J. Software Engineering & Applications, 2010, 3, 603-609
doi:10.4236/jsea.2010.36070 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

Dynamic Two-phase Truncated Rayleigh Model for Release Date Prediction of Software

Lianfen Qian[1], Qingchuan Yao[2], Taghi M. Khoshgoftaar[2]

[1]Department of Mathematical Sciences, Florida Atlantic University, Boca Raton, USA; [2]Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, USA.
Email: lqian@fau.edu, qingchuan_yao@yahoo.com, taghi@cse.fau.edu

Received October 23rd, 2009; revised November 13th, 2009; accepted November 15th, 2009.

ABSTRACT

Software reliability modeling and prediction are important issues during software development, especially when one has to reach a desired reliability prior to software release. Various techniques, both static and dynamic, are used for reliability modeling and prediction in the context of software risk management. The single-phase Rayleigh model is a dynamic reliability model; however, it is not suitable for software release date prediction. We propose a new multi-phase truncated Rayleigh model and obtain parameter estimation using the nonlinear least squares method. The proposed model has been successfully tested in a large software company for several software projects. It is shown that the two-phase truncated Rayleigh model outperforms the traditional single-phase Rayleigh model in modeling weekly software defect arrival data. The model is useful for project management in planning release times and defect management.

Keywords: *Software Testing, Weekly Defect Arrival Data, Single-Phase Rayleigh Model, Two-Phase Truncated Rayleigh Model, Software Reliability*

1. Introduction

Software reliability is a key attribute of software quality. Various models have been developed for software reliability engineering [1]. The rising complexity, size and functionality of software systems make software reliability prediction difficult. The problem is compounded with short development times and strict release deadlines. Consequently, predicting the release date for achieving pre-specified system reliability has become a very important issue in software project development. Reliability modeling can not only assist in fulfilling commitments and project deadlines, but also aid in efficient resource management and planning.

Software reliability is the probability of failure-free software operation for a given period of time in a given operating environment. The key attribute in software reliability engineering is the number of defects observed in specified time intervals (e.g. weeks). Software reliability prediction models assess a software product's reliability or estimate the number of latent defects when it is released to the customers. Such an estimate is important for two reasons: 1) as an objective statement of the quality of the product and 2) for resource planning in the software maintenance phase.

There are two categories of software reliability models: static and dynamic models. Among the static models, Bayesian belief networks [2] and utilizing software process metrics are relatively popular. Related literature also proposes various models for software defect prediction which can be used to indirectly gauge software reliability [3,4]. The primary drawback among static models can not effectively capture the software process and its variations during the course of software project development. On the other hand, a dynamic software reliability model is reflective of the software testing phase and is generally applicable before product release.

Among dynamic models, the (single-phase) Rayleigh model has been shown suitable to fit software defect arrival patterns [5,6]. A single-phase Rayleigh model divides the whole software development life cycle into six stages that are in chronological order: High Level Design (HLD), Low Level Design (LLD), CODING, Unit Testing (UT), Integration Testing (IT) and System Testing (ST). The six stages are assigned to a sequence of numerical scales. That is: HLD = 0.5, LLD = 1.5, CODING = 2.5, UT = 3.5, IT = 4.5 and ST = 5.5 [5]. Those numerical assignments seem rather *ad hoc*. Instead, we

could assign the six stages to, for instance, $\{t_1, t_2, t_3, t_4, t_5, t_6\}$, as long as $\{t_i\}_{i=1}^{6}$ satisfies $t_1 < t_2 < t_3 < t_4 < t_5 < t_6$. With different numerical assignments for the six stages, the fitted single-phase Rayleigh models could show a much different accuracy pattern, as shown in **Figure 1**.

The defects/KLOC in **Figure 1** is reconstructed from the work of Thangarajan *et al*. [5]. The quadratic fit is shown to illustrate that the small pairs of data could be fitted well by an arbitrary model such as quadratic model, rather than just single-phase Rayleigh model. Also, by assigning one numerical number to each stage, the data set now contain only six pairs at most. For prediction purposes, most likely 3, 4 and 5 pairs available, such a small sample size offers no confidence in the reliability prediction.

