Computing (2011) 92:33–48
DOI 10.1007/s00607-010-0128-6

Transfer function modelling in software reliability

S. Chatterjee · S. Nigam · J. B. Singh · L. N. Upadhyaya

Received: 29 May 2009 / Accepted: 21 October 2010 / Published online: 13 November 2010

Abstract This paper demonstrates the applicability of transfer function model in the field of software reliability. Here a stepwise procedure for fitting a transfer function model has been described and then the prediction of remaining faults in software has been done using the built in model. Some real life data have been used for illustration purpose.

Keywords Software reliability · Transfer function · Faults · TBF

Mathematics Subject Classification (2000) 68N30 Mathematical aspects of software engineering (specification, verification, metrics, requirements, etc.)

1 Introduction

Computer has become an essential part of our modern society and is used in our day to day life, starting from data processing, payment system, etc., to many complex and safety critical systems like air traffic control system, telecommunications, defence, medical, instrumentation, etc. All these activities are mainly executed by software. Due to the rapid growth of technology and high level of industrial competition, the software development process has also become very complex. As a result cost of software is also increasing. With time, the demand of highly reliable software at a reasonable price is increasing. Hence, these days, study of software reliability has become a very important and necessary issue. Measurement in software is still in its infancy. Since 1970, various approaches have been used to make software reliability growth models (SRGMs) to quantify the reliability, cost, release time, etc. of software.

S. Chatterjee (✉) · S. Nigam · J. B. Singh · L. N. Upadhyaya
Department of Applied Mathematics, ISM, Dhanbad 826004, India
e-mail: chatterjee_subhashis@rediffmail.com

The primary reason for performing reliability analysis is to improve the operational profile of software. This is achieved by applying SRGMs to failure data obtained during testing of software. The purpose of using SRGMs is to determine whether the test results obtained is satisfying the desired requirements and to improve the reliability of software sufficiently. Use of SRGMs are not only for estimating reliability, but also these models are used to predict the remaining errors, time between failures (TBFs), cost and release time of a software. Detailed study about software reliability modelling and its applications may be found in [1–3].

SRGMs have been classified in different categories by various researchers [1,4]. Goel [4] has classified the SRGMs in four categories: (1) time between failure class model, (2) failure count model, (3) fault seeding model, and (4) input domain based model. Most of the SRGMs developed falls under categories (1) and (2). In time between failure class models, the TBF has been considered as a random variable and assumed that it follows some distribution. Similarly in failure count category number of failures has been considered as a random variable and assumed that it follows some stochastic process, like non homogeneous Poisson process. All these models are mainly used for predicting remaining errors and reliability of software. During development of these models various assumptions are made like faults are removed immediately, software debugging process is perfect, faults are not correlated, etc. In perfect debugging category some important models are [5–15]. Later, different SRGMs have been developed using the concept of imperfect debugging and considering various aspects of software development process [16–32]. Unfortunately none of these models can be used for all types of software failure data. To overcome this serious drawback the use of time series modelling in software reliability was first proposed by Singpurwalla and Soyer [33]. According to Goel [4], time series analysis can be used for software reliability modelling. The same idea was also ratified in [2,3]. Later some more time series models in the area of software reliability have been described in [34,35]. In reality, software debugging process is imperfect because any change or modification(s) made in software during testing phase leads to the introduction of some new errors in software. Also due to functional and logical relationship in software the software faults are correlated. Hence, one error will influence the other. As a result, the removal of an error will remove more errors. Hence, introduction or removal of errors implies that, the reliability of software is decreasing or increasing. The advantage of time series modelling in software reliability is that these models are assumptions free and they can very well describe the dependency in faults as well as imperfect debugging. All these time series models in software reliability are either auto regressive (AR) or integrated auto regressive moving average (ARIMA) models. These models have very well predicted the TBFs of software but these models cannot be used for prediction of faults corresponding to each testing time. To overcome this limitation an attempt has been made to develop transfer function model in this paper. Applications of transfer function modelling in various areas of scientific and engineering problem have already been established. Some of them are given in [36–38]. Unfortunately, the application of transfer function modelling in reliability analysis is very less. Use of transfer function model in hardware reliability was first described by Singpurwalla [39]. In this paper the TBFs are considered as input series and the numbers of faults or failures as output series and a transfer function model has been developed to predict the number of faults

or failures present in the software corresponding to each TBF. The proposed model has been validated using some real software failure data. Considering number of faults as input series, TBFs of a software can be predicted as output series using transfer function model. This is another advantage of the transfer function modelling.

