Software Release and Patching Time with Warranty Using Change Point

Chetna Choudhary, P. K. Kapur, A. K. Shrivastava, and Sunil K. Khatri

1 Introduction

With the enormous use of information technology, computer systems are widely used to control safety-critical systems and day-to-day entities. Increased demand of high-quality software is the necessity in these application areas. To determine reliability of the system, the software reliability must be evaluated with utmost care. Developing reliable software is the main concern among the software industry. Proper scheduling of processes, limitation of resources, unrealistic requirements, etc. had a great impact on software reliability. Many softwares nowadays are interdependent among the modules; it is very hard to develop the consistent software. It is particularly tough as soon as there is interdependence, and it is rather difficult to identify whether there is reliable software being delivered. Once the software is delivered, one way to major its reliability is through customer feedback like problem being reported, system outages, complaints or compliments, etc. However, by then it is too late; software vendors need to know whether their products are reliable before they are shipped to customers. Software reliability models attempt to provide that information.

C. Choudhary (✉)
Amity School of Engineering & Technology, Amity University, Noida, UP, India
e-mail: chetna.choudhary13@gmail.com

P.K. Kapur • A.K. Shrivastava
Amity Center for Interdisciplinary Research, Amity University, Noida, UP, India
e-mail: pkkapur1@gmail.com; kavinash1987@gmail.com

S.K. Khatri
Amity Institute of Information Technology, Amity University, Noida, UP, India
e-mail: sunilkkhatri@gmail.com

P.K. Kapur et al. (eds.), *Quality, IT and Business Operations*, Springer Proceedings in Business and Economics, DOI 10.1007/978-981-10-5577-5_30

Various researchers are working on software reliability growth models to predict the reliability under different sets of parameters. These software reliability growth models generally follow non-homogenous Poisson process (NHPP) with consideration that a constant fault detection rate (FDR) and a fault detection process depend only on the residual fault content [3]. With consideration to specific environment, software programs are executed and improved with detection and correction of bugs/errors. Generally SRGMs take into consideration that, in fault detection process, the faults which occurred due to each failure are independent and random with time according to the distribution given by Musa et al. [12]. Factors such as running environment, testing strategy, defect density, and resource allocation can affect the failure distribution. To cover more faults within the shorter period of time, testing teams keep including new approaches and techniques not yet been used, or in consultation with some experts, we need to preform software risk analysis. Hence, the fault detection rate may not be continuously smooth and can be changed at some time moment τ known as change point. Foremost, Zhao [8] used the change point in software and hardware reliability. Then Huang et al. [13] included the change point to calculate the software reliability growth modeling with testing effort functions. Kapur et al. proposed the multiple change points in software reliability growth modeling for fielded software [13]. Looking into the importance of the change point models in the reliability estimation, an SRGM based on stochastic differential equations incorporating change point concept has been proposed by Kapur et al. [16].

Software requires testing to overcome the faults which in turn increase reliability. However which results in consumption of resources like work force, processing hours etc. which acquire more cost to the organization. However, in testing phase if bugs are not eliminated, then users will face the consequences of faults in the operational time which will increase the debugging cost of the software compared to the testing phase [2]. The delayed testing not only results in increase in reliability of software but also the development cost of software. This will further delay the software from releasing which cause thrashing in terms of market opportunity. On the other hand, insufficient testing intended for fault confiscation shortens the cost of software improvement but increase the hazard of more faults during operational phase. Hence it is significant to determine the optimal time to release the software from the point of developer.

Therefore, in order to minimize the associated risk, the organization needs proper scheduling of different prerelease and post-release phases with due consideration to the change point. From the many decades, researchers are working on software release time problem [1–3, 12, 19]. Yamada [15] anticipated a software cost model under warranty to determine the optimal time to release software. Dohi [4] projected release policies with due consideration to debugging time lag. Yamada and Osaki [5] proposed problem related to release time on the basis of deprecated cost and optimized reliability objective. Kapur and Garg presented the release policies based on testing effort by means of inclination toward intensity of failure [6]. They furthermore incorporated the concept of cost of penalty while considering software release time problem [7]. Huang and Huang and Lyu anticipated release time policies by considering the consequence of testing effort overheads [8]. Yun and Bai projected the software release time problems presuming software life cycle to be random [9]. With competitive market surroundings, a need to deliver a better product has increased. This need is captured by the users in terms of what we can call a software warranty, which acts as an indicator to judge its reliability with presumption that a longer warranty period indicates higher reliability [1]. Warranty is an assurance between the seller and buyer that a product is error-free and will work as stated, and if defect occurs, it's a vendor's job to rectify those defects [2]. The duration of warranty is decided by the developer. In warranty phase rectifying a bug then reporting a fault is negligible for the user. Pham and Zhang included service contract and hazard cost to the conventional cost function [10]. Kapur [11] refined a multi-objective maximization problem to determine release time. Kapur [11] formulated release policy for the exponential change point SRGM.

