Software Reliability Estimation using Yamada Delayed S Shaped Model under Imperfect Debugging and Time Lag

Dr. Ajay Gupta
Asstt.Professor
Department of Mathematics
Bhagwant Institute of Technology,
Muzaffarnagar (U.P.), India.

Dr. Digvijay Choudhary
Asstt.Professor
Department of Mathematics
J.P. Institute of Engineering Technology,
Meerut(U.P), India.

Dr. Suneet Saxena
Asstt.Professor
Department of Mathematics
J.P. Institute of Engineering Technology.
Meerut(U.P), India.

ABSTRACT
Reliability of software has been analyzed using some existing mathematical models often termed as software reliability growth models(SRGM). We have considered *Yamada Delayed S shaped* model and incorporated the fault dependency, debugging time lag and imperfect debugging . Results shows that reliability of software gets improved under imperfect debugging

General Terms
Reliability, SRGM.

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

1. INTRODUCTION
Software reliability is the probability of failure free software operation for a specified period of time in specified environment. Over past thirty years, many mathematical models have been proposed for estimation of reliability growth of product during software development process [4,6]. Such models often referred as software reliability growth models(SRGM).

The reliability of software depends on fault detection and correction process. Removing all detected faults will presumably increase the reliability of the software. Ohba [10] 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 exiting 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 [12] 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 Yamada Delayed S shaped model and generalized it by involving imperfect debugging (b=0.1) and time delay function

$$\Phi(\mathrm{t}) = \frac{1}{\mathrm{r}(1-\mathrm{b})} \ln \{1 + (1-b) r t\}.$$

2. 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.
θ Fault detection rate of dependent faults.
p Proportion of independent faults.
Ψ 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.

3. ASSUMPTIONS
(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.

4. SOFTWARE RELIABILITY GROWTH MODEL

The total detected faults in time (0, t). are given by

$$m(t) = m_d(t) + m_i(t) \qquad (1)$$

4.1 Independent faults $m_i(t)$

The rate of independent faults detected is proportional to the remaining faults. We have following differential equation.

$$\frac{d}{dt} m_i(t) = r[f_i - m_i(t)] \qquad f_i > 0, 0 < r < 1$$

Under imperfect debugging the differential equation becomes

$$\frac{d}{dt} m_i(t) = r[f_i + bm_i(t) - m_i(t)] \qquad f_i > 0, 0 < r < 1$$

Solving equation using initial condition $m_i(0) = 0$ and involving time delay function $\Phi(t)$, we propose

$$m_i(t) = \frac{f_i}{(1-b)}[1 - \exp\{-r(1-b)(t - \Phi(t))\}] \qquad (2)$$

4.2 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

$$\frac{d}{dt} m_d(t) = \theta[f_d - m_d(t)]\frac{m_i(t)}{f_0} \qquad 0 < \theta < 1$$

Putting the $m_i(t)$ we get,

$$\frac{d}{dt} m_d(t) = \frac{f_i}{(1-b)f_0}\theta[f_d - m_d(t)]\,[1 - \exp - r1 - bt - \Phi t \qquad (3)$$

4.3 Yamada Delayed S shaped model

This model describe S shaped curve for the cumulative number of faults detected such that failure rate initially increases, and later decays. The S shaped curve can be regarded as a learning process because the testers' skills will gradually improve as time progresses.

Assuming $\Phi(t) = \frac{1}{r(1-b)}\ln\{1 + (1-b)rt\}$, simplifying equations (2) and (3) , and using $f_i = pf_0$ $f_d = (1-p)f_0$, we get

$$m_i(t) = \frac{pf_0}{(1-b)}[1 - (1 + r(1-b)t)\exp\{-r(1 - bt]$$

$$m_d(t) = (1-p)f_0[-\exp\left\{\frac{-p\theta[A+B]}{r(1-b)^2}\right\}]$$

$$A = r(1-b)t + 2\exp\{-r(1-b)t\}$$

$$B = r(1-b)t\exp\{-r(1-b)t\} - 2$$

5. 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

$$R\left(\frac{\Delta t}{t}\right) = \exp[-\{m(t + \Delta t) - m(t)\}] \qquad t \geq 0, \Delta t \geq 0$$

Assuming $f_0 = 400$, $r = 0.225$, $\theta = 0.0833$, $p = 0.55$, $\Psi = 2.84$
(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.