2 Proposed model

Detail study about transfer function modelling is available in [40]. Transfer function models represent a dynamic relationship between a continuous input and a continuous output. The relationship between the continuous input X_t and the continuous output Y_t, i.e., the transfer between X_t and Y_t is represented by a linear differential equation. In transfer function model building observations must be considered in pairs (X_t, Y_t), each measured at equispaced times. In discrete transfer function model X_t, Y_t both are discrete and the transfer between them is represented parsimoniously by the linear difference equation

$$(1 + \xi_1\nabla + \cdots + \xi_r\nabla^r)Y_t = (\eta_0 + \eta_1\nabla + \cdots \eta_r\nabla^s)X_{t-b} \tag{1}$$

In Eq. (1) the backward difference operator ∇ is used in place of the differential operator $\mathrm{D} = \frac{d}{dt}$, since $X_t$and Y_t is discrete.

Here $\xi(\nabla) = 1 + \xi_1\nabla + \cdots + \xi_r\nabla^r$, $\eta(\nabla) = \eta_0 + \eta_1\nabla + \cdots \eta_r\nabla^s$ are different operators, $\xi_1, \xi_2, \ldots \xi_r$ and $\eta_0, \eta_1, \ldots, \eta_s$ are unknown parameters, which in practice, have to be estimated from the data. Constant b the delay parameter, associated with the leading indicator series X_t indicates which of the previous values X_t affect the present Y_t. In [40] the delay parameter b has been defined as delay or dead time before the response to a given input change begins to take effect. In general if b unit delay is assumed, then the index t is replaced by $t - b$. Here, in this paper the input series X_t represents the TBFs of a software. Since there is a delay in fault identification and fault correction process during software testing, X_t cannot be a actual TBF. Practically after error detection, error correction consumes some time. Hence there is a delay and this delay has been represented by b here.

Equation (1) may be written equivalently in terms of past values of the input and output by substituting $B = 1 - \nabla$, where B is the backward shift operator defined as $BX_t = X_{t-1}$ and $B^bX_t = X_{t-b}$. Therefore, Eq. (1) becomes:

$$\begin{aligned}(1 - \delta_1 B - \cdots - \delta_r B^r)Y_t &= (\omega_0 - \omega_1 B - \cdots - \omega_s B^s)X_{t-b} \\ &= (\omega_0 B^b - \omega_1 B^{b+1} - \cdots - \omega_s B^{b+s})X_t\end{aligned}$$

or $\delta(B)Y_t = \omega(B)B^bX_t = \Omega(B)X_t$ where

$$\begin{aligned}\delta(B) &= 1 - \delta_1 B - \delta_2 B^2 - \cdots - \delta_r B^r, \\ \omega(\mathrm{B}) &= \omega_0 - \omega_1 B - \omega_2 B^2 - \cdots - \omega_s B^s \quad \text{and} \quad \Omega(\mathrm{B}) = \omega(\mathrm{B})\mathrm{B}^b\end{aligned}$$

are different operators used in time series analysis [40].

Alternatively, the pair of observations (X_t, Y_t) is represented by a linear filter

$$Y_t = \nu_0 X_t + \nu_1 X_{t-1} + \nu_2 X_{t-2} + \cdots = \nu(B) X_t$$

for which the transfer function

$$\nu(B) = \nu_0 + \nu_1 B + \nu_2 B^2 + \cdots$$

can be expressed as a ratio of two polynomials

$$\nu(B) = \frac{\Omega(B)}{\delta(B)} = \delta^{-1}(B)\Omega(B).$$

The final transfer function model is

$$Y_t = \frac{\Omega(B)}{\delta(B)} X_t \tag{2}$$

of order (r,s).

Stepwise procedure:

(1) Prior to model building a first step in the analysis of the data is its careful screening. This is done by normalizing the data with suitable transformation like log transfer to remove the nonstationarity in the data.
(2) After normalizing the data, for the development of a transfer function model a tentative identification of the values of r, b and s is done where r is the order of the AR model, s is the order of the MA model and b is the delay parameter. This can be accomplished by an examination of the partial autocorrelation, auto-correlation and cross-correlation [40].
(3) After identification of r, s and b the maximum likelihood estimation technique is used to estimate the model parameters by minimising the conditional sum of square function as given in [40].
(4) After estimating the values of r, s and b fit a transfer function model as given in Eq. (2) and predict the remaining faults present in the software.

3 Results and discussion

Five real data sets have been used for model development and validation. From the cross-correlation function of all five data sets used here, the estimated value of delay b obtained is one.

Data set: 1 The data set used for the development of a transfer function model is a Control System data set given in [3]. Here, the cumulative time between failures has been considered as input series and the number of faults has been considered as output series. Since the input series contained the fluctuations the natural log transformation has been used to normalize it. This data set contains 136 observations. The first 100

