ソフトウェア工学 100−11
（1994．9．8）

ソフトウェアフォールトの修正数に着目した不完全デバッグモデル

得能貢一，山田茂

鳥取大学工学部社会開発システム工学科
〒680 鳥取市湖山町南 4-101

実際のソフトウェア開発におけるテスト工程でのデバック作業は，常に確実に実施されているとは限らない．換言すると，発見されたソフトウェアフォールトのすべてが修正・除去されているわけではない．一般に，これを不完全デバッグと呼ぶ．本研究では，発見されたフォールトの修正には必ずしも成功しないという不完全デバッグ環境を考慮したソフトウェア信頼度成長モデルについて議論する．ここでは，あるテスト時刻までに修正された累積フォールト数を表す確率変数を定義して，このモデルをマルコフ過程により記述する．また，このモデルに基づいて，ソフトウェアの信頼性評価に有用な定量的尺度を導出し，数値例を示す．

An Imperfect Debugging Model Based on the Number of Corrected Software Faults

Koichi Tokuno, Shigeru Yamada

Department of Social Systems Engineering
Faculty of Engineering, Tottori University
Tottori, 680 Japan

Actual debugging actions during the testing phase in the software development are not always performed perfectly. In other words, all detected software faults are not corrected and removed certainly. Generally, this is called imperfect debugging. In this paper, we discuss a software reliability growth model considering imperfect debugging that the faults are not always corrected/removed when they are detected. Defining a random variable representing the cumulative number of faults corrected up to a specified testing time, this model is described by a semi-Markov process. We derive several quantitative measures for software reliability assessment and show their numerical examples.

1. Introduction

Most software reliability growth models proposed so far are based on the assumption of perfect debugging that all faults detected during the testing and operation phases are corrected and removed perfectly (see Yamada [9 ~ 11]). However, debugging actions in real testing and operation environments are not always performed perfectly. For example, type misses invalidate the fault correction activity or fault removal is not carried out precisely due to wrong analysis for the obtained testing results (see Shooman [8]). That is, they are in imperfect debugging environment. Therefore, the faults are not always corrected and removed perfectly when they are detected. Then, it is interesting to develop a software reliability growth model considering imperfect debugging environment (cf. Ohba and Chou [5] and Shanthikumar [7]). Such an imperfect debugging model is expected to estimate reliability assessment measures more accurately.

In this paper, we discuss a software reliability growth model with imperfect debugging that the faults are not corrected/removed certainly when they are detected. Defining a random variable representing the number of faults corrected by a given time point, this model is formulated by a semi-Markov process (see Goel and Okumoto [2] and Ross [6]). We derive various interesting quantities for software reliability measurement and show their numerical illustrations.

2. Model Description

For imperfect debugging environment, the software reliability model developed in this paper is based on the following assumptions.

1. Each fault which causes a software failure is corrected perfectly with probability p ($0 \leq p \leq 1$), while it is not corrected with probability $q(=1-p)$ (see Shanthikumar [7]). We call p the perfect debugging rate.

2. The hazard rate is constant between software failures caused by a fault in the software system, and geometrically decreases whenever each detected fault is corrected (see Moranda [3]).

3. The probability that two or more software failures occur simultaneously is negligible.

4. No new faults are introduced during the debugging. (In general, this assumption may not be true. However, Goel [1] claimed that if the additional faults introduced constitute a very small fraction of the fault population, the practical effect on model results would be minimal.) At most one fault is removed when it is corrected and the correction time is not considered.

Now, we consider a stochastic process $(\boldsymbol{Y}, \boldsymbol{T})$ where fault count vector $\boldsymbol{Y}= \{Y(l);\ l = 1,2,\cdots\}$ and time series vector $\boldsymbol{T}= \{T_l;\ l = 1,2,\cdots\}$ (see Ross [6]). Let $i = 0,1,2,\cdots$ be the state space, where i represents the cumulative number of corrected faults. Then, the events $\{Y(l) = i\}$ means that i faults have been corrected at the l-th software failure occurrence and T_l represents the l-th software failure occurrence time, where $Y(0) = 0$ and $T_0 = 0$. A sample function of $(\boldsymbol{Y}, \boldsymbol{T})$ is shown in Fig. 1. For example, Fig. 1 shows that a fault is detected at T_3 but the fault correction fails (i.e. the fault is imperfectly debugged). Further, let $X(t)$ be a random variable representing the cumulative number of faults corrected up to the testing time t. Then, $X(t)$ forms a semi-Markov process (see Goel and Okumoto [2]). That is, from assumption 1, when i faults have been corrected by arbitrary testing time t,

$$X(t) = \begin{cases} i & \text{(with probability } q\text{)} \\ i+1 & \text{(with probability } p\text{),} \end{cases} \qquad (1)$$

