Release and Testing Stop Time of a Software: A New Insight

P.K. Kapur[1], A.K. Shrivastava[2]

[1]*Center for Interdisciplinary Research, Amity University, Noida, U.P. , India*
[2]*Department of Operational Research, University of Delhi, Delhi, India*
[1]*pkkapur1@gmail.com* ,[2]*kavinash1987@gmail.com*

***Abstract*: Testing is a vital phase in the software development life cycle. But, the way it is performed, varies from one organization to another.** *One of the prime concern in software industry is to determine the optimal duration of testing.* **Both researchers as well as software developers have been working towards solving this issue since long. The duration of testing is directly proportional to its reliability level but prolonged testing costs a lot in terms of higher testing and market opportunity cost. Therefore determination of optimal testing time has become an important optimization problem in the field of software development. As a common industrial practice, software release also marks the end of testing phase of a software. But, this often accompanies issues like delayed release in case the testing is continued to ensure a high reliability level or a low reliability level in case the software is released early. To counter these problems, now a days testing is divided into two phases i.e. pre-release and post release testing phase. During post release testing phase organization aims at treating remaining software faults and subsequently enhance product experience for customers. In this paper we present a generalized approach of optimal scheduling policy to determine the optimal release and testing stop time of a software while minimizing overall testing cost. In our proposed work, software testing & operational phases are governed by different distribution functions in distinct phases, i.e. in prerelease, post release phase (before and after testing stop time) in our proposed cost model. Numerical example is given to support our findings with the help of a real life software failure data set of Tandem Computers.**

Keywords: **Software Reliability, Release, Testing Stop Time**

I. INTRODUCTION

During the software development process, there are various questions on which management have to take decision. One such crucial question is deciding onto the release policies and the optimal time to stop software testing. Release of software is dependent on testing phase as, software testing points out the defects that are embedded during the development cycle of a software. Rigorous testing ensures software reliability and hence increases customer's confidence in the product. Testing plays an important role in determination of software release time. Management is dependent upon its testing team which is responsible for finding bugs in the software and their removal before the software is unveiled in the market. Hence, before software release it should be tested well so that it satisfies users requirement as well as management expectations.

978-1-4673-7231-2/15/$31.00 ©2015 IEEE

For this software is tested by a testing team which is not the part of software development. They test all the software from all aspects so that software works properly in any circumstances. Testers use the software like users do in the field to check if there are any unexpected results. This testing is done to ensure the quality of the software product in the operational phase.

There are several reasons which clearly tell us as why Software testing is important. Some of the major reasons are listed below:

1. Software testing is really required to point out the defects and errors that were made during the development phases.

2. It's essential since it makes sure of the Customer's reliability and their satisfaction in the application.

3. It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence.

4. Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.

5. Testing is required for an effective performance of software application or product.

6. It's important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development.

7. It's required to stay in the business.

Moreover, many bugs that are found during the testing phase can be used as ideas for feature enhancement. Errors found in the earlier stages of the development reduce the cost of production. Hence it is very important to find the errors as early as possible. This could be done by reviewing the specification documents or by walkthrough. The downward flow of the defect will increase the cost of production. Hence testing can prove to be corrective as well as innovative phase of a software lifecycle.

But prolonged testing to find all the bugs in the software is not feasible in the current market situation. As spending time on testing guarantee software reliability but on the other hand it costs a lot to the firm in terms of market opportunity cost and due to market competition. Hence there is a trade-off between minimizing cost and at the same time providing reliable software to the users.

In the existing literature on software testing, it was assumed that testing stops with release of the software. But, this process can be further improved by releasing the software earlier to get the advantage of market opportunity cost and the continue testing even after software release. Market opportunity cost is the benefit in terms of monetary value, market capture, sales etc. which the firm could have received by releasing the software earlier. But early release of the software may land in low software reliability which can cause more number of failures in field and hence goodwill loss fro firm. Hence testing after release is also important to ensure reliability in field. On the other hand, a late release leads to consumption of more resources and hence increase the production cost. Moreover, late release might end up in the loss of market opportunities. Therefore the decision of release and testing stop time has utmost importance from firm's point of view.