During the software development life cycle, collecting one single representative number for each stage results in a very small sample size. Furthermore, it is more likely that the data of major software defects are followed weekly, hence allowing project management to monitor the dynamic progress of the software development process. Our motivated weekly software development defects data set, **Figure 2**, shows the serious inadequacy of the single-phase Rayleigh model. This leads to our research on developing a better dynamic software reliability model to estimate the number of major defects, hence predict software release date.

The existing organizational reliability prediction model for software release date prediction at a large software company, where the weekly data in **Figure 2** were collected, is the dynamic single-phase Rayleigh model [5,6]. The software process in the organization consisted of two or more development phases. This is due to the software production cycles, availability of supporting hardware (e.g. wingboard/test phones) in the earlier software development stages, man-power management (e.g. testers' rearrangement) during the software development phases, and other dynamic issues during development. **Figure 2**, for instance, shows that the scatter plot overlaid with the single and the newly proposed two-phase truncated (piecewise, for short) Rayleigh models for the data set from the large software company. It is clear that the two-phase truncated Rayleigh model fits the data much better than the single-phase Rayleigh model.

Motivated by the example, we propose a multiple-phase truncated Rayleigh model in this paper. Such a model is better suited to fit the weekly defect arrival patterns during software development process. For simplicity reasons, we focus on the two-phase truncated (piecewise) Rayleigh model. The model can be extended to include additional phases reflecting the development process. It is shown through empirical modeling that the model accuracy is significantly improved. Furthermore, using the two-phase truncated Rayleigh model, the release date is predicted with a much higher confidence level.

The paper is organized as follows: Section 2 summarizes the single-phase Rayleigh model and proposes the multi-phase model, with a focus on the two-phase truncated Rayleigh model. Section 3 presents the algorithms of nonlinear least squares estimators of the model parameters and flowcharts of the dynamic process. Section 4 applies the proposed two-phase truncated Rayleigh model to defect arrival data of a large real-world software project from the large software organization. Finally, Section 5 concludes the paper and provides suggestions for future work.

2. Multi-Phase Truncated Rayleigh Models for Software Reliability Prediction

The dynamic single-phase Rayleigh model is a standard technique for software reliability modeling, and has been widely used for the software project and quality management in the software industry. The software organization, from which our case study data is obtained, has utilized the dynamic single-phase Rayleigh model for several of their previous software project developments.

The single-phase Rayleigh model is a parametric regression model with the regression function specified by the Rayleigh distribution with a multiplier coefficient. When the parameters of the Rayleigh distribution are estimated based on the updated data from a software project, dynamic projections about the number of defects for the software can be made based on the model over the software development life cycle.

The Rayleigh distribution is a special case of Weibull distribution, and has various applications including reliability estimation and life cycle pattern modeling [7,8] in developing software projects, life testing experiments in clinical studies dealing with cancer patients [9]. We now summarize the Rayleigh distribution. Denote t_m be the time at which the single-phase Rayleigh density curve reaches its peak. The cumulative distribution function of Rayleigh distribution with the constant multiplier K (the total number of latent defects) is

$$F(t; K, \theta) = K\left[1 - e^{-\theta t^2}\right],$$

where $\theta = 1/(2t_m^2)$ is the scale parameter. The single-phase Rayleigh model has a regression function parameterized as,

$$f(t; K, \theta) = 2K\theta t e^{-\theta t^2} \quad (1)$$

where both K and θ are the two parameters that need to be estimated using the data.