(see Fig. 2). Further, from assumption 2, when i faults have been corrected, the hazard rate for

the next software failure occurrence is given by

$$z_i(t) = Dk^i \ (i = 0, 1, 2, \cdots;\ D > 0,\ 0 < k < 1), \tag{2}$$

where D and k are the initial hazard rate and the decreasing ratio, respectively. Noting that some specified functions are executed frequently, (2) reflects that the faults cause software failures with high frequency in execution during the early stage of the testing and the hazard rate decreases rapidly by correcting them. This assumption is a practically modified one (see Musa et al. [4]). Early imperfect debugging models such as Goel and Okumoto [2] and Ohba and Chou [5] often assume that the hazard rate changes at each fault correction by a constant amount. Then, the distribution function for the next software failure occurrence time is given by

$$F_i(t) = 1 - e^{-Dk^i t}. \tag{3}$$

Let $Q_{ij}(t)$ denote the one step transition probability that after making a transition into state i, the process $\{X(t),\ t \geq 0\}$ makes a transition into state j by time t. Then, $Q_{ij}(t)$, which represents the probability that if i faults have been corrected at time zero, j faults are corrected by time t after the next failure occurs, is given by

$$Q_{ij}(t) = P_{ij}(1 - e^{-Dk^i t}), \tag{4}$$

where P_{ij} are the transition probabilities from state i to state j and given by

$$P_{ij} = \begin{cases} q\ (j = i) \\ p\ (j = i+1) \\ 0\ (\text{elsewhere}) \end{cases} \quad (i, j = 0, 1, 2, \cdots). \tag{5}$$

3. Derivation of Reliability Measures

3.1 Distribution of the First Passage Time to the Specified Number of Corrected Faults

Suppose that i faults have been corrected at some testing time. Let $G_{i,n}(t)$ denote a distribution function of the first passage time from state i to state n. In other words, $G_{i,n}(t)$ is the probability that n faults are corrected in the time interval $(0,\ t]$ on the condition that i faults have been already corrected at time zero. Then, we get the following renewal equation:

$$G_{i,n}(t) = Q_{i,i+1} * G_{i+1,n}(t) + Q_{i,i} * G_{i,n}(t) \quad (i = 0,\ 1,\ 2,\ \cdots,\ n-1), \tag{6}$$

where $*$ denotes a Stieltjes convolution and $G_{n,n}(t) = 1\ (n = 1, 2, \cdots)$.

We use Laplace–Stieltjes (L–S) transforms to solve (6), where the L–S transform of $G_{i,n}(t)$ is defined as

$$\widetilde{G}_{i,n}(s) \equiv \int_0^\infty e^{-st} dG_{i,n}(t). \tag{7}$$

From (6) we get

$$\widetilde{G}_{i,n}(s) = \widetilde{Q}_{i,i+1}(s)\widetilde{G}_{i+1,n}(s) + \widetilde{Q}_{i,i}(s)\widetilde{G}_{i,n}(s) \quad (i = 0,\ 1,\ 2,\ \cdots,\ n-1). \tag{8}$$

From (4) the transforms of $Q_{i,i+1}(t)$ and $Q_{i,i}(t)$ are respectively given as

$$\widetilde{Q}_{i,i+1}(s) = \frac{pDk^i}{s + Dk^i}, \tag{9}$$

$$\widetilde{Q}_{i,i}(s) = \frac{qDk^i}{s + Dk^i}. \tag{10}$$

Substituting (9) and (10) into (8) yields

$$\widetilde{G}_{i,n}(s) = \frac{pDk^i}{s + pDk^i}\widetilde{G}_{i+1,n}(s) \quad (i = 0, 1, 2, \cdots, n-1). \tag{11}$$