Though it is a difficult job for a testing team to develop a software which is completely free of errors. But in order to ensure that users face least number of errors during operation phases, today software industry continue to test even after release it. As software releases, it is noted that either the rate of failure decreases or else it is invariable, depending on whether there is augmentation consistency of software throughout the operational phase or not (see Figs. 1 and 2).

Figure 1 illustrates graph for reliability of software for the case when no testing is done after release. Hence there is no reliability growth. On the other hand, in Fig. 2 we see that there is growth in reliability after a certain time interval due to

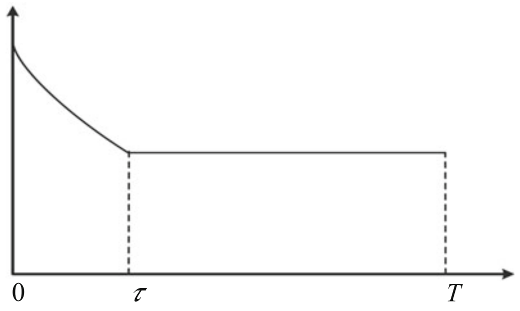

Fig. 1 Growth of software reliability without patching

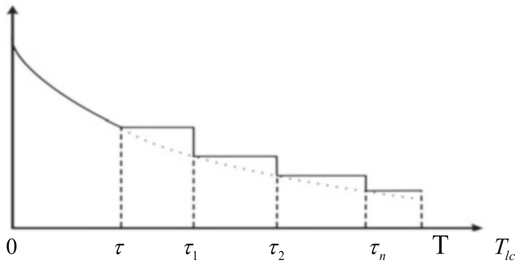

Fig. 2 Growth of software reliability including patching

release of patches/updates after release. This time interval is the difference between two patching times after release, and the update can also be termed as fix, an instruction set that is used to repair flaws which successfully deceives the testing side during prerelease testing phase of software. Well-timed delivery for updates is able to serve intention and will stipulate the premature release of patch results in indecent fixation of failures, and delayed updates would lead to more failures during operational phase and will incur more charge to the development side. Cavusoglu [20] considered the patch release for security and executive tribulations from the insight of both the seller and the organization. They formed a game theoretic model to gain knowledge of balancing the update, and cost management takes place between the merchant and organization. Consequently, Okamura [24] and Luo [21] considered a bug discovery process which is nonhomogeneous and examines both sporadic and asporadic update management models. Lately Dev [3] developed a cost model for finding optimal release policies for security patch management. Arora [22] has shown the transposition between releasing buggy software and saving in patching the buggy software later through a cost model. They have also worked on showing the disadvantage of releasing the software before time and updating later on. Jiang [17] estimated scheduling strategies of software where they have shown the advantage of early software release with continued testing after the release. Kapur [24] outlined a comprehensive framework to determine the optimum time to release and to stop testing software along with the benefit of early release and continuing testing even after release.

Based on the above literature, we see that no research was conducted to consider the effect of change point in finding the optimal time to release the software and updating the software under warranty. This gap analysis has led us to develop a model that can offer a solution to the above problem. In the proposed work, a generalized outline is provided to develop a cost model for determining the optimum time to release and to update software under warranty using change point; therefore, the total expected cost can be minimized. Remaining sections are structured as follows: In Sect. 2, we will talk about model formation specifying necessary presumptions essential for the model. Section 3 will give you an idea about the developed cost model using change point. In Sect. 4, numerical illustration is catered using a real-life data set to authenticate the projected work. Section 5 will discuss the final conclusion.

2 Framework of Modeling

This part of the section pays more attention on the structuring outlines for the proposed work. The following are the notations and presumptions worn in structuring the proposed work.

2.1 Notations

$m(t)$	Expected number of faults that are to be removed by time t or mean value of fault
A	Number of faults initially lying dormant in the software
b1	Rate of fault detection before change point
b2	Rate of fault detection after change point
$F(t)$	Fault distribution function
τ_c	Change point
τ	Release time of software
τ_p	Release time of patch
W	Length of warranty
T_{lc}	Life cycle of the software
C_1	Testing cost per unit time
C_2	Market opportunity cost
C_3	Fault detection/removal cost by tester before releasing the software
C_4	Fault detection/removal cost communicated by means of the user after release of software in warranty
C_5	Fault detection/removal cost communicated by means of the consumer after patch releasing in warranty period
C_6	Fault removal cost communicated by means of the user after warranty