In current industry practices, testing after release for fixing the bug by patch release is in trend. Patching is a common phenomenon now that is followed by almost all the software firms. A patch, sometimes also called a *fix*, is a small program of software that is used to correct errors that successfully deceived the testing team during pre-release testing phase of a software. Patches are used to update our software and helps to protect the software from unwanted or unexpected functioning which can cause failure. These patches serve a number of different functions like fixing security holes, optimizing the utilization of resources in the software system, add newer and more secure features, remove old and unprotected features, update drivers to increase software efficiency and many more.

Now, the important question that strikes the mind of every product manager is how to decide the optimal periods of these phases i.e. release and testing stop time of software. Figure 1 and 2 shows the existing and current situations of software release and testing stop time problem. In figure 1 software release time coincides with the testing stop time of software. Whereas figure 2 shows the current practices involved in software industry, in which software is released earlier and testing continues after release and stops at a point where testing team is sure for the software reliability.

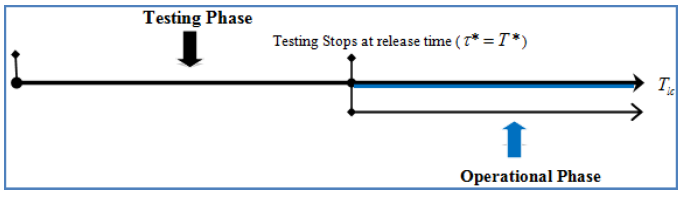

Fig. 1 Optimal Release Time Without Patching

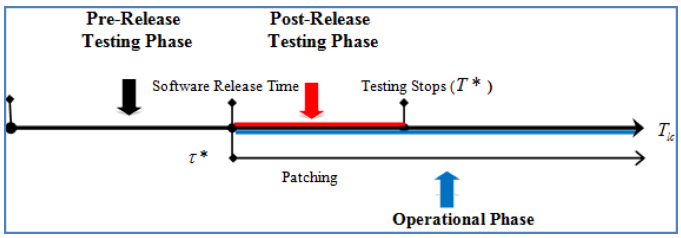

Fig. 2 Optimal Release Time With Patching

In the second strategy errors encountered are corrected and bundled as a software patch which is presented to the user. The idea of early release of software and to continue testing after release is to increase their testing base from a limited number of testers to significantly large number of testers which contributes to customer side testing of the software. To find the solution of above mentioned release time problem mathematical models are used. These models help in the process of making these crucial decisions for the organization. Software Reliability Growth Models (SRGMs) have been put to use in order to formulate and solve many optimization problems in literature. These models help to determine the reliability growth by testing with time.

Since the software release time is such a crucial decision to make, the topic has been extensively studied in the past two decades and a number of optimal release policies have been proposed in the literature [1,2,3,11,12,26]. There are many attributes on which a firm has to think of before making the final releasing a software. As, shipping the software too early might result in pendency of faults in the software and on the other hand, if testing proceeds for a very long time, the surety of reliable product increases but the cost of testing, contract penalty and loss of market initiative may constitute an even larger portion of the cost of late delivery. Hence, both, economic factors and technical factors have to be taken care of while deciding the optimal release time of the software product. Due to the complexity of software architectures, it is becoming increasingly difficult for the software developers to produce reliable systems efficiently. There are many examples in the existing literature that shows the importance of working on software reliability and its failure. Some recent major computer system failures caused by software bugs are as follows [6]:

a) In early 2014, a Swiss bank found itself in huge trouble when their customers reported of receiving statements containing details for other bank clients along with their own statement due to a software glitch which led to issuing of bank statements addressed to the wrong people.

b) Thousands of law students across United States poured their efforts during an exam only to find that the software denied accepting their submissions. This happened during August 2014 which took a turmoil both on the students and organizing council.

c) In late 2014, Amazon faced a huge setback of $100,000 due to a bug in its price comparison software which landed up as a festive bonanza for buyers and they picked up items as expensive as mobile phones for a single penny.