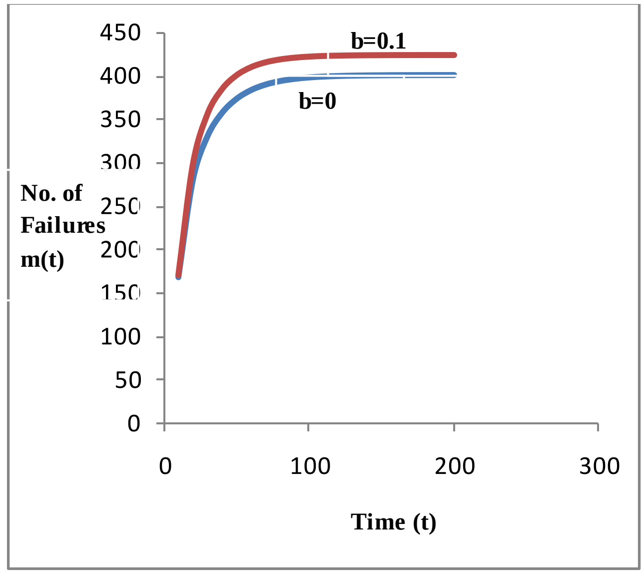
6. CONCLUSION

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.

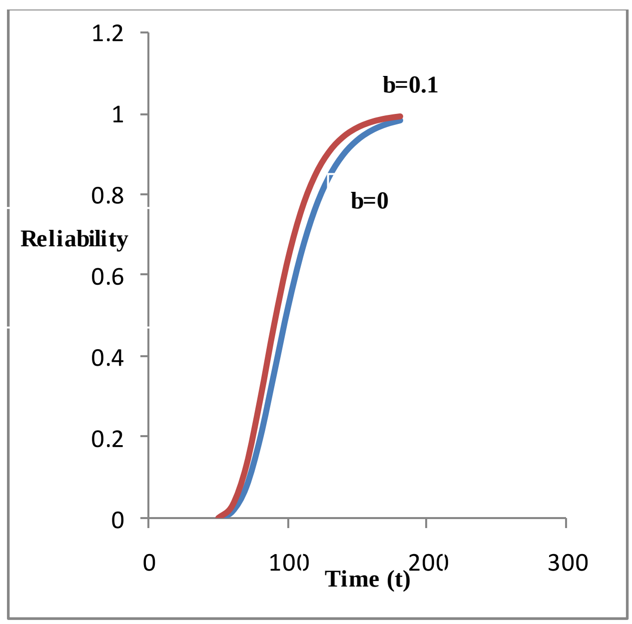
Graph 2 shows the variation of software reliability with respect to testing time. Software reliability increases rapidly with testing time during initial phase . Under imperfect debugging (b=0.1) after 140 units of testing time the probability of failure free execution of software in 10 units time interval is 90 % whereas under perfect debugging (b=0) the probability is 84%. 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: Imperfect Debugging and Software Reliability

