Int. J. Appl. Math. Comput. Sci., 2017, Vol. 27, No. 4, 777–783
DOI: 10.1515/amcs-2017-0054

SOFTWARE RELIABILITY GROWTH MODELING FOR AGILE SOFTWARE DEVELOPMENT

SHUBHAM RAWAT [a,*], NUPUR GOYAL [b], MANGEY RAM [b,a]

[a] Department of Computer Science and Engineering
Graphic Era University, Dehradun, Uttarakhand, India
e-mail: geu.shubham@gmail.com

[b] Department of Mathematics
Graphic Era University, Dehradun, Uttarakhand, India

The frequent incremental release of software in agile development impacts the overall reliability of the product. In this paper, we propose a generic software reliability model for the agile process, taking permanent and transient faults into consideration. The proposed model is implemented using the NHPP (non-homogenous Poisson process) and the Musa model. The comparison of the two implementations yields an effective, empirical and reliable model for agile software development.

Keywords: software reliability, testing, non-homogeneous poison process, transient faults, permanent faults, Musa model.

1. Introduction

Today, software has become indispensable both as a product and as a driving force for the creation of new technologies and the refinement of the existing ones. In the past decades the role of software has amplified from generating mathematical data to controlling and monitoring modern systems such as broadcasting, financial transactions systems, national defense organizations, medical systems, household appliances, automobiles, and many more. As software becomes more requisite and convoluted day-by-day, its reliability grows into a critical factor in the determination of software quality (Wilson, 1997).

Software reliability is defined in statistical terms as *the probability of failure-free operation of a computer program in a specified environment for a specified time* (Musa *et al.*, 1987). Calculation and estimation of software reliability is an essential tool for developing reliable software systems. Several methods regarding assessment and investigation through metrics, models and tools have been introduced in the last four decades (Rawat *et al.*, 2015).

With the drastic growth of software industry, competitive pressure, convulsive market forces and rapidly evolving requirements have become an inevitable part of any software project. To ensure success, software systems need to be rapidly developed and delivered, and must also accommodate the ever changing requirements. The traditional plan-driven approach focuses on a complete requirement specification prior to designing, constructing, and testing the system. It is more inclined towards planning, designing, and documenting the system than on the software development process. Therefore, planning dominates over the actual program development and testing. In the rapid development and for delivery to be in place, there is a need for a more flexible and adaptable development process (Sommerville, 2011). Therefore, most firms have changed their approach from traditional plan-driven to agile for software development. According to a survey by Forrester Research (West *et al.*, 2010), about half of software engineers use agile processes for information systems development.

The agile way is a more informal approach to software development directed by a set of developmental guidelines. It accentuates customer collaboration, highly motivated team members, incremental delivery of software, adaptability to changes and maintaining development simplicity (Agile Alliance, 2017). The agile approach follows the philosophy of *release early, release often*, which emphasizes the importance of early

*Corresponding author

and frequent releases of the system. Early releases of software deliver a core product with a set of limited functionalities and each subsequent release incrementally adds new functionalities, repairs existing faults or adapts new technologies.

Since every release adds new code to the system, it is capable of introducing new faults. Although a new release is meant to improve the system, there is always a possibility of degeneration due to the adjunct of new bugs.

Essentially, every release adds some fault content to the existing system. A possible software failure curve is shown in Fig. 1. In the useful life phase, frequent releases are made, which soars the overall failure rate, hence dropping the reliability of the system.

Software reliability modeling (SRM) is one of the key areas of research in software reliability. An SR model offers an efficient method of evaluating and forecasting software reliability based on certain assumptions about the fault in software and fault exposure in a given usage environment (Palma *et al*., 1993).

Jelinski and Moranda (1972) proposed the first software reliability (SR) model and hundreds of models have been introduced so far (Rawat *et al*., 2015). Some that have received much attention are exponential SRGMs (software reliability growth models) (Goel and Okumoto, 1979), which assume a constant disaster force of all the faults. In S-shaped SRGMs (Yamada *et al*., 1983), failure intensity varies with time; and in hyper-exponential SRGMs (Matsumoto *et al*., 1988), failure intensity of faults is constant within the same time interval. The imperfect debugging model of Goel and Okumoto stressed the fact that not all faults in the system are removed when they are detected (Goel, 1985). The fault detection rate, described by a constant (Pham and Pham, 2000) or by a learning phenomenon of developers (Fang and Yeh, 2016), is also studied in the literature.

