Software Reliability Model Selection Based on Deep Learning with Application to the Optimal Release Problem

Yoshinobu Tamura[1] and Shigeru Yamada[2]

[1]*Information Science and Engineering Section, Department of Engineering, Graduate School of Sciences and Technology for Innovation, Yamaguchi University, Ube-shi, Japan*
[2]*Department of Social Management Engineering, Graduate School of Engineering, Tottori University, Tottori-shi, Japan*
E-mail: tamura@yamaguchi-u.ac.jp; yamada@sse.tottori-u.ac.jp

Received 1 September 2016; Accepted 10 October 2016
Publication 21 October 2016

Abstract

In the past, many software reliability models have been proposed by several researchers. Also, several model selection criteria such as Akaike's information criterion, mean square errors, predicted relative error and so on, have been used for the selection of optimal software reliability models. These assessment criteria can be useful for the software managers to assess the past trend of fault data. However, it is very important to assess the prediction accuracy of model after the end of fault data observation in the actual software project. In this paper, we propose a method of optimal software reliability model selection based on the deep learning. Moreover, we show several numerical examples of software reliability assessment in the actual software projects. In particular, we discuss the optimal release time and total expected software cost in terms of the model selection based on the deep learning.

Keywords: Software reliability model, optimal model selection, deep learning, optimal release time, software cost.

Journal of Industrial Engineering and Management Science, Vol. 1, 43–58.
doi: 10.13052/jiems2446-1822.2016.003

1 Introduction

In actual software development projects, the comprehensive use of the technologies and methodologies in software engineering is needed for improving software quality/reliability, i.e., requirement specification, design, coding, and, testing. In particular, the waterfall model is well known as the sequential phases in software development process. In the past, many software reliability models [1–3] have been applied to assess the reliability for quality management and testing-progress control of software development. However, it is difficult for the software managers to select the optimal software reliability model for the actual software development project. As an example, the software managers can assess the software reliability for the past data sets by using usual model evaluation criteria. On the other hand, the estimation results based on the past fault data cannot be guaranteed for the future data sets of actual software projects.

The selection method of the optimal software reliability model based on the deep learning is proposed in this paper. Also, several numerical examples of software reliability assessment by using the fault data in the actual software projects are shown. Moreover, we compare the methods to estimate the cumulative numbers of detected faults based on the deep learning with that based on neural network. Furthermore, we discuss the optimal release time and total expected software cost in terms of the model selection.

2 Software Reliability Growth Models

Many software reliability models have been applied to assess the reliability for quality management and testing-progress control of software development. As an example, we show the typical software reliability models with the mean value function representing the expected number of detected faults during $(0,\ t)$ as follows:

Exponential NHPP (nonhomogeneous Poisson process) model

$$E(t) = a(1 - e^{-bt}), \tag{1}$$

Delayed S-shaped NHPP model

$$S(t) = a\{1 - (1 + bt)e^{-bt}\}, \tag{2}$$

Logarithmic Poisson execution time model

$$\mu(t) = \frac{1}{\theta} \ln[\lambda_0 \theta t + 1], \tag{3}$$

Exponential SDE (stochastic differential equation) model

$$SDE_e(t) = a \left\{ 1 - e^{-bt + \frac{\sigma^2}{2} t} \right\}, \tag{4}$$

S-shaped SDE model

$$SDE_s(t) = a \left\{ 1 - (1 + bt)\, e^{-bt + \frac{\sigma^2}{2} t} \right\}, \tag{5}$$

where a is the expected number of initial inherent faults, b a fault-detection rate per unit time per fault, λ_0 the intensity of initial inherent failure, θ the reduction rate of the failure intensity rate per inherent fault, and σ a positive constant representing a magnitude of the irregular fluctuation. In this paper, we discuss above software reliability growth models.

