Fault Identification and Reliability Assessment Tool Based on Deep Learning for Fault Big Data

Yoshinobu Tamura[1] and Shigeru Yamada[2]

[1]*Tokyo City University*
[2]*Tottori University*
E-mail: {*tamuray@tcu; yamada@sse.tottori-u*}*.ac.jp*

Received 28 February 2017; Accepted 22 May 2017;
Publication 2 June 2017

Abstract

Recently, many open source software (OSS) are developed under several OSS projects. Then, the software faults detected in OSS projects are managed by the bug tracking systems. Also, many data sets are recorded on the bug tracking systems by many users and project members. In this paper, we propose the useful method based on the deep learning for the improvement activities of OSS reliability. Moreover, we apply the existing software reliability model to the fault data recorded on the bug tracking system. In particular, we develop an application software for visualization and reliability assessment of fault data recorded on OSS. Furthermore, several numerical illustrations of the developed application software in the actual OSS project are shown in this paper. Then, we discuss the analysis results based on the developed application software by using the fault data sets of actual OSS projects.

Keywords: Software tool, fault identification, reliability assessment, fault big data, deep learning.

1 Introduction

Considering the software reliability, many software reliability models [1–4] have been energetically applied to assess the reliability for quality management

Journal of Software Networking, 161–176.
doi: 10.13052/jsn2445-9739.2017.008

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. In particular, the software managers will need to convert the fault data on bug tracking system from the raw data to the fault count data. It is efficiently compared with the conventional method based on software reliability growth models if the software managers can use all raw data of bug tracking system. In some cases, the poor handling of quality problem prohibits the progress of open source software (OSS), because the development cycle of OSS has no specified testing-phase. In OSS projects, the observed data recorded on the bug tracking systems are used for quality management. Also, the fault big data are recorded on the bug tracking system of open source project. Then, the quality of OSS will be improved significantly if the software managers can make an effective use of these data sets on the bug tracking systems.

In particular, it is important for the software managers to identify the critical failures caused of software faults in case of the OSS. In this paper, we focus on the fault identification of critical software failures caused by software fault. Then, we propose the useful method based on the deep learning for the improvement activities of OSS reliability. Moreover, we apply the existing software reliability model to the fault data recorded on the bug tracking system. Furthermore, we develop the application software in order to analyze the fault data recoded on the bug tracking system. It will be very useful for the OSS users and software managers to understand the various status of OSS by using the developed software tool. In particular, we show several numerical illustrations of the developed software tool.

Finally, we show the organization of this paper as follows: Section 2 introduces the bug tracking system. Also, the methods of reliability assessment based on the deep learning and hazard rate model are discussed in Sections 3 and 4. These methods of software reliability assessment will be useful for the software managers to control the quality of OSS. Moreover, we show the specification requirement of our tool in Section 5. Section 6 shows the performance illustrations of our tool.

2 Fault Data Recorded on Bug Tracking System

Generally, the fault data of many OSS have been managed on the software system for development support called as the bug tracking system. There are the proprietary software such as BugLister, and the OSS such as Bugzilla. The data items recorded as the fault data of OSS is as follows:

An Example of Data Item

Bug ID, Product, Component, Assignee, Status, Resolution, Summary, Changed, Alias, Assignee Real Name, Hardware, Keywords, Number of Comments, Opened, OS, Priority, Reporter, Reporter Real Name, Severity, Summary, Tags, Target Milestone, URL, Version, Votes.

It will be useful for the OSS users and software managers if we can visually analyze above fault data.

There are several tools in order to make a visualization the fault data for specified organizations. However, there is not software tool in order to analyze the fault data recorded on the bug tracking system.

For examples, there are software tools for fault management such as Redmine, Jira, and YouTrack. Jira and YouTrack are the proprietary software. On the other hand, Redmine is the OSS. Moreover, it is well known as the continuous integration server such as TeamCity and Jenkins. However, all of these support tool for software management cannot automatically predict the software fault. We propose the prediction method of software fault. In particular, the identification method of software fault based on deep learning is discussed in this paper. Also, the incorporable software tool for the existing bug tracking system is developed by using the HTML5, CSS3, and JavaScript.

3 Identification of Critical Fault Based on Deep Learning

Several algorithms in terms of deep learning have been proposed [5–10]. In this paper, we apply the deep neural network to learn the fault data on bug tracking systems of open source projects.