The single-phase Rayleigh model (1) does not fit the

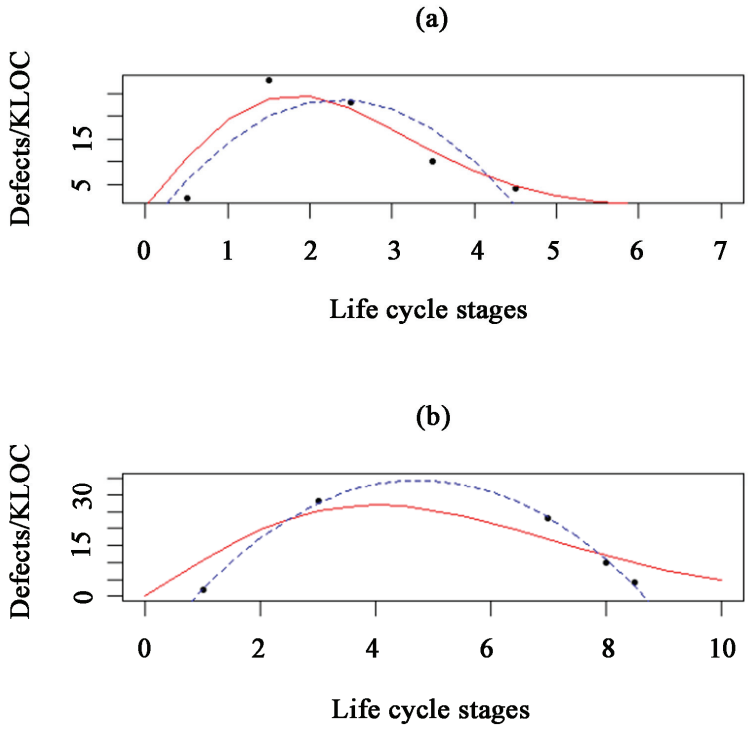

Figure 1. Single-Rayleigh model vs. quadratic model for two *ad hoc* numerical assignments for the ordinal stages in the software development life cycle. Solid line is for the single-phase Rayleigh model, while the dashed line is for the quadratic model. (a) (HLD,LLD, CODING, UT, IT, ST) = (0.5,1.5,2.5,3.5,4.5,5.5); (b) (HLD,LLD, CODING, UT, IT, ST) = (1,3,7,8,8.5,9)

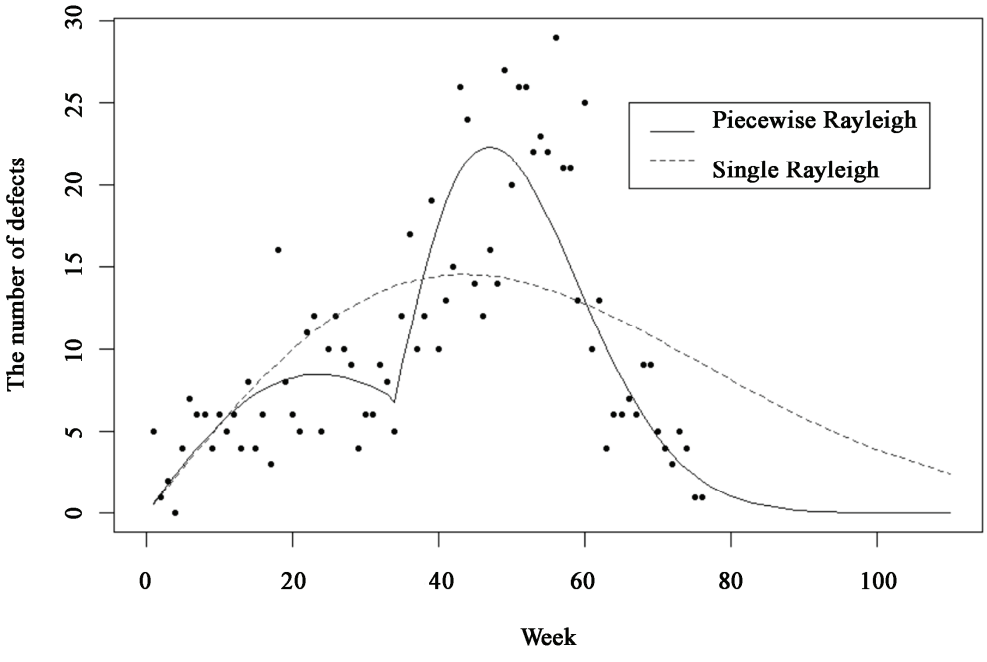

Figure 2. Major defects vs. development time in weeks

case study data set well. Actually it is a very poor fit as seen in **Figure 2**, and makes the case for a much needed improvement in modeling software defect arrival patterns. We propose a new multi-phase truncated Rayleigh model

defined as below:

$$g(t;\alpha)=\begin{cases} f(t;K_1,\theta_1), & 0\le t\le \tau_1 \\ f(t-\nu_1;K_2,\theta_2), & \nu_1\le \tau_1 < t\le \tau_2 \\ \cdots & \\ f(t-\nu_{d-1};K_d,\theta_d), & \nu_{d-1}\le \tau_{d-1}<t<\infty, \end{cases}$$