Table 1 Predicted failures corresponding to each cumulative TBF

Cumulative TBF	Log (cumulative TBF)	Faults	Predicted faults	Cumulative TBF	Log (cumulative TBF)	Faults	Predicted faults
42,045	10.6465	101	101.2275	56,463	10.9413	119	119.4174
42,188	10.6499	102	102.2380	56,485	10.9417	120	120.4280
42,296	10.6524	103	103.2486	56,560	10.9431	121	121.4386
42,296	10.6524	104	104.2592	57,042	10.9515	122	122.4491
45,406	10.7234	105	105.2698	62,551	11.0437	123	123.4597
46,653	10.7505	106	106.2802	62,651	11.0453	124	124.4702
47,596	10.7705	107	107.2908	62,661	11.0455	125	125.4807
48,296	10.7851	108	108.3013	63,732	11.0624	126	126.4913
49,171	10.8031	109	109.3119	64,103	11.0682	127	127.5018
49,416	10.8080	110	110.3224	64,893	11.0805	128	128.5124
50,145	10.8227	111	111.3334	71,043	11.1710	129	129.5230
52,042	10.8598	112	112.3436	74,364	11.2167	130	130.5334
52,489	10.8684	113	113.3541	75,409	11.2307	131	131.5440
52,875	10.8757	114	114.3646	76,057	11.2392	132	132.5545
53,321	10.8841	115	115.3752	81,542	11.3089	133	133.5651
53,443	10.8864	116	116.3858	82,702	11.3230	134	134.5756
54,433	10.9047	117	117.3963	84,566	11.3453	135	135.5861
55,381	10.9220	118	118.4069	88,682	11.3298	136	136.5967

observations are used for model fitting and the next 36 observations are used for the prediction. The estimated parameter values are

$$\delta_1 = 1.063, \quad \delta_2 = 0.02889, \quad \delta_3 = -0.03306, \quad \delta_4 = -0.04826, \quad \omega_0 = -0.001083.$$

The proposed transfer function model of order (4,1) using Eq. (2) is as follows

$$Y_t = \frac{-0.001083}{1 - 1.063B - 0.02889B^2 + 0.03306B^3 + 0.04826B^4} X_t \tag{3}$$

The remaining failures in the software are predicted using the model proposed in (3). The original and predicted failures corresponding to each cumulative TBF have been computed and tabulated in Table 1. The corresponding graph is given in Fig. 1.

Data set: 2 The data set 2 used for the development of a transfer function model is a Real-Time Control Systems data given in [3]. The software for monitor and real-time control systems consists of about 200 modules and each module has, on average, 1,000 lines of codes developed using a high level language like FORTRAN. This data set records the software failures detected during the 111-day testing period. Here, the failure per day has been considered as input series and cumulative faults have been considered as output series. The first 70 observations are used for model fitting and

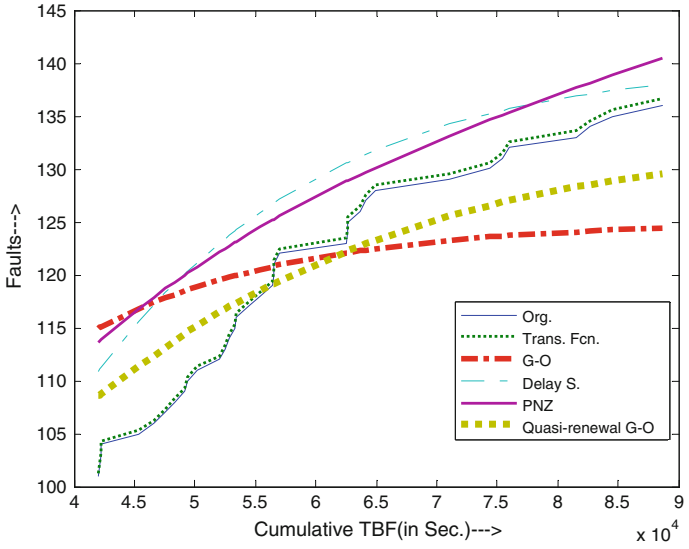

Fig. 1 Faults corresponding to cumulative TBF

the next 41 observations are used for the prediction. The estimated parameter values are

$$\delta_1 = 2, \quad \delta_2 = -1, \quad \delta_3 = -0.03306, \quad \omega_0 = -2.643 \times 10^{-16},$$
$$\omega_1 = 1.97 \times 10^{-16}, \quad \omega_2 = 4.608 \times 10^{-17}.$$

The proposed transfer function model of order (3,3) using Eq. (2) is as follows

$$Y_t = \frac{-2.643 \times 10^{-16} - 1.97 \times 10^{-16} B - 4.608 \times 10^{-17} B^2}{1 - 2B + B^2 + 0.03306 B^3} X_t \tag{4}$$

The remaining faults in the software are predicted using the model proposed in (4). The original and predicted faults corresponding to each day have been computed and tabulated in Table 2. The corresponding graph is given in Fig. 2.