Considering the identification method of fault, it is difficult to apply the method such as the neural network, because the input data sets need to be selected by deciding the amount of characteristics for the objective variable in advance. On the other hand, the amounts of characteristics for the object variable are automatically decided in case of the deep learning.

We apply the following amount of information to the parameters of pre-training units. Then, the objective variable is given as the fault level as shown in Table 1. We apply two kinds of fault levels to the amount of compressed characteristics, i.e., Critical and Major, and Others respectively. The following 8 kinds of explanatory variables are set to the amount of pre-training units [11–13].

Table 1 The indexed fault levels recorded on bug tracking system

Index Number	Fault Level
1	Trivial
1	Enhancement
1	Minor
1	Normal
1	Blocker
2	Major
2	Critical

- Date recorded on bug tracking system
- Name of software product
- Name of software component
- Number of software version
- Nickname of fault reporter
- Nickname of fault assignee
- Status of software fault
- Name of operating system

Then, each data of explanatory variable is converted from the character data to the numerical value such as the rate of occurrence.

4 Software Reliability Model

We apply the hazard rate model to assess the quality and reliability for the fault data recorded on the bug tracking system of open source software. The time-interval between successive software failures of $(k-1)$-st and k-th can represent as the random variable $X_k(k = 1, 2, \ldots)$. Then, the hazard rate [14–16] $z_k(x)$ at time x during the operation-phase of software for X_k is given as follows:

$$z_k(x) = \phi\{N - (k - l)\} \quad (k = 1, 2, \ldots, N; \phi > 0, N > 0), \tag{1}$$

where each parameter is defined as follows:

$z_k(x)$: the hazard rate for the software for X_k,
N : the total number of latent faults,
ϕ : the hazard rate per inherent fault.

Equation (1) is the existing model [14–16]. The property of the frequency of software failure-occurrence can assess by using MTBF. Then, MTBF is defined by

$$\mathrm{E}[X_k] = \frac{1}{\phi(N-k+1)}. \tag{2}$$

5 Fault Identification and Reliability Assessment Tool

The specification requirement of the fault identification tool based on deep learning are shown as follows:

1. This tool should be operated by clicking the mouse button and typing on the keyboard to input the data through GUI system.
2. HTML5, CSS3, and JavaScript should be used to implement the program. This tool is developed as a stand-alone application on Windows, Unix, and Mac OS X operating system. Also, this tool operates as Web application. Then, NW.js [17] and NVD3 [18] technologies are used in this tool.
3. This tool treats the proposed method of fault identification for open source fault data, and illustrate the analysis result of software product, one of software component, one of software versions, one of software reporter, one of software assignee, one of software severity, one of the testing data recorded on bug tracking system, and the estimation results of fault levels based on deep learning.
4. The hazard rate model is applied to the fault data recorded on the bug tracking system in order to assess the reliability.

The procedures of fault identification and reliability assessment tool built into the developed software for OSS are shown as follows:

1. This tool processes the data file in terms of the software fault data recorded on the bug tracking system of OSS in order to identify the fault severity.
2. The data set obtained from the bug tracking system is analyzed.
3. This tool estimates the fault severity based on the proposed method of fault identification.
4. This tool illustrates several results of fault analyses.
5. By using the hazard rate model, this tool estimates the MTBF.
6. In particular, this tool focuses on the fault identification based on the deep learning. The testing data recorded on bug tracking system and the estimation results of fault levels based on deep learning are plotted.

This tool is composed of several function components such as fault analysis, estimation based on deep learning, reliability assessment based on hazard rate model, and graphical representation of fault data.

6 Performance Examples for the Developed Software Tool

This section focuses on Apache HTTP server [19] in order to evaluate the performance of the developed application software. The OSS such as Apache HTTP server is closely watched from the point of view of the standardization, the quick delivery, and the cost reduction. In this paper, we show numerical illustrations by using the data sets of Apache HTTP server known as OSS. The data used in this paper are collected in the bug tracking system on the website of Apache HTTP server open source project. We obtain 10,000 fault data sets from the data recorded on bug tracking system of Apache HTTP server. Then, 90% of the recorded data is used as the learning data of the developed application software. Also, our application software is developed as the cross-platform application software by using NW.js and NVD3 technologies. Thereby, the developed application software is available as the Web application by using the same source code. The developed application software is released in the following URL: http://www.comm.tcu.ac.jp/tamura/.

