International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 4, 273–287, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.4-021

Dependability Analysis Tool Based on Multi-Dimensional Stochastic Noisy Model for Cloud Computing with Big Data

Yoshinobu Tamura
Tokyo City University
Tamazutsumi 1-28-1, Setagaya-ku
Tokyo 158-8557, Japan
Email: tamuray@tcu.ac.jp

Shigeru Yamada*
Tottori University
Minami 4-101, Koyama
Tottori-shi, 680-8552, Japan
E-mail: yamada@tottori-u.ac.jp
**Corresponding author*

(Received November 20, 2016; Accepted March 3, 2017)

Abstract
This paper focuses on a big data on cloud computing environment by using open source software such as Open Stack and Eucalyptus because of the unification management of data and low cost. We propose a new approach to software dependability assessment based on stochastic differential equation modelling and jump diffusion process modelling in order to consider the interesting aspect of the numbers of components, cloud applications, and users. Moreover, we discuss the determination of an optimum software maintenance time minimizing the total expected software cost. In particular, we develop the three-dimensional AIR application for reliability and cost optimization analysis based on the proposed method. Then, we show numerical performance of the developed AIR application to evaluate the method of software reliability assessment for the big data on cloud computing.

Keywords- Cloud computing, Reliability modelling, Jump diffusion process, Cost optimization

1. Introduction

The cloud computing with big data are used as the next-generation software service paradigm. The cloud computing service is connected by many mobile devices. Recently, the mobile clouds based on cloud service become known as the next-generation software service paradigm. Then, the installer software developed by the third-party developers indirectly effect on the reliability of a mobile device by using such mobile clouds. Especially, OSS (Open Source Software) serves as key components of critical infrastructures in our society. The OSS project contains special features so-called software composition by which many distributed components are developed in all parts of the world. However, the poor handling of quality problem and customer support has limited the progress of OSS, because the OSS development cycle has no specific testing phase in order to detect and remove software faults introduced in the development process. Also, the mobile OSS known as one of OSS has been gaining a lot of attention in the embedded system area, i.e., Android (Open Handset Alliance, 2016), Busy Box (Andersen, 2016), Firefox OS (Firefox OS, 2016), etc. Therefore, it is difficult for many companies to assess the reliability in mobile clouds, because a mobile OSS includes the different software versions, the vulnerability issues, the opened source code, the security hole, etc.

Many Software Reliability Growth Models (SRGM's) (Musa, 1987; Lyu, 1996; Kapur et al., 2011; Yamada, 2014) have been applied to assess the reliability for quality management and testing progress control of software development. On the other hand, the effective methods assisting dynamic testing management for a new distributed development paradigm as typified by the cloud computing have only a few presented (Li et al., 2011; Ullah et al., 2012; Cotroneo et al., 2013). Also, there are some interesting research papers in terms of the cloud hardware, cloud service, mobile clouds, and cloud performance evaluation (Iosup et al., 2011; Khalifa and Eltoweissy, 2013). However, most of them have focused on the case studies of cloud service and cloud data storage technologies. The effective methods of dynamic reliability assessment considering the cloud computing with big data by using OSS have been only a few presented. In particular, it is very important to consider the status of fault-detection and big data in terms of the reliability assessment for cloud computing in the following standpoint:

- The big data as the results from the huge and complicated data by using the internet network cause the system-wide failures because of the complexity of data management.
- The cloud computing has the characteristics of particular maintenance phase such as the provisioning processes.
- The cloud computing provides the cloud service to the various mobile devices.
- The various mobile devices reconfigure the data storage areas for cloud computing.

Therefore, it is important to consider the indirect influences of big data on the reliability. We have proposed several methods of software reliability assessment for cloud computing in the past (Tamura and Yamada, 2010, 2012b, 2013b; Tamura et al., 2012a, 2013a). However, the effective methods of reliability assessment considering both the big data factor and fault one have been only a few presented, because it is very difficult to describe the indirect influence of big data and fault data as the reliability assessment measures. Then, we propose a new approach to describe the indirect effect on reliability by using three kinds of Brownian motions.

From above discussed points, we consider that all factors of *big data*, *cloud computing*, and *network access* have an effect on the cloud computing, directly and indirectly. In other words, the big data and cloud computing have a deeply complex issue in terms of the reliability. Therefore, it will be useful to consider the big data from the point of view of the reliability for cloud computing, i.e., it will be able to keep the stable operation of cloud computing if we can offer several reliability assessment measures considering the big data in terms of all factors of *cloud computing*, *mobile clouds*, and *open source software*. Then, we consider 3*V'smodel* defined by Gartner Group, Inc. (Pettey and Goasduff, 2011) for the big data.

Considering the reliability assessment for the cloud computing, mobile clouds, and OSS, it is difficult to consider the external factors from big data, because the external factors arising from the relationship between big data factor and cloud computing have an effect on the infrastructure software for cloud computing. Therefore, it is important for the software managers to consider these external factors from the big data, mobile clouds, and cloud computing.

