Software Reliability Estimation using Inflected S-shaped Model Involving Fault Dependency, Debugging Time Lag and Imperfect Debugging

1Dr. Ajay Gupta and 2Dr. Suneet Saxena

1Asstt. Director, Bhagwant Institute of Technology, Muzaffarnagar (U.P.), India
E-mail: ajaydr_gupta@yahoo.co.in

2Asstt. Professor, Department of Mathematics, J.P. Institute of Engineering Technology, Meerut, (U.P.), India
E-mail: sunsax75@yahoo.co.in

Abstract
The software responsible for running applications such as internet banking, railway reservation, air-traffic control system, etc are highly complex. The need and the importance for software quality are growing as the functionality of software becomes more complex and critical. The most important attribute in software quality is software reliability, which has attracted an increasing amount of attention in software engineering community. The most important task for software developer is to develop highly reliable software. Over past thirty years, many mathematical models have been proposed for estimation of reliability growth of product during software development process. Such models often referred as software reliability growth models(SRGM).

We have considered Inflected S-shaped SRGM and incorporated the fault dependency, debugging time lag and imperfect debugging. Results shows that reliability of software gets improved under imperfect debugging.

Keywords: Software Reliability, Imperfect debugging, Debugging time lag and fault dependency.

Introduction
Software reliability is the probability of failure free software operation for a specified period of time in specified environment. The reliability of software depends on fault detection and correction process. Removing all detected faults will presumably increase the reliability of the software. Ohba [8] conceived that there were two types of faults namely mutually independent and mutually dependent faults. Mutually independent faults can be detected and corrected immediately. There is no time delay between detection and correction. Mutually dependent faults cannot be removed immediately. Goel and Yang [2] analyzed the problem whether detected faults can be corrected immediately or not. Yang [13] reported that detected faults take months to remove for large software system.

Hung and Lin [5] incorporated fault dependencies and debugging time lag into existing SRGM. They analyzed problem of optimal release time for software system based on reliability and cost criterion. They also assumed detected faults were removed with certainty (perfect debugging). If some of the detected faults are not removed with certainty or new faults introduced during debugging process then it is called imperfect debugging. Yamada, Tokunou and Osaki [11] studied imperfect debugging model models with fault introduction rate. Xie and Yang [13] analyzed imperfect debugging on software development cost.

In this paper we have analyzed the software reliability using Inflected S-shaped Model model and generalized it by involving imperfect debugging (b=0.1) and time delay function

\[ \phi(t) = \frac{1}{r(1-b)} \ln \left[ \frac{\psi - 1}{1 + \exp \left\{ -r(1-b)t \right\} } \right] \]

Notations
- \( f_0 \) Expected number of initial faults.
- \( f_i \) Total number of independent faults.
- \( f_d \) Total number of dependent faults.
- \( r \) Fault detection rate of independent faults.
- \( \theta \) Fault detection rate of dependent faults.
- \( p \) Proportion of independent faults.
- \( \psi \) Inflection factor of inflected S shaped model.
- \( \phi(t) \) Delay effect factor.
- \( m(t) \) Mean value function (MVF) of the expected number of faults detected in time (0, t).
- \( m_d(t) \) MVF of the expected number of dependent faults detected in time (0, t).
- \( m_i(t) \) MVF of the expected number of independent faults detected in time (0, t).
- \( b \) Independent fault introduction rate while removing/fixing a detected fault.

Assumption
a) All detected faults are either independent or dependent. b) The total number of faults is finite. c) The detected dependent fault may not be removed immediately and it lags the fault detection process by \( \Phi(t) \). d) Introduction of new independent faults during debugging, process.
Software Reliability Growth Model
The total detected faults in time \((0, t)\) are given by
\[
m(t) = m_i(t) + m_d(t)\]  \( (1) \)

Independent Faults \(m_i(t)\)
The rate of independent faults detected is proportional to the remaining faults. We have following differential equation
\[
\frac{d}{dt} m_i(t) = r \left[ f_i - m_i(t) \right], \quad f_i > 0, \quad 0 < r < 1
\]

Under imperfect debugging, the differential equation becomes
\[
\frac{d}{dt} m_i(t) = r \left[ f_i + b m_i(t) - m_i(t) \right], \quad f_i > 0, \quad 0 < r < 1
\]

Solving equation using initial condition \(m_i(0) = 0\) and involving time delay function \(\phi(t)\), we propose
\[
m_i(t) = \frac{f_i}{(1-b)} \left[ 1 - \exp \left\{ -r(1-b)(t-\phi(t)) \right\} \right] \quad (2)
\]

Dependent Faults \(m_d(t)\)
The rate of dependent faults detected is proportional to the remaining dependent faults in the system and to the ratio of independent faults removed at time \(t\) to the total number of faults. Thus, we have
\[
\frac{d}{dt} m_d(t) = \theta \left[ f_d - m_d(t) \right] \frac{m_i(t)}{f_o}, \quad 0 < 0 < 1
\]

Putting the \(m_i(t)\) we get,
\[
\frac{d}{dt} m_d(t) = \frac{f_i \theta}{f_o (1-b)} \left[ f_d - m_d(t) \right] \left[ 1 - \exp \left\{ -r(1-b)t \right\} \exp \left\{ r(1-b) \phi(t) \right\} \right] \quad (3)
\]

Inflected S-shaped Model
This model was proposed by Ohba. Its underlying concept is that the observed software reliability growth becomes S-shaped if faults are mutually dependent.

