software reliability growth models and their applications. IEEE Trans Reliability 33(4): 289–292
27. Ohba M (1984) Software reliability analysis models. IBM J Res Dev 28: 428–443
28. Okumoto K, Goel AL (1983) Optimal release time for computer software. IEEE Trans Softw Eng SE 9(3): 323–327
29. Yamada S, Osaki S (1987) Optimal Software Release Policies with simultaneous Cost and Reliability Requirements. European Journal of Operational Research 31: 46-51
30. Kapur PK, Agarwal S, Garg RB (1994) Bicriterion release policy for exponential software reliability growth model. Proceedings of the 3rd International Symposium on Software Reliability Engineering 28:165–180
31. Jain M, Handa BR (2001) Cost analysis for repairable units under hybrid warranty. In: M.L. Agarwal, K. Sen, eds. Recent Developments in Operational Research. Narosa Publishing House: New Delhi 149–165
32. Singh O, Kapur PK, Anand A (2012) A Multi Attribute Approach for Release Time and Reliability Trend Analysis of a Software. International Journal of System Assurance and Engineering Management (IJSAEM) 3 (3): 246-254
33. Singh O, Aggrawal D, Kapur PK (2012) Reliability Analysis and Optimal Release Time for a Software using Multi-Attribute Utility Theory. Communications in Dependability and Quality Management -An International Journal 5(1): 50-64
34. Anand A, Agarwal M, Tamura Y, Yamada S (2016) Economic Impact of Software Patching and Optimal Release Scheduling. Quality and Reliability Engineering International. doi:10.1002/qre.1997
35. Das S, Anand A, Singh O, Singh J (2015) Influence of Patching on Optimal Planning for Software Release & Testing Time. Communication in Dependability and Quality Management: An International Journal 18(4): 82-93
36. Wiper MP, Wilson SP (2006) A Bayesian analysis of beta testing. TEST 15(1): 227-255
37. Mäkinen SJ, Kanniainen J, Peltola I (2014) Investigating Adoption of Free Beta Applications in a Platform-Based Business Ecosystem. Journal of Product Innovation Management 31(3): 451-465
38. Jiang Z, Sarkar S, Jacob VS (2012) Postrelease testing and software release policy for enterprise-level systems. Information Systems Research 23(3-part-1): 635-657

39. Kanoun K, Martini M, Souza J (1991) A method for software reliability analysis and prediction application to the TROPICO-R switching system. IEEE Trans. on Software Engineering 17 (4): 334–344
40. Huang CY, Lyu MR (2011) Estimation and analysis of some generalized multiple change-point software reliability models. IEEE Transactions on Reliability 60(2): 498-514
41. SAS Institute Inc (2010) SAS/ETS® 9.22 User's Guide. Cary, NC