Data set: 3 The data set 3 used for the development of a transfer function model was reported by Zhang (2002) based on system test data for a telecommunication system. System test data consists of two releases (Phases 1 and 2) given in [3]. Both of them consist of software failure data detected during 21 weeks of testing individually. Here, the weeks have been considered as input series and a cumulative fault have been considered as output series. Both phases (Phase 1 and Phase 2) the first 11 observations are used for fitting a transfer function model and the next 10 observations are used for the prediction. The estimated parameter for Phase I data set is

$$\delta_1 = 1.486, \quad \omega_0 = -0.1081, \quad \omega_1 = -0.2297$$

Table 2 Predicted failures corresponding to each day

Time (in days)	Original cumulative failures	Predicted cumulative failures	Time (in days)	Original cumulative failures	Predicted cumulative failures
71	467	467	92	475	475
72	468	467	93	475	475
73	469	469	94	475	475
74	469	470	95	475	475
75	469	469	96	476	475
76	469	469	97	476	477
77	470	469	98	476	476
78	472	471	99	476	476
79	472	474	100	477	476
80	473	472	101	477	478
81	473	474	102	477	477
82	473	473	103	478	477
83	473	473	104	478	479
84	473	473	105	478	478
85	473	473	106	479	478
86	473	473	107	479	480
87	475	473	108	479	479
88	475	477	109	480	479
89	475	475	110	480	481
90	475	475	111	481	480
91	475	475			

The proposed transfer function model of order (1,2) using Eq. (2) is as follows

$$Y_t = \frac{-0.1081 + 0.2297B}{1 - 1.486B} X_t \tag{5}$$

The remaining faults in the software are predicted using the model proposed in (5). The original and predicted faults corresponding to each week have been computed and tabulated in Table 3. The corresponding graph is given in Fig. 3.

For the phase 2 data set the estimated parameter values are

$$\delta_1 = 1.132, \quad \omega_0 = 0.05498, \quad \omega_1 = -0.05965$$

The proposed transfer function model of order (1,2) using Eq. (2) is as follows

$$Y_t = \frac{0.05498 + 0.05965B}{1 - 1.132B} X_t \tag{6}$$

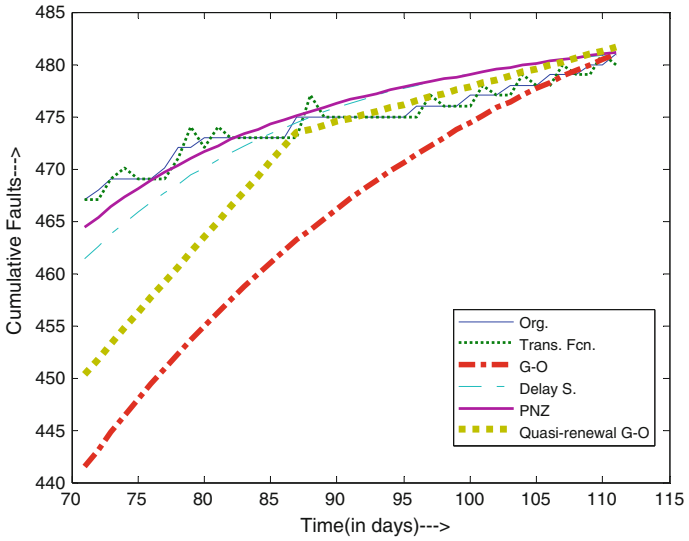

Fig. 2 Faults corresponding to time

Table 3 Predicted failures corresponding to each week for Phase 1

Time (in weeks)	Original cumulative faults	Predicted faults
12	15	17
13	19	20
14	19	25
15	22	25
16	22	29
17	23	29
18	24	31
19	24	32
20	24	32
21	26	31

The remaining faults in the software are predicted using the model proposed in (6). The original and predicted faults corresponding to each week have been computed and tabulated in Table 4. The corresponding graph is given in Fig. 4.

Data set: 4 Here a real time control application software failure data, Data 7, given in the CD ROM attached with the book published by Lyu [41] has been used for fitting a transfer function model. This is a real time control application software failure data. The software consists 870,000 lines of code. The test time reported in days and the cumulative faults captured corresponding each day.

Here, the execution time has been considered as input series and the corresponding cumulative fault has been considered as output series. This data set contains 109

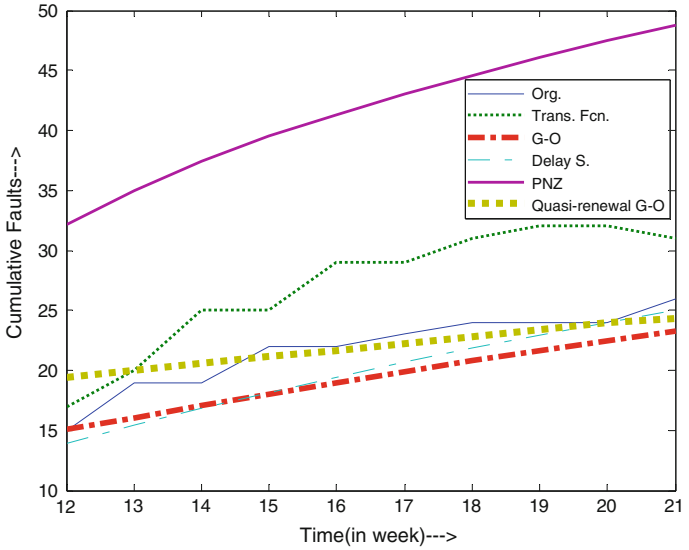

Fig. 3 Faults corresponding to time

Table 4 Predicted failures corresponding to each week for Phase 2

Time (in weeks)	Original cumulative faults	Predicted faults
12	25	25
13	30	27
14	32	33
15	36	35
16	37	35
17	39	39
18	39	43
19	39	42
20	42	42
21	43	46