Many NHPP based SRGMs have been proposed earlier. Yamada *et al*. (1984) suggested an exponential SRGM that considered two types of faults. The SRGM proposed by Pham and Zhang (2003) assumed multiple failure characteristics. Singh *et al*. (2014) classified faults subjected to the time of detection of a fault. The model due to Kapur *et al*. (1995) and Singh *et al*. (2008) categorized software faults as simple, hard and complex faults. Research has established that faults differ in testing effort and hence should be examined as a distinct entity.

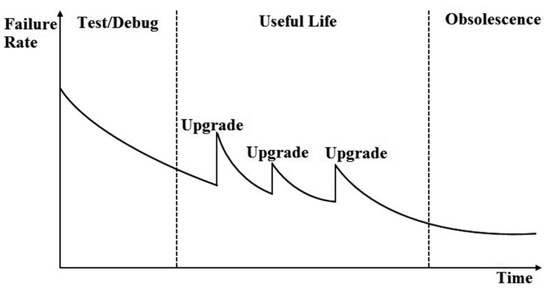

Fig. 1. Software failure curve.

In agile software development, the incremental delivery is coupled with continuous testing. In every release, existing faults are eliminated and potentially valuable features are delivered to the customer (Dingsøyr and Lassenius, 2016). The interaction between new implementation and the existing one usually soars the fault content of the system, hence dwindling the system reliability. In practice, when software is released, it contains some hidden faults which are reported post deployment and rectified in future releases.

In this paper, we propose a model to analyze software reliability through two different predefined models namely, the NHPP (non-homogenous Poisson process) and the Musa model. The enhancement in software reliability is obtained by a comparative study of the estimated results of both models.

In our model, we consider two types of faults: long-lasting (or permanent) faults and temporary (or transient) faults. The former exist in the system until they are eliminated by coding effort. On the other hand, they remain in the system for a short period of time and then disappear. Both types of faults are removed with varying testing effort and latter treated separately. The models in earlier research have ignored these two types of faults, their testing effort and potential to affect the overall reliability of the end product from the stance of agile development. In our model, permanent faults from the i-th release might be identified and removed in the $(i+1)$-th release or the $(i+2)$-th release or the $(i+3)$-th release, and so on. Transient faults, which can be removed by minimal testing effort, are removed in the release they arise. Also in the i-th release, we consider the remaining permanent faults of all the releases prior to it, i.e., the $(i-1)$-th, the $(i-2)$-th, $\ldots$, second, first. Any software which is a final product of a series of releases is vulnerable to critical reliability consequences. For such software, modeling becomes essential to estimate its reliability.

2. Model details

2.1. Model description. An SRGM provides a systematic way of assessing and predicting softwares reliability based on certain assumptions about the fault in the software and fault exposure in a given usage environment (Rawat *et al*., 2015). An SRGM for the agile approach must accommodate the release wise fault content of the software. The NHPP and the Musa model have inherent fault counting behavior and allow an expected number of faults to vary with time. Hence, we use the NHPP and the Musa model to formulate the SRGM.

2.1.1. NHPP model. NHPP based modeling has been extensively studied in the literature for its worthwhile detection of software faults and easy implementation of software reliability analysis (Lai and Garg, 2012). Hence, we formulate our model using the NHPP. Let us assume that faults can exist in software coding arbitrarily and their emergence is a function of time. The number of faults at any instant t in software coding is $N(t)$ which is a counting process and it denotes the increasing number of software disasters or faults at any instant t. $N(t)$ can be defined as a non-homogeneous Poison process (NHPP) if the following conditions are met:

(i) $N(0) = 0$.

(ii) Only one fault can occur in the time interval $(t, t + \mathrm{d}t)$.

(iii) The occurrence of faults does not depend on the previous faults.

