http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 9 (2): 331-336, 2010
ISSN 1812-5638

The Software Reliability Forecasting Method Using Fractals

Yong Cao and Qing-Xin Zhu
Institute of Computer Science and Engineering,
University of Electronic Science and Technology of China

Abstract: A forecasting method of software failure using fractals is proposed in this study. The empirical failure data (three data sets of Musa's and one of NTDS) are used to demonstrate the performance of the reliability forecasting. Compared with other method, our method is very effective. It should be noticed that the analyses and research methods in this study are differ from the conventional methods in the past and a new idea for the research of the software failure mechanism is presented.

Key words: Reliability, forecasting, time series, software failure

INTRODUCTION

Software reliability, namely the capability that a given component or system within a specified environment will operate correctly for a specified period of time, has been one of the most important qualities. Software-reliability models are used for the assessment and improvement of reliability in software systems and failure analysis is an important part of the research of software reliability. The important problem of the software reliability model is to calculate and predict the next failure time in advance. It was treated as random and statistical problem in the past and the modeling methods are usually based on probability statistics. The famous models are Jelinsky-Moranda based on time measurement, Littlewood and Goel-Okumoto etc.

A few papers have used time-series analysis in the construction of Software reliability models, by treating reliability growth data as an auto-regressive process. In literature (Lyu, 1996) a time series of software reliability data was assumed to arise from a power-law process

$$T_i = \delta_i T_{i-1}^{\theta(i)} \quad (1)$$

where, T_i is the time to the ith failure, $\theta(i)$ is an unknown constant and δ_i is a random coefficient. By assuming that the T_is and δ_is are lognormally distributed, we can take the logarithm of both sides and obtain a first-order auto-regressive process.

A more complex time-series model based on an auto-regressive integrated moving average process is presented in literature (Khoshgoftaar and Szabo, 1995). This study also stands out because a small number of software complexity metrics are integrated into the model along with reliability growth data.

Recently, along with the development of statistical theory, Bayesian approach has been applied in forecasting software reliability. El-Aroui and Soler (1996) first employed Bayesian statistical model to predict software reliability. They assumed the successive times between software failures follow the exponential distribution. Pham and Zhang (2003) proposed a software reliability forecasting model based on NHPP approaches. The experimental results indicate that proposed models have better goodness-of-fit than other exist NHPP models. Su and Huang (2006) showed how to apply neural networks to predict software reliability. Further they made use of the neural network approach to build a Dynamic Weighted Combinational Model (DWCM). The study of Kanmani *et al*. (2007) showed the neural network models which were applied for predicting faults in object-oriented software to be performing much better than the statistical methods.

The term fractal, which is a geometrical concept and means broken or irregular fragments, was originally coined by Mandelbrot to describe a family of complex shapes that possess an inherent self-similarity or self-affinity in their geometrical structure. It was generally believed by mathematicians and scientists that such complex natural phenomena were almost beyond rigorous description. Fractals are mathematical or natural objects that are made of parts similar to the whole in some ways. A fractal has a self-similar structure that occurs at different scales. For example, a small branch of a tree looks like the whole tree due to the existence of branching structures.

The word Fractal describes a new system of mathematics and Fractal objects are Self-Similar under some change in scale, either strictly, or statistically. For Strictly Self-Similar Fractals do not change their appearance significantly when viewed under a microscope

Corresponding Author: Yong Cao, No. 2, South Baihua Street, Chengdu City, Sichuan Province, China

of arbitrary magnifying power, whereas for statistically self-similar fractals, when a small portion of it is magnified, results into a fractal, which is seemingly but not exactly similar to the original fractal itself. Fractal objects, by definition, contain infinite datai1 i.e., they contain the same degree of detail in each part as is contained in the entire object, no matter how many times sections of it are enlarged.

