Reliability Assessment Methods and Optimal Bug-fix Release Problems Based on Deterministic Chaos Theory for an Open Source Software

Yoshinobu Tamura\textsuperscript{1} and Shigeru Yamada\textsuperscript{2}  
\textsuperscript{1}Department of Computer Science, Faculty of Applied Information Science, Hiroshima Institute of Technology  
Miyake 2–1–1, Saeki–ku, Hiroshima–shi, 731–5193 Japan  
\textsuperscript{2}Department of Management of Social Systems and Civil Engineering,  
Graduate School of Engineering, Tottori University  
Minami 4–101, Koyama, Tottori–shi, 680–8552 Japan  
E-mail : \textsuperscript{1}tam@cc.it-hiroshima.ac.jp, \textsuperscript{2}yamada@sse.tottori-u.ac.jp

Abstract: OSS(open source software) systems which serve as key components of critical infrastructures in our social life are still ever-expanding now. However, the poor handling of quality and customer support prohibit the progress of OSS. We focus on the management problems in the software quality issues that prohibit the progress of OSS.

In this paper, we propose the method of reliability assessment based on the deterministic chaos theory for OSS. Also, we analyze actual software fault-count data to show numerical examples of software reliability assessment for the OSS with application to optimal bug-fix release problems.

1. Introduction

The open source project contains special features so-called software composition by which several geographically-dispersed components are developed in all parts of the world. The successful experience of adopting the distributed development model in such open source projects includes GNU/Linux operating system, Apache Web server, and so on. However, the poor handling of quality and customer support prohibit the progress of OSS(open source software). We focus on the management problems in the software quality issues that prohibit the progress of OSS.

Many software reliability growth models (SRGM’s) \cite{1} have been applied to assess the reliability for quality management and testing-progress control of software development. On the other hand, the effective method of dynamic testing management for new distributed development paradigms as typified by the open source project has only a few presented \cite{2–5}. Also, the probabilistic methods based on SRGM’s need to consider from a physical standpoint of the software fault-reporting phenomena. Therefore, considering from a point of view of the SRGM, it is difficult to cover the software fault-reporting phenomena on the bug tracking system of open source software.

In this paper, we propose the method of reliability assessment based on the deterministic chaos theory for OSS. Also, we analyze actual software fault-count data to show numerical examples of software reliability assessment for the OSS. Furthermore, we find the optimal bug-fix release time based on the expected software maintenance effort.

2. Reliability Analysis Based on Deterministic Chaos Theory for OSS

2.1 Estimation method

In case that the fault data sets of OSS have the deterministic regularity, we can assess the reliability in the near future. We consider the following data vector obtained from the time-series data in bug tracking system of OSS:

\[ Z(t) = \{x(t), x(t-\tau), \cdots, x(t-(n-1)\tau)\}, \]

where \( \tau \) is the constant parameter, i.e., the arbitrary time-interval, and \( n \) the embedded dimension. In case that the specific chaotic dynamical system is \( m \), \( n \) is represented as follows:

\[ n = 2m + 1. \]

In this paper, we assume that the embedded dimension \( n \) is one-dimension. Then, \( Z(t) \) is given by the following equation:

\[ Z(t) = \{x(t), x(t-\tau), x(t-2\tau)\}. \]

First, we plot the data vector \( Z(t) \) in two-dimensions space. Second, we estimate the regression line based on two-dimensions space. Next, we select several regression lines in the coefficient of correlation of approximately 1. Finally, we estimate the next data from the latest data by estimating the orbit for the next steps. Above mentioned steps are based on the assumptions that the time-series data in next steps show the similar cases, if the specified time-series data has similar behavior. This method is based on the F. Takens theory \cite{6}.

2.2 Reliability assessment measures

We can give the following expressions as software reliability assessment measures derived from our chaos method \cite{7,8}:

\begin{itemize}
  \item Cumulative number of detected faults  
  The cumulative number of faults detected at the operational time \( t \), which is detected as \( C(t) \).
  \item Instantaneous mean time between software failures  
  The instantaneous mean time between software failures (MTBF\textsubscript{I}) is useful to measure the property of the frequency of software failure-occurrences, and is given by
  \[ MTBF_I(t) = \frac{1}{C(t+1) - C(t)}. \]
\end{itemize}
Figure 1. The estimated cumulative number of faults in the time-interval $\tau = 1$.

- Cumulative mean time between software failures
  
The cumulative mean time between software failures ($MTBF_C$) is given as follows:

$$MTBF_C(t) = \frac{t}{C(t)} \quad (5)$$

3. Numerical Examples

3.1 Assessment of chaos nature

We show the estimated Lyapunov exponent for each OSS and the conventional software system developed under the identical organization in Table 1. From Table 1, we consider that the chaos potential of the conventional software system developed under the identical organization is low level. On the other hand the fault data sets of OSS’s have the chaos potential.

3.2 Reliability assessment results

We show numerical examples for reliability assessment of Firefox Web browser. The estimated cumulative number of detected faults in the time-interval $\tau = 1$ is shown in Figure 1. Also, Figure 2 shows the estimated cumulative number of detected faults in the time-interval $\tau = 3$.

Similarly, the estimated cumulative MTBF in the time-interval $\tau = 1$ is shown in Figure 3. Also, Figure 4 shows the estimated cumulative MTBF in the time-interval $\tau = 3$. These figures show that the MTBF decrease as the operational procedures go on.

3.3 Comparison criteria of goodness-of-fit

We show the estimated cumulative number of detected faults for Firefox Web browser by using proposed method based on the proposed chaos theory, NHPP model [7], and SDE model [8] which are developed for OSS, in Figures 5 and 6, respectively.