Assuming
\[
\phi(t) = \frac{1}{r(1-b)} \ln \left[ \frac{\psi - 1}{1 + \psi \exp \left\{ -r(1-b)t \right\}} \right]
\]

simplifying equations (2) and (3), and using \(f_i = pf_o\), \(f_d = (1-p) f_o\)
we get
\[
m_i(t) = \frac{f_i}{(1-b)} \left[ 1 - \exp \left\{ -r(1-b)t \right\} \right] \quad \frac{1 + \psi \exp \left\{ -r(1-b)t \right\}}{r} \psi \exp \left\{ -r(1-b)t \right\}
\]

software reliability \( R(t) \)
\[
R(t) = \exp \left\{ -m(t) \right\}
\]

Reliability Analysis
Removing all detected faults will presumably increase the reliability of the software. The software reliability defined as the probability that a software failure does not occur in the time interval \((t, t+\Delta t)\) is
\[
R(\Delta t / t) = \exp \left[ - \left\{ m(t + \Delta t) - m(t) \right\} \right], \quad t \geq 0, \quad \Delta t \geq 0
\]

Assuming \(f_o = 400, \quad r = 0.225, \quad \theta = 0.0833, \quad p = 0.55, \quad \psi = 2.84 \)
(These numerical constants taken from reference paper [5]) Number of failures \(m(t)\) and software reliability \(R(10/t)\) have been evaluated under perfect debugging \((b=0)\) and imperfect debugging \((b=0.1)\). Further, graphs have also been plotted for \(m(t)\) and \(R(10/t)\) with respect to testing time \(t\).

Conclusion
Graph 1 reveals the variation of number of faults detected with respect to testing time. During initial phase of testing time the faults detected are very high and later on becomes constant. The number of faults debugged under imperfect debugging is higher than that in under perfect debugging. This is due to generation of new faults while debugging of detected faults.

Graph 2 shows the variation of software reliability with respect to testing time. Software reliability increases rapidly with testing time during initial phase . Under imperfect debugging \((b=0.1)\) after 140 units of testing time the probability of failure free execution of software in 10 units time interval is 91 % whereas under perfect debugging \((b=0)\) the probability is 85%. This shows that if we incorporate the factors fault dependency, debugging time lag and imperfect debugging into model, prediction of software reliability is more realistic and generalized. Also, we can predict when to stop testing based on reliability of software.

Table 1

<table>
<thead>
<tr>
<th>Time(t)</th>
<th>(b = 0)</th>
<th>(b = 0.1)</th>
<th>(b = 0)</th>
<th>(b = 0.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m(t))</td>
<td>(R(10/t))</td>
<td>(m(t))</td>
<td>(R(10/t))</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>178.0561658</td>
<td>180.9320033</td>
<td>2.92E-48</td>
<td>8.00086E-56</td>
</tr>
<tr>
<td>20</td>
<td>287.5088142</td>
<td>307.7972195</td>
<td>1.551E-20</td>
<td>1.07448E-23</td>
</tr>
<tr>
<td>30</td>
<td>333.1213605</td>
<td>360.6848379</td>
<td>1.2997E-11</td>
<td>3.53163E-12</td>
</tr>
</tbody>
</table>
Software Reliability Estimation using Inflected S-shaped Model Involving Fault Dependency, Debugging Time Lag and Imperfect Debugging

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>358.1876498</td>
<td>387.0540983</td>
<td>2.00858E-07</td>
</tr>
<tr>
<td>50</td>
<td>373.6083197</td>
<td>402.0983261</td>
<td>6.09271E-05</td>
</tr>
<tr>
<td>60</td>
<td>383.3141526</td>
<td>411.030208</td>
<td>0.002169106</td>
</tr>
<tr>
<td>70</td>
<td>393.3261662</td>
<td>416.3839362</td>
<td>0.020680304</td>
</tr>
<tr>
<td>80</td>
<td>399.4475927</td>
<td>419.5998759</td>
<td>0.086038795</td>
</tr>
<tr>
<td>90</td>
<td>405.7791232</td>
<td>421.5325939</td>
<td>0.211956758</td>
</tr>
<tr>
<td>100</td>
<td>409.3116655</td>
<td>424.0643887</td>
<td>0.374872523</td>
</tr>
<tr>
<td>110</td>
<td>411.9322084</td>
<td>424.216007</td>
<td>0.537652426</td>
</tr>
<tr>
<td>120</td>
<td>413.7298715</td>
<td>424.3071392</td>
<td>0.675390651</td>
</tr>
<tr>
<td>130</td>
<td>415.3246724</td>
<td>424.3619153</td>
<td>0.780192274</td>
</tr>
<tr>
<td>140</td>
<td>416.5728873</td>
<td>424.216007</td>
<td>0.854717606</td>
</tr>
<tr>
<td>150</td>
<td>417.7928715</td>
<td>424.3071392</td>
<td>0.905484578</td>
</tr>
<tr>
<td>160</td>
<td>418.8291565</td>
<td>424.3619153</td>
<td>0.939137782</td>
</tr>
<tr>
<td>170</td>
<td>419.8919496</td>
<td>424.398392</td>
<td>0.961064605</td>
</tr>
<tr>
<td>180</td>
<td>419.9316632</td>
<td>424.4146286</td>
<td>0.975195809</td>
</tr>
<tr>
<td>190</td>
<td>419.9567802</td>
<td>424.4265232</td>
<td>0.984240195</td>
</tr>
<tr>
<td>200</td>
<td>419.9726655</td>
<td>424.4336727</td>
<td></td>
</tr>
</tbody>
</table>

Graph 1

Graph 2

References