This paper focuses on the method of reliability analysis for the cloud computing. Then, we propose a method of software reliability analysis considering the big data on cloud computing. Moreover, we formulate the total expected software cost considering the big data on cloud computing. In particular, we develop the three-dimensional AIR application for reliability and cost optimization analysis based on the proposed method. Then, we show numerical performance

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 4, 273–287, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.4-021

of the developed AIR application to evaluate the method of software reliability assessment for the big data on cloud computing.

2. Wiener Process Modeling for Reliability Assessment Considering the Cloud Computing

By using Ito's formula (Wong, 1971; Arnold 1974; Yamada et al., 1994; Mikosch, 1998), we have proposed the following stochastic equation model (Tamura and Yamada, 2015a, 2015b):

$$M(t) = R(t)\left[1 - \exp\left\{-\int_0^t b(s)\, bs - \sigma_1\omega_1(t) - \sigma_2\omega_2(t)\right\}\right] \quad (1)$$

where $b(t)$ is the software fault-detection rate at operation time $t$and a non-negative function, $R(t)$, means the amount of changes of requirements specification.

As the noise-by-noise sample path for each factor, we can represent as the following equations. First, the sample path in terms of fault factor is given as

$$M_1(t) = R(t)\left[1 - \frac{1+c}{1+c\cdot\exp(-bt)}\cdot\exp\{-bt - \sigma_1\omega_1(t)\}\right] \quad (2)$$

Second, the sample path in terms of network factor is given as

$$M_2(t) = R(t)\left[1 - \frac{1+c}{1+c\cdot\exp(-bt)}\cdot\exp\{-bt - \sigma_2\omega_2(t)\}\right] \quad (3)$$

Therefore, the cumulative numbers of detected faults are obtained as follows:

$$M(t) = R(t)\left[1 - \frac{1+c}{1+c\cdot\exp(-bt)}\cdot\exp\{-bt - \sigma_1\omega_1(t) - \sigma_2\omega_2(t)\}\right] \quad (4)$$

In the proposed model, we assume that the parameter σ_1 depends on the parameter b resulting from the failure-occurrence phenomenon. Similarly, we assume that the parameter σ_2 depends on the parameter c resulting from the network environment of cloud computing.

3. Jump-Diffusion Modeling for Reliability Assessment Considering the Big Data

Similarly, as the noise-by-noise sample path for each factor, we can represent as the following equations. First, the sample path in terms of fault factor is given as

$$M_j^1(t) = R(t)\left[1 - \frac{1+c}{1+c\cdot\exp(-bt)}\cdot\exp\left\{-bt - \sigma_1\omega_1(t) - \sum_{i=1}^{Y_t(\gamma)}\log V_i\right\}\right] \quad (5)$$

where $Y_t(\gamma)$ is a Poisson point process with parameter γ at operation time t. Also, $Y_t(\gamma)$ is the number of occurred jumps, γ the jump rate. $Y_t(\gamma)$, $\omega_1(t)$, $\omega_2(t)$, and V_i are assumed to be mutually independent. Moreover, V_i is $i^{\text{-th}}$ jump range.

Second, the sample path in terms of network factor is given as

$$M_j^2(t) = R(t)\left[1 - \frac{1+c}{1+c\cdot\exp(-bt)}\cdot\exp\left\{-bt - \sigma_2\omega_2(t) - \sum_{i=1}^{Y_t(\gamma)}\log V_i\right\}\right] \quad (6)$$

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 4, 273–287, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.4-021

By using Ito's formula (Arnold, 1974; Wong, 1971), the solution of the former equation can be obtained as follows:

$$M_j(t) = R(t)\left[1 - \frac{1+c}{1+c\cdot\exp(-bt)}\cdot\exp\left\{-bt - \sigma_1\omega_1(t) - \sigma_2\omega_2(t) - \sum_{i=1}^{Y_t(\gamma)}\log V_i\right\}\right] \quad (7)$$

The proposed model in Eq. (7) includes the noise with jump term V_i. The cloud managers can assess several characteristics of the cloud with big data by using the size and shape of the noises with jump term, because the proposed model can totally comprehend the provisioning process, the change of users, the change of cloud applications, the indirectly effects as the results from the huge-complicated data in cloud computing with big data as the noise.