3 Optimal Software Release Problem

Several optimal software release problems considering software development process have been proposed by many researchers [4, 5]. It is interesting for software developers to predict and estimate the time when we should stop testing in order to develop a highly reliable software system efficiently.

We define the following:

c_1: the fixing cost per fault during the testing phase,
c_2: the cost per unit time during the testing phase,
c_3: the maintenance cost per fault after the testing phase.

Then, the expected software cost in the testing phase can be formulated as:

$$C_1(t) = c_1 H(t) + c_2 t, \tag{6}$$

where $H(t)$ means the mean value function of software reliability growth model.

Also, the expected maintenance cost after the release of software is represented as follows:

$$C_2(t) = c_3 \{a - H(t)\}. \tag{7}$$

Consequently, from Equations (6) and (7), the total expected software cost is given by

$$C(t) = C_1(t) + C_2(t). \tag{8}$$

The optimum release time t^* is obtained by minimizing $C(t)$ in Equation (8).

4 Optimal Software Reliability Model Selection Based on Neural Network

Let $w_{ij}^1(i = 1, 2, \ldots, I; j = 1, 2, \ldots, J)$ be the connection weights from i-th unit on the sensory layer to j-th unit on the association layer, and $w_{jk}^2(j = 1, 2, \ldots, J; k = 1, 2, \ldots, K)$ denote the connection weights from j-th unit on the association layer to k-th unit on the response layer. Moreover, $x_i(i = 1, 2, \ldots, I)$ represent the normalized input values of i-th unit on the sensory layer, and $y_k(k = 1, 2, \ldots, K)$ are the output values. We apply the actual number of detected faults per unit time $N_i(i = 1, 2, \ldots, I)$ to the input values $x_i(i = 1, 2, \ldots, I)$.

Considering the amount of characteristics for the software reliability models, we apply the following amount of information as parameters $\lambda_i(i = 1, 2, \ldots, I)$ to the input data $x_i(i = 1, 2, \ldots, I)$.

- The estimated values of Akaike's information criterion
- The estimated values of mean square errors
- The estimate of all parameters included in model
- The estimated error between the estimated and actual values of the total number of detected faults in the testing time point of 25%, 50%, and 75% for data sets

The input-output rules of each unit on each layer are given by

$$h_j = f\left(\sum_{i=1}^{I} w_{ij}^1 x_i\right), \tag{9}$$

$$y_k = f\left(\sum_{j=1}^{J} w_{jk}^2 h_j\right), \tag{10}$$

where a logistic activation function $f(\cdot)$ which is widely-known as a sigmoid function given by the following equation:

$$f(x) = \frac{1}{1 + e^{-\theta x}}, \tag{11}$$

where θ is the gain of sigmoid function. We apply the multi-layered neural networks by back-propagation in order to learn the interaction among input and output [6]. We define the error function given by the following equation:

$$E = \frac{1}{2}\sum_{k=1}^{K}(y_k - d_k)^2, \tag{12}$$

where $d_k(k = 1, 2, \ldots, K)$ are the target input values for the output values. We apply 5 kinds of model type to the target input values $d_k(k = 1, 2, \ldots, 5)$ for the output values, i.e., the exponential NHPP model, the delayed S-shaped NHPP model, the Logarithmic Poisson execution time model, the exponential SDE model, and the S-shaped SDE model, respectively. Then, the number of units I in sensory layer is 37 because of 5 models.

5 Optimal Software Reliability Model Selection Based on Deep Learning

The structure of the deep learning in this paper is shown in Figure 1. In Figure 1, $z_l(l = 1, 2, \ldots, L)$ and $z_m(m = 1, 2, \ldots, M)$ mean the pre-training units. Also, $o_n(n = 1, 2, \ldots, N)$ is the amount of compressed characteristics. Several algorithms in terms of deep learning have been proposed [7–12]. In this paper, we apply the deep neural network to learn the software testing phase.