2.2 Assumptions

1. Phenomenon for fault removal pursues NHPP whole time of the software life cycle.
2. Every time a fault is experienced, an instantaneous debugging attempt takes place to discover the cause of failure in order to confiscate it.
3. Number of faults left in the software equally affects the failure rate.
4. Fault fixation process is ideal and no fresh faults are being introduced during debugging process.
5. Process of fault detection autonomously and identically circulated.
6. Number of bug's latent in the software is unchanging.
7. Life cycle of software is finite.
8. Patch cost is negligible.
9. Opportunity cost is pretended to be invariably growing, twofold continually differentiable convex function of r. As per approximated outcome of the learning, the results to a great extent accelerated through definite functional outline of market opportunity cost; hence, we are taking into consideration the form given by Jiang and Sarkar (2011).

2.3 Devising the Model

Through hazard rate we formulate the mean value function for the augmented amount of faults eliminated which is shown as

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = \left(\frac{f(t)}{1 - F(t)}\right)(a - m(t)) \tag{1}$$

where $\frac{f(t)}{1-F(t)}$ is the failure detection rate.

On solving the above equation by considering the initial condition as $m(0) = 0$, we get

$$m(t) = \mathrm{a} \cdot F(t) \tag{2}$$

2.4 Model Framework

In the proposed framework, we have subdivided the software life cycle $[0, T_{\mathrm{lc}}]$ into five subparts, i.e., testing phase of software before change point phase $[0, \tau_c]$, software testing phase after change point $[\tau_c, \tau]$, releasing of patch $[\tau, \tau_p]$, phase of warranty $[\tau, \tau + w]$, and software operational phase $[\tau + \mathrm{w}, T_{\mathrm{lc}}]$. Software time frame can be depicted as in Fig. 3. Also it is to be mentioned that each subdivision of fault detection process follows NHPP phenomenon.

First Phase [0, $\boldsymbol{\tau}_c$] During the phase, testers are detecting or removing the bugs. The number of fault detected is given by

$$m\,(\tau_c) = a \cdot F\,(\tau_c) \tag{3}$$

where $F_1(\tau_{\mathrm{c}})$ rate of fault detection using these bugs is detached commencing from software before change point in $[0, \tau_{\mathrm{c}}]$.

Second Phase [$\boldsymbol{\tau}_c$, $\boldsymbol{\tau}$] For this interval change point has been considered as software and is ready to be released by the developer. The developers at some moment change the testing strategies to find more number of bugs in product. The number of bugs detached in this phase is shown as

$$m\,(\tau - \tau_c) = a \cdot (1 - F_1\,(\tau_c)) \cdot \left(1 - \frac{1 - F_2\,(\tau)}{1 - F_2\,(\tau_c)}\right) \tag{4}$$

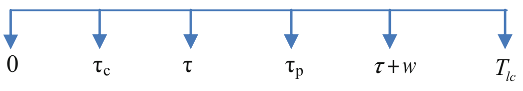

Fig. 3 Life cycle of software

where $a{\cdot}(1 - F_1(\tau_c))$ is the number of bugs removed after change point. $\left(1 - \frac{(1-F_2(\tau))}{1-F_2(\tau_c)}\right)$ is the fault removal rate after change point.

Third Phase [$\boldsymbol{\tau}$, $\boldsymbol{\tau}_p$] During this phase update is prepared in reference to the bugs detected by users. At τ_p, a fix/update will be provided to fasten the number of faults reported by users. The total number of faults detected in this phase is given below as

$$\begin{aligned} m\left(\left(\tau + \tau_p\right) - \tau\right) &= \left(a - m\left(\tau\right)\right) \cdot F_3\left(\left(\tau + \tau_p\right) - \tau\right) \\ &= \left(1 - \left[1 - \tfrac{1-F_1(\tau_c)\cdot(1-F_2(\tau))}{1-F_2(\tau_c)}\right]\right) \cdot F_3\left(\left(\tau + \tau_p\right) - \tau\right) \end{aligned} \tag{5}$$

In the above equation, $a\left(1 - \left[1 - \frac{(1-F_1(\tau_c))(1-F_2(\tau))}{1-F_2(\tau_c)}\right]\right)$ represents the outstanding bugs which were left hidden during the first and second phases. $F_3((\tau + \tau_\mathrm{p}) - \tau)$ is the rate for fault detection/removal by the user in $[\tau, \tau_\mathrm{p}]$.