Solving (11) recursively, we obtain the L–S transform of $G_{0,n}(t)$ as

$$\widetilde{G}_{0,n}(s) = \prod_{i=0}^{n-1} \frac{pDk^i}{s + pDk^i} = \sum_{i=0}^{n-1} A_{k,i,n} \frac{pDk^i}{s + pDk^i}, \tag{12}$$

where

$$\left.\begin{aligned} A_{k,0,1} &\equiv 1 \\ A_{k,i,n} &= \frac{k^{\frac{1}{2}n(n-1)-i}}{\prod_{\substack{j=0 \\ j \neq i}}^{n-1} (k^j - k^i)} \\ (n = 2, 3, \cdots,\ i = 0, 1, 2, \cdots, n-1) & \end{aligned}\right\}. \tag{13}$$

By inverting (12) with respect to t and rewriting $G_{0,n}(t)$ as $G_n(t)$, we have the distribution function of the first passage time when n faults are corrected

$$G_n(t) = \sum_{i=0}^{n-1} A_{k,i,n}(1 - e^{-pDk^i t}), \qquad (14)$$

where $G_0(t) \equiv 1$.

3.2 Distribution of the Number of Faults Corrected Up to a Specified Time

Let S_n $(n = 1, 2, \cdots)$ be random variables representing the n-th successful correction time of detected faults. Since $X(t)$ is a counting process (see Fig. 1), we have the following equivalent relation:

$$\{S_n \le t\} \iff \{X(t) \ge n\}. \qquad (15)$$

Therefore, we get

$$\Pr\{S_n \le t\} = \Pr\{X(t) \ge n\}. \qquad (16)$$

Let $P_n(t)$ denote the probability that n faults are corrected up to testing time t. From (14) and (16), we obtain the probability mass function $P_n(t)$ as

$$\begin{aligned} P_n(t) &= \Pr\{X(t) = n\} \\ &= G_n(t) - G_{n+1}(t). \end{aligned} \qquad (17)$$

Suppose that the initial fault content in the system prior to the testing, N, is known. Using (17), we can derive the expectation and variance of $X(t)$, respectively as

$$\mathrm{E}[X(t) \mid N] = \sum_{n=1}^{N} G_n(t), \qquad (18)$$

$$\mathrm{Var}[X(t) \mid N] = \sum_{n=1}^{N} (2n-1) G_n(t) - \left\{ \sum_{n=1}^{N} G_n(t) \right\}^2, \qquad (19)$$

where it is noted that $P_N(t) = G_N(t)$ since $G_{N+1}(t) = 0$.

3.3 Expected Number of Faults Detected Up to a Specified Time

We introduce a new random variable $Z(t)$ representing the cumulative number of faults detected up to testing time t. Let $M_i(t)$ be the expected number of faults detected up to time t on the condition that i faults have been already corrected at time zero, i.e.

$$M_i(t) = \mathrm{E}[Z(t) \mid X(0) = i], \qquad (20)$$

which is called a Markov renewal function. In similar to the preceding subsection, supposing that the initial fault content N is known, we obtain the following renewal equations:

$$\begin{aligned} M_i(t) = F_i(t) + Q_{i,i} * M_i(t) + Q_{i,i+1} * M_{i+1}(t) \\ (i = 0, 1, 2, \cdots, N-1), \end{aligned} \qquad (21)$$

where $M_N(t) = 0$. Using the L-S transforms of $M_i(t)$ $(i = 0, 1, 2, \cdots, N-1)$, we get from (12)

$$\begin{aligned} \widetilde{M_0}(s) &= \frac{1}{p} \sum_{n=1}^{N} \prod_{i=0}^{n-1} \frac{pDk^i}{s + pDk^i} \\ &= \frac{1}{p} \sum_{n=1}^{N} \widetilde{G}_n(s). \end{aligned} \qquad (22)$$

Inverting (22) with respect to t and rewriting $M_0(t)$ as $M(t \mid N)$, we have

$$\begin{aligned} M(t \mid N) &= \frac{1}{p} \sum_{n=1}^{N} G_n(t) \\ &= \frac{1}{p} \mathrm{E}[X(t) \mid N]. \end{aligned} \qquad (23)$$

Now we consider that all faults detected by the testing are divided into two types. One are the faults which are successfully corrected, and the other are the faults detected again due to imperfect debugging. Then, the expected number of faults debugged imperfectly is given by

$$\begin{aligned} D(t \mid N) &= M(t \mid N) - \mathrm{E}[X(t) \mid N] \\ &= \frac{q}{p} \mathrm{E}[X(t) \mid N]. \end{aligned} \qquad (24)$$

3.4 Distribution of the Time between Software Failures

Let X_l $(l = 1, 2, \cdots)$ be a random variable representing the time interval between the $(l-1)$-st and the l-th software failure occurrences and $\Phi_l(x)$ be a distribution function of X_l. It is noted that X_l depends on the number of the faults corrected up to the $(l-1)$-st software failure occurrence, which is not explicitly known.