As with the neural network, we apply the 4 characteristic quantities of pre-training units discussed above. Moreover, we apply 5 kinds of model type to the amount of compressed characteristics, i.e., the exponential NHPP model, the delayed S-shaped NHPP model, the Logarithmic Poisson execution time model, the exponential SDE model, and the S-shaped SDE model, respectively.

6 Numerical Examples

We focus on actual software project data sets [13] in order to assess the performance of our method.

6.1 Estimation Results Based on Trend Analysis for Each Model

We apply 5 models as the typical two kinds of curve types, i.e., exponential and S-shaped reliability growth model curves for the total number of detected

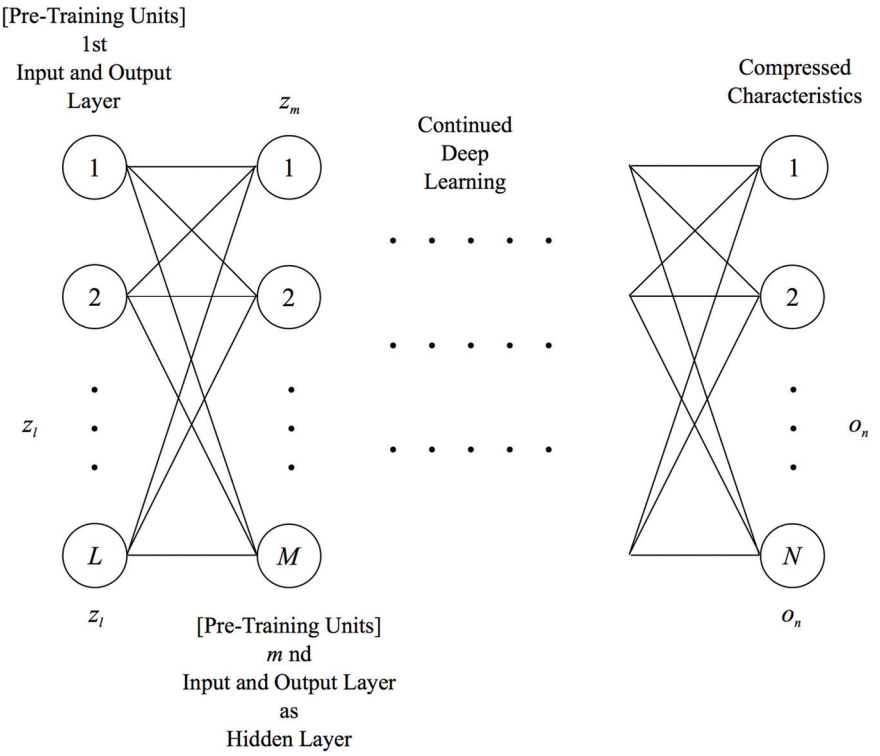

Figure 1 The structure of deep learning.

faults, to the actual fault data sets. The estimation results by using 90% of data set for each software development project are shown in Figures 2–4. Similarly, the estimation results by using 80% of data set for each software development project are shown in Figures 5–7. From Figures 2–7, we can confirm that the model best fitted for the past data are not always fitted for the future in fact. Then, it is necessary for the software managers to offer the model best fitted for the future.

6.2 Estimated Results of the Optimal Model Based on the Deep Learning

The estimated results of the optimal model based on the neural network and deep learning are shown in Tables 1 and 2. From Tables 1 and 2, we found that the estimated recognition rates based on the deep learning perform better than that of the neural network. In particular, the estimation results based on the deep learning give a high-recognition rates in terms of the type

of model. Moreover, the estimation results based on the deep learning for the long-term prediction by using 80% of data sets perform significantly better than that of neural network.