Fractals can be applicable in depicting natural complexities, which have been applied to biology, geophysics, solid physics, chemistry and astronomy etc. It is also well known that fractals have been applied to analyze and prediction of some random affairs like earthquakes. They are also used in random movements like random walk or Brownian motion so that we can discover the character of fractional dimension. Qin *et al*. (2006) proposed a burst detection algorithm over data streams through building monotonic search space based on fractals. Theoretical analysis and experimental results show that this algorithm can achieve a higher precision with less space and time complexity as compared with the existing methods. Durga *et al*. (2008) proposed a filter-based feature selection method based on Correlation Fractal Dimension (CFD) discrimination measure and successfully applied the proposed algorithm on a challenging classification problem in bioinformatics.

Software failures can also be treated as random events and fractal can be applied to software failure prediction. We may investigate the fractal relationship between the cumulative time of software failure and the accumulative number of software failure in time series.

POWER LAW IN FRACTALS

A power law is a relationship between two scalar variables x and y, which can be written as follows:

$$y = Cx^k \quad (2)$$

where, C is the constant of proportionality and k is the exponent of the power law. Such a power law relationship shows as a straight line on a log-log plot since, taking logs of both sides, the above equation is equivalent to

$$\log(y) = k\log(x) + \log(C) \quad (3)$$

which has the same form as the equation for a straight line

$$Y = kX + c \quad (4)$$

The equation $f(x) = Cx^k$ has a property that relative scale change $f(sx)/f(x) = s^k$ is independent of x. In this sense, f (x) is scale invariant or lacks a characteristic scale. Consequently, f (x) can be related to fractals because of its scale invariance. The k is called fractal dimension. For strictly self-similar fractals, two scalar variables x and y fit power law strictly described as Eq. 2, for statistically self-similar fractals, the x and y fit power law statistically and fractal dimension k is slope of linear regression on a log-log plot described as Eq. 2.

THE SOFTWARE RELIABILITY FORECASTING METHOD USING FRACTALS

Time series are able to characterize the prediction of software systems. The ith software failure can be described as pair (the ith failure, the ith failure time), namely the accumulative number of software failure and the cumulative time of corresponding software failure.

Fractals can be measured by dimensional measures, such as the Hausdorff dimension etc. The fractal dimension is often noninteger and smaller than the embedding topological dimension. Previously we always adopt scale variation method to analyze fractal dimension of data such as earthquakes etc. We select time section t as scale ϵ, divide the time section into some time subsections and take count of $N(\epsilon)$ which is the number of subsections the earthquake happen. Then we change the time section ϵ to obtain a new $N(\epsilon)$ and we repeat the above steps to obtain a series of ϵ-$N(\epsilon)$. We regard a ϵ-$N(\epsilon)$ pair of the series as a point in log-log coordinate and draw a log ϵ-log $N(\epsilon)$ graph to analyze the data. This method is inapplicable to analyzing the data of software failure time for its key is prediction of next failure time whereas the next failure time is non- determinate. There will make a lot of waste if we adopt scale variation method to analyze the data of software failure. According to Eq. 3 let $y = N(r) = i$ which is the accumulative number of software failure and $x = T_i$ which is the cumulative time of the ith software failure. The formula will be described as:

$$\log(T_i) = \tfrac{1}{k}\log(i) - \tfrac{1}{k}\log(C) \quad (5)$$

Let d = 1/k. The Eq. 5 can be transformed into:

$$\log(T_i) = d\log(i) - d\log(C) \quad (6)$$

$$T_i = si^d \quad (7)$$

where, let $s = C^{-d}$. When software failures happen, if there exists fractal relationship between the cumulative time of software failure and the accumulative number of software failure, the time fractal dimension will be computed.