Further, let C_l be a random variable representing the number of faults corrected up to the $(l-1)$-st software failure occurrence. Then, C_l follows a binomial distribution having the following probability mass function:

$$\Pr\{C_l = i\} = \binom{l-1}{i} p^i q^{l-1-i} \quad (i = 0, 1, 2, \cdots, l-1), \tag{25}$$

where $\binom{l-1}{i}$ is a binomial coefficient denoted as $\binom{l-1}{i} = (l-1)!/[(l-1-i)!i!]$. From (25), at the $(l-1)$-st software failure occurrence, the expected number of corrected faults is given by $p(l-1)$.

Further, it is evident that

$$\Pr\{X_l \le x \mid C_l = i\} = F_i(x), \tag{26}$$

which is given by (3). Accordingly, we can get the distribution function for X_l as

$$\begin{aligned}\Phi_l(x) &= \Pr\{X_l \le x\} \\ &= \sum_{i=0}^{l-1} \Pr\{X_l \le x \mid C_l = i\} \Pr\{C_l = i\} \\ &= \sum_{i=0}^{l-1} \binom{l-1}{i} p^i q^{l-1-i} (1 - e^{-Dk^i x}).\end{aligned} \tag{27}$$

Then, we have the reliability function for X_l as

$$\begin{aligned}R_l(x) &\equiv \Pr\{X_l > x\} \\ &= 1 - \Phi_l(x) \\ &= \sum_{i=0}^{l-1} \binom{l-1}{i} p^i q^{l-1-i} e^{-Dk^i x}.\end{aligned} \tag{28}$$

The expectation of random variable X_l is defined by

$$\mathrm{E}[X_l] \equiv \int_0^\infty R_l(x)dx. \tag{29}$$

We call (29) the mean time between software failures (MTBF). From (28), we can derive $\mathrm{E}[X_l]$ as

$$\mathrm{E}[X_l] = \frac{(p/k + q)^{l-1}}{D}. \tag{30}$$

Apparently, the following inequality holds for arbitrary natural number l:

$$\mathrm{E}[X_l] < \mathrm{E}[X_{l+1}] \quad (l = 1, 2, \cdots). \tag{31}$$

That is, a software reliability growth occurs whenever a software failure is observed.

4. Numerical Examples

Using the imperfect debugging model discussed above, we show numerical illustrations for software reliability measurement.

The distribution functions of the first passage time to the specified number of corrected faults, $G_n(t)$ in (14), are shown in Fig. 3 for various perfect debugging rates, p's, where $n = 10$, $D = 0.2$, and $k = 0.9$. We can see that the smaller perfect debugging rate p becomes, the more difficult it is to correct faults.

The expected numbers of faults corrected up to testing time t, $\mathrm{E}[X(t) \mid N]$ in (18), for various p's are shown in Fig. 4 where $N = 20$, $D = 0.2$, and $k = 0.9$. The variances of the number of faults corrected up to testing time t, $\mathrm{Var}[X(t) \mid N]$ in (19), for various p's are shown in Fig. 5 where $N = 20$, $D = 0.2$, and $k = 0.9$. As shown in Fig. 5, $\mathrm{Var}[X(t) \mid N]$ is a convex function with respect to testing time t with

$$\mathrm{Var}[X(0) \mid N] = \mathrm{Var}[X(\infty) \mid N] = 0. \tag{32}$$

This means that the correctability of faults in debugging is unstable during the early stage of the testing, and as the testing is in progress, it becomes stable. As shown in Fig. 5, we can see that the smaller the perfect debugging rate p

becomes, the more difficult it is to stabilize the fault-correctability.

Further, the coefficients of variation (CV) of $X(t)$, $\mathrm{cv}[X(t) \mid N]$, defined as

$$\mathrm{cv}[X(t) \mid N] \equiv \frac{\sqrt{\mathrm{Var}[X(t) \mid N]}}{\mathrm{E}[X(t) \mid N]}, \qquad (33)$$

for various p's are shown in Fig. 6 where $N = 20$, $D = 0.2$, and $k = 0.9$. As shown in Fig. 6, $\mathrm{cv}[X(t) \mid N]$ is a monotone decreasing function with respect to testing time t.