Time(t)	No. of Failures under perfect and imperfect debugging m(t)		Software Reliability under perfect and imperfect debugging R(10/t)	
	b=0	b=0.1	b=0	b=0.1
10	168.486071	170.3327 30	3.92292E-49	2.83521E-56
20	279.9459043	298.2353796	2.41674E-22	7.0759E-26
30	329.7203558	356.1458972	2.42796E-12	3.07245E-13
40	356.4643255	384.9570294	9.86708E-08	1.08693E-07
50	372.5958026	400.991772	4.15088E-05	8.24615E-05
60	382.6854085	410.3949515	0.001718807	0.003652453
70	389.0515334	416.0073079	0.017875033	0.034491484
80	393.0758837	419.3743507	0.078475136	0.132281549
90	395.6208572	421.3971734	0.199978303	0.296503826
100	397.2304036	422.6128685	0.36133188	0.481577918
110	398.248362	423.3435558	0.525287239	0.644561008
120	398.892172	423.7827416	0.665524857	0.767990546
130	399.2993513	424.0467194	0.772964933	0.853280241
140	399.5568729	424.2053867	0.84970147	0.909037438
150	399.7197431	424.3007557	0.902120042	0.944289228
160	399.8227508	424.3580784	0.936929277	0.966132204
170	399.8878983	424.392533	0.9596346	0.979503584
180	399.929101	424.4132424	0.97427785	0.987629508
190	399.9551597	424.4256901	0.983654144	0.992546101
200	399.9716407	424.4331719		



Graph 1: No. of Failures m(t) and time (t)



Graph 2: Reliability R(10/t) and time (t)

7. REFERENCES

[1] Dwyer, D. & D'Onofrio, P., (2011). Improvements in estimating software reliability from growth test data. Reliability and Maintainability Symposium (RAMS), 2011 Proceedings - Annual, 1 – 5.

[2] Goel, A. L. & Yang, K. J. (1997). Software reliability And readiness assessment based on the non-homogenous Poisson process, *Advances in Computers*, *45*, 197-267.

[3] Gokhale, S.S., Lyu, M.R. & Trivedi, K.S. (2006). Incorporating fault debugging activities into software reliability models: a simulation approach. Reliability, IEEE Transactions on, 55(2), 281 – 292.

[4] Hung, C. Y., Lyu, M. R. & Kuo, S. Y. (2003). A unified scheme of some non-homogeneous poisson process models for software reliability estimation, *IEEE Trans. On Software Engineering 29*(3), 261-269.

[5] Hung, C. Y. & Lin, C. T. (2006).Software reliability analysis by considering fault dependency and debugging time lag, *IEEE Trans. on Reliability*, 55(3), 436-450.

[6] Hung, C. Y., Lin, C. T., Kuo, S. Y., Lyu, M. R. & Sue, C. C. (2004).Software reliability growth models incorporating fault dependency with various debugging time lags,*Proceedings of the 28th Annual International Computer Software and Application Conference, Hong Kong, China*, 186-191.

[7] Hung, C. Y., Lin, C. T., Lo, J. H. & Sue, C. C. (2004). Effect of fault dependency and debugging time lag on software error models, *Proceedings of the 2004 IEEE Region 10 Conference, Thailand*, 243-246.

[8] Kapur, P.K., Pham, H., Anand, S. & Yadav, K. (2011). A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation Reliability, IEEE Transactions on , 60 (1), 331 – 340.

[9] Lyu, M. R. (1993) .*Handbook of Software Reliability Engineering* : McGraw-Hill. 428-443.

[10] Ohba, M. (1984). Software reliability analysis models, *IBM Journal of Research and Development, 28*(4),

[11] Ohba, M. & Chou, X. (1989). Does imperfect debugging affect software reliability growth, *Proceedings of the 11th International Conference on Software Engineering, Pittsburgh, USA*, 237-244.

[12] Yamada, S., Tokunou, K. & Osaki, S. (1992). Imperfect debugging models with fault introduction rate for software reliability assessment, *International Journal of System Science*, 23(12), 2241-2252.

[13] Yang, K. Z. (1996). An infinite server queuing model for software readiness and related performance measures, Ph.D. Dissertation, *Department of Electrical Engineering and Computer Science, Syracuse University.*

[14] Xie, M. & Yang, B. (2003). A study of the effect of imperfect debugging on software development cost, IEEE Trans. Software Engineering, 29(5) ,471-473.

[15] Zhang, X. Liu, .,Gao, Y., Zhang, T.,& Liu, H.(2009).The Prediction Model of Software Reliability Based on the Modular. Information Technology and Applications, 2009. IFITA '09. International Forum on, 2 , 315 – 318.