4. Optimal Maintenance Problem

Considering the existing optimal software release problems (Yamada and Osaki, 1985, 1987), we define the following cost parameters:

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

Then, the expected software cost in the operation of cloud OSS can be formulated as:

$$C_1(t) = c_1\mathrm{E}[M(t)] + c_2 t \quad (8)$$

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

$$C_2(t) = c_3\{R(t)\text{-}\mathrm{E}[M(t)]\} \quad (9)$$

Consequently, from Eqs. (8) and (9), the total expected software maintenance cost is given by

$$C(t) = C_1(t) + C_2(t) \quad (10)$$

Moreover, as the noise-by-noise sample path for each factor, we can represent as the following equations. First, the sample path in terms of fault factor is given as

$$C_j^1(t) = c_1 M_j^1(t) + c_2 t + c_3\{R(t)\text{-}M_j^1(t)\} \quad (11)$$

Second, the sample path in terms of network factor is given as

$$C_j^2(t) = c_1 M_j^2(t) + c_2 t + c_3\{R(t)\text{-}M_j^2(t)\} \quad (12)$$

5. Multi-Dimensional AIR Application for Reliability Analysis

The specification requirement of the reliability and optimization analysis tool for big data on cloud computing are shown as follows:

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 4, 273–287, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.4-021

a) This tool should be operated by clicking the mouse button and typing on the keyboard to input the data through GUI system. In particular, the user experience design is adopted as the important element of our tool. Moreover, the three-dimensional space is used in the proposed tool in order to represent multi-dimensional stochastic jump diffusion processes model.

b) Open source Apache Flex SDK (Flex.org, 2016) should be used to implement the program. This tool is developed as a stand-alone Adobe AIR application on Windows, UNIX, and Mac OS operating system. Also, this tool operates as Web application. Moreover, the three-dimensional space is implemented by illustrating several three-dimensional graphs.

c) The method of maximum-likelihood and Genetic Algorithm (GA) are used as the estimation of unknown parameters in our model (Tamura and Yamada, 2015a, 2015b).

d) This tool treats the proposed multi-dimensional stochastic processes model considering the cloud computing with big data, and illustrate the cumulative numbers of detected faults at arbitrary time t, the mean time between software failures, the number of remaining faults, and the total expected software cost as software dependability and optimization assessment measures.

It is known the following items as the elements of user experience design.

“The elements of user experience design”.

Visual Design; Information Architecture; Information; Structuring, Organization and Labeling; Finding and Managing; Interaction Design; Usability; Accessibility; Human-Computer Interaction.

We develop the dynamic reliability analysis tool based on “Visual Design” and “Interaction Design” by using the animation effects of Flex. As an example, we use the following source code as the “Visual Design” and “Interaction Design”.

```
<fx:Declarations>

   <view:BasicView id="world"/>

   <mx:Parallel id="blur">
   <mx:Blur duration="400" blurXFrom="40.0" />
   </mx:Parallel>

   <mx:Glow id="glowImage" duration="500" alphaFrom="0.7"
   alphaTo="0.0" blurXFrom="40.0" blurYFrom="40.0" color="0xff0000"/>

</fx:Declarations>
```

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 4, 273–287, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.4-021

```
<mx:Accordion id="accordion" width="100%" height="100%"
   headerStyleName="aHeaderStyles" openEasingFunction="{Bounce.easeOut}"
   openDuration="1000">

   <s:NavigatorContent label="Control Panel" backgroundAlpha="1.0" width="100%"
   height="100%">
         <s:VGroup width="100%" height="100%" verticalAlign="middle"
         horizontalAlign="center" paddingLeft="0"
   paddingRight="0" paddingTop="0" paddingBottom="0">
         <s:SWFLoader width="100%" height="100%" source="edit.swf" />
         </s:VGroup>
   </s:NavigatorContent>

   <s:NavigatorContent label="Cumulative Number of Detected Faults"
   backgroundAlpha="1.0" width="100%" height="100%">
         <s:VGroup width="100%" height="100%" verticalAlign="middle"
         horizontalAlign="center" paddingLeft="0"
         paddingRight="0" paddingTop="0" paddingBottom="0">
         <s:SWFLoader width="100%" height="100%" source="num_3d.swf" />
         </s:VGroup>
   </s:NavigatorContent>

   <s:NavigatorContent label="Mean Time Between Faults" backgroundAlpha="1.0"
   width="100%" height="100%">
         <s:VGroup width="100%" height="100%" verticalAlign="middle"
         horizontalAlign="center" paddingLeft="0"
         paddingRight="0" paddingTop="0" paddingBottom="0">
         <s:SWFLoader width="100%" height="100%" source="mtbf_3d.swf" />
         </s:VGroup>
   </s:NavigatorContent>

   <s:NavigatorContent label="Cumulative Number of Remaining Faults"
backgroundAlpha="1.0" width="100%" height="100%">
         <s:VGroup width="100%" height="100%" verticalAlign="middle"
         horizontalAlign="center" paddingLeft="0"
         paddingRight="0" paddingTop="0" paddingBottom="0">
         <s:SWFLoader width="100%" height="100%" source="remain_3d.swf" />
         </s:VGroup>
         </s:NavigatorContent>

   <s:NavigatorContent label="Total Software Cost" backgroundAlpha="1.0"
   width="100%" height="100%">
         <s:VGroup width="100%" height="100%" verticalAlign="middle"
         horizontalAlign="center" paddingLeft="0"
         paddingRight="0" paddingTop="0" paddingBottom="0">
         <s:SWFLoader width="100%" height="100%" source="cost_3d.swf" />
         </s:VGroup>
   </s:NavigatorContent>
```

