Optimal Release Policy for Covariate Software Reliability Models

Ebenezer Yawlui, University of Massachusetts Dartmouth
Priscila Silva, University of Massachusetts Dartmouth
Vidhyashree Nagaraju, Ph. D., University of Tulsa
Lance Fiondella, Ph. D., University of Massachusetts Dartmouth

Key Words: optimal time to release, covariate software reliability model, optimal effort allocation

SUMMARY & CONCLUSIONS

The optimal time to release a software is a common problem of broad concern to software engineers, where the goal is to minimize cost by balancing the cost of fixing defects before or after release as well as the cost of testing. However, the vast majority of these models are based on defect discovery models that are a function of time and can therefore only provide guidance on the amount of additional effort required. To overcome this limitation, this paper presents a software optimal release model based on cost criteria, incorporating the covariate software defect detection model based on the Discrete Cox Proportional Hazards Model. The proposed model provides more detailed guidance recommending the amount of each distinct test activity performed to discover defects.

1 INTRODUCTION

Software reliability growth models (SRGM) [1-3] predict future defects discovered as a function of testing time as well as decreasing trends in defect discovery intensity and increasing trends in mean time to failure. They can also be used to gauge schedule and cost risk as well as guide release planning strategies [4] with models to minimize cost and satisfy reliability constraints. However, the traditional nonhomogeneous Poisson process (NHPP) SRGM upon which these release planning models are based are generally unappealing to software engineers because they only tell the user of such models how much additional testing effort to perform, yet both reliability and security focused software testing is a rich and multi-faceted activity that can benefit from multiple testing techniques and tools. As a result, release problems based on traditional NHPP SRGM lack the practical ability to provide more detailed guidance to software engineers on how much of each activity to perform during the release planning process, which would align more closely with efforts undertaken by practicing software engineers.

Past research on the optimal release time of software systems, includes the work of Okumoto and Goel [4] who developed release models based on reliability and cost criterion, considering factors such as the number of defects discovered before and after release as well as the cost of testing. Yamada and Osaki [5] extended this to an optimal software release problem for both cost and reliability requirements. Dozens of papers have derived methods similar to [4] for alternative NHPP SRGM, including models incorporating test coverage [6], environmental factors [7], and testing effort [8], and test efficiency [9]. Pham and Wang [10] introduced software reliability and cost models based on quasi-renewal processes, while Pham and Zhang [11] developed a model with warranty cost, implementing the equations in Excel.

Other early statistical studies include the work of Ross [12] who presented a method to estimate the error rate of software at a given time t and developed a stopping rule to determine when to discontinue the testing and declare that the software is ready for use. Dalal and Mallows [13] considered two variations of the optimal release problem where the distribution underlying defect discovery is not entirely known. Singpurwalla [14] described an approach based on the principles of decision making under uncertainty and maximization of expected utility to determine how long to test and debug software prior to release. Yang and Chao [15] compared various stopping rules according to their ability to identify an optimal release time and proposed two new rules, discovering that rules based on cost were more stable than other rules for a variety of bug structures.

Optimal release problems based on architecture-based software reliability [16], where the application reliability is computed in terms of the component reliabilities and transition probabilities among the components include Lyu et al. [17] who formulated reliability and cost optimization for software based on its architecture, characterizing the failure rates of components and cost to decrease this rate with SRGM. Pietrantuono et al. [18] proposed a test resource allocation model to provide solutions at various levels of detail, depending upon the information available to the engineer about the system in order to identify the most critical components within the software architecture so that testing resources can be assigned to proportional to their criticality.

Optimal release problems incorporating uncertain or iterative adaptation include Zhao and Xie [19] who studied the robustness of optimum release time procedures computing the variation of the optimum release time as a function of the variation of the estimated parameters. Yang et al. [20] studied the uncertainty in software cost and its impact on optimal software release in terms of variance and risk functions. Xie et

978-1-6654-6053-8/23/$31.00 ©2023 IEEE

al. [21] developed an approach for optimal software release under parameter uncertainty and performed simulation studies to demonstrate its ability to consider multiple risk levels. Nagaraju and Fiondella [22] developed an online optimal release planning strategy in which one or more models are periodically during testing to assess if the optimal release time has passed or if additional testing should be performed.