The expected number of faults detected up to testing time t, $M(t \mid N)$ in (23), is shown in Fig. 7 along with the expected number of imperfect debugging faults, $D(t \mid N)$ in (24), where $N = 30$, $D = 0.2$, $k = 0.9$, and $p = 0.9$. In this case,

$$M(\infty \mid 30) = 33.3, \; D(\infty \mid 30) = \frac{q}{p} \cdot 30 = 33.3q. \qquad (34)$$

Then, $(100q)\%$ of the cumulative number of faults detected eventually is imperfectly debugged.

The reliability functions, $R_l(x)$ in (28), for various l's are shown in Fig. 8 where $D = 0.2$, $k = 0.9$, and $p = 0.9$, and the values of MTBF for various l's are shown in Table 1. Fig. 8 and Table 1 show that a software reliability growth during the testing occurs whenever a software failure occurs.

5. Conclusion

In this paper, from a viewpoint that whether a fault correction activity succeeds or not is uncertain, based on a semi-Markov process, we have developed a software reliability growth model for imperfect debugging environment in which the faults detected by testing are not always corrected/removed. Various interesting quantities for software reliability measurement have been derived from the model and their numerical examples have been illustrated.

Acknowledgment

Shigeru Yamada is pleased to acknowledge the support of a Grant-in-Aid for Scientific Research from the Ministry of Education, Science and Culture of Japan under Grant No. 06680323.

References

[1] A. L. Goel, "Software reliability models: Assumptions, limitations, and applicability", *IEEE Trans. Software Engineering*, vol. SE-11, no. 12, pp. 1411-1423, Dec. 1985.

[2] A. L. Goel and K. Okumoto, "An Imperfect Debugging Model for Reliability and Other Quantitative Measures of Software Systems", Technical Report No. 78-1, Department of Industrial Engineering and Operations Research, Syracuse University, New York, 1978.

[3] P. B. Moranda, "Event-altered rate models for general reliability analysis", *IEEE Trans. Reliability*, vol. R-28, no. 5, pp. 376-381, Dec. 1979.

[4] J. D. Musa, A. Iannino, and K. Okumoto, *"Software Reliability: Measurement, Prediction, Application"*, McGraw-Hill, New York (1987).

[5] M. Ohba and X. Chou, "Does imperfect debugging affect software reliability growth ?", *Proc. 11th Int. Conf. Software Engineering*, pp. 237-244, 1989.

[6] S. M. Ross, *"Stochastic Processes"*, John Wiley & Sons, New York, 1983.

[7] J. G. Shanthikumar, "A state- and time-dependent error occurrence-rate software reliability model with imperfect debugging", *Proc. National Computer Conf.*, pp. 311-315, 1981.

[8] M. L. Shooman, *"Software Engineering: Design, Reliability, and Management"*, McGraw-Hill, New York, 1983.

[9] S. Yamada, "*Software Reliability Assessment Technology*" (in Japanese), HBJ Japan, Tokyo, 1989.

[10] S. Yamada, "Software quality/reliability measurement and assessment: Software reliability growth models and data analysis", *J. Information Processing*, vol. 14, no. 3, pp. 254-266, 1991.

[11] S. Yamada, "*Software Reliability Models: Fundamentals and Applications*", (in Japanese), JUSE Press, Tokyo, 1994.

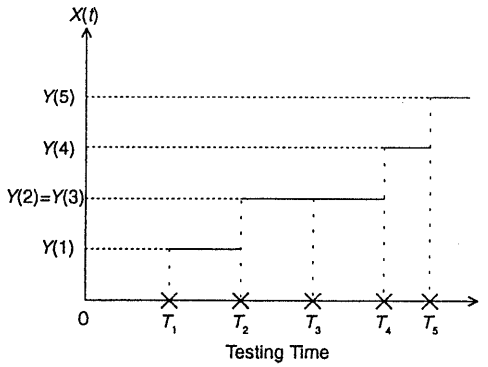

Fig. 1. A sample function of (Y, T).