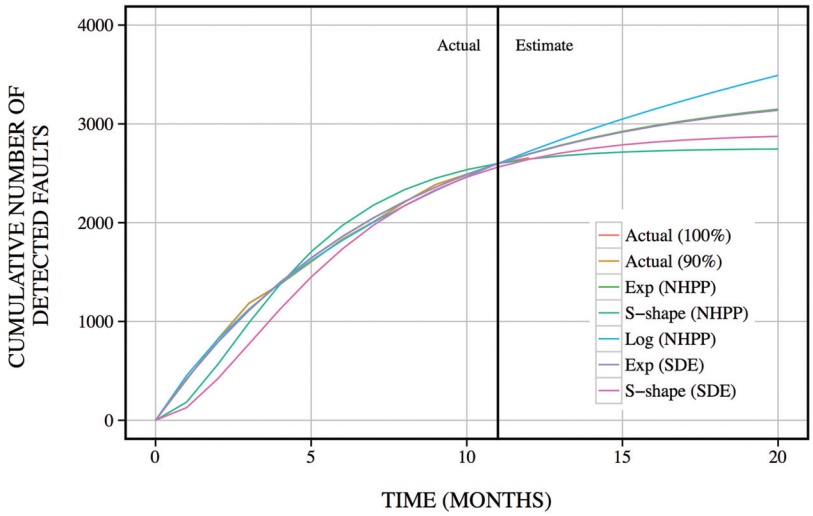

Figure 2 The estimation results by using 90% of data set of software project 1.

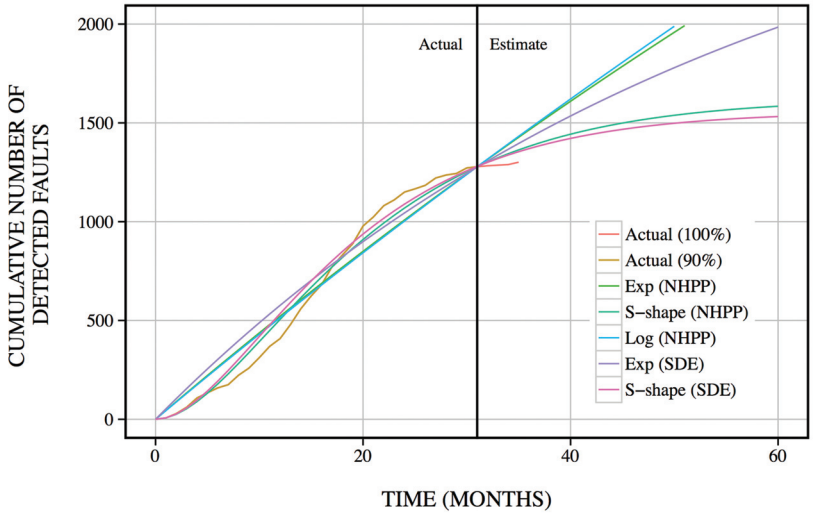

Figure 3 The estimation results by using 90% of data set of software project 2.

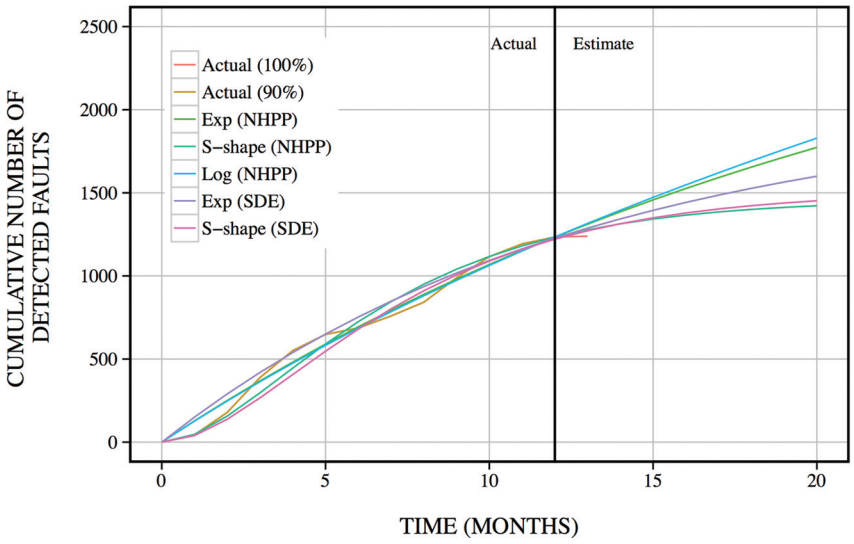

Figure 4 The estimation results by using 90% of data set of software project 3.