Practically it is impossible for a testing team to come out with an error free software. Moreover, the smallest of bug can drive an organization towards great losses. The best thing an organization can do in order to minimize the associated risk is by proper scheduling of different prerelease and post release phases.

Several researchers have proposed quantitative methods to estimate the number of defects and their distribution in a software [11,19,23,26]. These models known as Software Reliability Growth Models (SRGMs) are used to frame several software related optimization problems. SRGMs provide mathematical relationship between various attributes that affect the process of testing. The simplest SRGM was suggested by Okumoto and Goel is based on exponential distribution function [5]. They considered an unconstrained cost objective in the first method and an unconstrained reliability objective in another. Dohi proposed optimal software release policies with debugging time lag[1]. Yamada and Osaki [28] examined NHPP based release time problems with cost minimization and reliability maximization objective. Kapur and Garg [16] discussed testing effort based with release policies with aspiration level of failure intensity. They also introduced the concept of penalty cost in modeling releasing time problem of software [17]. Huang and Lyu [7,8] discussed release policies considering the effect of testing effort expenditure. Yun and Bai [30] proposed that software release time problems assuming software life cycle to be random. Later Kapur [14] developed release time problem to minimize cost of a software having random life cycle, subject to achieve a desired level of intensity assuming. Further, Kapur and Garg [15] developed a software cost model incorporating the cost of dependent faults along with independent faults. Pham and Zhang [27] incorporated warranty and risk cost to the traditional cost function. Later on some researchers worked on bi-criterion release time problem, Such as Kapur [13] developed a multi-objective optimization problem for determination of release time.

Today, no software ends up with a single release. Multiple upgradations of software is a general phenomenon. Many researchers [18,20, 24] analyzed this practice and worked to model this phenomenon mathematically. They worked towards finding the optimal release time of multi up-graded software. Kapur [20,21] proposed a cost model for removing the faults when there are successive releases of the software. As a general practice in today's software industry, software fixes are made available to the users in between two successive releases to ensure a bug free operation of the software. These fixes are remedial programs for errors detected and corrected during post release testing phase of a software. They are made available to the users as patches. In this paper, we analyzed the idea of patching and related policies as a software developer and aimed at minimizing the cost at the vendor side. As per the concept of patching, a software can be released before testing is complete (as investigated using earlier NHPP based software reliability models). This idea is further supported by Jiang and Sarkar [9]. Following this concept, there is a time period when both developer and users simultaneously perform the task of testing the software. Rest of the article is organized as follows: section-II consists of notations, assumptions along with the modeling framework for the cost without and with patching. Numerical analysis of the proposed mathematical cost is supplemented in section-III. Finally Conclusion is drawn in section-IV.

II. MODELING FRAMEWORK

A. Notations

$m(t)$	The mean value function or the expected number of faults removed by time t.
a	Constant, representing the initial number of faults lying dormant in the software when the testing starts.
$m(T_{lc})$	Number of faults removed during the lifecycle of the software
b	Failure detection/correction rate.
$F(t)$	Distribution functions for fault correction.
$h(t)$	Time dependent fault correction rate per remaining faults.
τ	Release Time of the software
T	Testing stop time of the software
c_1	Cost of testing per unit time
c_2	Market opportunity cost
c_3	Cost of debugging a fault by the testers before the release of the software
c_4	Cost of debugging a fault by the testers after the release of the software (i.e in operational phase) when it is being reported by one of the users
c_5	Cost of debugging a fault by the testing team after release of the software when the failure is detected by the testing team
c_6	Cost of debugging a fault reported by user after testing stop time

B. Assumptions

The proposed model is based upon the following basic assumptions:

1. Failure observation / fault removal phenomenon is modeled by NHPP.

2. Software is subject to failures during execution caused by faults remaining in the software.

3. Each time a failure is observed, an immediate debugging effort takes place to find the cause of the failure in order to remove it.