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 4, 273–287, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.4-021

</mx:Accordion>

Moreover, the three-dimensional space is implemented in the developed tool by using action script programming language. In particular, the library of paper vision 3D (papervision3d, 2016) is used in the developed tool. In order to illustrate three-dimensional graphs, the developed tool can execute the 3D rendering processing with high speed by using papervision3D library. As an example, we show the source code in terms of these libraries as follows:

```
import org.papervision3d.view.*
import org.papervision3d.objects.*;
import org.papervision3d.materials.*
import org.papervision3d.objects.primitives.*
import org.papervision3d.core.geom.*;
import org.papervision3d.core.geom.renderables.*;
import org.papervision3d.materials.utils.MaterialsList;
import org.papervision3d.core.geom.Lines3D;
import org.papervision3d.core.geom.renderables.Line3D;
import org.papervision3d.core.geom.renderables.Vertex3D;
import org.papervision3d.materials.special.LineMaterial;
import org.papervision3d.materials.special.Letter3DMaterial;
import org.papervision3d.typography.Text3D;
import org.papervision3d.typography.fonts.HelveticaBold;
import org.papervision3d.materials.ColorMaterial;

private function enterFrame(e:Event):void {
      rot += (targetRot - rot) * 0.05;
      pitch += (targetPitch - pitch) * 0.05;
      pitch = Math.max(pitch, -90);
      pitch = Math.min(pitch, 90);
      world.camera.x = (lab_colors.length*1.5) * Math.sin(rot * Math.PI / 180);
      world.camera.z = (lab_colors.length*1.5) * Math.cos(rot * Math.PI / 180);
      world.camera.y = (lab_colors.length*1.5) * Math.sin(pitch * Math.PI / 180);
      }

private function init():void {
      world.startRendering();
      myUI.addChild(world);

      var afile:File=File.applicationDirectory;
      var file:File=File.applicationDirectory;;

      afile = afile.resolvePath("3d_actual.csv");
      afileStream.open(afile, FileMode.READ);
      astr = afileStream.readMultiByte(afileStream.bytesAvailable, "UTF-8");
      astr = (astr.split("\r\n")).join("\n");
      astr = (astr.split("\r")).join("\n");
```

```
        actual = astr.split("\n");
        amakeParticles();

        file = file.resolvePath("3d_data.csv");
        fileStream.open(file, FileMode.READ);
        str = fileStream.readMultiByte(fileStream.bytesAvailable, "UTF-8");
        str = (str.split("\r\n")).join("\n");
        str = (str.split("\r")).join("\n");
        lab_colors = str.split("\n");
        makeParticles();

        addEventListener(Event.ADDED_TO_STAGE, onStage);
        makeWorld();
        }
    private function makeWorld():void {
        var line_mat:WireframeMaterial = new WireframeMaterial(0x444444);
        line_mat.doubleSided = true;
        var line_a:Plane = new Plane(line_mat, 1, 700, 1, 1);
        world.scene.addChild(line_a);
        line_a.y = 0;
        line_a.z = 0;
        line_a.rotationZ = 90;
        var line_b:Plane = new Plane(line_mat, 1, 700, 1, 1);
        world.scene.addChild(line_b);
        line_b.y = 0;
        line_b.x = 0;
        line_b.rotationX = 90;
        var line_c:Plane = new Plane(line_mat, 1, 700, 1, 1);
        world.scene.addChild(line_c);
        line_c.y = 0;
        line_c.y = 0;
        line_c.rotationY = 90;

        world.scene.addChild(alines);
        world.scene.addChild(lines);
        }

    private function downHandler(e:MouseEvent):void {
        isMouseDown = true;
        oldX = mouseX;
        oldY = mouseY;
        }

    private function upHandler(e:MouseEvent):void {
        isMouseDown = false;
        }

    private function moveHandler(e:MouseEvent):void {
```

```
        if(isMouseDown){
                var dx:Number = e.stageX - oldX;
                var dy:Number = e.stageY - oldY;
                targetRot += dx * 0.5;
                targetPitch += dy * 0.5;
                oldX = e.stageX;
                oldY = e.stageY;
        }
}
```

6. Numerical Performance Illustration

We focus on Hadoop (The Apache Software Foundation, 2016) and Open Stack (The Open Stack project, 2016) in order to evaluate the performance of our models. In this paper, we show numerical examples of software reliability assessment by using the data sets for Hadoop of database software and Open Stack of cloud software. The data sets used in this paper are collected in the bug tracking systems on the websites of Hadoop and Open Stack open source projects.