6.1 Fault Data Analysis and Reliability Assessment

We show several results of fault data analyses. The analysis results for software versions and fault severity are shown in Figures 3 and 6. Also, the periods of main software versions are shown as follows:

1. Version 2.0 (06 Apr., 2002)
2. Version 2.2 (01 Dec., 2005)
3. Version 2.4 (21 Feb., 2012)

From Figures 1 and 6, we found that the rate of fault occurrence becomes large around the time of each release date. In particular, many software faults are continuously detected before the release of version 2.2.

Moreover, Figures 7 and 8 show the actual MTBF and estimated MTBF. From Figures 7 and 8, we found that the MTBF grows as operation procedure.

6.2 Estimation Results Based on the Deep Learning

The developed software estimates the critical fault by using the deep learning package of R statistical programming. Figure 9 shows the testing data obtained from 10,000 fault data. The estimation result for 1,000 testing data sets

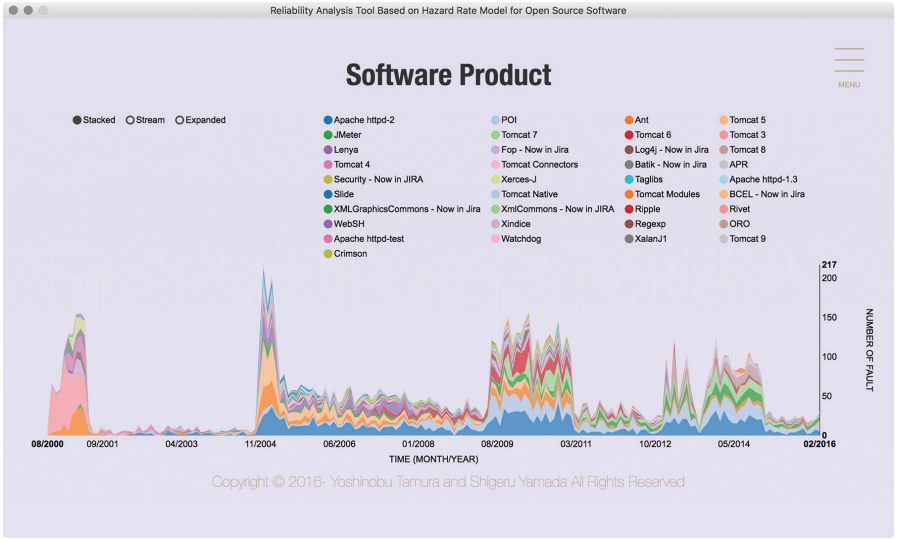


Figure 1 The analysis result for software product.

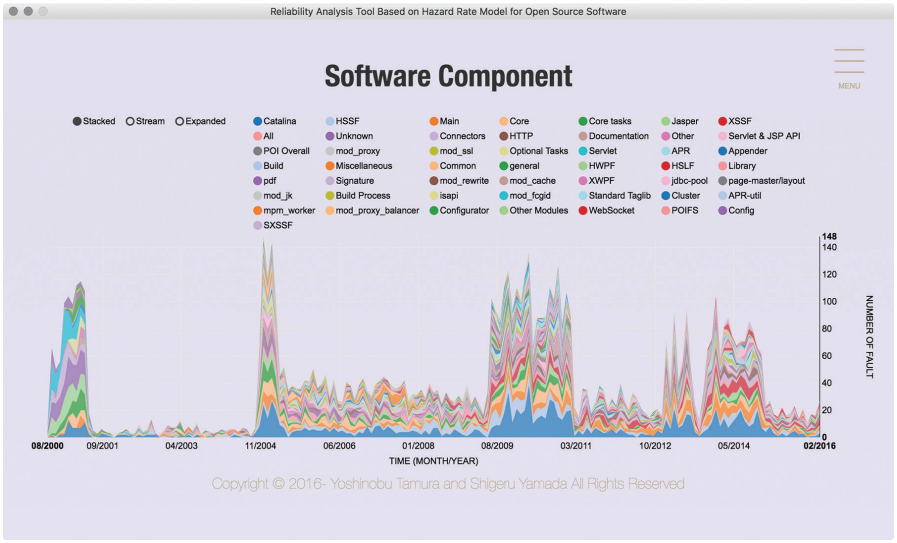


Figure 2 The analysis result for software component.

based on deep learning by using the set of 9,000 learning data is shown in Figure 10. In particular, Figure 10 shows that the proposed method can recognize “Major” and “Critical” in fault levels.