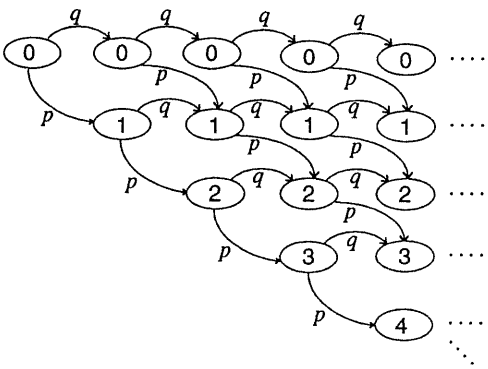

Fig. 2. A diagrammatic representation of transitions between states of $X(t)$.

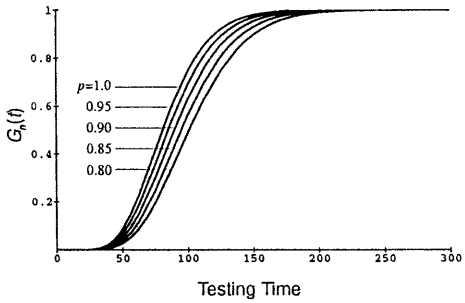

Fig. 3. Dependence of perfect debugging rate p in $G_{10}(t)$ (D=0.2, k=0.9).

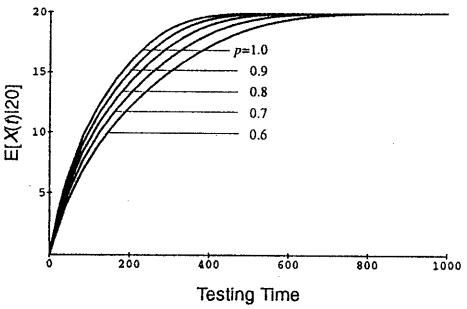

Fig. 4. Dependence of perfect debugging rate p in E[$X(t)$|20] (D=0.2, k=0.9).

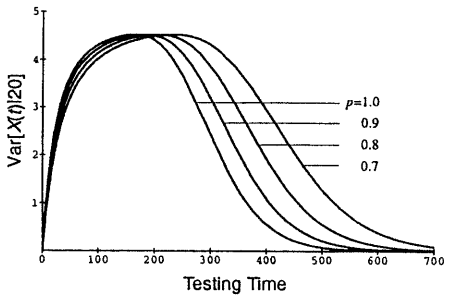

Fig. 5. Dependence of perfect debugging rate p in Var[$X(t)$|20] (D=0.2, k=0.9).

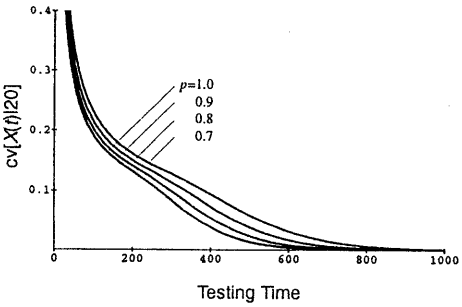

Fig. 6. Dependence of perfect debugging rate p in cv[$X(t)$|20] (D=0.2, k=0.9).

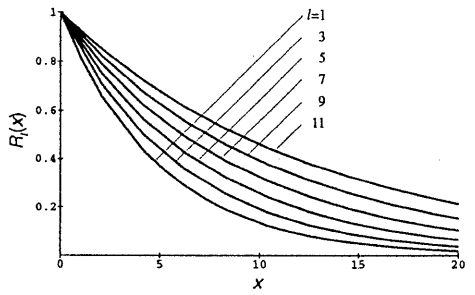

Fig. 8. Dependence of number of failures l in software reliability $R_l(x)$ (D=0.2, k=0.9, p=0.9).

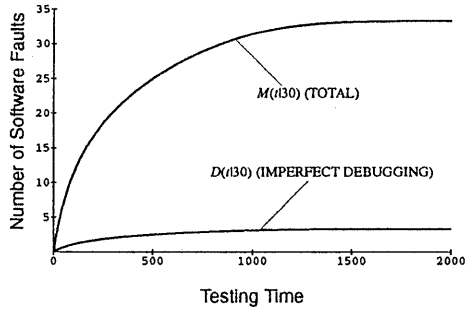

Fig. 7. $M(t|30)$ and $D(t|30)$ (D=0.2, k=0.9, p=0.9).

Table 1. MTBF $E[X_l]$ (p=0.9, D=0.2, k=0.9).

l	$E[X_l]$
1	5.000
2	5.500
3	6.050
4	6.655
5	7.321
6	8.053
7	8.858
8	9.744
9	10.72
10	11.79