Moreover, we compare the goodness-of-fit of the proposed method based on chaos theory with the conventional SRGM’s for OSS. We adopt the value of the Mean Square Error (MSE) as a comparison criterion of goodness-of-fit. Table 2 shows the result of goodness-of-fit comparison in terms of the MSE.
Table 1. Comparison of the Lyapunov exponent.

<table>
<thead>
<tr>
<th>Software System Developed under the Identical Organization</th>
<th>OSS</th>
<th>Firefox Web Browser</th>
<th>0.3453</th>
<th>Project 1</th>
<th>-0.1034</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thunderbird Mailer</td>
<td>OSS</td>
<td>Project 2</td>
<td>0.0460</td>
<td>-0.1290</td>
<td></td>
</tr>
<tr>
<td>Calender Scheduler</td>
<td>OSS</td>
<td>Project 3</td>
<td>0.1379</td>
<td>-0.1773</td>
<td></td>
</tr>
<tr>
<td>Fedora Linux OS</td>
<td>OSS</td>
<td>Project 4</td>
<td>0.0660</td>
<td>-0.1878</td>
<td></td>
</tr>
<tr>
<td>Apache HTTP Server</td>
<td>OSS</td>
<td>Project 5</td>
<td>0.1328</td>
<td>-0.2615</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Comparison of the MSE for the cumulative number of detected faults.

<table>
<thead>
<tr>
<th>Compared methods</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method (τ = 3)</td>
<td>302.74</td>
</tr>
<tr>
<td>NHPP model</td>
<td>565.52</td>
</tr>
<tr>
<td>SDE model</td>
<td>437.48</td>
</tr>
</tbody>
</table>

Figure 5. The estimated cumulative number of faults in the time-interval τ = 1.

Figure 6. The estimated cumulative number of faults in the time-interval τ = 3.

4. Optimal Bug-fix Release Problem

Recently, it becomes more difficult for software developers to produce highly-reliable software systems efficiently, because of the more diversified and complicated software requirements. Thus, it has been necessary to control the software development process in terms of reliability, cost, and delivery time. On the other hand, the optimal release problem for OSS has only a few presented.

Several optimal software release problems considering host-concentrated software development process have been proposed by many researchers [9, 10]. However, the effective optimal software bug-fix release problems for OSS have not been proposed. Therefore, we formulate a maintenance effort model based on our method, and analyze the optimal bug-fix release problem minimizing the total expected maintenance effort.

It is interesting for the software developers to predict and estimate the time when we should stop bug-fixing in order to develop a highly reliable software system efficiently. Hence, we discuss about the determination of OSS bug-fix release times minimizing the total expected software effort.

4.1 Formulation of bug-fix release problem

It is very important in terms of software management that we decide for the optimal length of bug-fix release period for OSS. We find the optimal bug-fix release time based on the expected software maintenance effort in this section. We formulate a maintenance effort model based on our proposed method using the chaos theory.

We define the following:

- \( m_0 \): the fixing effort per fault.
- \( m_1 \): the maintenance effort per fault.
- \( m_2 \): the effort per time for fixing faults.

Then, the expected software effort of OSS during the bug-fix period can be formulated as:

\[
E_1(t) = m_0 C(t).
\]

where \( C(t) \) is the estimated cumulative number of detected faults by using our proposed method.

Also, the expected software maintenance effort after the bug-fix release is represented as follows:

\[
E_2(t) = m_1 |C(t_0) - C(t)| + m_2 t,
\]

where, \( t_0 \) is the mean time of the past bug-fix release period.

Moreover, if the software components are added to entire system after the bug-fix release, the penalty effort is imposed.
We define the penalty effort function as follows:
\[ G(t) = (1 - c) \exp \left[ \frac{t - t_0}{\nu} \right], \tag{8} \]
where \( c \) is the ratio of the number of new components to entire system after the bug-fix release, and \( \nu \) the mean number of the previous bug-fix release.

Consequently, from Eqs. (6), (7), and (8), the total expected software effort is given by
\[ E(t) = E_1(t) + E_2(t) + G(t). \tag{9} \]

The optimum bug fix release time \( t^* \) is obtained by minimizing \( E(t) \) in Eq. (9).

4.2 Numerical illustrations for optimal bug-fix release time

In this section, we show several numerical examples based on the optimal bug-fix release problems. Figure 7 is shown the estimated total expected software effort where \( t_0 = 120, \nu = 6, c = 0.1, m_0 = 2, m_1 = 5, \) and \( m_2 = 3 \) as an example. From Figure 7, we find that the optimum bug-fix release time is derived as \( t^* = 125 \) days after the security hole was detected at \( t^* = 120 \). Then, the total expected software effort \( E(t^*) \) is 2180.1 man-days effort.

Figure 7. The estimated total expected software effort

5. Conclusion

In this paper, we have discussed the method of reliability assessment based on the deterministic chaos theory for the OSS. By using the deterministic chaos theory, we have proposed the method of reliability assessment incorporating the complicated situation of registration for bug tracking system, the degree of maturation of OSS, and so on. As the results of comparison, we have concluded that the proposed method based on chaos theory fits better than the conventional NHPP model and SDE model for OSS in terms of the actual data set.

Furthermore, it has been necessary to control the software development process in terms of reliability, development effort, and bug-fix release time for OSS. We have formulated the maintenance effort model based on our method and analyzed the optimal bug-fix release problem minimizing the total expected maintenance effort. Also, we have derived the optimum bug-fix release time.

We consider that the proposed method can assist improvement of quality for OSS systems developed under the open source project.

Acknowledgments

This work was supported in part by the Grant-in-Aid for Scientific Research (C), Grant No. 18510124 from the Ministry of Education, Culture, Sports, Science, and Technology of Japan.

References