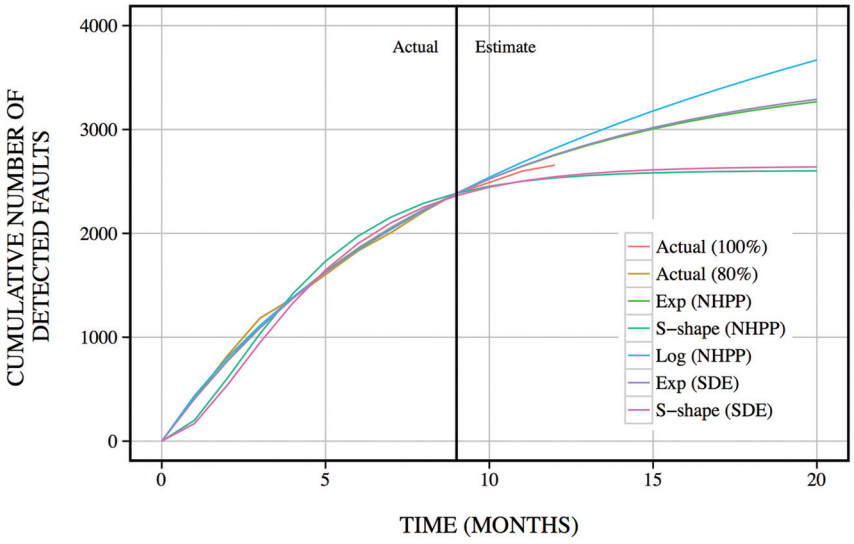

Figure 5 The estimation results by using 80% of data set of software project 1.

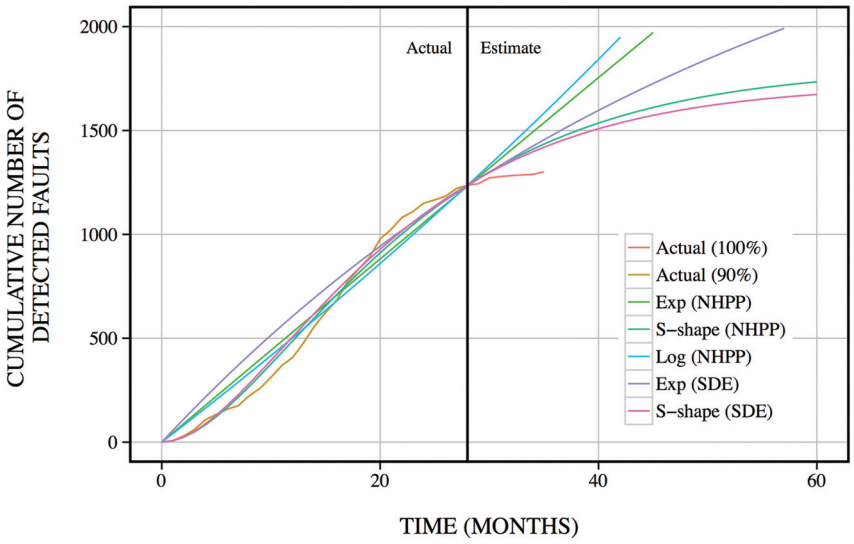

Figure 6 The estimation results by using 80% of data set of software project 2.