This paper presents a software optimal release model based on cost criteria, incorporating the covariate software defect detection model based on the Discrete Cox Proportional Hazards Model (DCPH) [23]. In contrast to past formulations based on a NHPP model that only consider a single dimension of time or other measures such as testing effort or coverage, the proposed model is multivariate. Specifically, the covariate software defect detection model based on the DCPH considers the number of defects detected as a function of one or more testing activities (covariates). Therefore, the problem is to allocate a constrained budget to these multiple alternative test activities of potentially different unit cost in order to maximize defect discover, so that they can be removed prior to release. This model is based on the assumption that resolving defects after release is more costly than those discovered to release. The practical advantage of this generalized approach is that the optimal policy identified by the model can provide direct guidance to software engineers on how to allocate efforts across multiple activities, instead of simply advising on the additional amount of time required to minimize costs. The approach is demonstrated through a data set from the literature [24]. Our results indicate that the approach can be utilized to allocate effort among alternative test activities in order to minimize cost.

The remainder of the paper is organized as follows: Section 2 reviews defect discovery models incorporating covariates. Section 3 formulates a model to minimize life cycle cost in terms of testing activities within the covariate model. Section 4 illustrates the cost minimization model with numerical examples. Section 5 concludes and suggests future research.

2 COVARIATE DEFECT DISCOVERY MODELS

This section presents defect detection models with covariates.

2.1 *Covariate Software Defect Detection Model based on the Discrete Cox Proportional Hazards Model*

The discrete Cox proportional hazards NHPP SRGM [23] correlates r covariates to the number of events in each of n intervals. In the context of software defect discovery, these covariates can be distinct defect testing activities allocated to multiple tools or techniques employed for this purpose. The matrix $\mathbf{x}_{n\times r}$ quantifies the amount of effort dedicated to each activity in each interval. For example, $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ denotes the amount of each activity $(1 \le j \le r)$ performed in the i^{th} interval.

The mean value function predicts the number of defects discovered up to and including the n^{th} interval given covariates $\mathbf{x}$ according to

$$m(\mathbf{x}) = \omega \sum_{i=1}^{n} p_{i,\mathbf{x}_i} \quad (1)$$

where $\omega > 0$ denotes the number of defects that would be discovered with infinite testing and

$$p_{i,\mathbf{x}_i} = \left(1 - (1 - h(i))^{g(\mathbf{x}_i;\boldsymbol{\beta})}\right) \prod_{k=1}^{i-1} (1 - h(k))^{g(\mathbf{x}_k;\boldsymbol{\beta})} \quad (2)$$

is the probability that a defect is discovered in the i^{th} interval, given that it was not discovered in the first $(i-1)$ intervals, $h(\cdot)$ is the baseline hazard function, and $\boldsymbol{\beta}$ is the vector of r parameters contained within the Cox proportional hazards model.

$$g(\mathbf{x}_i;\beta) = \exp(\beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_r x_{ir}) \quad (3)$$

2.2 *Hazard functions*

This section presents examples of hazard functions that can be incorporated into Equation (2). The following three hazard functions were originally employed in the covariate software reliability model of [24].

1) Geometric (GM):

$$h(b) = b \quad (4)$$

2) Negative binomial of order two (NB2):

$$h(i;b) = \frac{ib^2}{1 + b(i-1)} \quad (5)$$

where $b \in (0,1)$ is the probability of detecting a defect.

3) Discrete Weibull of order two (DW2):

$$h(i;b) = 1 - b^{i^2-(i-1)^2} \quad 6)$$

To estimate the parameters of the discrete Cox proportional hazard model (DCPH) with covariates, the log-likelihood function [23] is

$$LL(\gamma,\beta,\omega) = -\omega \sum_{i=1}^{n} p_{i,\mathbf{x}_i} + \sum_{i=1}^{n} y_i \ln(\omega) + \sum_{i=1}^{n} y_i \ln\left(p_{i,\mathbf{x}_i}\right) - \sum_{i=1}^{n} \ln(y_i!) \quad (7)$$

where γ is the vector of model parameters contained in the hazard function and y_i is the number of defects discovered in the i^{th} interval. Substituting one of the hazard functions specified in section 2.2 into Equation (2) produces unique log-likelihood functions. Thus, given covariates data $\mathbf{x}$ and vector of defects discovered in each of the n intervals (y_n), the model fitting step identifies the numerical values of the total number of vulnerabilities to be discovered (ω), vector of m covariates coefficients (β), and hazard function parameters (γ).