For the software reliability growth model, $N(t)$ is exposed to be an NHPP with a mean value function $m(t)$, where $m(t)$ is the total number of faults in each release. The NHPP gives

$$P[N(t)] = n = \frac{[m(t)]^n}{n!}\mathrm{e}^{(-m)(t)}, \quad n \geq 0. \tag{1}$$

Based on the definition of the mean value function, $m(t)$ is the average quantity of errors occurring in the time interval $(0, t)$ and can be expressed in terms of failure rates, i.e.,

$$m(t) = \int_0^t \lambda(s)\,\mathrm{d}s \tag{2}$$

and

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = \frac{f(t)}{1 - F(t)}(a - m(t)) \tag{3}$$

with the initial condition

$$m(0) = 0.$$

After solving Eqns. (2) and (3) with the help of the initial condition, we have

$$m(t) = aF(t),$$

where a is constant and $F(t)$ is a probability distribution function.

2.1.2. Musa model. The Musa model uses program execution time as the independent variable. This model has had the widest distribution among software reliability models and applicability (Farr, 1996). A simplified version of the Musa model is

$$n = N_0\Big[1 - \exp\Big(\frac{-Ct}{N_0T_0}\Big)\Big], \tag{4}$$

where N_0 is the inherent number of errors, T_0 the mean time to failure (MTTF) at the start of testing and C is the testing compression factor equal to the ratio of the equivalent operating time to the testing time.

The present MTTF,

$$T = T_0 \exp\Big(\frac{Ct}{N_0T_0}\Big), \tag{5}$$

yields

$$R(t) = \exp\Big(\frac{-t}{T}\Big). \tag{6}$$

2.2. Assumptions and nomenclature. This model has the following assumptions:

(i) Fault removal in any release depends upon the faults from all previous releases.

(ii) The number of faults at the start of testing is finite.

(iii) The only source of software failure is a coding fault.

(iv) Every release adds new features to the system which remove some existing faults, but also add some new faults.

(v) The system is under imperfect debugging.

(vi) The failure rates for permanent and transient faults follow exponential and Weibull distributions, respectively.

The notation used throughout this work is shown in Table 1.

3. Formulation of the SRGM

3.1. Using the NHPP model. In the proposed model, the NHPP is used to describe the time dependent nature of $N(t)$ detected by specific testing. Since two types of fault have been considered, the total number of faults in the n-th release becomes

$$m_n(t) = m_{n1}(t) + m_{n2}(t), \tag{7}$$

where 1 and 2 represent the permanent and transient faults, respectively, and n denotes the release number. We assume that the mean value function of permanent faults follows the exponential distribution given by Goel and Okumoto (1980), and the mean value function of transient

Table 1. Notation.

$m(t)$	Total number of faults
$f(t)$	Probability density function
$F(t)$	Probability distribution function (pdf)
t_{n-1}	Time for n-th release
a_n	Initial fault content
b_{n1}	Detection rate for permanent fault
b_{n2}	Detection rate for transient fault
RL_n	n-th release
λ	Failure rate for n-th release
P	Probability of fault occurrence
F_{n1}	Pdf for permanent faults in n-th release
F_{n2}	Pdf for transient faults in n-th release
R	Reliability
C	Testing compression factor
T	MTTF

faults follows a Weibull distribution (Yamada, 1994). Thus, the mean value function of the software reliability growth model can be written as (Kapur *et al.*, 2014)

$$m_n(t) = \lambda a F_{n1}(t) + (1-\lambda) a F_{n2}(t), \tag{8}$$

where

$$F_{n1}(t) = 1 - \exp(-b_{n1}t), \tag{9}$$

$$F_{n2}(t) = 1 - (1 + b_{n2}t)\exp(-b_{n2}t). \tag{10}$$

Additionally, we assume that the software has some permanent faults that get propagated to future releases. Thus, the generalized equation for the total number of expected faults in the n-th release can be expressed as

$$\begin{aligned} m_n(t) &= \lambda_n a_n F_{n1}(t - t_{n-1}) + (1-\lambda)_n a_n F_{n2}(t - t_{n-1}) \\ &\quad + \sum_{i=1}^{n-1} \lambda_i a_i (1 - F_{i1}(t_{n-1} - t_{n-2})) F_{n1}(t - t_{n-1}). \end{aligned} \tag{11}$$