We show the main screen of the developed dynamic reliability analysis tool in Fig. 1. The items of several reliability assessment measures are shown in the left side of screen. Also, the right side of screen shows the data edit function.

The sample path of the estimated number of detected faults for the fault and network factors in Eq. (7) is shown in Fig. 2. From Fig. 2, we can confirm that the jump noise of network factor becomes large in the early operating phase of cloud computing. Also, we found that the noise becomes large from 100 to 200 days. On the other hand, we can confirm that the noise of fault factor becomes small in all operating phase of cloud computing. The developed tool will be useful to assess the reliability of the characteristics of big data on cloud computing.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 4, 273–287, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.4-021

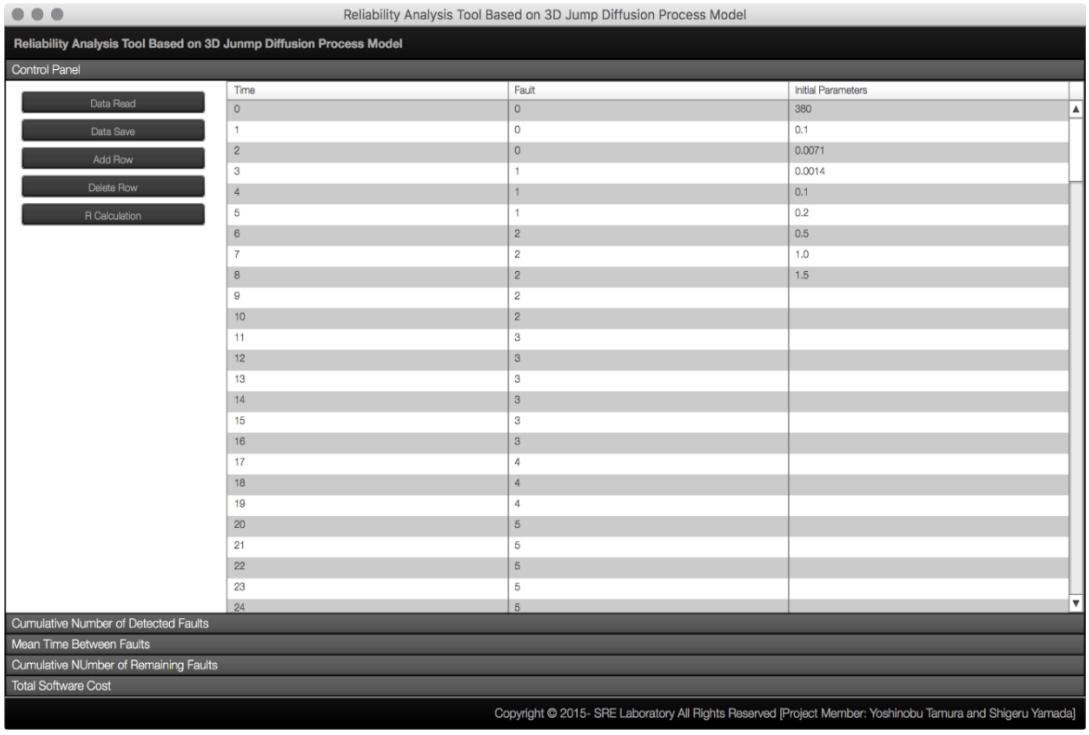

Fig. 1. The main screen of the developed AIR application

Furthermore, we show the sample path of the mean time between software failures and the sample path of the estimated number of remaining faults in Figs. 3 and 4. The software managers will be able to comprehend easily the motion of noises by using the cubic graph in these figures.

We show several numerical examples based on the optimal maintenance problems which are discussed in the section IV. The sample path of the estimated total expected software cost for the fault and network factors in Eqs. (11) and (12) is shown in Fig. 5. From Fig. 5, we can confirm that the noise of network factor becomes large in the early operation phase of cloud computing. On the other hand, we can confirm that the noise of fault factor becomes small in all operation phase of cloud computing. Therefore, we find that the cloud computing environment in this case keeps in stable condition in terms of the reliability. Moreover, we find that the optimum maintenance time is given as t^*=367days considering the noise. Then, the total expected software cost is 345.

The developed cost optimization tool is described by using several noise parameters based on the Wiener and jump diffusion processes. From the mentioned results above, the software managers will be able to visually confirm the stability of cloud computing by using the sample path of the proposed model. In particular, the developed tool can assess the reliability and evaluate cost optimization in terms of multi-factors for big data on cloud computing.