Fourth Phase [$\boldsymbol{\tau}_p$, $\boldsymbol{\tau} + w$] In warranty phase the user faces failure because of the faults which were not corrected in the first patch. The amount of faults detected/removed by end of this phase is summarized as

$$\begin{aligned} m\left((\tau + w) - \left(\tau + \tau_p\right)\right) = a \cdot \left(1 - \left[1 - \tfrac{(1-F_1(\tau_c))(1-F_2(\tau))}{1-F_2(\tau_c)}\right]\right) \\ \cdot \left(1 - F_3\left(\tau + \tau_p\right) - (\tau)\right) \\ \cdot F_4\left((\tau + w) - \left(\tau + \tau_p\right)\right) \end{aligned} \tag{6}$$

In the above equation, $a \cdot \left(1 - \left[1 - \frac{(1-F_1(\tau_\mathrm{c}))(1-F_2(\tau))}{1-F_2(\tau_\mathrm{c})}\right]\right) \cdot \left(1 - F_3\left(\tau + \tau_\mathrm{p}\right) - (\tau)\right)$ represent the outstanding faults remained hidden during the second and third intervals. $F_4((\tau + w) - (\tau + \tau_\mathrm{p}))$ is the rate of fault removal during the interval $[\tau_\mathrm{p}, \tau + w]$.

Fifth Phase [$\boldsymbol{\tau} + w, T_{lc}$] During the post warranty interval, the amount of bugs discovered is summarized

$$\begin{aligned} m\left(T_{lc} - (\tau + w)\right) = a\left(1 - \left[1 - \tfrac{(1-F_1(\tau_c))(1-F_2(\tau))}{1-F_2(\tau_c)}\right]\right) \\ \left(1 - F_3\left(\tau + \tau_\mathrm{p}\right) - \tau\right)\right)\left(1 - F_4\left((\tau + w) - \left(\tau + \tau_p\right)\right)\right) \\ F_5\left(T_\mathrm{lc} - (\tau + w)\right) \end{aligned} \tag{7}$$

In this interval, the faults, those that are left undiscovered during last intervals, can cause failure, and the same will be communicated by the user for elimination. The amount of faults left is substituted as

$$a \cdot \left(1 - \left[1 - \frac{(1-F_1(\tau_c))\cdot(1-F_2(\tau))}{1-F_2(\tau_c)}\right]\right)$$
$$\left(1 - F_3\left(\tau + \tau_p\right) - \tau\right)\right)\left(1 - F_4\left((\tau + w) - \left(\tau + \tau_p\right)\right)\right).$$

$F_5(T_{\mathrm{lc}} - (\tau + w))$ expresses the rate of fault removal during the interval $[\tau + w, T_{\mathrm{lc}}]$.

Now, the cost model will be formulated in the next section to calculate the total cost consumed for the fault removal as per the Eqs. (3, 4, 5, 6, 7) phase wise.

3 Cost Model

Through this section we put emphasis on finding the amount of cost incurred for "a" initial number of faults detected throughout the software life cycle. In this we will discuss briefly cost related to testing, market opportunity, and amount spent in removing the faults per phase.

Cost of Testing The amount incurred while testing the software till release via testing side is defined as cost of testing. The cost for this phase is measured by doing product of testing cost with the time duration for which the testing is conducted. Let c_1 denote the testing cost per unit time, and τ is the release time of the software as per the proposed model. So the sum of testing cost is represented by

$$c_1 \cdot \tau \tag{8}$$

Market Opportunity Cost This can be termed as the cost incurred due to postponement in release of the software. Market opportunity cost is inversely proportional to the release of the software. More time taken for the release will increase the opportunity cost. This can be represented by the below equation, which is same as provided by Jiang and Sarkar [17]:

$$c_2 \cdot \tau^2 \tag{9}$$

Software Release Cost Before Change Point For the interval $[0, \tau_c]$, testers work separately for detecting the bugs in the software. This amount, i.e., the fault removed by testers per unit time is expressed asc_3. As obtained using Eq. (3), the number of faults detected/removed by the testing team before change point is represented by $m(\tau_c)$. Hence the cost can be formulated in the interval $[0, \tau_c]$as

$$Cost_{0,\tau_c} = c_3 \cdot m\left(\tau_c\right) \tag{10}$$

Software Release Cost After Change Point In the given interval $[\tau_c, \tau]$, the tester is functioning independently for detection process. Therefore, the cost of removing the bugs by testers is expressed asc_4. Now, the cost incurred for the interval is deliberated by multiplying cost with the faults detected. The total cost of removing the faults is specified by

$$Cost_{\tau,\tau_c} = c_4 \cdot m\,(\tau - \tau_c) \tag{11}$$

Patch Release Cost Because of the increased market competition, organizations cannot risk in delaying software release. Hence software is released with some faults still present in phase of testing. Because of this the users are provided warranty on the software to fix the faults with updates/patch by the developer during operational phase. Service contract is assurance provided to users about software that it will continue to perform efficiently during operational phase; otherwise, the organization will provide solution in the form of updates for the failure-causing faults, or they will replace the product. On facing failure within software, the users report the problem to the organization for fixation. The organization has to again test the software which is subject to additional charge to the firm. Using Eq. (5), the total amount of bugs is expressed by $m((\tau + \tau_p) - \tau)$ communicated by the user and solved by the tester. Suppose c_5 represents the fault removal, cost per unit time and the cost incurred for the interval are shown as

$$cost_{\tau,\tau_p} = c_5 \cdot m\left(\left(\tau + \tau_p\right) - \tau\right) \tag{12}$$

