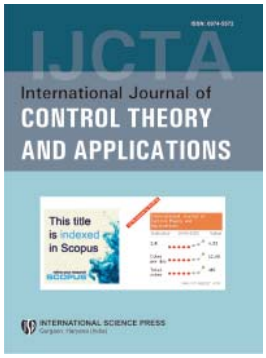

International Journal of Control Theory and Applications

ISSN : 0974–5572

Volume 9 • Number 40 • 2016

Importance of Release Time Determination for Success in Software Development Projects

Deiva Preetha C.A.S[a] and Subburaj Ramasamy[a]

[a]School of Computing, SRM University, Kattankulathur 603203, Tamilnadu, India;
E-mail: s_preetha@yahoo.com, subburaj.r@ktr.srmuniv.ac.in

Abstract: Success of a software project largely depends on the solution to the well known stopping rule problem. If a product is released too soon with inadequate testing, it might lead to dissatisfaction of the customer due to poor quality of the software. On the other hand, an organization may not be able to carry out too much of testing, due to high cost and significant impact on Time to Market (TTM), a critical component in software projects. Hence, it is essential to determine the optimal release time for the Software under Test (SUT). Extensive research has been carried out with respect to release time determination. We present a review of different software release time determination methodologies with brief details on the criteria considered in each one of them. The software industry may choose an appropriate technique, to determine release time, so as to improve the success rate of the project with confidence.

Keywords: *Software Release Time Determination, Stop Testing, Release Time Optimization*

1. INTRODUCTION

In the Software Development Life Cycle (SDLC) testing is an important phase. It consumes a major portion of the software development expenses. While it is essential to remove faults, exhaustive testing covering all possible inputs and paths in a large system is not practical, considering the related efforts and the cost involved in it. Therefore when to stop the testing is a critical decision to be taken for every software development project. The software development team has to determine whether the testing should be extended or not and this is termed as "Stopping rule problem" [1,2,3,4]. Optimum testing time should be determined to solve this problem and researchers have proposed different criteria for this. In this paper, we discuss on some approaches that address this optimization problem.

In Section 2, we summarize approaches which consider reliability goals as the criteria for determining the release time [5-12]. In Section 3, we present release policies that are proposed based on cost criteria [13-16]. Section 4 discusses methods which use a combination of reliability and the cost as criteria for determining optimal release time [17-22]. Concluding remarks are made in Section 5.

2. RELIABILITY CRITERIA

2.1. Faults Remaining in the System as a Criterion

One approach to find the release time is to depend on number of faults that remain in the product at a specific time. A decision on stopping the testing is taken when the number of faults remaining in the system is less than or equal to the target number of faults. The model assumes that fault and failure has a direct one-to-one relationship. The number of faults expected to be removed is assumed to be equal to the number of failures that may occur. Real time software projects defy this rule [23,24]. Also there are cases where a software product may contain a large number of faults, but the chances of these faults to cause failures could be relatively small. On the contrary, a product may contain only few faults but these faults might be in the areas of the software program that are most frequently used, leading to large number of failures. Hence number of faults remaining in the system is not a good measure of reliability.

2.2. Failure Intensity Function as a Criterion

Failure intensity is a good indicator of reliability of the software and would be a better choice than the number of remaining faults. According to Kapur [25] there is a definite and repeatable relationship between reliability and failure intensity for any software project.

Failure intensity can be effectively used as a criterion to determine the optimal release time by deriving from appropriate Software Reliability Growth Models (SRGM). After eliminating the nth fault, if the estimated failure intensity is $(n+1)$ then it should be decided to stop the testing when:

$$\lambda(n+1) \leq \lambda_{\circ}$$

where, $\lambda_{\circ}$ is the target failure intensity

2.3. Fault Detection Rate and Types of Debugging

The early SRGM models [26,27,28] assume a constant Fault Detection Rate (FDR) per fault. They assume that every fault has the same probability of being identified during the testing cycle. But in reality, FDR is a trend based metric and takes different values as time progresses in a Software Development Life Cycle [29]. Also assumptions made on the debugging mode by early SRGMs do not hold well in real time scenarios. Some early SRGMs consider perfect debugging mode and some consider imperfect debugging to be tightly linked with learning phenomenon. In practice, perfect debugging is not always possible and quality of imperfect debugging is not related to learning phenomena as most of the software projects have two different teams that are involved in debugging and testing and the respective quality measures would vary. To address this problem, Subburaj et al.[24] have classified quality of debugging into three categories: imperfect, perfect, and efficient debugging. Number of failures will be same as the number of faults detected only in the case of perfect debugging. In the case of efficient debugging, some faults are detected without causing a failure and hence the cumulative number of failures at infinite testing time may be less than the total number of faults detected. Subburaj et al. recommend a NHPP model that describes the learning phenomenon of the testing team and the various patterns of debugging. The model demonstrates an improved Goodness of Fit (GoF) catering to all patterns of variations of failure intensity function. These models have good predictive validity and hence can be used for taking an objective decision to stop the testing.

2.4. Conditional Reliability

Testing reliability is in fact conditional reliability. For instance $R(x \mid t)$ means that the software was repaired and was working with a reliability of 1 at the time t and we wish to find out the reliability at $(t+x)$ units of time. In this case we can stop testing when $R(x \mid t) \geq R_{0,}$ where R_0 is the target reliability.

2.5. Sensitivity analysis of release time

Probabilistic models are used for predicting the reliability level achieved. Most of the results assume that parameters of the model are clearly defined. In practice, depending on the information that is available at the particular point of time, these parameters are estimated. In most of the cases, only limited information will be available and hence the parameter estimates may not be accurate. It is essential to know how these estimated parameters could influence the decisions on the software release. Since parameter values would change with the availability of more information, change in some parameter values may be highly sensitive for determining the software release time. If the release time is more closely influenced by a specific parameter, immense care should be taken to estimate the respective parameter with high level of accuracy. [30,31,32,33]

3. COST CRITERIA

Okumoto and Goel suggested a simple cost model [17]. At a specific testing time T, the optimal release time can be determined by minimizing the "total cost function" that is estimated. Thereby, optimal failure intensity at release time can be given expressed as:

$$\lambda(T) = \frac{ct_3}{ct_2 - ct_1}$$

where

ct_1 = cost estimated for removal of fault in the testing phase

ct_2 = cost estimated for removal of fault in the operation phase

ct_3 = cost estimated for the software testing per unit time

Researchers have proposed various formulations of cost components. The above listed three parameters are the regular cost components that are used by most of them. Additional components are identified and added to this generalized cost function model to help on the release time determination.

3.1. Testing-Effort and Test Efficiency

By running more test cases we can ensure that the software reliability increases. But testing with ineffective test cases may lead to an undesirable increase in the cost. If additional faults are to be detected during the testing cycle, the project might require more experienced test engineers or more advanced testing tools. Test Engineers generally rely on automated techniques or tools for detecting additional faults during the testing cycle. These tools can be considered as valuable provided they reduce the testing period without affecting the reliability goals. Chin-Yu Huang et al.[34,35] recommend a generalized logistic testing-effort function (TEF) using which the resource utilization during the software development process can be described. The cost involved in procuring new techniques or tools may not be a constant and will take various values over the course of the project, based on the performance of the testing system, testing effort (TE) metrics and test efficiency. When P, the additional faults identified in the testing phase is relatively small, the total expected cost will be larger. This is because of the cost incurred on developing or procuring new testing tools. As P increases, there is an increase in the optimal release time, but the total expected cost reduces. The reason is more faults are being detected in the testing cycle, reducing the cost of detecting the faults in the operational phase. When the value of P is the same, with varied cost functions, higher the cost functions, lower is the optimal release time. In order to arrive on an optimal release time, software development team has to choose an appropriate policy considering cost, effort involved in the testing and the efficiency of testing cycle.