To examine the influence of errors on reliability owing to add-ons at a different instant, a multi-release software reliability model has been established. Software reliability is an attribute of any software that consistently performs the required tasks according to its specifications (Goyal and Ram, 2014). In other words, software reliability is the probability, or quantity, of faults existing in the software (Pandey and Goyal, 2015). Thus, software reliability can be described as

$$R(t) = \exp(-m(t)t). \tag{12}$$

The generalized expression of software reliability in the n-th release is

$$R_n(t) = \exp(-m_n(t)t). \tag{13}$$

3.2. Using the Musa model. For modeling the software developed using the agile approach, the reliability of different software releases can be evaluated using the Musa model. The generalized formula of software reliability in the n-th release is

$$R(t) = \exp\left(\frac{-t}{T}\right). \tag{14}$$

4. Numerical computations

4.1. NHPP based SRGM. For the practical utility of the SRGM, we analyze the software reliability trend for four successive releases, taking $n = 1, 2, 3, 4$ in Eqn. (11). Setting the other parameters as $a_1 = 100, a_2 = 80, a_3 = 60, a_4 = 40, b_{11} = 0.1, b_{12} = 0.12, b_{21} = 0.01, b_{22} = 0.14, b_{31} = 0.001, b_{32} = 0.16, b_{41} = 0.0001, b_{42} = 0.18, \lambda_1 = 0.47690, \lambda_2 = 0.17920, \lambda_3 = 0.01420, \lambda_4 = 0.09057$ (Aggarwal *et al.*, 2011; Singh *et al.*, 2014), software reliability for four releases is summarized in Table 2. Figure 2 depicts the boost in reliability with advancing releases.

Table 2. Reliability of software developed using an agile process under the NHPP.

Time (t)	Reliability $R(t)$			
	RL1	RL2	RL3	RL4
0	1.00000	1.00000	1.00000	1.00000
0.1	0.95330	0.99618	0.99987	0.99998
0.2	0.82545	0.97751	0.99648	0.99924
0.3	0.64871	0.94136	0.98548	0.99436
0.4	0.46240	0.88654	0.96301	0.98218
0.5	0.29874	0.81372	0.92612	0.95996
0.6	0.17484	0.72556	0.87309	0.92565
0.7	0.09264	0.62649	0.80389	0.87809
0.8	0.04441	0.52227	0.72035	0.81730
0.9	0.01926	0.41913	0.62608	0.74449
1	0.00755	0.32291	0.52608	0.66212

4.2. Musa based SRGM. Using the formulations of the Musa model, we derive the reliability values for the agile process model. Setting $T_1 = 2.09687, C_1 = 0.041666, N_1 = 100, t_1 = 0.041666, T_2 = 2.09687, C_2 = 0.083333, N_2 = 80, t_2 = 0.083333, T_3 = 70.42253, C_3 = 0.012500, N_3 = 60.79, t_3 = 0.012500, T_4 = 110.41183, C_4 = 0.166666, N_4 = 40, t_4 = 0.166666$ in Eqn. (14), we determine software reliability for four releases. The values are indicated in Table 3.

Figure 3 shows reliability trends in different releases of the software using the Musa model. It is evident from the graph that the reliability strengthens in consecutive releases.

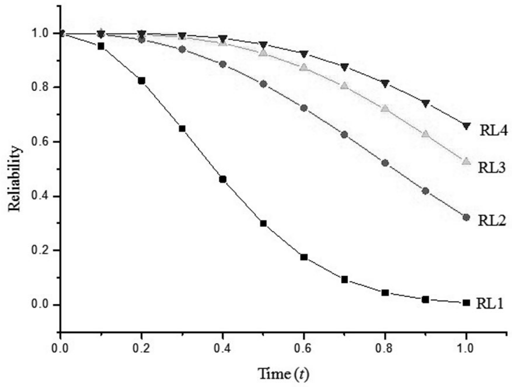

Fig. 2. Reliability trends through the NHPP model.

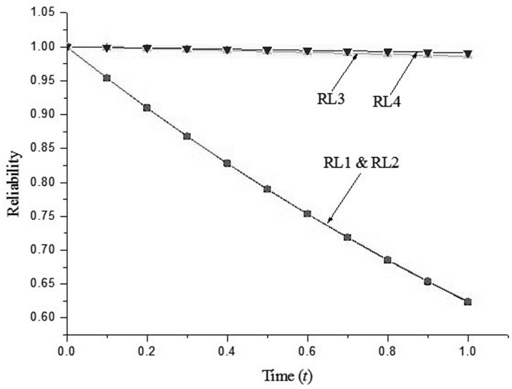

Fig. 3. Reliability trends through the Musa model.