Software Under Warranty Cost With the patch release, the user still may find some faults during the interval $[\tau_p, \tau + w]$ because of the faults remaining in the patch phase. Due to warranty of the software, firms need to confiscate the failure-causing faults. By using Eq. (6), the amount of fault detached subsequent to update release and is denoted by $m((\tau + \mathrm{w}) - (\tau + \tau_p))$. Suppose c_6 represents the removal cost per unit fault, then the detection cost can be expressed as

$$cost_{\tau_p,\tau+w} = c_6 \cdot m\left((\tau + w) - \left(\tau + \tau_p\right)\right) \tag{13}$$

Post Warranty Cost As the software warranty gets over, the organization does not provide any support to any kind of failures that occurred, and sometimes they make agreement with the user for removing the faults all the way through the software life cycle. This cost has been incorporated while modeling the cost function. During $[\tau + w, T_{lc}]$, failures are encountered only by the users due to number of faults remained unobserved post warranty period. Using Eq. (7), we can obtain the total number of faults. Let $c_7$denote the removal cost of faults so the total cost incurred is given by

$$Cost_{\tau+w,T_{lc}} = c_7 \cdot m\,(T_{lc} - (\tau + w)) \tag{14}$$

Total Cost This can be summed as all the cost used throughout the life cycle of software. By adding together the entire cost as per Eqs. (8, 9, 10, 11, 12, 13, 14), the total cost function can be expressed as

$$totalcost = c_1 \cdot \tau + c_2 \cdot \tau^2 + Cost_{0,\tau_c} + Cost_{\tau,\tau_c} + Cost_{\tau,\tau_p} + Cost_{\tau_p,\tau+w} + Cost_{\tau+w,T_{lc}} \tag{15}$$

In the subsequent section, validation of the projected model has been performed through a numerical illustration.

4 Numerical Illustration

On the basis of supposition that the fault is identical and is autonomously distributed, it is clear that the rate of fault removal follows exponential distribution given as $F_i(t) = 1-e^{-b_i \cdot t}$ where b_i denotes rate of fault detection in "i" interval. The parameters are estimated using Woods [18] data set. As per the data, the software is being tested for 20 weeks to detect 100 bugs for the given duration. The change point analyzer is used to find the change point of the data sets which is taken to be eighth week. The parameter estimation is done using the SPSS on the basis of least square method on the data set that gives 'a' = 356.937 and $b_1 = 0.0419$ and $b_2 = 0.1326$. This depicts that software initially has 357 faults which were detected/removed by the testing team with rate b_1 and b_2. We have assumed the software life cycle $T_{lc} = 100$.Usually rate of fault detection by the user and by tester is dissimilar due to different competence. Suppose $r1$, $r2$, $r3$ denotes fault detection rate ratio for the user with respect to tester in third, fourth, and fifth intervals.

The sum of faults detected/removed in interval $[0, \tau_c]$ represented as

$$m(\tau_c) = a\left(1 - e^{-b_1 \cdot \tau_c}\right)$$

Therefore from Eq. (10) the cost associated with $m(\tau_c)$ number of faults is specified as

$$c_3 \cdot a(1 - \exp(-b_1 \cdot \tau_c)) \tag{16}$$

Using Eq. (11), we get the faults removed in interval $[\tau_c, \tau]$ as $a \cdot (1-(1-\exp(-b_1 \cdot \tau_c)))\left(1 - \frac{(1-(1-\exp(-b_2 \cdot \tau)))}{1-(1-\exp(-b_2 \cdot \tau_c))}\right)$ and the cost associated cost with $m(\tau - \tau_c)$ is specified as

$$c_4 \cdot a(1-(1-\exp(-b_1\tau_c)))\left(1 - \frac{(1-(1-\exp(-b_2\tau)))}{1-(1-\exp(-b_2\tau_c))}\right) \tag{17}$$