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 4, 273–287, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.4-021

Fig. 2. The sample path of the estimated cumulative number of detected faults

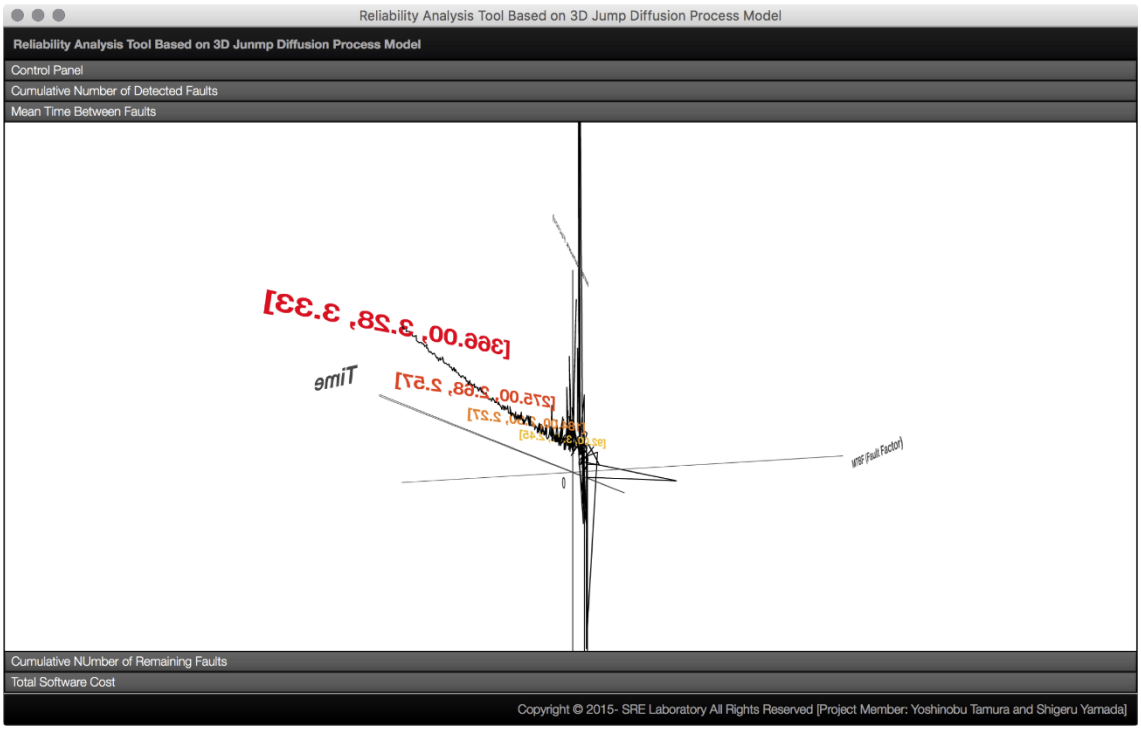

Fig. 3. The sample path of the mean time between software failures

International Journal of Mathematical, Engineering and Management Sciences
Vol. 2, No. 4, 273–287, 2017
https://dx.doi.org/10.33889/IJMEMS.2017.2.4-021