3.2. Cost of detecting and debugging a fault

Most of the software reliability models assume that the debugging process is perfect and a fault is debugged immediately after it is identified. According to Gokhale et al.[36] the time needed to debug a fault is also a critical parameter to be considered for any software development project. In practice, resolving a failure

involves two major steps. Fault detection process is the first step when a change request is raised and the fault causing the failure is diagnosed. This is followed by the debugging process when the respective fault is removed. The numbers of faults debugged at a given time are lesser than the number of faults identified. For determining optimal release time it is important to consider the cost involved in both the fault detection and fault removal process.

3.3. Scheduled Delivery Time

While arriving on the optimal release time for software, it is important to consider the uncertainty involved in determining the total cost. If an organization fails to release software at the scheduled delivery time, it incurs additional cost as penalty cost. Yamada et al.[37] propose a model to determine the expected penalty cost in a project due to delay in software release. This helps in minimizing the risk and on arriving with better estimates on the total cost of the project.

4. COMBINED CRITERIA – COST AND RELIABILITY

Quality of software is usually decided by the effectiveness of the testing cycle. If more time is spent on testing with appropriate testing resources, more errors can be removed and software with good reliability metrics can be released. However, organization may not be able to invest such huge time on the testing cycle as it directly affects the cost. In contrast, reducing the testing time for saving the cost increases the risk of releasing an unreliable product to the customer. Hence it is important to determine the release time considering both cost and the reliability requirement of the software system.

Zhao and Xie [38] propose a general method to obtain the optimum release time depending on both the reliability and cost factors, which considers two parameters as below

1. The time when the expected cost is minimized
2. The minimum time when the reliability reaches a required level

It may be noted that Subburaj[39] and Musa[40] recommend failure intensity instead of reliability in the above criterion.

4.1. Fuzzy Environment

In most of the scenarios, determining release time using these classical optimization methods need well defined criteria. However real time system environment is not deterministic and there is always an uncertainty related to some parameters. Recent research recommends fuzzy environment to be used for determining optimal release time [41,42]. Most of the release time models obtain cost and reliability coefficients based on various factors which are not deterministic. Components of cost function rely on dynamic factors like the test approach, test setup, team composition and competency of testing resources. Due to high attrition rates in the software industry, team composition changes more frequently, leaving ambiguous information for the decision makers. Fuzzy environment provides a chance for subjective computation of the model parameters. Kapur et al.[42] have formulated release time determination problem considering cost and reliability goal by defining it using fuzzy functions. The problem definition considers the dual objectives; reduction of cost and failure intensity. The defined “fuzzy cost function” includes the uncertainty in the failure cost during operational phase, considering the reliability goal.

4.2. Pre and Post Release Testing Phases

It is a common phenomenon to release the software earlier and continue with the testing after release until the testing team is sure of the software reliability. Software industry now considers the testing in two phases; pre-release and post-release testing. Kapur et al.[43,44,45] propose a model considering this post release testing

phase when an organization focuses on the remaining software faults to improve the product experience for the customers. Software fixes are made available to the user between two successive releases. The errors which were missed in the pre-release testing phase of the software are detected and corrected during the post-release testing phase and the fixes are made available to the users as patches.

5. CONCLUSION

Different software projects may need to follow different approaches to determine the release time. Some software projects have a reliability requirement that needs to be met for the product to be released. For some other projects the organization likes to obtain the release time based on the cost-benefit analysis. There are also cases where the optimal release time determination is driven by both reliability and cost requirements. In this paper we have summarized related approaches which an organization may study and apply to determine an optimal release time for the software system. The traditional models as well as recent ones which consider the current trends in software industry are discussed and analyzed. The review may be neither comprehensive nor complete. The software release time determination problem is undergoing extensive research in the last few decades and new approaches are evolved as a continuous ongoing process. Size and nature of the system being developed, the environment in which the system is deployed and the effectiveness of the resources utilized during testing cycle are the driving factors for decisions on the release time of the product. Industry professionals should choose an appropriate approach for determining software release time based on these factors in order to increase the success rate of their project and enhance customer satisfaction.