Letting $\theta = \{\gamma, \beta, \omega\}$ denote the vector of all model parameters, the log-likelihood expression can be reduced from $|\theta|$ to $|\theta| - 1$ parameters by differentiating the log-likelihood function with respect to ω, equating the results to zero, solving for ω to produce

$$\hat{\omega} = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} p_{i,\mathbf{x}_i}} \quad (8)$$

and substituting Equation (8) into the log-likelihood function to obtain a reduced log-likelihood (RLL) function.

The maximum likelihood estimates of the remaining $|\theta| - 1$ parameters, denoted $\theta' \coloneqq \theta/\omega$ for compactness, is determined by computing partial derivatives

$$\frac{\partial RLL}{\partial \beta} = 0 \tag{9}$$

and

$$\frac{\partial RLL}{\partial \gamma} = 0 \tag{10}$$

These steps can be applied to the alternative hazard functions in Section 2.2 to obtain the corresponding maximum likelihood estimates of the models. Solving this system of equations and substituting the numerical values $\hat{\beta}$ and $\hat{\gamma}$ into Equation (8) produces the maximum likelihood $\widehat{\omega}$.

3 OPTIMAL TEST ALLOCATION BASED ON COST CRITERION

This section formulates a software optimal release model based on cost criteria, incorporating the covariate software defect detection model based on the Discrete Cox Proportional Hazards Model. In practice, this cost minimization model uses a covariate software defect detection model fitted according to the methods described in Section 2. Specifically, given model $\widehat{m}(\mathbf{x})$ fitted to test activities $\mathbf{x}_{n\times r}$ and vector of observed failures $\mathbf{y}$, we seek to allocate resources to test activities in interval $n+1$ in a manner that minimizes cost

$$\text{argmin } \mathbf{C}(\mathbf{X}) \tag{11}$$

subject to

$$\sum_{i=1}^{r} \alpha_i \mathbf{X}_{i,n+1} \leq B \tag{12}$$

where $\mathbf{X}$ is the effort dedicated to each of the r covariates in each of the intervals up to and including the $(n+1)^{st}$ interval, α_i is the cost associated with a unit of test activity i, $\mathbf{X}_{i,n+1}$ is the number of units of test activity i allocated in the $n+1$ interval, and $B > 0$ is a budget constraint.

The specific form of the cost function for the optimal release model based on cost criteria, incorporating the covariate software defect detection model based on the Discrete Cox Proportional Hazards Model is

$$\mathbf{C}(\mathbf{X}) = \alpha_{pre}\widehat{m}(\mathbf{X}) + \alpha_{post}\big(\widehat{m}(\mathbf{x}) - \widehat{m}(\mathbf{X})\big) + \sum_{i=1}^{r} \alpha_i \sum_{j=1}^{n+1} \mathbf{X}_{i,j} \tag{13}$$

where α_{pre} and α_{post} respectively denote the cost of removing a defects before or after release ($\alpha_{post} > \alpha_{pre}$), $\widehat{m}(\mathbf{X})$ is the estimated number of defects discovered prior to release according to the fitted model, $\widehat{m}(\mathbf{x})$ is the estimated number of defects to be discovered throughout the software lifecycle experiencing test activities $\mathbf{x} > \mathbf{X}$, and the final term is the cost of all test activities applied in all $n+1$ intervals prior to release.

In order to estimate the optimal effort allocation of multiple test activities $\mathbf{X}^*$ that minimizes cost of release, we take partial derivatives of the total cost Equation (13) with respect to the amount of effort to be allocated to each covariate in interval $n+1$ to produce the following system of equations

$$\frac{\partial \mathbf{C}(\mathbf{X}_{i,n+1})}{\partial \mathbf{X}_{i,n+1}} = \mathbf{0} \tag{14}$$

and solve numerically, since this system of equations lacks closed form solution, due to the nonlinear nature of Equations (2) and (3).