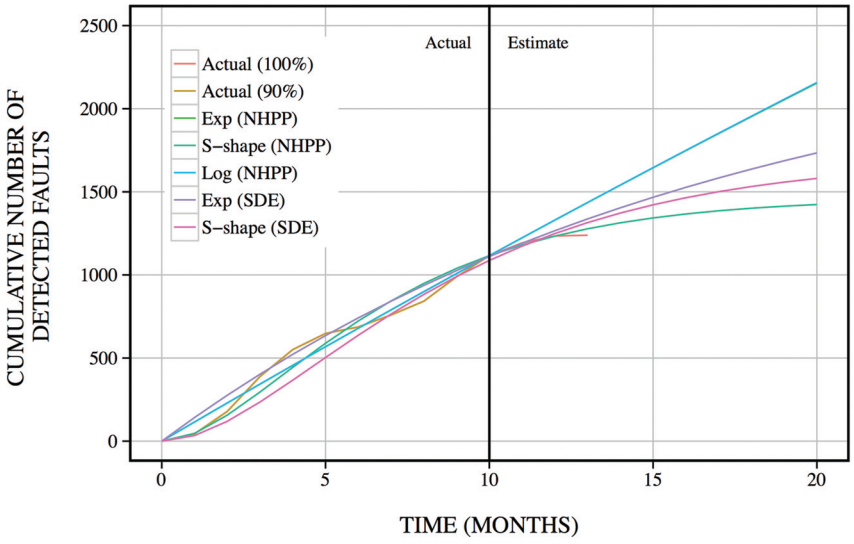

Figure 7 The estimation results by using 80% of data set of software project 3.

Table 1 The comparison of prediction accuracy after 90% of actual fault data

Accurate Model	Model Selected by Neural Network	Model Selected by Deep Learning
S-shape (NHPP)	Exp (SDE)	Exp (NHPP)
S-shape (SDE)	Exp (SDE)	S-shape (SDE)
S-shape (SDE)	Exp (SDE)	S-shape (NHPP)
	Recognition Rate by Neural Network	Recognition Rate by Deep Learning
Type of Model	0%	67%
Name of Model	0%	33%

Table 2 The comparison of prediction accuracy after 80% of actual fault data

Accurate Model	Model Selected by Neural Network	Model Selected by Deep Learning
Exp (NHPP)	Exp (SDE)	Log/Exp (NHPP)
S-shape (NHPP)	Exp (SDE)	Exp (SDE)
S-shape (SDE)	Exp (SDE)	S-shape (SDE)
S-shape (SDE)	S-shape (SDE)	S-shape (SDE)
S-shape (NHPP)	S-shape (SDE)	S-shape (SDE)
S-shape (SDE)	S-shape (SDE)	S-shape (SDE)
	Recognition Rate by Neural Network	Recognition Rate by Deep Learning
Type of Model	67%	83%
Name of Model	33%	50%

6.3 Discussion of the Optimal Release Time

We show the comparison results of the estimated optimal software release time and total expected software cost for each software project data in Tables 3–5, respectively. Then, we focus on 4 models without the logarithmic Poisson execution time model because this model assumes the initial inherent faults of infinity. Considering the prediction accuracy after 90% of actual fault data, the best fitted models are "S-shape (NHPP)" and "S-shape (SDE)" as shown in Table 1. Then, we found that "S-shape (NHPP)" and "S-shape (SDE)" is the shortest period of release time and the minimal cost from Tables 3–5. From these results, it is very important to decide the best model in terms of optimal release time, because the model selection has a great influence on the optimal release time and total software cost.