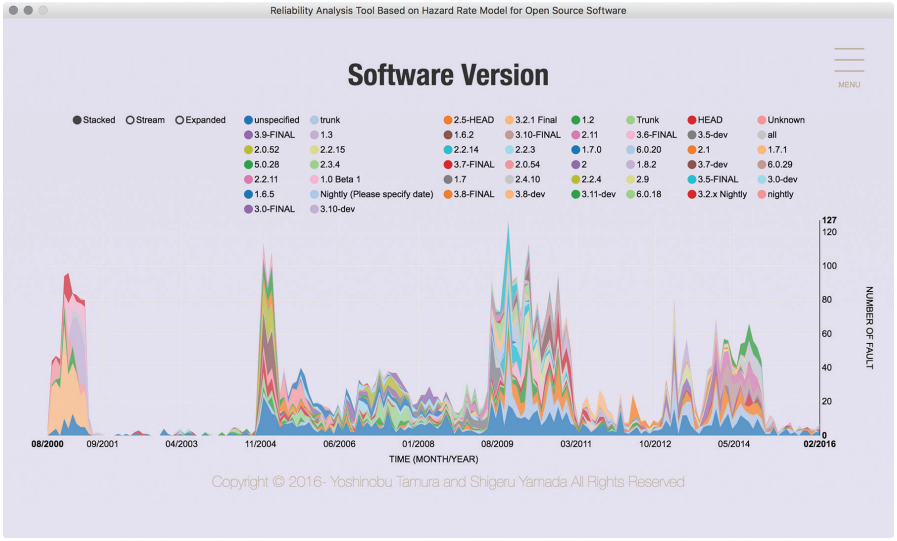


Figure 3 The analysis result for software versions.

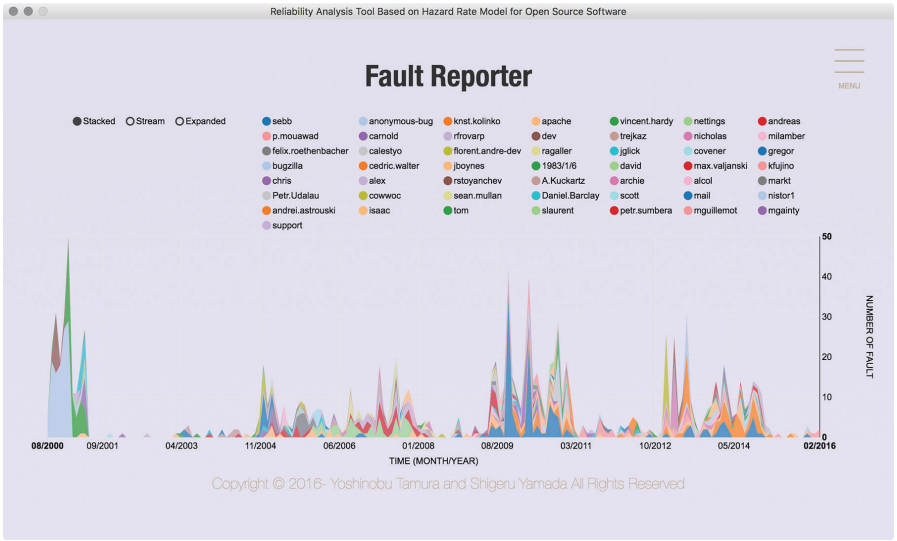


Figure 4 The analysis result for fault reporter.