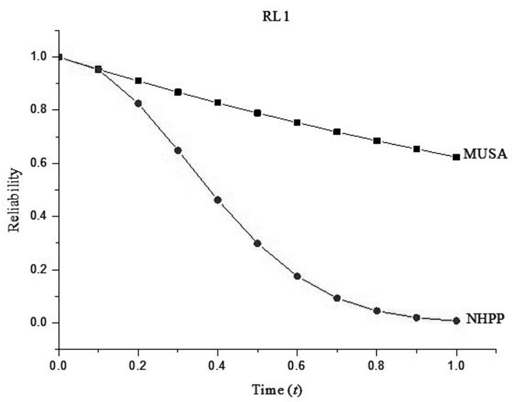

Fig. 4. Comparison of reliability through NHPP and Musa model based SRGMs in Release 1.

5. Comparative study

The results of the NHPP based SRGM and the Musa model based SRGM were analyzed and compared. The release wise reliability trend of releases 1, 2, 3 and 4 for the two models is shown in Figs. 4–7, respectively. We observe that the reliability growth in every release in the

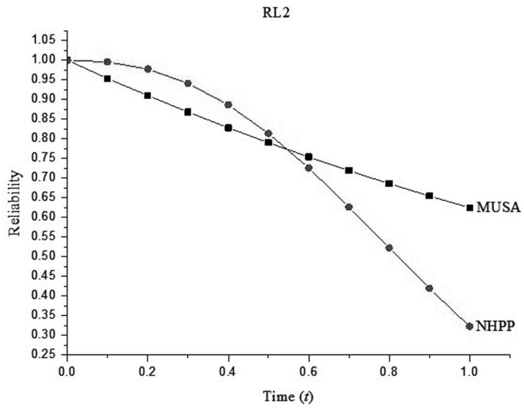

Fig. 5. Comparison of reliability through NHPP and Musa model based SRGMs in Release 2.

Table 3. Reliability of software developed using an agile process under the Musa model.

Time (t)	Reliability $R(t)$			
	RL1	RL2	RL3	RL4
0	1.00000	1.00000	1.00000	1.00000
0.1	0.95388	0.95399	0.99858	0.99912
0.2	0.90989	0.91010	0.99717	0.99823
0.3	0.86793	0.86823	0.99576	0.99735
0.4	0.82790	0.82829	0.99434	0.99647
0.5	0.78972	0.79018	0.99294	0.99559
0.6	0.75330	0.75383	0.99153	0.99471
0.7	0.71856	0.71915	0.99012	0.99383
0.8	0.68542	0.68606	0.98872	0.99296
0.9	0.65381	0.65450	0.98732	0.99208
1	0.62366	0.62439	0.98592	0.99120

Musa model is higher than that in the NHPP based model. Also, in the former the reliability attains constancy in the consecutive releases.

The graphs outline the contrast in reliability through NHPP and Musa based SRGMs, respectively, in various releases of the software. The pictorial representation of the reliability trend makes it apparent that the Musa model based SRGM has more promising reliability out-turn than the NHPP based SRGM.

6. Conclusions

In anticipation of a better market position and customer satisfaction, many software development firms are adopting agile practices. In this paper, we have proposed an SRGM for software under agile development using the NHPP and the Musa model. Two types of faults, i.e., permanent and transient, have been treated independently for each release. Our comparison of the reliability of the two SRGMs indicates that the Musa model based

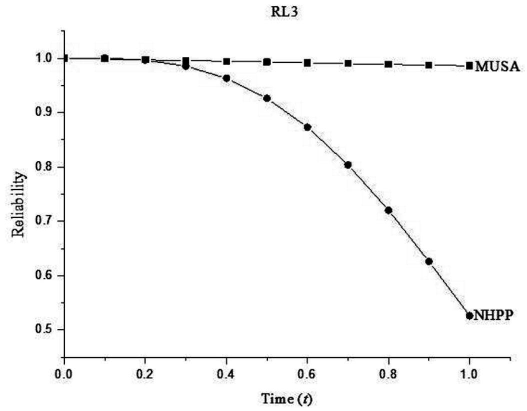

Fig. 6. Comparison of reliability through NHPP and Musa model based SRGMs in Release 3.