REFERENCES

[1] M. Xie,"On the Determination of Optimum Software Release Time," Proceedings of the 1991 International Symposium on Software Reliability Engineering.218-224,1991

[2] S. Ross,"Software Reliability: the Stopping Rule Problem," IEEE Trans. Software Eng., vol. SE-11, pp. 1472-1476, 1985.

[3] S. R. Dalal, C. I.Mallows,"When Should One Stop Testing Software?," J. American Statistical Assoc., vol. 83, no. 403, pp. 872-879, 1988.

[4] P. A. Caspi, E. F. Kouka, "Stopping Rules for a Debugging Process Based on Different Software Reliability Models," Proc. Int. Conf. on Fault - Tolerant Computing, pp. 114-119, 1984.

[5] P. K. Kapur, R. B. Garg, "Optimal Software Release Policies for Software Reliability Growth Models under Imperfect Debugging," R.A.I.R.O., vol. 24, pp. 295-305, 1990.

[6] H. S. Koch, P. Kubat, "Optimal Release Time of Computer Software," IEEE Trans. Soft-ware Eng., vol. SE-9, pp. 323-327, 1983.

[7] K. D. Levin, O. Yadid, "Optimal Release Time of Improved Versions of Software Packages," Inf. & Software Techn., vol. 32, pp. 65-70, 1990.

[8] J. D. Musa, A. F. Ackerman, "Quantifying Software Validation: When to Stop Testing?, " IEEE Software, vol. 2, pp. 19-22, 1990.

[9] M. Xie, "On the Determination of Optimum Software Release Time," Proceedings of the 1991 International Symposium on Software Reliability Engineering,218-224

[10] S. Zheng, "Dynamic release policies for software systems with a reliability constraint," IIE Transactions, vol. 34, no. 3, pp. 253-262, Mar. 2002.

[11] J.G. Shanthikumar, and S. Tufekci, "Application of a software reliability model to decide software release time", Microelectron. Heliab., Vol. 23, pp. 41-59 (1983).

[12] R. Brettschneider, "Is Your Software Ready for Release?," IEEE Software, pages 100–108, 1989.

[13] W. Ehrlich, B. Prasanna, J. Stampfel, and J. Wu, "Determining the cost of a stop-testing decision, " IEEE Trans. Softw. Eng., vol. 19, pp. 33-42, Mar. 1993.

[14] H. Pham, X. Zhang, "Software release policies with gain in reliability justifying the costs, Ann, " Softw. Eng., vol. 8, pp. 147-166, 1999.

[15] Y. W. Leung, "Optimum software release time with a given budget", J. Syst. Softw., vol. 17, pp. 233-242, 1992.

[16] B. Yang, H. Hu, and, L. Jia, "A Study of Uncertainty in Software Cost and its Impact on Optimal Software Release Time," IEEE Transactions on Software Engineering, 4(6):813–825, 2008.

[17] K. Okumoto, A. L. Goel, "Optimum Release Time for Software Systems Based on Reliability and Cost Criteria," J. Systems and Software, vol. 1, pp. 315-318, 1980.

[18] S. Yamada, S. Osaki, "Cost-Reliability Optimum Release Policies for Software Systems," IEEE Trans. Rel., vol. R-34, pp. 422-424, 1985.

[19] S. Yamada, S. Osaki, "Cost-Reliability Optimum Release Policies for Software Systems," IEEE Trans. Rel., vol. R-34, pp. 422-424, 1985.

[20] S.Yamada. and, S.Osaki, "Cost-reliability optimal release policies for software systems, " IEEE Transactions on Reliability, Vol. R-34 No. 5, pp. 422-4.

[21] K. Okumoto, A. L. Goel, "Optimum release time for software systems based on reliability and cost criteria," J. System Software, vol. 1, pp. 315-318, 1980.