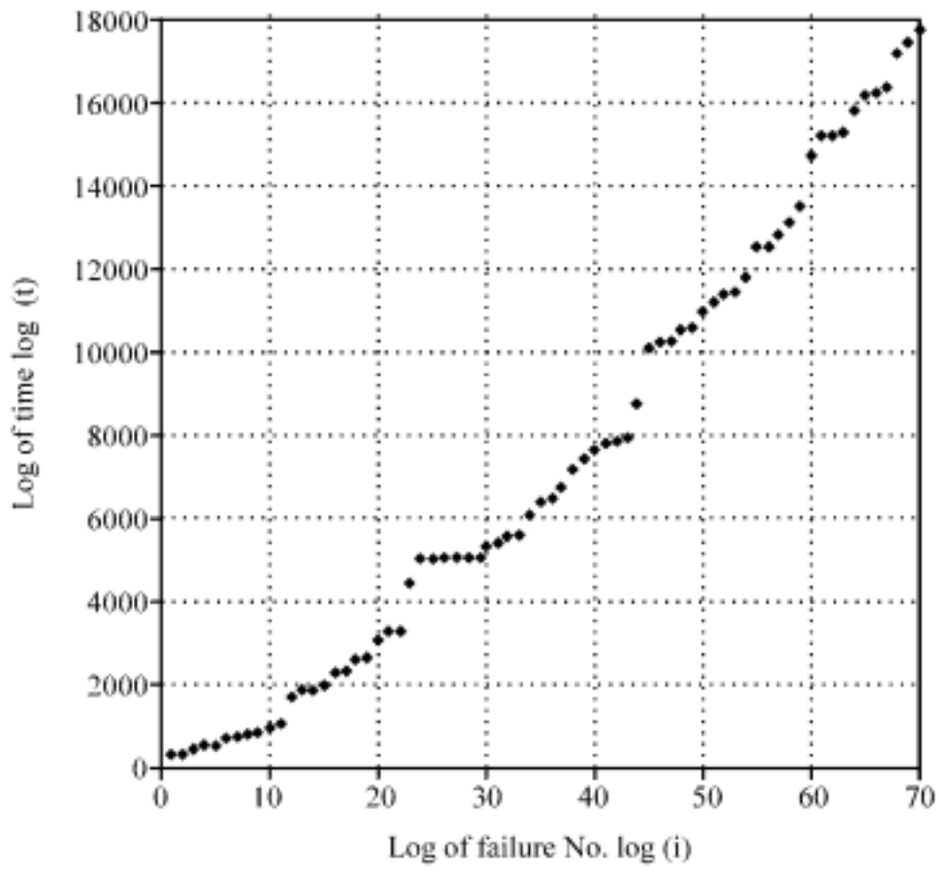

Fig. 1: The double log coordinates of the cumulative time of software failure and the accumulative number of software failure log t -log i of Musa's failure data set 1

$$d = \frac{\log(t_{ij}) - \log(t_{ik})}{\log(i_j) - \log(i_k)} \quad (8)$$

where, i_j, i_k denote the ordinal number of software failure and t_{ij}, t_{ik} denote the corresponding failure time.

At first, we compute double log coordinates of the cumulative time of software failure and the accumulative number of software failure log t-log i in Musa's data set 1 (Musa *et al*., 1987), namely Appendix 1 (The results shown Fig. 1). The slope d of each beeline connects point pairs (3, 20), (3, 21), (3, 22),… (3, 80) is between 1.52-0.07 and 1.52+0.07. All points are almost in a beeline, where, d = 1/k and k is fractal dimension. It is observed easily that the there exists fractal relationship between the cumulative time of software failure and the accumulative number of software failure in part. Whether the relationship always exists need further experiments.

Prediction of scalar time-series {x (n)} refers to the task of finding an estimate $\overline{x}(n+1)$ of the next future sample x (n + 1) based on the knowledge of the history of the time-series. Introducing a general nonlinear function f (.): $R^N \rightarrow R$ applied to the vector X (n) = [x (n), x (n-1),..., x (N-1))]^T of past samples, we arrive at the nonlinear forecasting approach:

$$\overline{x}(n + 1) = f(X(n)) \quad (9)$$

In this way, we adopt method of sliding window to compute. The prediction algorithm of sliding-window fractal model of software failure will be described as follow:

Algorithm 1:

Begin

Initialization: suppose the size of slide window m, k=1 and A is a array of the number of failure corresponding failure time;