Table 3 The comparison of the estimated optimal software release time and total expected software cost in case of software project 1

Models	Optimal Release Time	Total Expected Software Cost
Exp (NHPP)	46.070	6827.9
S-shape (NHPP)	22.112	5528.0
Exp (SDE)	45.642	6791.7
S-shape (SDE)	27.040	5832.1

Table 4 The comparison of the estimated optimal software release time and total expected software cost in case of software project 2

Models	Optimal Release Time	Total Expected Software Cost
Exp (NHPP)	678.35	16872.7
S-shape (NHPP)	74.124	3331.3
Exp (SDE)	229.46	6429.4
S-shape (SDE)	68.989	3193.9

Table 5 The comparison of the estimated optimal software release time and total expected software cost in case of software project 3

Models	Optimal Release Time	Total Expected Software Cost
Exp (NHPP)	116.99	6415.2
S-shape (NHPP)	29.124	2951.2
Exp (SDE)	65.768	4155.1
S-shape (SDE)	31.475	3051.7

7 Conclusion

In the past, many software reliability models have been proposed. In fact, these software reliability models have been applied to many software development projects. However, it is very difficult for the software managers to select the optimal software reliability model for the actual software development project. As an example, the software managers can assess the software reliability for the past data sets by using usual model evaluation criteria such as the Akaike's information criterion, mean square errors, predicted relative errors, and so on. On the other hand, the estimation results based on the past fault data cannot guarantee for the prediction of future data sets in actual software projects.

This paper focuses on the learning for the software development projects. We have proposed the selection method of optimal software reliability model based on the deep learning. In particular, it is difficult to assess the reliability by only using the past fault data, because the estimation results based on the

past fault data cannot be guaranteed for the future data sets of actual software projects. In this paper, we have compared the methods of reliability assessment based on neural network with that of deep learning. Moreover, we have shown that the proposed method based on the deep learning can assess better than that based on neural network. Thereby, we have found that our method can assess the software reliability in the future with high accuracy based on the past software project data.

Furthermore, we have discussed the optimal release time and total expected software cost in terms of the model selection based on the deep learning. In particular, we have found that it is very important to decide the best model in terms of optimal release time, because the model selection has a great influence on the optimal release time and total software cost.

Acknowledgments

This work was supported in part by the Telecommunications Advancement Foundation in Japan, the Okawa Foundation for Information and Telecommunications in Japan, and the JSPS KAKENHI Grant No. 15K00102 and No. 16K01242 in Japan.

References

[1] Lyu, M. R., ed. (1996). *Handbook of Software Reliability Engineering*. Los Alamitos, CA: IEEE Computer Society Press.

[2] Yamada, S. (2014). *Software Reliability Modeling: Fundamentals and Applications*. Tokyo/Heidelberg: Springer.

[3] Kapur, P. K., Pham, H., Gupta, A., and Jha, P. C. (2011). *Software Reliability Assessment with or Applications*. London: Springer.

[4] Yamada, S., and Osaki, S. (1985). Cost-reliability optimal software release policies for software systems. *IEEE Trans. Reliab.* R-34, 422–424.

[5] Yamada, S., and Osaki, S. (1987). Optimal software release policies with simultaneous cost and reliability requirements. *Eur J. Operat. Res.* 31, 46–51.

[6] Karnin, E. D. (1990). A simple procedure for pruning back-propagation trained neural networks. *IEEE Trans. Neural Netw*. 1, 239–242.

[7] Kingma, D. P., Rezende, D. J., Mohamed, S., Welling, M. (2014). "Semi-supervised learning with deep generative models," in *Proceedings of Neural Information Processing Systems*, 3581–3589.
[8] Blum, A., Lafferty, J., Rwebangira, M. R., and Reddy, R. (2004). "Semi-supervised learning using randomized mincuts," in *Proceedings of the International Conference on Machine Learning*, 1–13.
[9] George, E. D., Dong, Y., Li, D., and Alex, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans. Audio Speech Lang. Process.* 20: 30–42.
[10] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P. A. (2010). Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* 11, 3371–3408.
[11] Martinez, H. P., Bengio, Y., and Yannakakis, G. N. (2013). Learning deep physiological models of affect. *IEEE Comput. Intell. Mag.* 8, 20–33.
[12] Hutchinson, B., Deng, L., and Yu, D. (2013). Tensor deep stacking networks. *IEEE Trans Pattern Anal. Mach Intell.* 35, 1944–1957.
[13] Brooks, W. D., and Motley R. W. (1980). *Analysis of Discrete Software Reliability Models*. Technical Report RADC-TR-80-84, Rome Air Development Center, Berlin.