where d is the number of phases and $\alpha^T=(\theta_1,\ldots,\theta_d,K_1,\ldots,K_d,\nu_1,\ldots,\nu_{d-1},\tau_1,\ldots,\tau_{d-1})$ is the model parameter vector. For simplicity, we will discuss the case with $d = 2$, the two-phase truncated (piecewise) Rayleigh model with regression function parameterized as follows:

$$g(t;\alpha)=\begin{cases} f(t;K_1,\theta_1), & 0\le t\le \tau \\ f(t-\nu;K_2,\theta_2), & \nu\le \tau < t, \end{cases} \quad (2)$$

where τ is the location of the phase change, ν is the starting location for the second phase. Due to the nature of the software defect data, we suggest to use the left truncated Rayleigh model for the second phase. Then $\alpha^T=(\theta_1,\theta_2,K_1,K_2,\nu,\tau)$ is the parameter vector, need to be estimated.

3. Algorithms for Piecewise Rayleigh Models

In this section, we describe the nonlinear least squares estimator of the model parameters. Let $\{(t_i,d_i)\}_{i=1}^n$ be the defect arrival data collected over time, where $t_i=i/n$ is the time index for the ith week, d_i is the total number of software defects detected during the ith week, and n is the number of weeks observed. Let

$$S(\alpha)=\sum_{i=1}^{n}\left[d_i-g(t_i;\alpha)\right]^2.$$

Then the nonlinear least squares estimator is the minimization of $S(\cdot)$. Notice that $S(\cdot)$ is not differentiable in the location of phase change point τ and the starting point of second phase ν. In conjunction with nonlinear least squares method and Gauss-Newton algorithm, we utilize a four-step technique (described below) to obtain the estimators of the parameter vector α. The package *nls* in **R** language is used to obtain the estimates of the model parameters.

Step 1: For any given location of phase change τ in (0, 1), fix a ν such as $0<\nu\le\tau<1$, we compute the nonlinear least squares estimators [10], $\tilde{\alpha}_{1n}(\nu,\tau)$ for the smooth parameters $\alpha_1^T=(\theta_1,\theta_2,K_1,K_2)$, by minimizing $S(\alpha)$ over α_1.

Step 2: Substitute $\tilde{\alpha}_{1n}(\nu,\tau)$ into $S(\alpha)$ to obtain the profile objective function, $\tilde{S}(\nu,\tau)$. Then we minimize $\tilde{S}(\nu,\tau)$ over $0<\nu\le\tau<1$ for the given τ to obtain $\tilde{\nu}(\tau)$. Notice that the minimizer $\tilde{\nu}(\tau)$ is a function of τ.

Step 3: Substitute $\tilde{\nu}(\tau)$ into $\tilde{S}(\nu,\tau)$ to get $\tilde{S}(\tau)$. The minimizer of $\tilde{S}(\tau)$ over $\tau\in(0,1)$ is called the change point estimator, denoted by $\hat{\tau}$.

Step 4: Substitute $\hat{\tau}$ into $\tilde{\nu}(\tau)$ to get $\hat{\nu}$ and $\tilde{\alpha}_{1n}(\hat{\nu},\hat{\tau})$ to get $\hat{\alpha}_{1n}$. Put them together, we obtain the nonlinear least squares estimator, $\hat{\alpha}_n^T=(\hat{\alpha}_{1n}^T,\hat{\nu},\hat{\tau})$ of α.

Figures 3 and **4** illustrate the flow charts of the dynamic process of the algorithm for single-phase and multi-phase truncated Rayleigh models, respectively. We provide the flowchart for the single-phase Rayleigh model for comparison purpose.

4. Application to a Real Software Defect Data Set

The data set motivated our research were collected from Feb-25-06 to Aug-04-07 at a large software company. There are 76 weeks software defects arrival data. Number of major defects during a week is reported.

4.1 Single-phase vs. Piecewise Rayleigh Models

We illustrate the two-phase truncated Rayleigh model by fitting the software defect arrival data set. From **Figure 2**, it is observed that using two-phase truncated Rayleigh model improves the model fitting significantly compared to the single-phase Rayleigh model with respect to model accuracy and model goodness-of-fit. For comparison