Repeat for i=k to m+k-1 {

B(i)=log(A(i));/*the logarithm of practical failure time in the slide window.*/

C(i)=log(i);/*the logarithm of failure number in the slide window.*/

}

- According to Eq. 5 and method of linear regression, compute the slope of linear regression in the slide window b = d = 1/k and constant a = log(s) = -dlog (C)
- Make a prediction of next point out of the slide window using the above coefficients b and a
- Add the practical failure time of the next point to A; k++; /*the slide window move backwards.*/

Until test over

End

After software failure happens, maintenance personnel will repair software system, correct mistake and reliability of software will be changed. The fractal dimension k will change and d will also change. Apparently in the sliding window the double log coordinates of the cumulate time of software failure and the accumulative number of software failure log t-log i preserve linearity well. Although with the sliding of window the slope that we use linear regression to obtain will change, the linearity still exists. When window slides, the points in the anterior window and the points in the posterior window may not be in a beeline approximately, but the points in same window are in a beeline approximately. We can see that forecasting data almost fits actual data in Fig. 2. Next we will compare our method with other methods through experiments.

RESULTS

The forecasting algorithm and a one-step-ahead forecasting policy are applied in four examples. The software failure data are obtained from Musa's data set 1 (Appendix 1), Musa's data set 2 (Appendix 2), Musa's data set 3 (Appendix 3) and NTDS data set (Appendix 4) (Musa *et al*., 1987; Singpurwalla and Soyer, 1985; Jelinsky and Moranda, 1972). The performance of

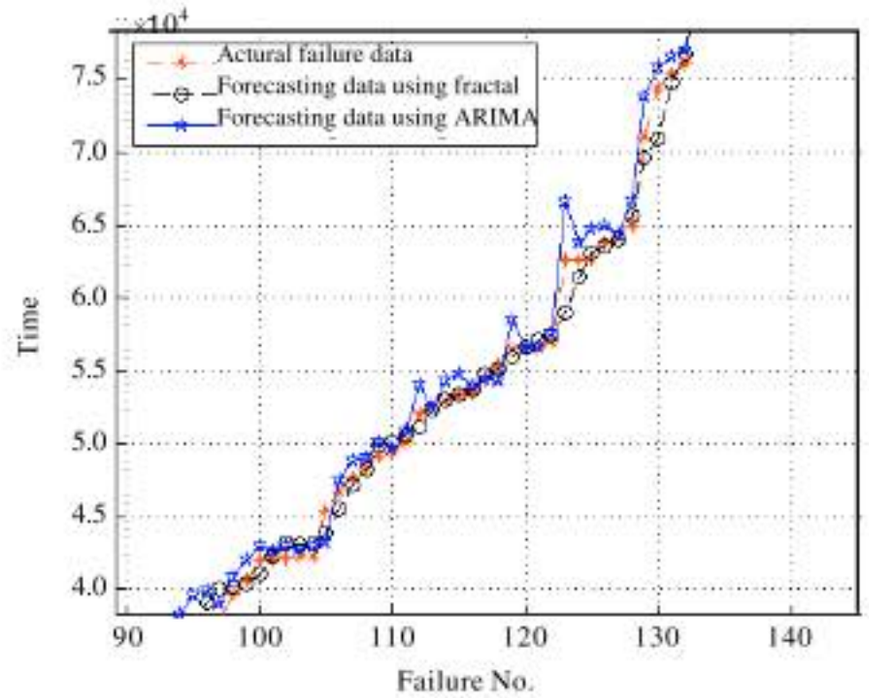

Fig. 2: Forecasting results of different models (ARIMA and Fractal) of Musa's data set 1 (Appendix 1) of software failure (Sliding window size m = 5)