Amount of faults detached in $[\tau, \tau_p]$ is expressed as

$$m\left(\left(\tau + \tau_p\right) - \tau\right) = a\left(1 - \frac{(1-(1-\exp(-b_1 \cdot \tau_c))) \cdot (1-(1-\exp(-b_2 \cdot \tau)))}{1-(1-\exp(-b_2 \cdot \tau_c))}\right) \cdot \left(1 - \exp\left(-b_2 \cdot r_1 \cdot \left(\tau + \tau_p - \tau\right)\right)\right)$$

and associated cost associated with $m((\tau+\tau_{\mathrm{p}})-\tau)$ number of faults is represented by

$$c_5\cdot a\cdot\left(1-\tfrac{(1-(1-\exp(-b_1\cdot\tau_{\mathrm{c}})))\cdot(1-(1-\exp(-b_2\cdot\tau)))}{1-(1-\exp(-b_2\cdot\tau_{\mathrm{c}}))}\right)\cdot\left(1-\exp\left(-b_2\cdot r_1\cdot\left(\tau+\tau_{\mathrm{p}}-\tau\right)\right)\right) \tag{18}$$

Amount of faults detached in $[\tau_{\mathrm{p}},\tau+w]$ is represented by

$$\begin{aligned}&m\left((\tau+\mathrm{w})-\left(\tau+\tau_p\right)\right)\\&=a\cdot\left(1-\frac{(1-(1-\exp(-b_1\cdot\tau_c)))\cdot(1-(1-\exp(-b_2\cdot\tau)))}{1-(1-\exp(-b_2\cdot\tau_{\mathrm{c}}))}\right)\\&\cdot\left(1-\left(1-\exp\left(-b_2\cdot r_1\cdot\left(\tau+\tau_p-\tau\right)\right)\right)\right)\\&\cdot\left(1-\exp\left(-b_2\cdot r_2\cdot\left((\tau+w)-\left(\tau+\tau_p\right)\right)\right)\right)\end{aligned}$$

and the associated cost associated with $m((\tau+w)-(\tau+\tau_P))$given number of faults is given by

$$\begin{aligned}&c_6\cdot a\cdot\left(1-\tfrac{(1-(1-\exp(-b_1\cdot\tau_c)))\cdot(1-(1-\exp(-b_2\cdot\tau)))}{1-(1-\exp(-b_2\cdot\tau_{\mathrm{c}}))}\right)\\&\cdot\left(1-\left(1-\exp\left(-b_2\cdot r_1\cdot\left(\tau+\tau_{\mathrm{p}}-\tau\right)\right)\right)\right)\\&\cdot\left(1-\exp\left(-b_2\cdot r_2\cdot\left((\tau+w)-\left(\tau+\tau_{\mathrm{p}}\right)\right)\right)\right)\end{aligned} \tag{19}$$

Also the amount of faults removed in $[\tau+w,T_{\mathrm{lc}}]$ is given by

$$\begin{aligned}m\left(T_{\mathrm{lc}}-(\tau+w)\right)=a\cdot&\left(1-\tfrac{(1-(1-\exp(-b_1\cdot\tau_{\mathrm{c}})))\cdot(1-(1-\exp(-b_2\cdot\tau)))}{1-(1-\exp(-b_2\cdot\tau_{\mathrm{c}}))}\right)\\&\cdot\left(1-\left(1-\exp\left(-b_2\cdot r_1\cdot\left(\tau+\tau_{\mathrm{p}}-\tau\right)\right)\right)\right)\\&\cdot\left(1-\left(1-\exp\left(-b_2\cdot r_2\cdot\left((\tau+w)-\left(\tau+\tau_{\mathrm{p}}\right)\right)\right)\right)\right)\end{aligned}$$

and the cost associated with $m(T_{\mathrm{lc}}-\tau+\mathrm{w})$ given amount of faults is shown as

$$\begin{aligned}&c_7\cdot a\cdot\left(1-\tfrac{(1-(1-\exp(-b_1\cdot\tau_{\mathrm{c}})))\cdot(1-(1-\exp(-b_2\cdot\tau)))}{1-(1-\exp(-b_2\cdot\tau_{\mathrm{c}}))}\right)\\&\cdot\left(1-\left(1-\exp\left(-b_2\cdot r_1\cdot\left(\tau+\tau_p-\tau\right)\right)\right)\right)\\&\cdot\left(1-\left(1-\exp\left(-b_2\cdot r_2\cdot\left((\tau+w)-\left(\tau+\tau_p\right)\right)\right)\right)\right)\end{aligned} \tag{20}$$