3.1 *Application of Optimal Test Allocation based on Cost Criterion*

This section describes how to apply optimal test allocation based on cost criterion. Toward this end, we contrast this process with the steps performed in the context of traditional approaches [4] for optimal software release models based on the NHPP without covariates. In this simpler context, the following three steps to estimate the optimal release time are typically taken:

- (S.1) Fit a model to the complete failure data to obtain parameter estimates using a software reliability growth model without covariates.
- (S.2) Establish a numerical version of the optimal release equation, by substituting the maximum likelihood estimates obtained in step (S.1) into a single dimensional version of Equation (13) with $\widehat{m}(t)$ and $\widehat{m}(T)$ for the lifecycle (t) and release time (T), where $t > T$, as well as term $c \times T$ that accounts for the cost of testing prior to release.
- (S.3) Plot this numerical version of the single dimensional cost as a function of time according to the cost equation established in step (S.2) to graphically illustrate the trend and identify the minimum with a numerical solver. This trend is typically bathtub shaped because the cost of failures initially motivates additional testing, but eventually increases because the cost of testing outweighs the risk associated with releasing the software with a small number of remaining defects.

In contrast to the traditional procedure for identifying the optimal release time of software, the software optimal release model based on cost criteria, incorporating the covariate software defect detection model based on the Discrete Cox Proportional Hazards Model generalizes time (T) to the multiple test activities performed ($\mathbf{X}$). The proposed approach thus constitutes a portfolio allocation problem in which limited resources are divided among multiple alternative test activities.

4 ILLUSTRATIONS

This section illustrates the software optimal release model based on cost criteria, incorporating the covariate software defect detection model based on the Discrete Cox Proportional Hazards Model formulated in Section 3. All illustrations are based on the DS1 dataset [24], which consists of $n = 17$ intervals (weeks) and composed of three covariates, including execution time (E) in hours, failure identification work (F) in person hours, and computer time failure identification (C) in hours as well as the corresponding number of defects discovered as a result of these test activities. To simplify the exposition and enable clear visualizations, two covariates are

used. Specifically, the illustrations use the E and C covariates with the negative binomial hazard function, which is appropriate, since past studies [23] identified that this subset of covariates produced the most accurate predictions of future defect discovery. For concreteness, the cost parameters were set to $\alpha_{pre} = \$10$, $\alpha_{post} = \$500$, and $\alpha_1 = \alpha_2 = \$3$, while the software lifecycle was set to $\widehat{\omega} = 60.0153$, which denotes the estimated number of defects that would be discovered with infinite testing ($n \to \infty$ in Equation (1)) when only the first 9 of 17 intervals were used to fit the model. This approach of fitting the model with only 9 intervals was taken to illustrate how to use the optimal release model based on cost criteria, incorporating the covariate software defect detection model interactively by predicting how to allocate test activities for interval 10. In practice, the model could then be refit based on the results obtained by applying effort in interval 10 and the process repeated for subsequent intervals.

The first example graphically illustrates the estimated cost curve as a function of additional application of the E and C covariates, while the second example performs a sensitivity study on the optimal cost attainable under budgets of various sizes.

4.1 *Optimal test activity allocation based on cost criterion*

This example graphically illustrates the optimal test activity allocation for two covariates. Toward this end, nine of 17 intervals of the DS1 data set were fit to the negative binomial hazard function using the E and C covariates.

Figure 1 shows a plot of the cost curve based on this fitted model and the numerical parameters given in Section 4.

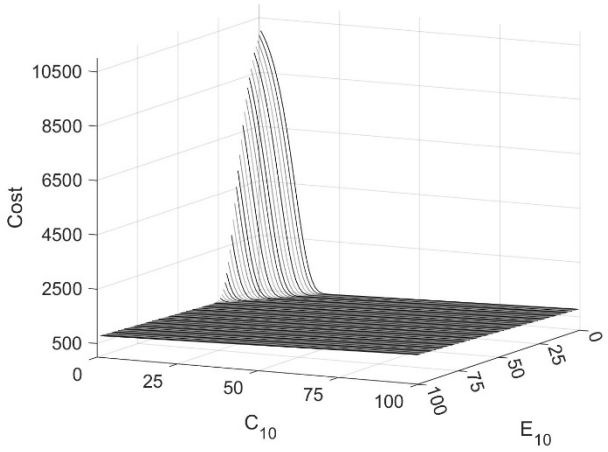

Figure 1 - Impact of test activity allocation on cost under negative binomial hazard function fitted to E and C covariates of DS1