4. Failure rate is equally affected by all the faults remaining in the software.

5. All faults are removed perfectly.

6. Total numbers of faults lying dormant in the software are finite.

7. Lifecycle of the software is finite.

8. Cost of patching is negligible.

Market opportunity cost which is assumed to be monotonically increasing, twice continuously differentiable convex function of τ . Since the qualitative conclusion of the study is not much affected by the actual functional form market opportunity cost therefore we will use the form used by Jiang and Sarkar [9].

Using above assumptions and hazard rate approach for deriving the mean value function of cumulative number of faults removed, we have:

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

Where $h(t) = \frac{f(t)}{1-F(t)}$ is the time dependent fault correction rate per remaining faults.

Solving equation (1) we get

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

Now by taking different distribution function we get different mean value functions.

A. Cost Model Without Patching

In the cost models considered so far, the release time of a software marks the end of testing phase and idea of patching is not considered. Going by the mathematical notations, $\tau^* = T^*$. Hence there are two phase i.e $[0,\tau] and [\tau, T_{lc}]$ .Also in the existing literature we assume that testing and operational phases are governed by the same distribution function. But in the cost model proposed below we have taken different distribution function for testing and operational phases which is more practical due to several reasons viz. testing team may carry on the task of testing in pre and post release testing phases with different priorities, objectives and experience level.

In the first phase $[0,\tau]$ the total number of faults removed by the testing team is $m(\tau) = a.F_1(\tau)$ (3)

Cost incurred in this phase is given by $c_3.m(\tau)$

Where $F_1(\tau)$ is the rate by which the faults are removed from the software by the testing team in the interval $[0,\tau]$.

In the second phase $[\tau, T_{lc}]$ researchers assumed that all the remaining faults in the software are reported from the software users during its lifecycle are removed by the testing team. Hence the total number of faults removed by the testing team in the second phase is given by

$$m(T_{lc} - \tau) = a(1 - F_1(\tau)).F_2(T_{lc} - \tau) \tag{4}$$

Where $F_2(T_{lc} - \tau)$ is the rate by which the faults are removed from the software by the users in operational phase $[\tau, T_{lc}]$.

Cost incurred in this phase is given by $c_4.m(T_{lc} - \tau)$

Total cost incurred in this case when patching is not considered is given by:

$$C(\tau) = c_1.\tau + c_2.\tau^2 + c_3.m(\tau) + c_4.m(T_{lc} - \tau) \tag{5}$$

Where τ^2 is the functional form of market opportunity cost used by Jiang and Sarkar [9].

B. Cost Model With Patching

In this case, lifecycle of the software is divided into three phases viz. $[0,\tau],[\tau,T] and [T, T_{lc}]$.

Phase1 : $[0,\tau]$

In this phase testing team is working to detect/correct failure/fault and the total number of faults removed in this interval is given by

$$m(\tau) = a.F_1(\tau) \tag{6}$$

So, the expenses in this phase are only due to tester and is given by $c_3 \cdot m(\tau)$

Where $F_1(\tau)$ is the fault removal rate in the interval $[0,\tau]$.

Phase 2 : $[\tau, T]$

In this phase there are two testing groups working simultaneously i.e. testers and users. However, both of them are expected to detect the bugs at a different rate owing to different intensity and efficiency of testing of these two groups. In addition, the sizes of the both the groups (testers and users) are also different. Intensity of testing (usage) refers to the average amount of time spent on testing in one day. Since the purpose of testing team is to find bugs, so they are continuously work on the software to detect maximum number of bugs before testing stops. On the other hand, software users spend only limited amount of time on on the software. In other words users spend considerably less amount of time as compared to testers. Thus, both these teams differ in their testing intensities. Efficiency of testing refers to the measure of effective time spent in detecting errors. When the same amount of time is spent in professional testing and customer usage, we expect that a bug is more likely to be detected by a

dedicated member of testing team than by a customer. This is due to the availability of better skills, tools and professional training to dedicated testers. Also prior to the software release, these software testers have already spent a significant amount of time on failure detection and removal of its cause. Therefore rate of failure detection with normal users will be less as compared to dedicated testers.