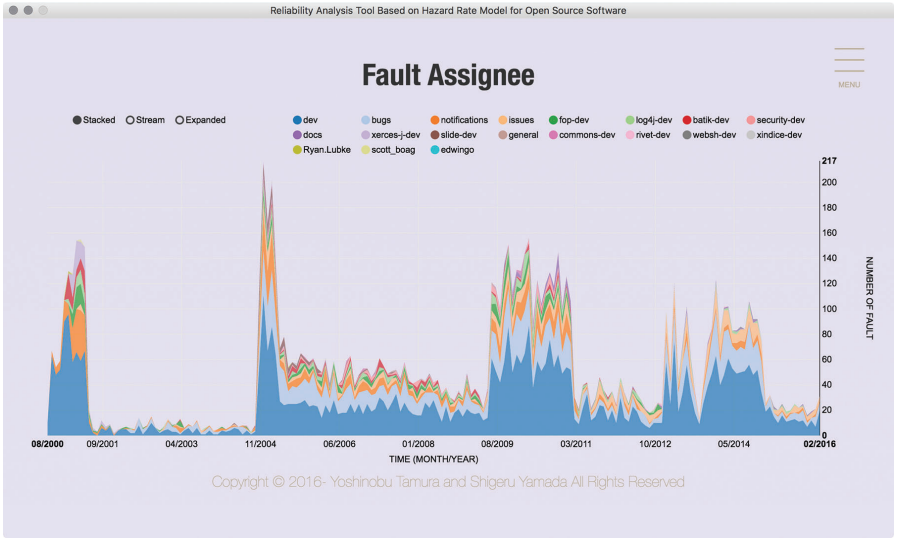


Figure 5 The analysis result for fault assignee.

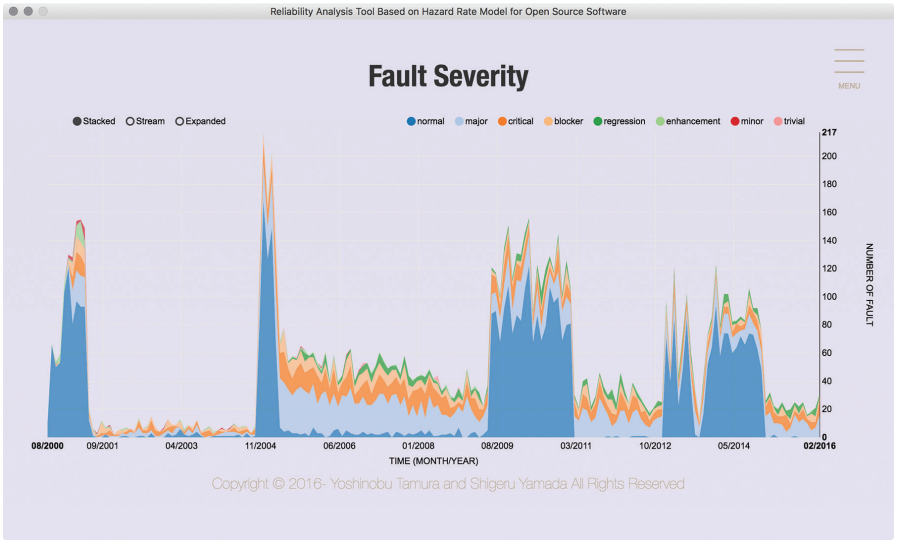


Figure 6 The analysis result for fault severity.

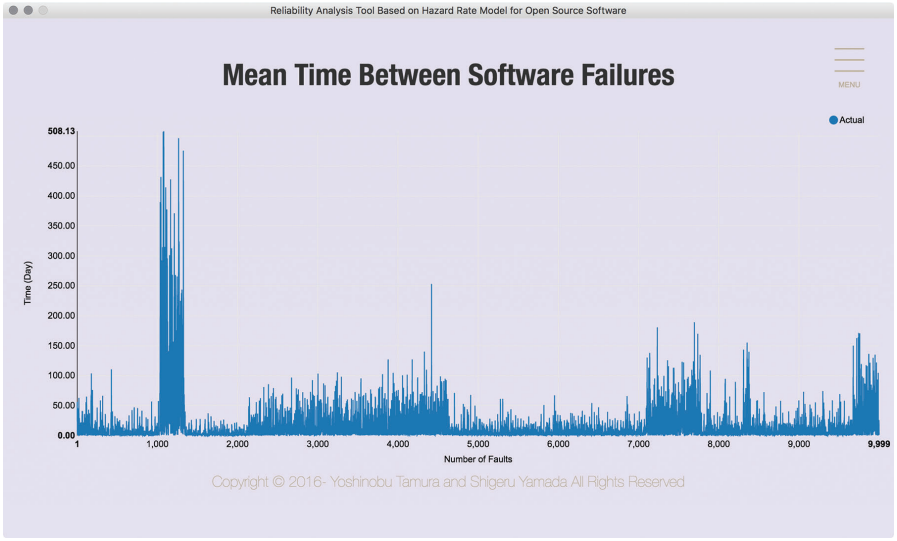


Figure 7 The actual MTBF.