Figure 1 indicates that the model predicts that allocating no additional testing to activity E or C would result in a cost exceeding 10,000. The minimum of Equation (13) according to the fitted model is $\mathbf{C}^* = 561.12$ is obtained when $\mathrm{E}_{10}^* = 0$ and $\mathrm{C}_{10}^* = 22.51$. The interpretation of $\mathrm{E}_{10}^* = 0$ is that no additional effort should be allocated to the test activity corresponding to execution time, but that some additional effort should be allocated to computer time failure identification (C). E_{10}^* may be zero because $\hat{\beta}_E \ll \hat{\beta}_C$, which would indicate that the rate of defect discovery given application of activity E is substantially lower than the corresponding rate for activity C. If both E_{10}^* and C_{10}^* were equal to zero, then the model would effectively be recommending software release because no additional effort is anticipated to be required. Figure 1 also indicates that for values of $\mathrm{C}_{10} > \mathrm{C}_{10}^*$, the cost increases. A less pronounced increase is also visible for test activity allocations where $\mathrm{E}_{10} > \mathrm{E}_{10}^*$. These increasing trends occur because the number of remaining defects that would be discovered between release and the end of the software lifecycle denoted by the term $\left(\widehat{m}(\mathbf{x}) - \widehat{m}(\mathbf{X})\right)$ in Equation (13) is relatively small compare to the cost of allocating additional effort to test activities, even when the number of remaining defects is scaled by the relatively high cost of post release failures (α_{post}).

4.2 *Sensitivity of optimal cost to test activity allocation based on cost criterion under different budget constraints*

This example illustrates the impact of the size of the budget B on the optimal test activity allocation recommended by the fitted model. Toward this end, we solved Equation (13) for values $B \in \{10,11,\dots,40\}$, allocating effort to covariates E and C in order to identify the cost attainable given B. The results of this analysis are shown in Figure 2.

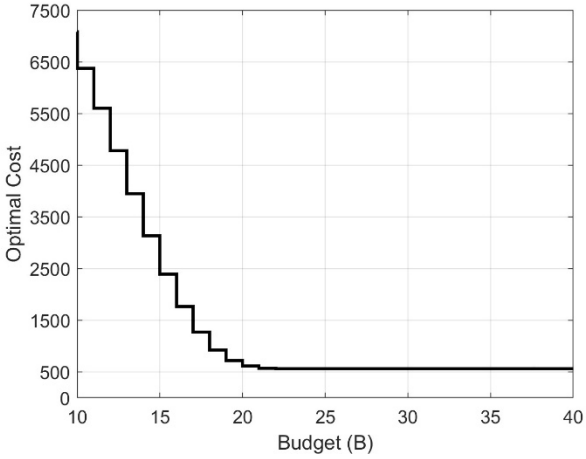

Figure 2 – Minimum cost attainable under optimal test activity allocation given budget B

Figure 2 indicates that cost decreases up until the optimal allocation $B = 22.51$ is reached. The figure also shows that the marginal utility of adding on additional unit to the budget is decreasing and that, for all values greater than $B = 23$, increasing the budget does not reduce the cost further.

To provide an alternative perspective to Figure 2, Figure 3 shows the contour plot of the cost function (Equation (13) shown in Figure 1) with the optimal allocation for each value of B superimposed on this contour plot as the black line with circle at the upper endpoint indicating the optimal allocation.

Here, the range of E_{10} and C_{10} were chosen to make the contours clear. Since all values of $\mathrm{E}_{10}^* = 0$, this indicates that the fitted model predicts that allocating additional time to this

activity will not compensate the cost of performing the testing because few additional defects would be found.