Failure observation rate in this phase will be higher as in the earlier phase, i.e. before the software release time τ, only testing team was involved in testing for detection of failure and correction of fault, but now after software release, testing team and users both are involved in the task of failure detection in the software. The total number of faults removed in this phase is given by :

$$m(T-\tau) = (a-m(\tau))\cdot \int_{\bullet}^{T-\tau} \overline{F}_2(x)\cdot f_3(x)dx + (a-m(\tau))\cdot \int_{\bullet}^{T-\tau} \overline{F}_3(x)\cdot f_2(x)dx$$

$$= m'(T-\tau) + m''(T-\tau) \quad (7)$$

Where $\overline{F}(t) = 1 - F(t)$ and $m'(T-\tau) = (a-m(\tau))\cdot \int_0^{T-\tau} \overline{F}_2(x)\cdot f_3(x)dx$ represents the total number of faults removed by the testing team when failure is reported by customers end and $m''(T-\tau) = (a-m(\tau))\cdot \int_0^{T-\tau} \overline{F}_3(x)\cdot f_2(x)dx$ represents the total number of faults removed by the testing team in the interval $[\tau, T]$. Also $(a-m(\tau))$ represents the remaining number of faults after τ and $F_2(T-\tau)\,\&\,F_3(T-\tau)$ are the failure detection rates of the two independent groups (testers and users) involved in detection of failure in the interval $[\tau, T]$.

Cost incurred in this phase is given by $c_4.m'(T-\tau) + c_5.m''(T-\tau)$

Phase 3 : $[T, T_{lc}]$

Now, in this phase, only users are involved in the detection of failure and report it to the software testers.

Total number of faults removal in this phase is given by:

$$m(T_{lc}-T) = (a-m(\tau))\cdot \left(1-\left(\int_0^{T-\tau} \overline{F}_2(x)\cdot f_3(x)dx + \int_0^{T-\tau} \overline{F}_3(x)\cdot f_2(x)dx\right)\right).F_4(T_{lc}-T) \quad (8)$$

where

$$(a-m(\tau))\cdot \left(1-\left(\int_0^{T-\tau} \overline{F}_2(x)\cdot f_3(x)dx + \int_0^{T-\tau} \overline{F}_3(x)\cdot f_2(x)dx\right)\right)$$

denotes the remaining number of faults after the testing stop time T and $F_4(T_{lc}-T)$ denotes the rate by which faults are removed in the third phase.

Cost incurred in this phase is given by $c_6.m(T_{lc}-T)$

Total cost in the case where patching is considered is given by

$$C(\tau,T) = c_1.\tau + c_2.\tau^2 + c_3.m(\tau) + c_4.m'(T-\tau) + c_5.m''(T-\tau) + c_6.m(T_{lc}-T) \quad (9)$$

C. Objective

In this paper, our motive is to determine the optimal release and testing stop time while minimizing the total expected cost when patching is considered. The objective function in both the cases is given below in (P1) and (P2) respectively.

1) Without Patching

$$Min\ C(\tau) = c_1.\tau + c_2.\tau^2 + c_3.m(\tau) + c_4.m(T_{lc}-\tau)$$

With Patching

$$Min\, C(\tau,T) = c_1.\tau + c_2.\tau^2 + c_3.m(\tau) + c_4.m'(T-\tau) + c_5.m''(T-\tau) + c_6.m(T_{lc}-T)$$

III. NUMERICAL ILLUSTRATION

For illustration purpose and also for the sake of simplicity we have taken exponential distribution function in this paper for fault detection/correction rate i.e. $F(t) = 1-e^{-b.t}$. Further we have considered that testing and operational phase are governed by the same distribution function i.e. $b_1 = b_2 = b(say)$. Also by the argument given in section II let $b_3 = rb$ be the rate by which users are detecting the failure. Where r is the ratio of fault detection rate under customers' usage with respect to tester's testing in the second phase. Also $b_4 = sb$ where s is the ratio of fault detection rate under customers' usage with respect to testers' testing in the third phase. Note that $s \geq r$ due to the fact that in third phase users base is increased hence the chance of detecting a failure in this phase is more as compared to the previous phase.