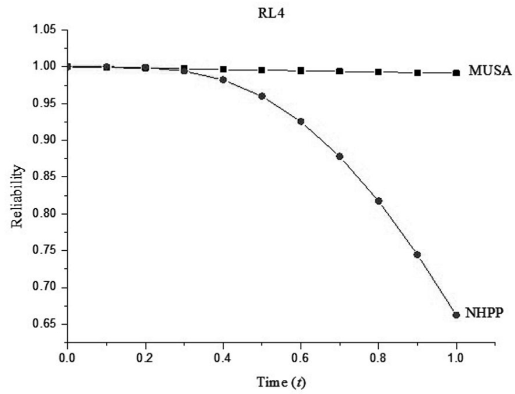

Fig. 7. Comparison of reliability through NHPP and Musa model based SRGMs in Release 4.

SRGM yields better reliability results than the NHPP based SRGM. The capability of the Musa model to attune substantial changes in software over time as faults are observed makes it perform better. In the future, we will study the reliability of various other agile practices, extending the ideas discussed in this paper.

References

Aggarwal, A.G., Kapur, P. and Garmabaki, A. (2011). Imperfect debugging software reliability growth model for multiple releases, *Proceedings of the 5th National Conference on Computing for Nation Development-INDIACOM, New Delhi, India*, pp. 337–344.

Agile Alliance (2017). `http://www.agilealliance.org/`.

Dingsøyr, T. and Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery, *Information and Software Technology* **77**(1): 56–60.

Fang, C.-C. and Yeh, C.-W. (2016). Effective confidence interval estimation of fault-detection process of software reliability growth models, *International Journal of Systems Science* **47**(12): 2878–2892.

Farr, W. (1996). Software reliability modeling survey, *in* M.R. Lyu (Eds.), *Handbook of Software Reliability Engineering*, McGraw-Hill, Inc. Hightstown, NJ, pp. 71–117.

Goel, A.L. (1985). Software reliability models: Assumptions, limitations, and applicability, *IEEE Transactions on Software Engineering* **11**(12): 1411–1423.

Goel, A.L. and Okumoto, K. (1979). Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Transactions on Reliability* **28**(3): 206–211.

Goel, A.L. and Okumoto, K. (1980). A time dependent error detection rate model for software performance assessment with applications, *Technical report*, DTIC Document, `http://www.dtic.mil/docs/citations/ADA088186`.

Goyal, N. and Ram, M. (2014). Software development life cycle testing analysis: A reliability approach, *Mathematics in Engineering, Science & Aerospace* **5**(3): 313–329.

Jelinski, Z. and Moranda, P. (1972). Software reliability research, *in* W. Freiberger (Ed.), *Statistical Computer Performance Evaluation*, Academic Press, New York, NY, pp. 465–484.

Kapur, P., Sachdeva, N. and Singh, J.N. (2014). Optimal cost: A criterion to release multiple versions of software, *International Journal of System Assurance Engineering and Management* **5**(2): 174–180.

Kapur, P., Younes, S. and Agarwala, S. (1995). Generalised Erlang model with n types of faults, *ASOR Bulletin* **14**(1): 5–11.

Lai, R. and Garg, M. (2012). A detailed study of NHPP software reliability models, *Journal of Software* **7**(6): 1296–1306.

Matsumoto, K.I., Inoue, K., Kikuno, T. and Torii, K. (1988). Experimental evaluation of software reliability growth models, *11th International Symposium FTCS-18, Tokyo, Japan*, pp. 148–153.

Musa, J.D., Iannino, A. and Okumoto, K. (1987). *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, Inc., New York, NY.

Palma, J., Tian, J. and Lu, P. (1993). Collecting data for software reliability analysis and modeling, *Proceedings of the 1993 Conference of the Centre for Advanced Studies on Collaborative Research: Software Engineering, Toronto, Canada,* Vol. 1, pp. 483–494.

Pandey, A.K.and Goyal, N.K. (2015). Background: Software quality and reliability prediction, *in* A.K. Pandey and N.K. Goyal (Eds.), *Early Software Reliability Prediction*, Studies in Fuzziness and Soft Computing, Vol. 303, Springer, New Delhi, pp. 17–33.