By taking phase wise testing cost consequently, we obtain the total cost with single patching specified by

$$
\begin{aligned}
\mathit{totalcost} = {} & c_1 \cdot \tau + c_2 \cdot \tau^2 + c_3 \cdot a\Big(1 - \exp\left(-b_1 \cdot \tau_c\right) \\
& + c_4 \cdot a\left(1 - \left(1 - \exp\left(-b_1 \tau_c\right)\right)\right)\left(1 - \tfrac{(1-(1-\exp(-b_2 \tau)))}{1-(1-\exp(-b_2 \tau_c))}\right) \\
& + c_5 \cdot a \cdot \left(1 - \tfrac{(1-(1-\exp(-b_1 \cdot \tau_c)))\cdot(1-(1-\exp(-b_2 \cdot \tau)))}{1-(1-\exp(-b_2 \cdot \tau_c))}\right) \\
& \quad \left(1 - \exp\left(-b_2 \cdot r_1 \cdot \left(\tau + \tau_p - \tau\right)\right)\right) \\
& + c_6 \cdot a \cdot \left(1 - \tfrac{(1-(1-\exp(-b_1 \cdot \tau_c)))\cdot(1-(1-\exp(-b_2 \cdot \tau)))}{1-(1-\exp(-b_2 \cdot \tau_c))}\right) \\
& \cdot \left(1 - \left(1 - \exp\left(-b_2 \cdot r_1 \cdot \left(\tau + \tau_{\mathrm{p}} - \tau\right)\right)\right)\right) \\
& \cdot \left(1 - \exp\left(-b_2 \cdot r_2 \cdot \left(\left(\tau + w\right) - \left(\tau + \tau_{\mathrm{p}}\right)\right)\right)\right) \\
& + c_7 a \cdot \left(1 - \tfrac{(1-(1-\exp(-b_1 \cdot \tau_c)))\cdot(1-(1-\exp(-b_2 \cdot \tau)))}{1-(1-\exp(-b_2 \cdot \tau_c))}\right) \\
& \cdot \left(1 - \left(1 - \exp\left(-b_2 \cdot r_1 \cdot \left(\tau + \tau_p - \tau\right)\right)\right)\right) \\
& \cdot \left(1 - \left(1 - \exp\left(-b_2 \cdot r_2 \cdot \left(\left(\tau + w\right) - \left(\tau + \tau_p\right)\right)\right)\right)\right) \\
& \cdot \left(1 - \exp\left(-b_2 \cdot r_3 \cdot \left(T_{\mathrm{lc}} - \left(\tau + w\right)\right)\right)\right)
\end{aligned}
\tag{21}
$$

now defining the cost parameter values before doing the optimization of the above given equation and providing all cost values as $c_1 = 30$, $c_2 = 5$, $c_3 = 20$, $c_4 = 30$, $c_5 = 55$, $c_6 = 55$, and $c_7 = 135$. Also let us suppose that the organization generally provides the user with warranty of 6 months, i.e., w = 24 (weeks) on the software and fault detection rate ratios r1 = 0.5, r2 = 0.5, and r3 = 0.6. It should be noted that values obtained are purely through experience. In general cost of per unit fault during a given phase is directly relative to the ratio of assets consumed to the removed fault within that period. It is to be noted that post warranty period of software is much longer as compared to any other phases of software life cycle; hence it consumes high amount of resources. Also the number of faults removed during this period is low as the maximum number of faults is debugged during testing and warranty phase. Hence cost per unit fault removal during post warranty period is highest. Based on the similar arguments, magnitude of cost per unit fault removal in other phases can be described. Now based on the above assumptions, we will describe the phase-wise fault detection with the associated cost. This is to inform that the service contract and update time be measured since time of release τ, because after release only updates and warranty can be provided to the customer. By substituting the cost values in Eq. (21) and performing optimization using MAPLE for the total cost function, we acquire optimal outcome as software release time $\tau^* = 21.19$ weeks; optimal time to release the patch $\tau_p^* = 1$ and the optimized cost value as 14385.69. The below given table (Table 1) summarizes the fault detection by the tester as well as user with the help of over mentioned values phase wise.

Table 1 Descriptions of faults phase wise

Phase	MVF	Number of removed faults
Software time to release before change point $[0, \tau_c]$	$m(\tau_c)$	101.83 (102 approx.)
Software release time before change point $[\tau_c, \tau]$	$m(\tau - \tau_c)$	210.7753 (211 approx.)
First patch release time $[\tau, \tau_p]$	$m((\tau + \tau_p) - \tau)$	2.8446 (3 approx.)
Warranty time $[\tau_p, \tau + w]$	$m((\tau + w) - (\tau + \tau_p))$	32.4592 (32 approx.)
Operational time $[\tau + w, T_{lc}]$	$m(T_{lc} - \tau + w)$	8.90770

5 Conclusion

Through this paper we anticipated a comprehensive cost model to find the optimum time for release and for patch of software sold beneath warranty using change point subsequently to reduce the total anticipated software expenditure. In the existing effort, we have carefully calculated the case of distinct patching; however, from the similar lines of anticipated model, we can broaden our work for multiple patches. In the future we are capable to expand our model by taking into consideration the budget and consistency constraint on the optimum time to release and to patch. This model provides information to the developers that how much more testing is required to test the software after changing the testing strategies after a moment of time τ, and due to the repetitive nature of updates, developers can release the subsequent patches less costly. As more failure reports are sent by the user or the tester, it will provide more opportunities for the developer to detect and further confiscate the corresponding fault. When the next update is released and executed, software failure rate will be reduced accordingly.