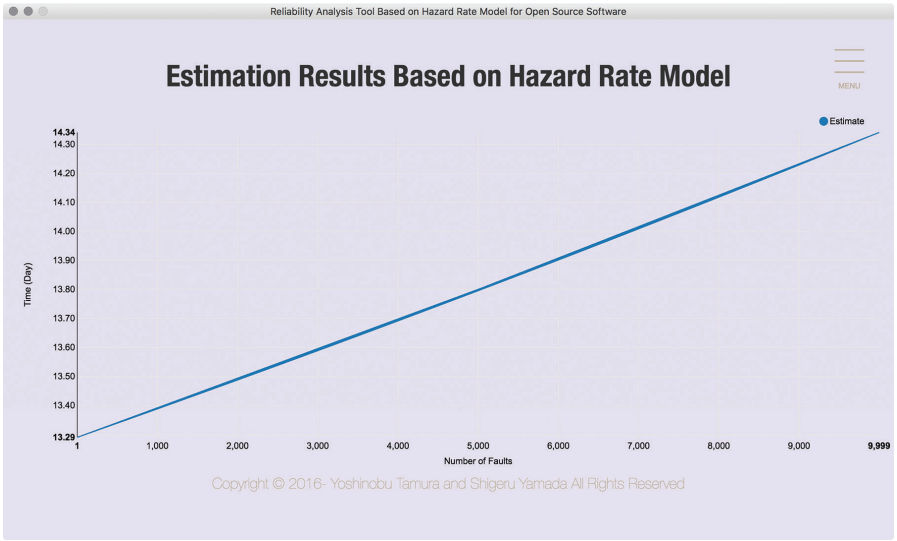


Figure 8 The estimated MTBF.

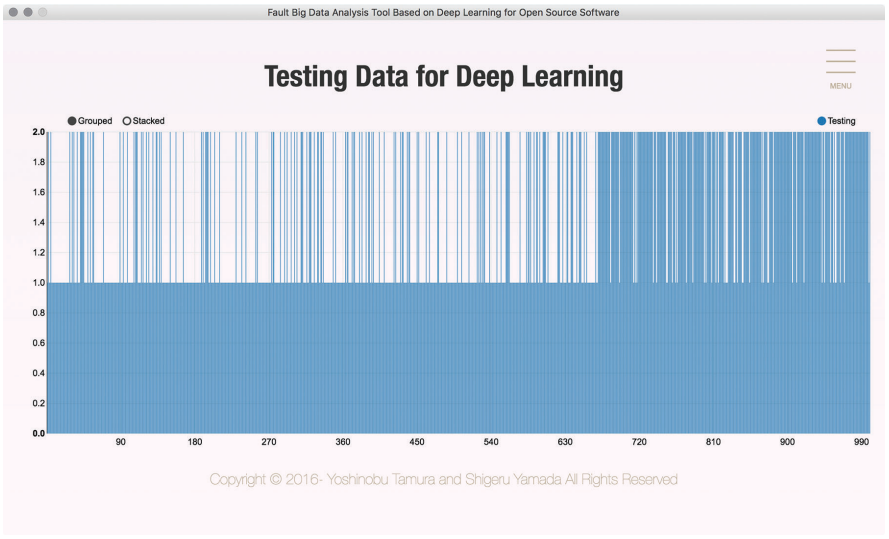


Figure 9 The 10% (1,000 lines) testing data recorded on bug tracking system.

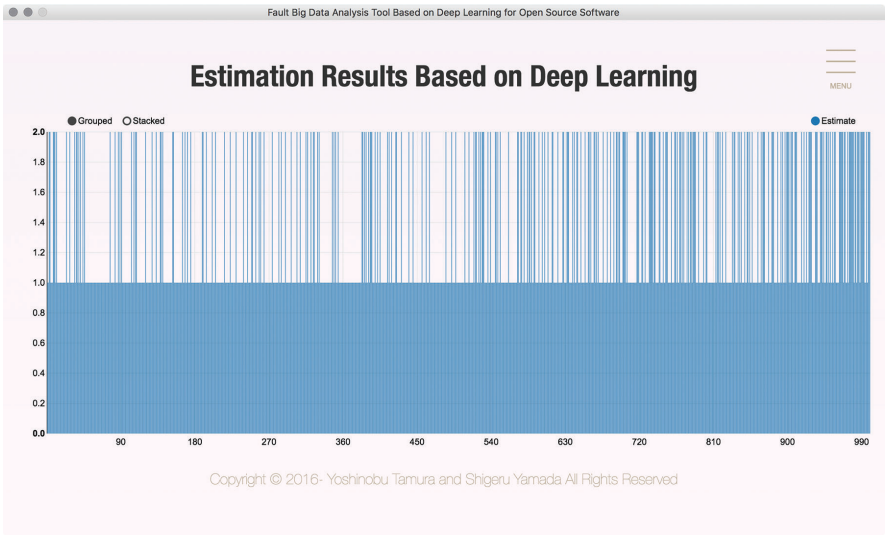


Figure 10 The estimation results of fault levels based on deep learning by using 90% (9,000 lines) learning data.