Pham, H. and Zhang, X. (2003). NHPP software reliability and cost models with testing coverage, *European Journal of Operational Research* **145**(2): 443–454.

Pham, L. and Pham, H. (2000). Software reliability models with time-dependent hazard function based on Bayesian approach, *IEEE Transactions on Systems, Man, and Cybernetics A: Systems and Humans* **30**(1): 25–35.

Rawat, S., Rawat, R.S. and Ram, M. (2015). A review on software reliability: Metrics, models and tools, *Mathematics in Engineering, Science & Aerospace* **6**(2): 135–156.

Singh, O., Anand, A., Aggrawal, D. and Singh, J. (2014). Modeling multi up-gradations of software with fault severity and measuring reliability for each release, *International Journal of System Assurance Engineering and Management* **5**(2): 195–203.

Singh, V., Kapur, P. and Mashaallh, B. (2008). Considering errors of different severity in software reliability growth modeling using fault dependency and various debugging time lag functions, *in* A.K. Verma *et al.* (Eds.), *Proceedings of Advances in Performance and Safety of Complex Systems*, MacMillan India, New Delhi, pp. 839–849.

Sommerville, I. (2011). *Software Engineering*, Addison-Wesley, Boston, MA.

West, D., Grant, T., Gerush, M. and Dsilva, D. (2010). Agile development: Mainstream adoption has changed agility, *Forrester Research* **2**(1): 41.

Wilson, T. (1997). Software failure: Management failure. Amazing stories and cautionary tales, *International Journal of Information Management* **17**(5): 387.

Yamada, S. (1994). Optimal release problems with warranty period based on a software maintenance cost model, *Transactions of the Information Processing Society of Japan* **35**(9): 2197–2202.

Yamada, S., Ohba, M. and Osaki, S. (1983). *S*-shaped reliability growth modeling for software error detection, *IEEE Transactions on Reliability* **32**(5): 475–484.

Yamada, S., Ohba, M. and Osaki, S. (1984). *S*-shaped software reliability growth models and their applications, *IEEE Transactions on Reliability* **33**(4): 289–292.

Shubham Rawat received his BTech degree (with honors) in computer science and engineering from Graphic Era University, India, in 2016. He is a software professional and has strong interests in software engineering, focusing on software process, requirement engineering and software quality.

Nupur Goyal received her BSc degree in computer science in 2009 from Kurukshetra University, Haryana, India. She received her MSc in mathematics in 2011 from H.N.B. Garhwal University, Srinagar, Uttarakhand, India, and the PhD degree from Graphic Era University, Dehradun, Uttarakhand, India, in 2016. Her research interests are in reliability theory. She has also been a faculty member of the Mathematics Department of the Suraj Degree College, Mahendergarh, Haryana.

Mangey Ram received the PhD degree (major in mathematics and minor in computer science) from the G.B. Pant University of Agriculture and Technology, Pantnagar, India, in 2008. He has been a faculty member for around nine years and has taught several core courses in pure and applied mathematics at undergraduate, postgraduate, and doctorate levels. He is currently a professor at Graphic Era University, Dehradun, India. Before joining the Graphic Era University, he was a deputy manager (probationary officer) with Syndicate Bank for a short period. He is the editor-in-chief of the *International Journal of Mathematical, Engineering and Management Sciences*, the executive editor of the *Journal of Graphic Era University*, an associate executive editor of the *Journal of Reliability and Statistical Studies*, and a guest editor or member of the editorial board of many journals. He is a regular reviewer for international journals. He has published 118 research works in many highly reputed national and international journals, and has also presented his contributions at national and international conferences. His fields of research are reliability theory and applied mathematics. Dr. Ram is a senior member of the IEEE, a member of the Operational Research Society of India, the Society for Reliability Engineering, Quality and Operations Management in India, the International Association of Engineers in Hong Kong, and the Emerald Literati Network in the UK. He has been a member of the organizing committee of a number of international and national conferences, seminars, and workshops. He has been granted the Young Scientist Award by the Uttarakhand State Council for Science and Technology, Dehradun, in 2009, the Best Faculty Award in 2011 and recently the Research Excellence Award for his significant contribution to research at Graphic Era University.

Received: 10 July 2016
Revised: 24 December 2016
Re-revised: 9 April 2017
Accepted: 27 April 2017