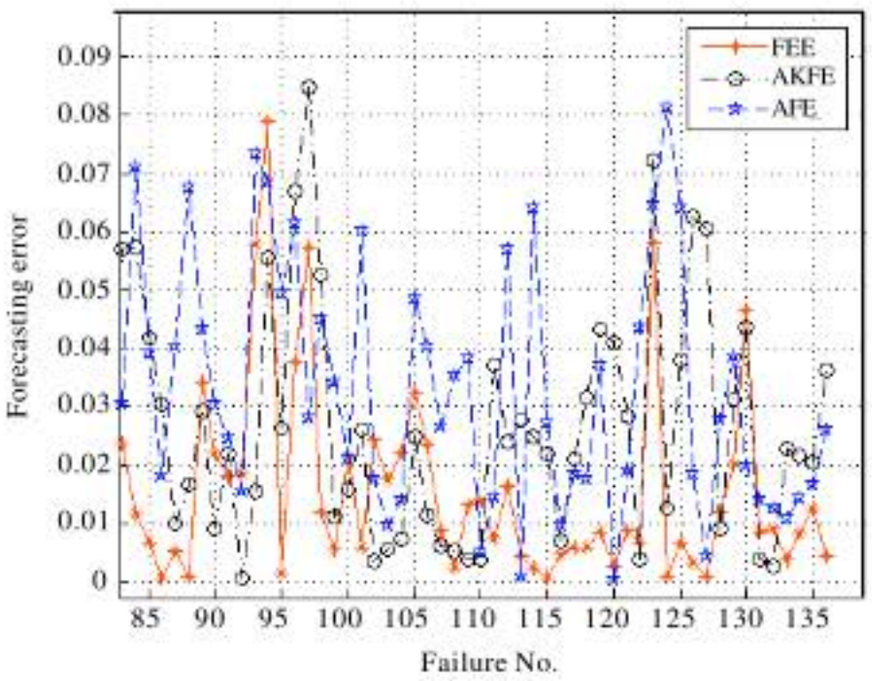

Fig. 3: Forecasting error of different models (Adaptive Kalman, ARIMA and Fractal) of Musa's data set 1 (Appendix 2) of software failure (FFE stands for Fractal Forecasting Error, AKFE stands for Adaptive Kalman Forecasting Error and AFE stands for ARIMA Forecasting Error)

the proposed model is compared with normal distribution (Singpurwalla and Soyer, 1985), Kalman filter (Singpurwalla and Soyer, 1985), adaptive Kalman filter (Singpurwalla and Soyer, 1985) and ARIMA (Khoshgoftaar and Szabo, 1995; Abdullah *et al*., 2008) forecasting methods. The experimental results are shown in Fig. 2-5, Table 1 and 2. In Fig. 2 the forecasting data using fractals fit actual data better than ARIMA; in Fig. 3 85% of the forecasting error using fractals is less than 3%, which is better than ARIMA (58%) and Adaptive Kalman filter (73%); in Fig. 4, 82% of the forecasting error using fractals is less than 6%, which is better