Hence $F_1, F_2, F_3\ and\ F_4$ in (6), (7) and (8) are given by $F_1(\tau) = a.(1-e^{-b.\tau})$ $F_2(T-\tau) = 1-e^{-b\cdot(T-\tau)}$, , $F_3(T-\tau) = 1-e^{-rb\cdot(T-\tau)}$ and $F_4(T_{lc}-T) = 1-e^{-sb(T_{lc}-T)}$. Substituting the values of $F_1, F_2, F_3\ and\ F_4$ in equation (6), (7) and (8) we get

$$m(\tau) = a.F_1(\tau) = a.(1-e^{-b.t}) \quad (10)$$

$$m(T-\tau) = (a-m(\tau))\cdot(1-(1-F_2(T-\tau))\cdot(1-F_3(T-\tau)))$$

$$= (a-m(\tau))\cdot(1-\overline{F_2}(T-\tau)\cdot\overline{F_3}(T-\tau))$$

$$= a.e^{-b\tau}(1-e^{-b(1+r)(T-\tau)}) \quad (11)$$

$$m'(T-\tau) = (a - m(\tau)) \cdot (1 - \overline{F_2}(T-\tau) \cdot \overline{F_3}(T-\tau)) . \frac{rb}{b+rb}$$

$$= a.e^{-b\tau} (1 - e^{-b(1+\delta)(T-\tau)}) . \frac{r}{1+r} \quad (12)$$

$$m''(T-\tau) = (a - m(\tau)) \cdot (1 - \overline{F_2}(T-\tau) \cdot \overline{F_3}(T-\tau)) . \frac{b}{b+rb}$$

$$= a.e^{-b\tau} (1 - e^{-b(1+\delta)(T-\tau)}) . \frac{1}{1+r} \quad (13)$$

$$m(T_{lc} - T) = (a - m(\tau)) . (1 - (1 - \overline{F_2}(T-\tau) . \overline{F_3}(T-\tau))) . F_4(T_{lc} - T)$$

$$= a.e^{-b\tau} . e^{-b(1+r)(T-\tau)} . (1 - e^{-sb(T_{lc}-T)}) \quad (14)$$

To find the optimal release time and testing stop time in both the cases of without and with patching is considered suppose $r = 0.4$, $s = 0.5$, $T_{lc} = 100$, $c_1 = 300$, $c_2 = 20$, $c_3 = 100$, $c_4 = 500$, $c_5 = 100$, $c_6 = 500$. Note that $a = m(T_{lc})$. Also the parameters of the above G-O model are estimated using Statistical Package for Social Sciences (SPSS) on Tandem data for first release [28]. First release of tandem data consists of 100 faults which were found during the testing of 20 weeks. The parameter estimates and R^2 values obtained using non linear regression method for the above SRGM G-O model are given in table 1.

TABLE 1: Parameter Estimates

SRGM	a	b	R^2
G-O Model	130.201	0.083	0.986

Using the above values in the cost function and optimizing it by using maple software we get