observations. The first 60 observations are used for model fitting and the next 49 observations are used for the prediction. The estimated parameter values are

$$\delta_1 = 2, \quad \delta_2 = -1, \quad \omega_0 = -5.185 \times 10^{-16},$$
$$\omega_1 = -1.242 \times 10^{-15}, \quad \omega_2 = -1.535 \times 10^{-15}$$

The proposed transfer function model of order (2,3) using Eq. (2) is as follows

$$Y_t = \frac{-5.185 \times 10^{-16} + 1.242 \times 10^{-15} B + 1.535 \times 10^{-15} B^2}{1 - 2B + B^2} X_t \tag{7}$$

The remaining failures in the software are predicted using the model proposed in (7). The original and predicted failures corresponding to each failure have been computed and tabulated in Table 5. The corresponding graph is given in Fig. 5.

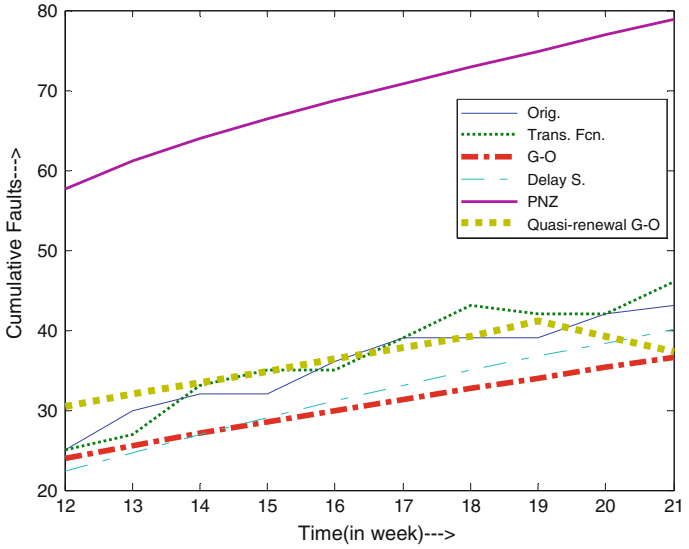

Fig. 4 Faults corresponding to time

Though various software failure data are available in [3,5] only five different data sets are used here for analysis purpose to restrict the length of the paper. The above mentioned procedure for developing transfer function model is applicable for any software failure data. The computed results shows, the number of faults predicted are almost very close to the original number of faults present in the software except in data set 3. But the result of data set 3 shows similar trend like original number of faults present in the software. The improvement in the prediction can be obtained by using pre whitening technique [40] in transfer function modelling. The modelling technique described here is applicable for any other data set.

4 Comparison

Many non homogeneous Poisson process based software reliability models are available in the literature. It is a time taking process to make comparative study with all these models. Therefore, a comparative study of the proposed transfer function model with some important existing non homogeneous Poisson process model[11,13,31,32] has been carried out in this section. The comparison has been done by evaluating some performance measures like, Akaike Information Criteria (AIC), Sum Square Error (SSE), Root Mean Square error (RMSE). AIC measure was proposed by Akaike [42]. The AIC (normalized by sample size n) as given in [40] is as follows:

$$\text{AIC} = \frac{-2\log(\text{maximizedlikelihood}) + 2\text{r}}{n} \approx \log\left(\sigma_e^2\right) + r\left(\frac{2}{n}\right) + \text{constant}$$

where σ_e^2 is the error variance and r is the number of parameters including the constant term. The second term in the AIC is a penalty factor for inclusion of more parameters.

Table 5 Predicted failures corresponding to each week

Time (in weeks)	Original failures	Predicted failures	Time (in weeks)	Original failures	Predicted failures
61	473	476	86	524	524
62	473	473	87	524	525
63	476	473	88	526	524
64	476	479	89	536	528
65	480	476	90	526	546
66	480	484	91	527	516
67	481	480	92	528	528
68	483	482	93	528	529
69	483	485	94	528	528
70	484	483	95	528	528
71	486	485	96	528	528
72	491	488	97	529	528
73	494	496	98	529	530
74	496	497	99	530	529
75	497	498	100	530	531
76	508	498	101	530	530
77	509	519	102	530	530
78	509	510	103	530	530
79	511	509	104	532	530
80	513	513	105	532	534
81	517	515	106	533	532
82	518	521	107	533	534
83	518	519	108	535	533
84	522	518	109	535	537
85	523	526			

The performance measure SSE given in [3] is defined as follows.