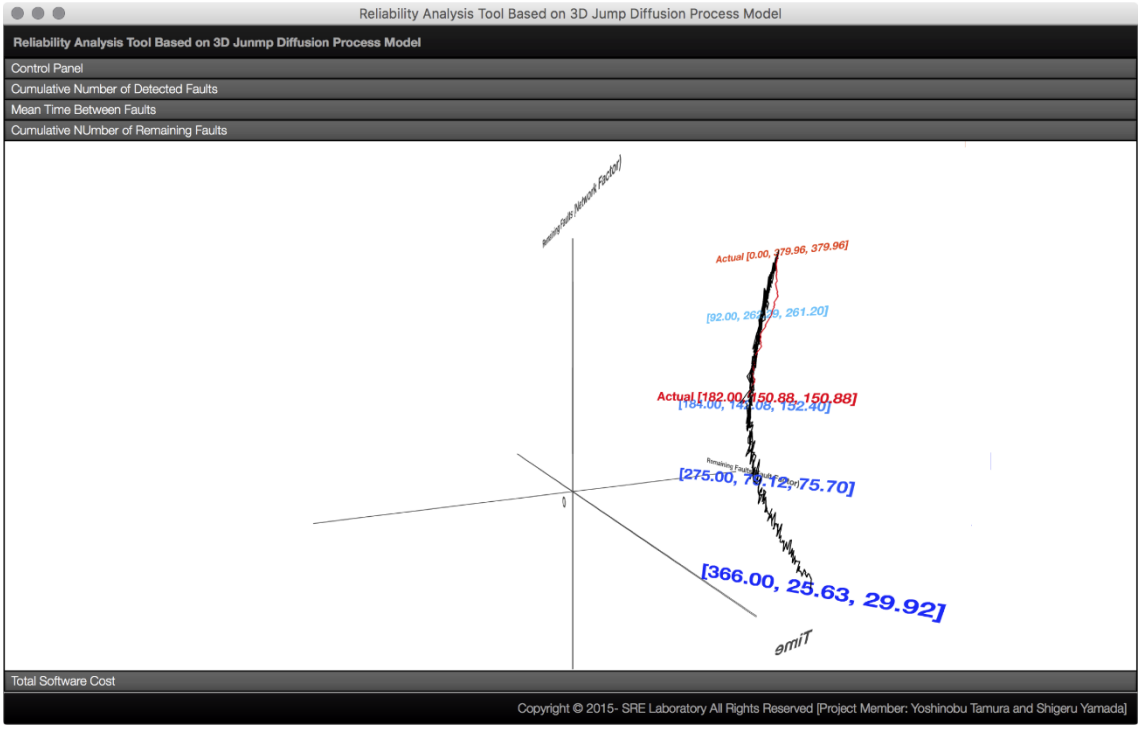

Fig. 4. The sample path of the estimated number of remaining faults

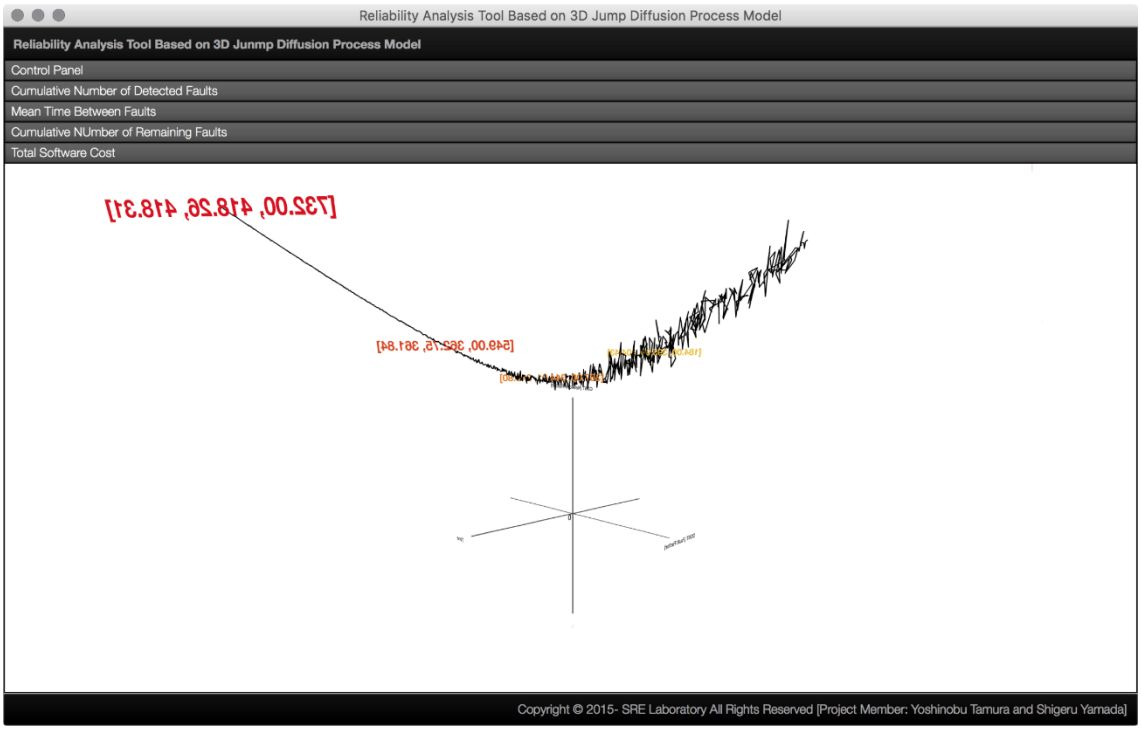

Fig. 5. Cost optimization analysis

7. Concluding Remarks

In this paper, we have focused on the big data on cloud computing. In particular, we have developed the cost optimization analysis tool in order to consider the characteristics of cloud computing under big data. Then, we have proposed the method of reliability assessment based on the jump diffusion process model incorporating the interaction among 3V's model on big data.

Moreover, we have developed the three-dimensional AIR application based on the proposed method. Additionally, we have presented several performance illustrations of the developed AIR application and the proposed method for the actual data. Furthermore, it is important for software managers to assess the reliability and cost optimization for the big data on cloud computing. We have shown the estimated cumulative numbers of detected faults and total expected software cost considering the cloud computing. Thereby, we have found that the developed AIR application can assess integrated reliability considering both software failure and network traffic.

In case of considering the effect of external factors on entire system in the development of software reliability assessment methods for cloud computing, it is necessary to grasp the deeply-intertwined factors. In this paper, we have shown that the proposed method can grasp such deeply-intertwined factors by assuming 3V's model of big data. Also, we have analyzed actual data to show numerical performance of software reliability assessment for the cloud computing. Moreover, we have formulated the total expected software cost considering the characteristics of big data on cloud computing. Furthermore, we have found that the developed tool can evaluate the optimum software maintenance time for the big data on cloud computing.