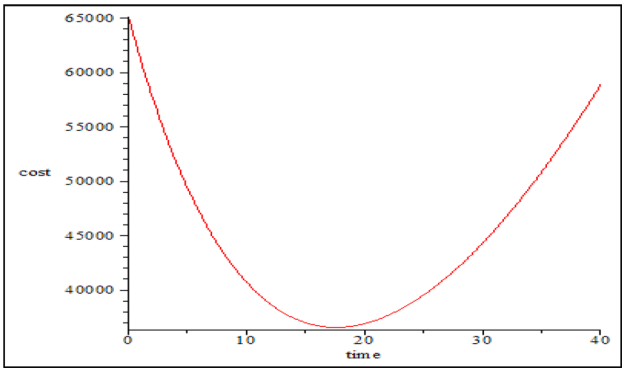

Fig. 3. Graph of cost function without patching

$\tau^* = T^* = 17.59$, $c^*(\tau) = 36574.34$ for the case of without patching and $\tau_p^* = 10.67$, $T_p^* = 24.55$ and $c^*(\tau, T) = 31600.07$ for the case of with patching. As evident from the above results that optimal release time in the case when patching is considered is earlier than the testing stop time when patching is not considered i.e. $\tau_p^* < \tau^*$ and optimal testing stop time T_p^* is later than the T^* optimal testing stop time when patching is not considered. Also there is a significant amount (13.6%) of reduction in the cost when patching is considered. This shows that our proposed framework is beneficial to the firm in terms of cost reduction. Figure 3 and 4 represents the graph of cost function for without and with patching respectively.

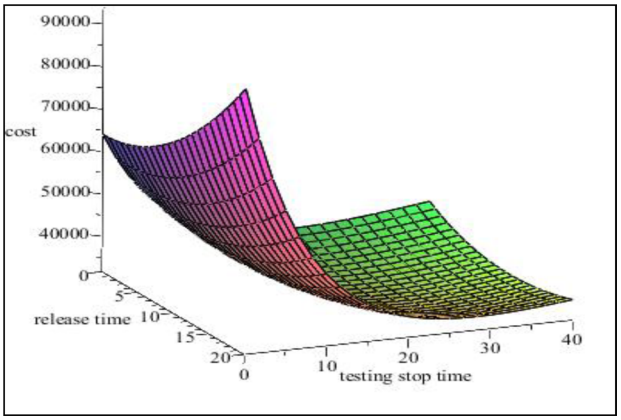

Fig. 4. Graph of cost function with patching

IV. CONCLUSION

Now a day's patch release has become an integral part of software development strategies. Vendors frequently release patches for fixing bugs and functional updates during post release testing phase of the software. Numerical analysis included in the paper shows that if a firm is providing software patches, it should release the software before its scheduled release time (without patching) and can opt for post release testing; which leads to significant decrease in the software testing cost thus making it more profitable for the organization. In future we can extend our model to find optimal release and testing stop time under reliability and budgetary constraints.

REFERENCES

[1] Dohi T., Kaio N. "Osaki S. "Optimal Software Release Policies with Debugging Time Lag" Published in International Journal of Reliability, Quality and Safety Engineering Volume 04, Issue 03, September 1997.

[2] Dalal, S.R. & Mallows, C.L., "When should one stop testing software?" Journal of the American Statistical Association, 83, 872-879 (1988).

[3] Ehrlich, W., B. Prasanna, J. Statmpfel, & J. Wu, , "Determining the cost of a stop-test decision", IEEE Transactions on reliability, 28(3) , (1993).

[4] Goel, A.L., Okumoto, K., "Time dependent error detection rate model for software reliability and other performance measures", IEEE Transaction Reliability, R-28(3), 206-211(1979).

[5] Okumoto, K. & Goel, A. L. (1980), "Optimum Release Time for Software Systems Based on Reliability and Coat Criteria", Journal of system Software, 1, 315-318

[6] Hower, R., "What are some recent major computer system failures caused by software bugs?", http://www.softwareqatest.com/qatfaq1.html, (February 2015).

[7] Huang C. Y., "Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency", Journal of system Software, 77, 139-155(2005).

[8] 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), 583-591(2005).

[9] Huang C. Y., Kuo, S.Y. & Lyu, M. R., " Optimal Software Release Policy based on Cost and Reliability with Testing Efficiency", In: Proceedings of 23rd IEEE annual international computer software and applications conference, Phoenix, AZ, 468-473 (1999).