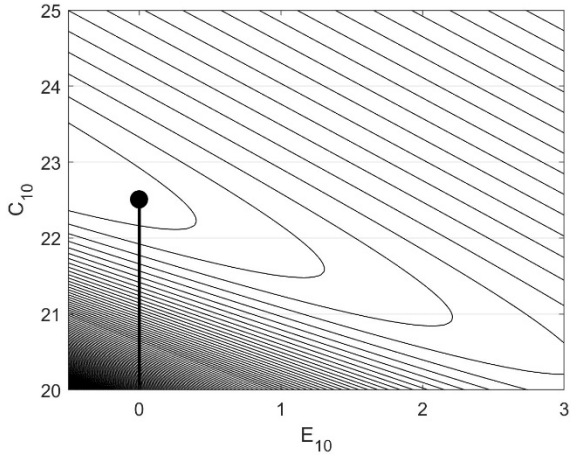

Figure 3 – Contour plot of cost function with optimal test activity allocation given budget B superimposed

5 CONCLUSIONS AND FUTURE RESEARCH

This paper presented a software optimal release model based on cost criteria, incorporating the covariate software defect detection model based on the Discrete Cox Proportional Hazards Model (DCPH). Unlike previous formulations which were based on a NHPP model that only considered a single dimension of time or other measures such as testing effort or coverage, the proposed model is multivariate. In particular, the covariate software defect detection model based on the DCPH considered the number of defects detected as a function of one or more testing activities (covariates). Therefore, the problem was to allocate a constrained budget to multiple alternative test activities of potentially different unit cost in order to maximize defect discover, so that they can be removed prior to release. The practical advantage of this generalized approach is that the optimal policy identified by the model provides direct guidance to software engineers on how to allocate efforts across multiple activities, instead of simply advising on the additional amount of time required to minimize costs. The approach was demonstrated through a data set from the literature. Our results indicated that the approach can be utilized to allocate effort among alternative test activities in order to minimize cost.

Future research will develop stable and efficient algorithms to identify the optimal test activity allocation when more covariates are considered and the dimensionality of the problem increases. Additional directions worthy of pursuit include (1) methods to consider parametric uncertainty [19-21] in order to establish confidence in release decisions and (2) online procedures [22] to iteratively allocate test activities, so that the approach aligns with application during the software testing process, as engineers work to improve the reliability of the software to a desired level in order to ensure it is suitable for release.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Number 1749635. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

1. M. Xie, *Software reliability modelling*, World Scientific, 1991.
2. H. Pham. *Software reliability*. Springer Science & Business Media, 2000.
3. K. Trivedi, *Probability & statistics with reliability, queuing and computer science applications*. John Wiley & Sons, 2008.
4. K. Okumoto, and A. Goel, "Optimum release time for software systems based on reliability and cost criteria," *Journal of Systems and Software*, vol. 1, pp 315-318, 1979-1980.
5. S. Yamada, and S. Osaki, "Cost–reliability optimal release policies for software systems," *IEEE Transactions on Reliability,* vol.34, pp. 422-424, (Dec.) 1985.
6. H. Pham, and X. Zhang, "NHPP software reliability and cost models with testing coverage," *European Journal of Operational Research,* vol. 145, pp 443-454, (Mar.) 2003.
7. H. Pham, "Software reliability and cost models: Perspectives, comparison, and practice." *European Journal of Operational Research*, vol. 149, pp 475-489, (Sep.) 2003.
8. R. Lai, M. Garg, P. K. Kapur, and S. Liu, "A study of when to release a software product from the perspective of software reliability models," *Journal of Software*, vol.6, pp. 651-661, (Apr.) 2011.
9. C.Y. Huang and M.R. Lyu, Optimal release time for software systems considering cost, testing-effort, and test efficiency. *IEEE Transactions on Reliability*, 54(4), pp.583-591, (Dec.) 2005.
10. H. Pham and H. Wang, "A quasi-renewal process for software reliability and testing costs," *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans,* vol. 31, pp. 623-631, (Nov.) 2001.
11. H. Pham, and X. Zhang, "A software cost model with warranty and risk costs," *IEEE Transactions on Computers,* vol. 48, pp 71–75, (Jan.) 1999.
12. S.M. Ross, "Software reliability: The stopping rule problem," *IEEE Transactions on Software Engineering*, 11(12), pp.1472-1476, (Dec.) 1985.
13. S.R. Dalal and C.L. Mallows, "Some graphical aids for deciding when to stop testing software," *IEEE Journal on Selected Areas in Communications*, 8(2), pp.169-175, (Feb.) 1990.
14. N.D. Singpurwalla, "Determining an optimal time interval for testing and debugging software," *IEEE Transactions on Software Engineering*, 17(4), p.313, (Apr.) 1991.
15. M. Yang and A. Chao, "Reliability-estimation & stopping-