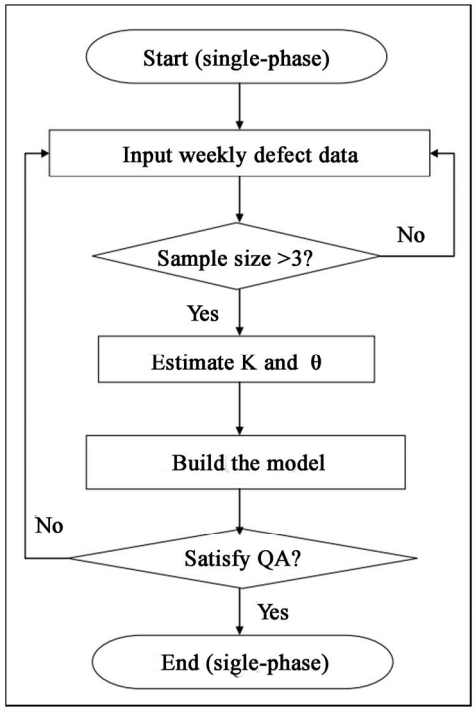

Figure 3. Algorithm for single-phase Rayleigh model

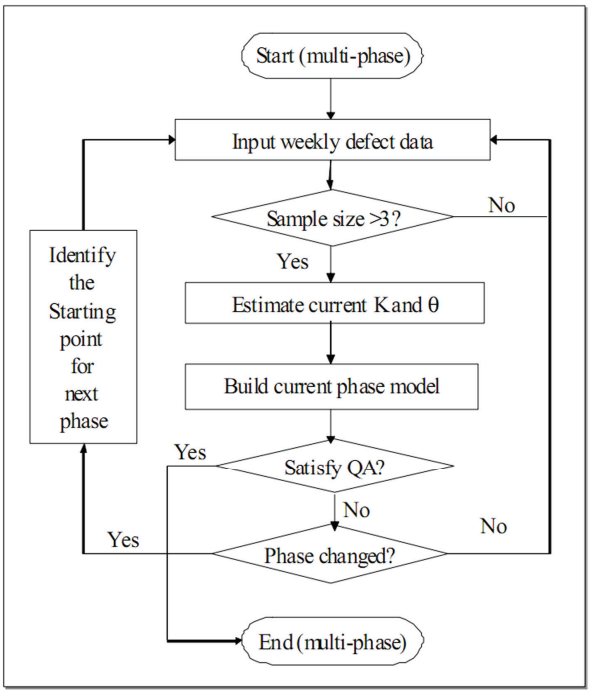

Figure 4. Algorithm for multi-phase Rayleigh model

purpose, the estimated single-phase Rayleigh regression function is given by,

$$\hat{g}(t) = 2\hat{K}\hat{\theta}te^{-\hat{\theta}t^2},$$

where $\hat{K} = 13.7111$ and $\hat{\theta} = 1.5297$.

For the two-phase truncated Rayleigh model (2), the estimated change is at the $\hat{\tau} = 33rd$ week with the starting point estimated at $\hat{\nu} = 31st$ week for the second phase. Hence phase one is from the first week to 33*rd* week and phase two is from the 34th week to the 76th week with estimated starting point at $\hat{\nu} = 31st$ week. The estimated first phase (right truncated) of the regression function is estimated as

$$\hat{g}(t) = 2\hat{K}_2\hat{\theta}_2(t-31/76)e^{-\hat{\theta}_2(t-31/76)^2},$$
$$\text{if } t > 33/76,$$

with $\hat{K}_1 = 4.3022, \hat{\theta}_1 = 5.2279,$ and the second phase (left truncated) of the regression function is estimated as

$$\hat{g}(t) = 2\hat{K}_2\hat{\theta}_2(t-31/76)e^{-\hat{\theta}_2(t-31/76)^2},$$
$$\text{if } t > 33/76,$$

with $\hat{K}_2 = 7.7731, \hat{\theta}_2 = 11.2445$. **Figure 2** shows the scatter plot overlaid with the two fitted curves using the single-phase and two-phase truncated Rayleigh models, respectively. From the fitted model, one can predict the future week's number of software defects and establish the quality assurance criterion and management for predicting the release date.

This proposed multi-phase truncated Rayleigh model can be utilized for modeling any future software development projects to obtain better prediction and provide more efficient estimation of the release date of the software product.

4.2 Quality Assurance Criterion for Release Date Prediction

In this section, we establish the quality assurance criterion for software release. The quality assurance criterion is determined by 95% and 99.9% confidence levels. That is, based on the fitted model, if the model shows that 95% or 99.9% of the total expected software defects has been detected, then we suggest that the software is ready for release.