7 Conclusion

The bug tracking systems are used in many open source projects. In particular, many fault are recorded as the fault big data on these bug tracking system. Then, the quality of OSS will be improved significantly if the software managers can make an effective use of these data sets on the bug tracking systems. In case of using the bug tracking systems, the software managers will be able to take prompt action for the debugging process with the use of the bug tracking system, if the software managers can judge with regard to the critical failures caused of software faults on OSS. This paper has focused on the identification method of critical software failures caused of software faults on OSS. We have proposed the useful method based on the deep learning for the improvement activities of OSS reliability. It is difficult to judge the critical software failures caused by software faults only with the use of the data on bug tracking system, because the contents of data recorded on bug tracking system cannot be guaranteed for the results occurred from the actual OSS, i.e., many general users as well as the major project members can report on the bug tracking system. Moreover, we apply the existing software reliability model to the fault data recorded on the bug tracking system. In particular, we have developed the application software as the cross-platform in order to analyze the fault data recorded on the bug tracking system by using NW.js technology. Also, we have shown several numerical illustrations of the developed application software.

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*. Heidelberg: Springer-Verlag, 2014.

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

[4] Yamada, S., and Tamura, Y. (2016). *OSS Reliability Measurement and Assessment*. London: Springer-Verlag.

[5] Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. (2014). "Semi-supervised learning with deep generative models," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, Montreal, QC, 3581–3589.

[6] 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*, (New York, NY: ACM), 1–13.

[7] 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. Processing* 20, 30–42.

[8] 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. Machine Learn. Res.* 11, 3371–3408.

[9] Martinez, H. P., Bengio, Y., and Yannakakis, G. N. (2013). Learning deep physiological models of affect. *IEEE Comput. Intellig. Magaz.* 8, 20–33.

[10] Hutchinson, B., Deng, L., and Yu, D. (2013). Tensor deep stacking networks. *IEEE Trans. Pattern Anal. Machine Intell.* 35, 1944–1957.

[11] Tamura, Y., Ashida, S., Matsumoto, M., and Yamada, S. (2016). "Identification method of fault level based on deep learning for open source software," *Software Engineering Research, Management and Applications, Studies in Computational Intelligence*, ed. R. Lee (Berlin: Springer), 65–76.

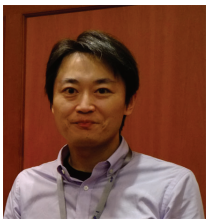
[12] Tamura, Y., Ashida, S., and Yamada, S. (2016). "Fault identification tool based on deep learning for fault big data," in *Proceedings of the 3rd International Conference on Information Science and Security*, Pattaya, 69–72.

[13] Tamura, Y., and Yamada, S. (2016). "Comparison of big data analyses for reliable open source Software," *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management*, Bali.

[14] Jelinski, Z., and Moranda, P. B. (1972). "Software reliability research," in *Statistical Computer Performance Evaluation*, (New York, NY: Academic Press), 465–484.

[15] Xie, M., and Bergman, B. (1988). "On modeling reliability growth for software," in *Proceedings of the 8th IFAC/IFORS Symposium. Identification and System Parameter Estimation*, Beijing. 567–570.
[16] Xie, M. (1989). "On a generalization of the J-M model," in *Proceedings of Reliability '89*, Ba/3/1–5 Ba/3/7, 5.
[17] NW.js community, NW.js. Available at: http://nwjs.io/
[18] Novus Partners, NVD3. Available at: http://nvd3.org/
[19] The Apache Software Foundation, The Apache HTTP Server Project. Available at: http://httpd.apache.org/

Biographies



Y. 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. From 2009 to 2017, he was an Associate Professor at the Graduate School of Sciences and Technology for Innovation, Yamaguchi University, Ube, Japan. Since 2017, he has been working as a Professor at the Department of Industrial & Management Systems Engineering, Faculty of Knowledge Engineering, Tokyo City University, Tokyo, 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.

S. 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* (*IEEM2008*) 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.