[22] H. Pham and, X. Zhang, "Software release policies with gain in reliability justifying costs," Annals of Software Engineering, vol. 8, 1999.

[23] R. Subburaj, G. Gopal and, P.K.Kapur, "A Software reliability growth model for vital quality metrics," South African Journal of Industrial Engineering Nov 2007 Vol 18(2): 93-108

[24] R. Subburaj, G. Gopal and, P.K.Kapur, "A Software Reliability Growth Model for Estimating Debugging and the Learning Indices," International Journal of Performability Engineering Vol. 8, No. 5, September 2012, pp. 539- 549

[25] P.K.Kapur, and, R.B. Garg, "Optimal Software Release Policies For Software Reliability

[26] Growth Models Under Imperfect Debugging," Operations Research, 1990; 24(3): 295-305. [26] A.L.Goel, "Software reliability models: Assumptions, limitations and applicability," IEEE Transactions on Software Engineering,SE-11 (12), pp.

[27] S.Yamada and, S.Osaki,"Software reliability growth modeling: Models and applications," IEEE Transactions on Software Engineering, Vol. SE-11(12), pp. 1431-1437, Dec. 1985.

[28] W.Farr, "Software Reliability Modeling Survey", IEEE Computer Society Press and McGraw-Hill, 1996.

[29] Sy-Yen Kuo, Chin-Yu Huang, and, Michael R. Lyu,"Framework for Modeling Software Reliability, Using Various Testing-Efforts and Fault-Detection Rates," IEEE Transactions on reliability, Vo.50,No,3,September 2001

[30] M. Xie G. Y. Hong, "A study of the sensitivity of software release time,"Journal of Systems and Software Volume 44 Issue 2 Dec. 1998

[31] Xiang Li, Min Xie and, Szu Hui Ng, "Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points," Applied Mathematical Modelling 34 (2010) 3560–3570

[32] M. Xie, B. Yang and, O. Gaudoin, "Sensitivity analysis in optimal software release time problems", Opsearch 41 (2004) 250–263.

[33] Chin-Yu Huang and, Michael R. Lyu, "Optimal Testing Resource Allocation, and Sensitivity Analysis in Software Development, "IEEE Transactions on Reliability,Vol.54, No.4,December 2005.

[34] Chin-Yu Huang, "Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency, " The Journal of Systems and Software 77 (2005) 139–155

[35] Chin-Yu Huang and, M.R. Lyu, "Optimal release time for software systems considering cost, testing-effort, and test efficiency, " IEEE Transactions on Reliability (Volume: 54, Issue: 4, Dec. 2005)

[36] S. S. Gokhale, M. R. Lyu, and, K. S. Trivedi., "Incorporating fault debugging activities into software reliability models:a simulation approach.," IEEE Transactions on Reliability,55(2):281–292, 2006.

[37] S. Yamada, "Optimum Release Policies for a Software System with a Scheduled Software Delivery Time," Int. J. Systems Sci., vol. 15, pp. 905-914, 1984.

[38] M. Zhao, M. Xie, "Robustness of optimum software release policies," Proceedings of the International Symposium on Software Reliability Engineering, 1993, pp. 218–225.

[39] JD .Musa, "Software reliability engineering," McGraw-Hill Publication; 1999.

[40] R .Subburaj, "Software reliability engineering," McGraw Hill Education (India) Private Limited, New Delhi; 2015. p.61–80.

[41] Bhoopendra Pachauri, Ajay Kumar and, Joydip Dhar, "Modeling optimal release policy under fuzzy paradigm in imperfect debugging environment," Information and Software Technology 55 (2013)

[42] P. Kapur, H. Pham, A. Gupta and, P. Jha, "Optimal release policy under fuzzy environment, " International Journal of Systems Assurance Engineering and Management 2 (1) (2011) 48–58.

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

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

[45] P.K. Kapur, A.K. Shrivastava, "Release and Testing Stop Time of a Software: A New Insight, " Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), Sep. 2-4, 2015