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where y_i is the original fault and $\hat{y}_i$ is the predicted fault.

The root mean square error (RMSE) is frequently used measure of differences between values predicted and the original values defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2}$$

where $\hat{y}_i$ and y_i are the predicted and actual faults respectively.

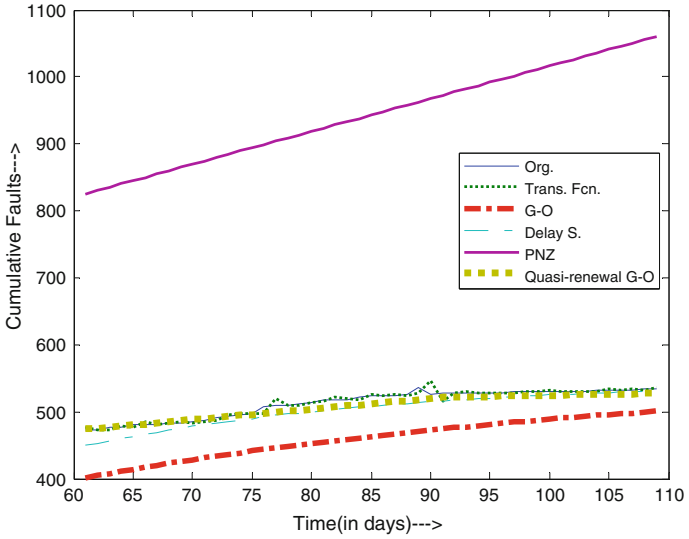

Fig. 5 Faults corresponding to time

The computed values of AIC, SSE and RMSE for proposed transfer function model and other NHPP models are given in Table 7. From the table it is clear that computed values of performance measures are less for proposed transfer function model in almost all data set used in this paper except for data set 3. One reason may be that the data set is very small. Also from the plots of prediction using different NHPP models and proposed model given in Figs. 1, 2, 3, 4, 5 it is observed that, the prediction made by the proposed transfer function model is very close to the original values and it also follows the same trend as original faults present in the software. Hence from the computed values of all the measures and the figures it is clear that the proposed model has better predictive ability.

5 Conclusion

The proposed transfer function model very well demonstrates the use of time series analysis for studying the interrelationship between TBFs and the number of errors. The results obtained in the previous section establishes the fact that, transfer function modelling is a better time series tool for the prediction of remaining software faults. It is better time series tool because transfer function models are useful for prediction of faults present in a software as well as for prediction of TBFs. Where as other time series models like AR, ARMA and ARIMA are used for prediction of TBF only because in these models only one time series is used. Also the proposed method is assumption free and it takes care of correlation between faults as well as imperfect debugging very well. The proposed time series model will help the decision maker to assess the correlation of faults. Results obtained shows that the method can be used for any software failure data.

Table 6 Mean value functions and estimated parameters

Model	Mean value function	Estimated parameters
Goel–Okumoto	$m(t) = a(1 - e^{-bt})$	$a = 125,\ b = 0.00006$ (data 1) $a = 497.282,\ b = 0.0308$ (data 2) $a = 50.4415,\ b = 0.0295$ (data 3-phase 1) $a = 71.8038,\ b = 0.0338$ (data 3-phase 2) $a = 557.6004,\ b = 0.0209$ (data 4)
Delay S Shaped	$m(t) = a(1 - (1 + bt)e^{-bt})$	$a = 140,\ b = 0.00007$ (data 1) $a = 483.039\, b = 0.06866$ (data 2) $a = 38.1189,\ b = 0.1068$ (data 3-phase 1) $a = 60.5933,\ b = 0.1077$ (data 3-phase 2) $a = 543.2680,\ b = 0.0522$ (data 4)
PNZ Model	$m(t) = \frac{a[1-e^{-bt}][1-\frac{\alpha}{\beta}]+\alpha at}{1+\beta e^{-bt}}$	$a = 121,\ b = 0.00005,\ \alpha = 2.5 \times 10^{-6},\ \beta = 0.002$ (data 1) $a = 470.759,\ b = 0.07497,\ \alpha = 0.00024,\ \beta = 4.69321$ (data 2) $a = 25.0931,\ b = 0.4247,\ \alpha = 0.0511,\ \beta = 26.3950$ (data 3-phase 1) $a = 40.0553,\ b = 0.6788,\ \alpha = 0.00496,\ \beta = 202.2334$ (data 3-phase 2) $a = 531.0082,\ b = 1.4800,\ \alpha = 0.0092,\ \beta = 221.4946$ (data 4)
Quasi -renewal Time-delay model based on G–O model	$m(t) = \sum_{k=1}^{n-1} \left(ab - b.m(s_1 \sum_{j=0}^{k-2} \alpha^j)\right) \left(s_1 \sum_{j=0}^{k-1} \alpha^j - s_1 \sum_{j=0}^{k-2} \alpha^j\right) + \left(ab - b.m[s_1 \sum_{j=0}^{n-2} \alpha^j]\right) \left(t - s_1 \sum_{j=0}^{n-2} \alpha^j\right)$	$a = 134.5,\ b = 0.00003,\ \alpha = 0.85,\ s_1 = 16000$ (data 1) $a = 490.31,\ b = 0.0187,\ \alpha = 0.93,\ s_1 = 45$ (data 2) $a = 28.5100,\ b = 0.05989,\ \alpha = 0.81,\ s_1 = 11$ (data 3-phase 1) $a = 48.2935,\ b = 0.06769,\ \alpha = 0.83,\ s_1 = 10$ (data 3-phase 2) $a = 540.2593,\ b = 0.01780,\ \alpha = 0.94,\ s_1 = 47$ (data 4)