Biographies

Yoshinobu Tamura received the B.S.E., M.S., and Ph.D. degrees from Tottori University in 1998, 2000, and 2003, respectively. From 2003 to 2006, he was a Research Assistant at Tottori University of Environmental Studies. From 2006 to 2009, he was a Lecturer and Associate Professor at Faculty of Applied Information Science of Hiroshima Institute of Technology, Hiroshima, Japan. Since 2009, he has been working as an Associate Professor at Yamaguchi University, Ube, Japan. His research interests include reliability assessment

for open source software. He is a regular member of the Institute of Electronics, Information and Communication Engineers of Japan, the Information Processing Society of Japan, the Operations Research Society of Japan, the Society of Project Management of Japan, and the IEEE. Dr. Tamura received the Presentation Award of the Seventh International Conference on Industrial Management in 2004, the IEEE Reliability Society Japan Chapter Awards in 2007, the Research Leadership Award in Area of Reliability from the ICRITO in 2010, and the Best Paper Award of the IEEE International Conference on Industrial Engineering and Engineering Management in 2012.

Shigeru Yamada received the B.S.E., M.S., and Ph.D. degrees from Hiroshima University, Japan, in 1975, 1977, and 1985, respectively. Since 1993, he has been working as a professor at the Department of Social Management Engineering, Graduate School of Engineering, Tottori University, Tottori-shi, Japan. He has published over 500 reviewed technical papers in the area of software reliability engineering, project management, reliability engineering, and quality control. He has authored several books entitled such as Introduction to Software Management Model (Kyoritsu Shuppan, 1993), Software Reliability Models: Fundamentals and Applications (JUSE, Tokyo, 1994), Statistical Quality Control for TQM (Corona Publishing, Tokyo, 1998), Software Reliability: Model, Tool, Management (The Society of Project Management, 2004), Quality-Oriented Software Management (Morikita Shuppan, 2007), Elements of Software Reliability Modeling Approach (Kyoritsu Shuppan, 2011), Project Management (Kyoritsu Shuppan, 2012), Software Engineering: Fundamentals and Applications (Suurikougaku Publishing, 2013), Software Reliability Modeling: Fundamentals and Applications (Springer-Verlag, 2014), and OSS Reliability Measurement and Assessment (Springer-Verlag, 2016). Dr. Yamada received the Best Author Award from the Information Processing Society of Japan in 1992, the

TELECOM System Technology Award from the Telecommunications Advancement Foundation in 1993, the Best Paper Award from the Reliability Engineering Association of Japan in 1999, the International Leadership Award in Reliability Engg. Research from the ICQRIT/SREQOM in 2003, the Best Paper Award at the 2004 International Computer Symposium, the Best Paper Award from the Society of Project Management in 2006, the Leadership Award from the ISSAT in 2007, the Outstanding Paper Award at the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM208) in 2008, the International Leadership and Pioneering Research Award in Software Reliability Engineering from the SREQOM/ICQRIT in 2009, the Exceptional International Leadership and Contribution Award in Software Reliability at the ICRITO'2010, 2011 Best Paper Award from the IEEE Reliability Society Japan Chapter in 2012, the Leadership Award from the ISSAT in 2014, and the Project Management Service Award from the SPM in 2014. He is a regular member of the IEICE, the Information Processing Society of Japan, the Operations Research Society of Japan, the Reliability Engineering Association of Japan, Japan Industrial Management Association, the Japanese Society for Quality Control, the Society of Project Management, and the IEEE.