References

1. Kapur PK, Khatri SK, Singh O, Shrivastava AK (2014) When to stop testing under warranty using SRGM with change point. In the IEEE Xplore conference proceeding of International Conference on IT in Business, Industry & Govt., CSIBIG held during March 8–9, 2014 at Sri Aurobindo Institute of Technology Indore Ujjain Highway Indore, pp 200–205
2. Kapur PK, Pham H, Gupta A, Jha PC Software reliability assessment with OR applications (Springer Series in Reliability Engineering)
3. Goel AL (1985) Software reliability models: assumptions, limitations and applicability. IEEE Trans Softw Eng SE-ll:1411–1423
4. Dohi T, Kaio N, Osaki S (1997) Optimal software release policies with debugging time lag. Int J Reliability, Quality and Safety Engineering 04(03)
5. Yamada S, Osaki S (1987) Optimal software release policies with simultaneous cost and reliability requirements. Eur J Oper Res 31:46–51
6. Kapur PK, Garg RB (1991) Optimal software release policies for software systems with testing effort. Int J Syst Sci 22(9):1563–1571

7. Kapur PK, Garg RB (1989) Cost-reliability optimum release policies for software system under penalty cost. Int J Syst Sci 20:2547–2562
8. Zhao M (1993) Change-point problems in software and reliability. Commun Stat Theory Methods 22(3):757–768
9. Yun WY, Bai DS (1990) Optimum software release policy with random life cycle. IEEE Trans Reliab 39(2):338–353
10. Pham H, Zhang X (1999) A software cost model with warranty and risk costs. IEEE Trans Comp 48(1):71–75
11. Kapur PK, Agarwal S, Garg RB (1994) Bi-criterion release policy for exponential software reliability growth models. Rech Oper/Oper Res 28:165–180
12. Musa JD, Iannino A, Okumoto K (1987) Software reliability: measurement, prediction, applications. Mc Graw Hill, New York
13. Kapur PK,Singh VB, Anand S (2007) Software reliability growth model of fielded software based on multiple change-point concept using a power function of testing time. In: Kapur PK, Verma AK (eds) Quality reliability and infocom technology. MacMillan India Ltd., New Delhi, pp 171–178
14. Kapur PK, Garg RB, Aggarwal AG, Tandon A (2009) General framework for change-point problem in software reliability and related release time problem. In proceedings of ICQRIT
15. Yamada S (1994) Optimal release problems with warranty period based on a software maintenance cost model. Trans IPS Jpn 35(9):2197–2202
16. KapurPK, SinghVB, Sameer A (2007) Effect of Change-Point on Software Reliability Growth Models Using Stochastic Differential Equations. Published in the proceedings of 3rd International Conference on Reliability and Safety Engineering. Misra RB, Naikan VNA, Chaturvedi SK Goyal NK (eds), (INCRESE-2007), Udaipur, pp 320–333
17. Jiang Z, Sarkar S, Jacob VS (2012) Post-release testing and software release policy for enterprise-level systems. Inf Syst Res 23(3), Part 1 of 2, 635–657
18. Wood A (1996) Predicting software reliability. IEEE Comput 29:69–77
19. Ompal S, Kapur PK, Shrivastava AK, Kumar V (2015) Release time problem with multiple constraints. Published in Int J Syst Assur Eng Manag 6(1):83–91
20. Cavusoglu H, Cavusoglu H, Zhang J (2008) Security patch management: share the burden or share the damage? INFORMS 54:657–670
21. Luo C, Okamura H, Dohi T (2015) Optimal planning for open source software updates. Proc IMechE Part O. doi:10.1177/1748006x15586507
22. Arora A, Caulkins JP, Telang R (2006) Research note: sell first, fix later: impact of patching on software quality. Manag Sci 52(3):465–471
23. Okamura H, Tokuzane M, Dohi T (2009) Optimal security patch release timing under non-homogeneous vulnerability-discovery processes. Proceedings of the 20th International Symposium on Software Reliability Engineering (ISSRE'09), Mysuru, pp 120–128
24. Kapur PK, Shrivastava AK (2015) When to release and stop testing of a software: a new insight, published in Conference Proceedings of International Conference on Reliability, Infocom Technology and Optimization (Trends and Future Directions), held during October 2–4, 2015 at Amity University, Uttar Pradesh, pp 1–6