Table 7 AIC, SSE and RMSE values

Models	AIC	SSE	RMSE
Goel–Okumotto	4.1882 (data 1)	2.2485e+003 (data 1)	7.9031 (data 1)
	4.3077 (data 2)	6.4631e+003 (data 2)	12.8733 (data 2)
	0.5191 (data 3-phase 1)	72.3876 (data 3-phase 1)	2.6905 (data 3-phase 1)
	1.6416 (data 3-phase 2)	359.4683 (data 3-phase 2)	5.9956 (data 3-phase 2)
	4.7367 (data 4)	1.4502e+005 (data 4)	54.4028 (data 4)
Delay shaped	2.1726 (data 1)	2.2077e+003 (data 1)	7.8310 (data 1)
	1.8823 (data 2)	244.3419 (data 2)	2.4412 (data 2)
	0.6907 (data 3-phase 1)	52.7967 (data 3-phase 1)	2.2978 (data 3-phase 1)
	1.2167 (data 3-phase 2)	219.7462 (data 3-phase 2)	4.6877 (data 3-phase 2)
	3.4098 (data 4)	6.6566e+003 (data 4)	11.6554 (data 4)
PNZ model	2.5711 (data 1)	2.1801e+003 (data 1)	7.7819 (data 1)
	1.0921 (data 2)	122.6356 (data 2)	1.7295 (data 2)
	2.5508 (data 3-phase 1)	3.9485e+003 (data 3-phase 1)	19.8708 (data 3-phase 1)
	2.0503 (data 3-phase 2)	1.0916e+004 (data 3-phase 2)	33.0388 (data 3-phase 2)
	7.9960 (data 4)	9.2033e+006 (data 4)	433.3851 (data 4)
Quasi -renewal Time-delay model based on G–O model	3.1713 (data 1)	696.6304 (data 1)	4.3990 (data 1)
	3.7498 (data 2)	1.9065e+003 (data 2)	6.8190 (data 2)
	1.8275 (data 3-phase 1)	28.1769 (data 3-phase 1)	1.6786 (data 3-phase 1)
	2.9090 (data 3-phase 2)	176.2544 (data 3-phase 2)	4.1983 (data 3-phase 2)
	3.3791 (data 4)	2.4516e+003 (data 4)	7.0734 (data 4)
Transfer function	−4.0888 (data 1)	6.5474 (data 1)	0.4265 (data 1)
	−0.0057 (data 2)	29 (data 2)	0.8410 (data 2)
	2.5246 (data 3-phase 1)	337 (data 3-phase 1)	5.8052 (data 3-phase 1)
	2.2816 (data 3-phase 2)	54 (data 3-phase 2)	2.3238 (data 3-phase 2)
	3.1977 (data 4)	939 (data 4)	4.3776 (data 4)

Acknowledgments This work is supported by University Grant Commission, New Delhi, India under grant F.No.33-115/2007 (SR). Also, the authors are thankful to Indian School of Mines, Dhanbad, India for providing facility. Authors are very much thankful to the reviewers for their valuable suggestions to improve the paper.

References

1. Musa JD, Iannino A, Okumoto K (1987) Software reliability measurement, prediction, application. McGraw-Hill Int, UK
2. Xie M (1991) Software reliability modelling. World Scientific Press, UK
3. Pham H (2006) System software reliability. Springer, UK
4. Goel AL (1985) Assumptions, limitations and applicability of software reliability modelling. IEEE Trans Software Eng 11:1411–1423
5. Jelinski Z, Moranda PB (1972) Software reliability research. In: Freiberger W (ed) Statistical computer performance evaluation. Academic Press, New York pp 465–484
6. Shooman ML (1972) Probabilistic models for software reliability prediction. In: Freiberger W (ed) Statistical computer performance evaluation. Academic Press, New York pp 485–502