Acknowledgement

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

Andersen, E. (2016). BUSYBOX, http://www.busybox.net/

Arnold, L. (1974). *Stochastic differential equations*. John Wiley & Sons, New York.

Cotroneo, D., Grottke, M., Natella, R., Pietrantuono, R., & Trivedi, K. S. (2013). Fault triggers in open-source software: An experience report. In *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on* (pp. 178-187). IEEE.

Firefox OS. (2016). Marketplace, Android--Partners--mozilla.org, Mozilla Foundation, http://www.mozilla.org/firefoxos/

Flex. org (2016). Adobe Flex Developer Resource, Adobe Systems Incorporated. [Online]. Available: http://flex.org/

Iosup, A., Ostermann, S., Yigitbasi, M. N., Prodan, R., Fahringer, T., & Epema, D. (2011). Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems*, *22*(6), 931-945.

Kapur, P. K., Pham, H., Gupta, A., & Jha, P. C. (2011). *Software reliability assessment with OR applications*. London: Springer.

Khalifa, A., & Eltoweissy, M., (2013). Collaborative autonomic resource management system for mobile cloud computing. *Proceedings of the Fourth International Conference on Cloud Computing, GRIDs, and Virtualization*, Valencia, Spain, pp. 115-121.

Li, X., Li, Y. F., Xie, M., & Ng, S. H. (2011). Reliability analysis and optimal version-updating for open source software. *Information and Software Technology*, *53*(9), 929-936.

Lyu, M. R. (1996). *Handbook of software reliability engineering* (Vol. 222). CA: IEEE Computer Society Press.

Mikosch, T. (1998). Elementary stochastic calculus with finance in view vol. 6 of Advanced Series on Statistical Science & Applied Probability.

Musa, J. D., Iannino, A., & Okumoto, K. (1987). *Software reliability: measurement, prediction, application*. McGraw-Hill, Inc.

Open Handset Alliance, (2016). Android, http://www.android.com.

Papervision3d, (2016). Open source real time 3D engine for flash, https://code.google.com/p/papervision3d/

Pettey, C., & Goasduff, L. (2011). *Gartner special report: examines how to leverage pattern-based strategy to gain value in Big Data*, 2011 Press Releases. Gartner Inc, 27.

Tamura, Y., & Yamada, S. (2015b). Three dimensional wiener processes model and optimal software maintenance planning. *Proceedings of the Ninth International Conference on Mathematical Methods in Reliability*, Tokyo, Japan, June 1-4, 863-870.

Tamura, Y., & Yamada, S. (2010). *Reliability analysis methods for an embedded open source software*. INTECH Open Access Publisher.

Tamura, Y., & Yamada, S. (2012b, October). Dependability analysis and optimal maintenance problem for open source cloud computing. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on* (pp. 1592-1597). IEEE.

Tamura, Y., & Yamada, S. (2013b, July). Service-oriented maintainability modeling and analysis for a cloud computing. In *Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual* (pp. 53-58). IEEE.

Tamura, Y., & Yamada, S. (2015a). Reliability analysis based on a jump diffusion model with two wiener processes for cloud computing with big data. *Entropy*, *17*(7), 4533-4546.

Tamura, Y., Kawakami, M., & Yamada, S. (2013a). Reliability modeling and analysis for open source cloud computing. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *227*(2), 179-186.

Tamura, Y., Miyahara, H., & Yamada, S. (2012a, December). Reliability analysis based on jump diffusion models for an open source cloud computing. In *Industrial Engineering and Engineering Management (IEEM), 2012 IEEE International Conference on* (pp. 752-756). IEEE.

The Apache Software Foundation, Apache Hadoop, (2016). http://hadoop.apache.org/

The OpenStack project, OpenStack, (2016). http://www.openstack.org/

Ullah, N., Morisio, M., & Vetro, A. (2012, October). A comparative analysis of software reliability growth models using defects data of closed and open source software. In *Software Engineering Workshop (SEW), 2012 35th Annual IEEE* (pp. 187-192). IEEE.

Wong, E. (1971). *Stochastic processes in information and dynamical systems*. McGraw-Hill.

Yamada, S. (2014). *Software reliability modeling: fundamentals and applications* (Vol. 5). Tokyo: Springer.

Yamada, S., & Osaki, S. (1985). Cost-reliability optimal release policies for software systems. *IEEE Transactions on Reliability*, *34*(5), 422-424.

Yamada, S., & Osaki, S. (1987). Optimal software release policies with simultaneous cost and reliability requirements. *European Journal of Operational Research*, *31*(1), 46-51.

Yamada, S., Kimura, M., Tanaka, H., & Osaki, S. (1994). Software reliability measurement and assessment with stochastic differential equations. *IEICE Transactions On Fundamentals of Electronics, Communications and Computer Sciences*, *77*(1), 109-116.