For the single-phase Rayleigh model, we estimate the release date with 95% confidence level. We set $F\left(t;\hat{K},\hat{\theta}\right) = 0.95\hat{K}$ and solve for *t* or equivalently

$$1 - e^{-\hat{\theta}t^2} = 0.95.$$

This implies that the release date equals to the ceiling of $n * \sqrt{-\ln(1-.95)/\hat{\theta}} = 107$ weeks, where $\hat{\theta}$=1.5297. Hence, with 95% confidence the software project will

need 107−76 = 31 weeks of further testing before releasing the software product. That is the predicted release date using the 76 weeks of data is Feb-29-08 based on the single-phase Rayleigh Model. With 99% confidence, it will require even much longer testing time.

Alternatively, utilizing the two-phase truncated Rayleigh model (2), we set

$$\int_0^{\hat{\tau}/n} \hat{g}(t)dt + \int_{\hat{\tau}/n}^{i/n} \hat{g}(t)dt = 0.999\int_0^1 \hat{g}(t)dt$$

and solve for i to get the estimated release week with 99.9% confidence level. Equivalently, the estimated release week number, i, satisfies that

$$e^{-\hat{\theta}_2\left(\frac{i-\hat{\nu}}{n}\right)^2} = e^{-\hat{\theta}_2\left(\frac{\hat{\tau}-\hat{\nu}}{n}\right)^2} - \frac{\left[0.999\int_0^1 \hat{g}(t)dt - \int_0^{\hat{\tau}/n} \hat{g}(t)dt\right]}{\hat{K}_2} = A_0 - \frac{0.999A - A_1}{\hat{K}_2}, \quad (3)$$

where

$$A_0 = e^{-\hat{\theta}_2\left(\frac{\hat{\tau}-\hat{\nu}}{n}\right)^2} = 0.9922,$$

$$A_1 = \int_0^{\hat{\tau}/n} \hat{g}(t)dt = 2.6967,$$

$$A = \int_0^1 \hat{g}(t)dt = 10.2586.$$

Solving Equation (3) to obtain the estimated release week number:

$$i = \hat{\nu} + \left\{ n * \sqrt{-\left(1/\hat{\theta}_2\right)\ln\left(A_0 - \frac{0.999A - A_1}{\hat{K}_2}\right)} \right\} = 76 \text{ weeks},$$

where $\{x\}$ is the smallest integer greater or equal to x. This indicates that with 99.9% confidence that the estimated release week is the end of 76th week. That is the software is ready for release, with almost 100% confidence based on the two-phase truncated Rayleigh model. We note that the large software organization has adopted our new two-phase truncated Rayleigh model and is using it to predict the number of software defects dynamically and release dates for ongoing software projects. Our new two-phase truncated Rayleigh model has improved the software release life cycle a great deal and has saved a lot of man-powered resource for the large software organization.

4.3 Model Performance Check

We utilize three measures of goodness-of-fit to assess the performance of the models: root mean square error (RMSE), magnitude of relative error (MRE), and adjusted coefficient of determination R^2_{adj}. The root mean square error measures the model accuracy defined as the square root of mean squared residuals. That is,

$$RMSE = \sqrt{\frac{1}{n-5}\sum_{i=1}^{n}\left(d_i - \hat{d}_i\right)^2},$$

where d_i is the number of defects detected during the ith week, $\hat{d}_i$ is the fitted (predicted) value of d_i. The smaller the RMSE, the better the model fits.

The second criterion for assessment of the performance of model fitting used in the reliability literature is the mean magnitude of relative error, defined as

$$MRE = \frac{\sum_{i=1}^{n}\left|\frac{d_i - \hat{d}_i}{d_i}\right| I(d_i > 0)}{\sum_{i=1}^{n} I(d_i > 0)}.$$

The implicit assumption in this summary measure is that the seriousness of the absolute error is proportional to the size of the observations. The smaller the MRE, the better the model fits.

The third measure of goodness-of-fit used is the adjusted determination of coefficient R^2_{adj} which is the adjusted percentage of variation in the number of defects per week explained by the model. That is

$$R^2_{adj} = 1 - \frac{SSE/(n-5)}{SSTO/(n-1)},$$

where $SSE = (n\text{-}5)(RMSE)^2$ and

$$SSTO = \sum_{i=1}^{n}\left(d_i - \bar{d}\right)^2 \text{ with } \bar{d} = \sum_{i=1}^{n} d_i \Big/ n.$$

The higher of the R^2_{adj}, the better the model fits.