rules for software testing, based on repeated appearance of bugs," *IEEE Transactions on Reliability*, 44(2), pp. 315-321, (June) 1995.
16. S.S. Gokhale, "Architecture-based software reliability analysis: Overview and limitations," *IEEE Transactions on Dependable and Secure Computing*, 4(1), pp.32-40, (Jan.-Mar.) 2007.
17. M. R. Lyu, S. Rangarajan, and A. P. A. van Moorsel., "Optimal allocation of test resources for software reliability growth modeling in software development" *IEEE Transactions on Reliability*, vol. 51, pp 183-192, (Jun.) 2002.
18. R. Pietrantuono, S. Russo and K.S. Trivedi, "Software reliability and testing time allocation: An architecture-based approach," *in IEEE Transactions on Software Engineering,* vol. 36, pp. 323-337, (May-June.) 2010.
19. M. Zhao, and M. Xie, "Robustness of optimum software release policies," In *Proc. of the IEEE International Symposium on Software Reliability Engineering*, pp. 218 – 225, 1993.
20. 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*, vol. 34, pp. 813-825, (Nov/Dec) 2008.
21. M. Xie, X. Li and S. H. Ng, "Risk-based software release policy under parameter uncertainty," In *Proc. of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 225(1), pp. 42-49, 2011.
22. V. Nagaraju, and L. Fiondella, "Online Optimal Release Time for Non-homogeneous Poisson Process Software Reliability Growth Model," In *Proc. of the Annual Reliability and Maintainability Symposium*, 2020**.**
23. V. Nagaraju, C. Jayasinghe, and L. Fiondella, "Optimal test activity allocation for covariate software reliability and security models," *Journal of Systems and Software*, vol. 168, p. 110643, (Oct.) 2020.
24. K. Shibata, K. Rinsaka, and T. Dohi, "Metrics-based software reliability models using non-homogeneous Poisson processes," In *Proc. of the IEEE International Symposium on Software Reliability Engineering,* pp. 52-61, (Nov.) 2006.

BIOGRAPHIES

Ebenezer Yawlui
Department of Electrical & Computer Engineering
University of Massachusetts – Dartmouth
285 Old Westport Road
North Dartmouth, MA 02747, USA

e-mail: eyawlui@umassd.edu

Ebenezer Yawlui is a MS student in the Department of Electrical & Computer Engineering at the University of Massachusetts Dartmouth. He received his BS (2020) in Electrical Engineering from Regional Maritime University, Ghana.

Priscila Silva, MS
Department of Electrical & Computer Engineering
University of Massachusetts – Dartmouth
285 Old Westport Road
North Dartmouth, MA 02747, USA

e-mail: psilva4@umassd.edu

Priscila Silva is a PhD student in the Department of Electrical & Computer Engineering at the University of Massachusetts Dartmouth. She received her MS (2022) in Computer Engineering from UMassD.

Vidhyashree Nagaraju, PhD
Tandy School of Computer Science, University of Tulsa
800 South Tucker Drive
Tulsa, OK 74104, USA

e-mail: vidhyashree-nagaraju@utulsa.edu

Vidhyashree Nagaraju is an Assistant Professor in the Tandy School of Computer Science at the University of Tulsa. She received her PhD (2020) in Computer Engineering from UMassD.

Lance Fiondella, PhD
Department of Electrical & Computer Engineering
University of Massachusetts – Dartmouth
285 Old Westport Road
North Dartmouth, MA 02747, USA

e-mail: lfiondella@umassd.edu

Lance Fiondella is an Associate Professor in the Department of Electrical & Computer Engineering at the University of Massachusetts Dartmouth and the Director of the University's Cybersecurity Center, a NSA/DHS Center of Academic Excellence in Cyber Research (CAE-R). He received his PhD (2012) in Computer Science & Engineering from the University of Connecticut.