[10] Jiang, S. & Sarkar, S., "Optimal Software Release Time with Patching considered", in Proc. 13th Annual Workshop Information technologies and Systems, Seattle, 61-66, (2003).

[11] Kapur P.K., Pham H., Gupta A., Jha P.C., "Software Reliability assessment with OR application", Springer, Berlin (2011).

[12] Kapur P.K., Garg R.B., Kumar S., "Contribution to hardware and software reliability", World Scientific publishing Co. Pvt. Ltd., Singapore, 1999..

[13] Kapur, P.K., Agarwal, S. & Garg, R. B., , "Bi-criterion Release Policy for Exponential Software Reliability Growth Models", Recherche Operationanelle / Operational Research, 28, 165-180, (1994).

[14] Kapur, P.K., Garg, R. B. & Bhalla, V. K., , "Release Policy with Random Software Life Cycle and Penalty Cost", Microelectron Reliability, 33(1), 7-12, (1993).

[15] Kapur, P.K. & Garg, R.B., "A Software Reliability Growth Model for an Error Removal Phenomenon", Software Reliability Journal, 291-294, (1992).

[16] Kapur, P.K. & Garg, R. B., "Optimal Software Release policies for Software Systems with testing Effort", International Journal System Science, 22(9), 1563-1571, (1991).

[17] Kapur, P.K. & Garg, R. B., "Cost-Reliability Optimum Release Policies for Software System under Penalty Cost", International Journal System Science, 20, 2547-2562, (1989).

[18] P.K. Kapur, H.Pham, Anu G. Aggarwal, Gurjeet Kaur, "Two dimensional multi-release software reliability modelling and optimal release planning" *IEEE Trans on Reliability*, Vol. 61 (3), pp. 758-768,2012.

[19] Kapur P.K., Pham H., Anand S. and Yadav K., "A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation reliability," IEEE Transactions on Reliability, vol. 60, no. 1, pp. 331 – 340, 2011.

[20] Kapur, P. K., Sachdeva, N., Singh, J. N. P., "Optimal Cost- A Criterion to Release Multiple Versions of Software", accepted for publication in International Journal of Software Assurance, Engineering and Management, , Volume 5, Issue 2, pp 174-180 June 2014.

[21] Kapur P.K., Pham H., Singh J.N.P & Sachdeva N. "When to stop testing multi up-gradations of software based on cost criteria" published in International Journal of Systems Science: Operations & Logistics Volume 1, Issue 2, 2014.

[22] McDaid, K. & Wilson, S.P., "Deciding how long to test Software", The Statistician, 50(2), 117-134, (2001).

[23] Musa J.D., Iannino A. and Okumoto K., Software Reliability: Measurement, Prediction, Applications, McGraw Hill, 1987.

[24] Ompal Singh, P. K. Kapur, A. K. Shrivastava, Vijay Kumar "Release time problem with multiple constraints" published in International Journal of System Assurance Engineering and Management March, Volume 6, Issue 1, pp 83-91,2015.

[25] Okumoto, K. & Goel, A. L., "Optimum Release Time for Software Systems Based on Reliability and Coat Criteria", Journal of system Software, 1, 315-318,1980.

[26] Pham H., System Software Reliability, Reliability Engineering Series, Springer, 2006.

[27] Pham, H., Zhang X., (1999), "A software cost model with warranty and risk costs", IEEE Trans Comp 48(1), 71-75.

[28] Wood, A. , "Predicting Software Reliability", IEEE Computers, 11, 69-77, (1996).

[29] Yamada, S. & Osaki, S., "Optimal Software Release Policies with simultaneous Cost and Reliability Requirements", European Journal of Operational Research, 31, 46-51, (1987).

[30] Yun, W.Y. & Bai, D.S. , "Optimum Software Release Policy with Random Life Cycle", IEEE transactions on Reliability, 39(2), 338-353, (1990).