Table 1 summarizes the three performance criteria for the real-world weekly software defects data set using both single-phase and two-phase truncated (piecewise) Rayleigh models. Based on the reported *RMSE, MRE* and R^2_{adj} values, the two-phase truncated Rayleigh model is much better than the single-phase Rayleigh model. The *MRE* is reduced by about 50%, while the goodness-of-fit measure R^2_{adj} is roughly doubled for the two-phase truncated compared to the single-phase Rayleigh models. The two-phase truncated Rayleigh model explains the almost doubled variation in the number of defects than the single-phase Rayleigh model does. Thus, based on the given data, we conclude that the two-phase truncated Rayleigh model is an attractive model for predicting weekly software defects and release date of software projects.

Table 1. Model comparisons using RMSE, MRE and R^2_{adj}

	Criterion		
Model	RMSE	MRE	R^2_{adj}
Single-phase	5.97	0.76	36.6%
Two-phase	4.13	0.36	70.4%

5. Conclusions

The research was motivated by a real-world software defect arrival data over many weeks from a large software organization. The paper proposes a new multi-phase truncated (focusing on a two-phase truncated model) Rayleigh model in fitting weekly defect arrival data.

It is shown that the proposed model is much more accurate than the existing single-phase Rayleigh model. The single-phase model was previously used by the organization during software development. Using both MRE and R^2_{adj} performance measures, the proposed model almost doubled the prediction accuracy, hence, shortening the release date prediction with a higher confidence level. From a software reliability perspective, our proposed two-phase truncated Rayleigh prediction model will help in the management and planning of project resources toward bettering the software release cycle time.

The two-phase truncated Rayleigh model can be easily extended to a multi-phase truncated Rayleigh model. Hence it can be used to predict release date for future software projects with a higher confidence level. A general multi-phase Rayleigh software release prediction model can be developed to automatically detect and reflect all the change locations and the starting points of the software development phases so that the multiple-phase truncated Rayleigh software prediction model can be generated to automatically forecast the software release time.

REFERENCES

[1] M. R. Lyu, "Software Reliability: To Use or not to Use?" *Proceedings of 5th International Symposium on Software Reliability Engineering*, 66-73 November 1994.

[2] Y. Wang and M. Smith, "Release Date Prediction for Telecommunication Software Using Bayesian Belief Networks," *Proceedings of the* 2002 *IEEE Canadian Conference on Electrical and Computer Engineering*, 2002, pp. 738-742.

[3] T. M. Khoshgoftaar and N. Seliya, "Fault Prediction Modeling for Software Quality Estimation: Comparing Commonly Used Techniques," *Empirical Software Engineering Journal*, Vol. 8, No. 3, 2003, pp. 255-283.

[4] T. M. Khoshgoftaar and N. Seliya, "Comparative Assessment of Software Quality Classification Techniques: An Empirical Case Study," *Empirical Software Engineering Journal*, Vol. 9, No. 3, 2004, pp. 229-257.

[5] M. Thangarajan and B. Biswas, "Mathematical Model for Defect Prediction across Software Development Life Cycle," *The SEPG* (*Software Engineering Process Group*) *Conference*, India, 2000. http://www.qaiindia.com/Conferences/SEPG2000/index.html

[6] S. H. Kan, "Metric and Models in Software Quality Engineering," 2nd Edition, Addison Wesley, Massachusetts, 2003.

[7] P. V. Norden, "Useful Tools for Project Management," *Operations Research in Research and Development*, B. V. Dean, Ed., John Wiley & Sons, New York, 1963.

[8] L. H. Putman, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transaction on Software Engineering*, Vol. SE-4, 1978, pp. 345-361.

[9] S. K. Bhattacharya and R. K. Tyagi, "Bayesian Survival Analysis Based on the Rayleigh Model," *Trabajos de Estadistica*, Vol. 5, No. 1, 1990, pp. 81-92.

[10] D. M. Bates and J. M. Chambers, "Nonlinear Models," Chapter 10 of Statistical Models in S. J. M. Chambers and T. J. Hastie, Eds., Wadsworth & Brooks/Cole, 1992.