7. Schick GJ, Wolverton RW (1978) An analysis of competing software reliability model. IEEE Trans Software Eng SE-4:104–120
8. Musa JD (1975) A theory of software reliability and its application. IEEE Trans Software Eng SE-1:312–327
9. Littlewood B, Verrall JL (1973) A Bayesian reliability growth model for computer software. Appl Stat 22:332–346
10. Xie M (1987) A shock model for software reliability. Microelectron Reliab 27:717–724
11. Goel AL, Okumoto K (1979) A time-dependent error detection rate model for software reliability and other performance measure. IEEE Trans Rel R-28:206–211
12. Chatterjee S, Misra RB, Alam SS (1997) Joint effect of test effort and learning factor on software reliability and optimal release policy. Int J Syst Sci 28(4):391–396
13. Yamada S, Ohba M, Osaki S (1984) S-shaped software reliability growth models and their applications. IEEE Trans Rel R-33:289–292
14. Pham L, Pham H (2001) A Bayesian predictive software reliability models with pseudo-failures. IEEE Trans Syst Man Cybern A Syst Hum 31(3):233–238
15. Pham L, Pham H (2000) Software reliability models with time dependent hazard rate based on Bayesian approach. IEEE Trans Syst Man Cybern A Syst Hum 30(1):25–35
16. Fakhre- Zakeri I, Slud E (1995) Mixture models for reliability of software with imperfect debugging: identifiably of parameters. IEEE Trans Rel 44:104–113
17. Zeephongsekul P, Xia G, Kumar S (1994) Software-reliability growth model: primary failures generate secondary-faults under imperfect debugging. IEEE Trans Rel 43:408–413
18. Xia G, Zeephongsekul P, Kumar S (1993) Optimal software release policy with a learning factor for imperfect debugging. Microelectron Rel 33:81–86
19. Pham H (1996) A software cost model with imperfect debugging random life cycle and penalty cost. Int J Syst Sci 27:455–463
20. Chatterjee S, Misra RB, Alam SS (1998) A generalized shock model for software reliability. Computer Electr Eng Int J 24:363–368
21. Gokhale SS, Lyu MR, Trivedi KS (2006) Incorporating fault debugging activities into software reliability models: a simulation approach. IEEE Trans Rel 55(2):281–292
22. Dai YS, Xie M, Poh KL (2005) Modelling and analysis of correlated software failures of multiple types. IEEE Trans Rel 54(1):100–106
23. Xie M, Dai YS, Poh KL, Lai CD (2004) Distributed system availability in the case of imperfect debugging process. Int J Syst Ind Eng Theory Appl Pract 11(4):396–405
24. Chatterjee S, Misra RB, Alam SS (2004) N-version programming with imperfect debugging. Comput Electr Eng 30(6):453–463
25. Park DH, Lee CH (2003) Markovian imperfect software debugging model and its performance measures. Stoch Anal Appl 21(4):849–864
26. Misra PN (1983) Software reliability analysis. IBM Syst J 22:262–272
27. Yamada S, Osaki S (1985) Cost-reliability optimal release policies for software systems. IEEE Trans Rel 31:422–424
28. Bhaskar T, Kumar UD (2006) A cost model for N-version programming with imperfect debugging. J Oper Res Soc 57(8):986–994
29. Zhang X, Teng X, Pham H (2003) Considering fault removal efficiency in software reliability assessment. IEEE Trans Syst Man Cybern A Syst Hum 33(1):114–120
30. Zhang X, Pham H (2000) Comparison of nonhomogeneous poisson process software reliability models and its applications. Int J Syst Sci 31(9):1115–1123
31. Pham H, Normann L, Zhang X (1999) A general imperfect software debugging model with S-shaped fault detection rate. IEEE Trans Rel 48(2):169–175
32. Hwang S, Pham H (2009) Quasi-Renewal Time-Delay Fault Removal Consideration in Software Reliability Modelling. IEEE Trans Syst Man Cybern A Syst Hum 39(1):200–209
33. Singpurwalla ND, Soyer R (1985) Assessing (Software) reliability growth using a random co-efficient autoregressive process and its ramifications. IEEE Trans Software Eng SE-11:1456–1464
34. Chatterjee S, Misra RB, Alam SS (1997) Prediction of software reliability using an auto regressive process. Int J Syst Sci 28:205–211
35. Walls LA, Bendell A (1987) Time series methods in reliability. Reliab Eng Syst Saf 18:239–265
36. Anselmo V, Ubertini L (1979) Transfer function-noise model applied to flow forecasting. Hydrol Sci 24:353–359

37. Nason GP, Sapatinas T (2002) Wavelet packet transfer function modeling of nonstationary time series. Stat Comput 12(1):45–56
38. Halas M, Kotta U (2007) Transfer Function of discrete-time Nonlinear Controls. Proc Estonian Acad Phys Math 56(4):322–335
39. Singpurwalla ND (1980) Analyzing availability using transfer function models and cross spectral analysis. Nav Res Logist Quart 27:1–16
40. Box GEP, Jenkins GM (1976) Time series analysis, forecasting, and control. Holden-Day, San Francisco
41. Lyu MR (1996) Handbook of software reliability engineering. IEEE Computer Society Press, McGraw Hill, New York
42. Akaike H (1974) A new look at the statistical model identification. IEEE Trans Automat Control 19:716–723