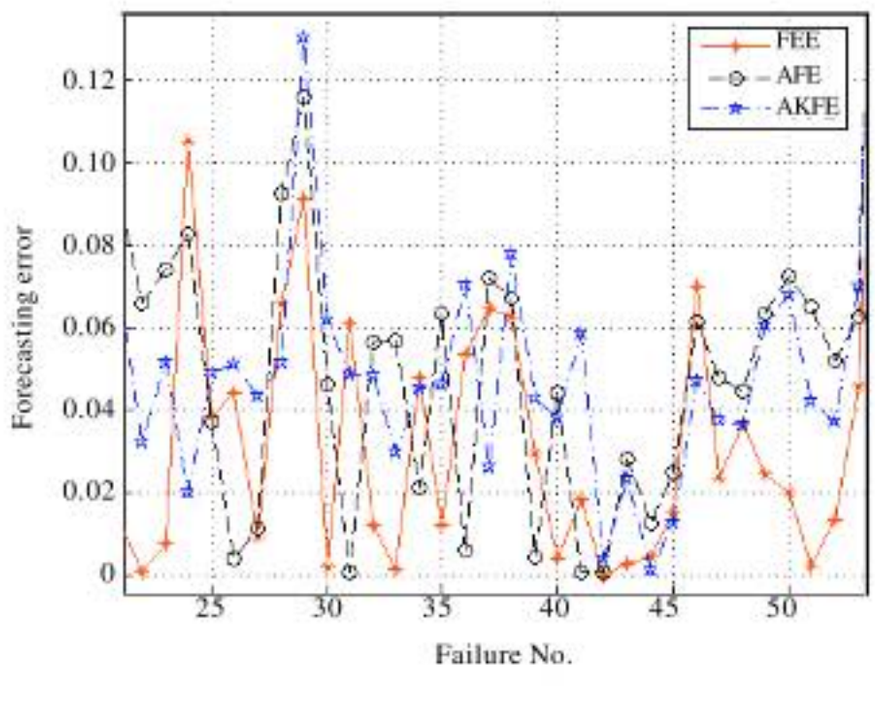

Fig. 4: Forecasting error of different models (Adaptive Kalman, ARIMA and Fractal) of Musa's data set 2 (Appendix 3) of software failure (Sliding window size m = 3. FFE stands for Fractal Forecasting Error, AKFE stands for Adaptive Kalman Forecasting Error and AFE stands for ARIMA Forecasting Error)

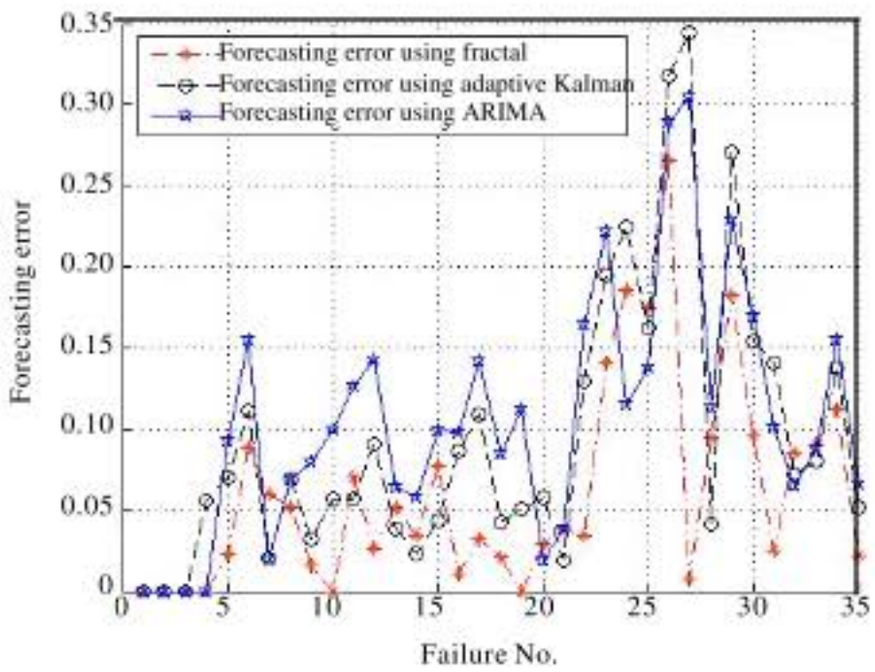

Fig. 5: Forecasting error of different models (Adaptive Kalman, ARIMA and Fractal) of NTDS data set (Appendix 4) of software failure (sliding window size m = 3)

Table 1: Forecasting results of different models of Musa's data set 1, 2 and NTDS data set

Data set	Error	Fractal	ARIMA	Kalman	Adaptive Kalman
Musa 1	MAE	0.0271	0.0432	0.0474	0.0425
	NRMSE	0.0312	0.0493	0.0541	0.0481
Musa 2	MAE	0.0574	0.0718	0.0687	0.0635
	NRMSE	0.0645	0.0824	0.0764	0.0702
NTDS	MAE	0.0625	0.1019	0.1034	0.0947
	NRMSE	0.0939	0.1349	0.1386	0.1203

than ARIMA (62%) and Adaptive Kalman filter (69%); in Fig. 5, 79% of the forecasting error using fractals is less

than 8%, which is better than ARIMA (48%) and Adaptive Kalman filter (65%). Obviously our method is effective. In the investigation, the values of mean absolute error:

Table 2: Frocasting results of different models of Musa's data set 3. Model I stands for normal distribution, Model II stands for Kalman filter and Model III stands for adaptive Kalman filter

No.	Actual data	Fractal	Model I	Model II	Model III
41	6.4615	6.7299	6.6196	5.8483	4.4709
42	7.6955	6.3346	6.6162	5.6402	5.9634
43	4.7005	4.8812	8.1919	8.8508	9.1343
44	10.0024	7.7381	4.7904	4.2833	2.5915
45	11.0129	10.3264	10.4177	13.3746	19.3881
46	10.8621	10.2448	11.4835	17.5081	11.0904
47	9.4372	10.0509	11.3140	11.3749	9.7274
48	6.6644	10.1561	9.7977	8.7832	7.4402
49	9.2294	10.3601	6.8764	5.2614	4.2686
50	8.9671	8.0196	9.5838	9.6011	11.5589
51	10.3534	9.6292	9.3004	10.5107	7.9377
52	10.0998	9.2025	10.7603	11.0118	10.8762
53	12.6078	9.3609	10.4852	10.7619	8.9795
54	7.1546	11.1223	13.1355	14.0128	14.3821
55	10.0033	11.0916	7.3952	6.5014	3.6967
56	9.8601	10.8053	10.4011	9.8018	12.6137
57	7.8675	9.7905	10.2433	11.7064	8.8370
58	10.5757	9.1107	8.1421	7.0314	5.6897
59	10.9294	9.7894	10.9974	11.3105	12.8783
60	10.6604	9.5571	11.3641	12.9563	10.2797
61	12.4972	11.5998	11.0736	10.7183	9.4502
62	11.3745	12.0479	13.0074	13.4218	13.3468
63	11.9158	12.6449	11.8162	11.8412	9.4337
64	9.575	12.2779	12.3801	11.6690	11.3776
65	10.4504	11.1209	9.9147	8.8680	7.0159
66	10.5866	10.4845	10.8297	9.8988	10.3937
67	12.7201	11.0893	10.9673	11.1270	9.8073
68	12.5982	11.7634	13.2073	14.0726	14.0040
69	12.0859	12.1176	13.0723	13.8034	11.4408
70	12.2766	12.6686	12.5277	11.7916	10.6270
71	11.9602	12.5548	12.7219	12.1274	11.4427
72	12.0246	12.9797	12.384	11.8994	10.7036
73	9.2873	11.4549	12.4459	11.9021	11.1151
74	12.495	11.1424	9.5809	8.2618	6.5952
75	14.5569	12.8428	12.9403	13.2075	15.4497
76	13.3279	12.7283	15.0996	18.2291	15.6550
77	8.9464	12.4730	13.8044	13.8709	11.2419
78	14.7824	11.7018	9.2257	7.1144	5.5249
79	14.8969	13.4620	15.3558	17.1115	22.4572
80	12.1399	14.9886	15.4692	19.7401	13.8009
81	9.7981	12.1403	12.5737	11.0814	9.0674
82	12.0907	10.0515	10.1218	7.3172	7.2440
83	13.0977	12.6443	12.5184	12.3172	13.6803
84	13.368	12.6054	13.5683	15.1443	13.0569
85	12.7206	11.5435	13.8459	14.0649	12.5528
86	14.192	12.3536	13.1633	12.5523	11.1382
87	11.3704	13.4333	14.6987	14.6675	14.5875
88	12.2021	13.7562	11.7469	10.8997	8.3918
89	12.2793	12.4227	12.6114	11.4370	12.0519
90	11.3667	11.8333	12.6876	12.7591	11.4003
91	11.3923	11.1775	11.7316	10.9819	9.7078
92	14.4113	10.9153	11.7545	10.9845	10.5370
93	8.3333	12.3353	14.9045	16.3186	16.8667
94	8.0709	10.8643	8.5791	7.6817	4.4545
95	12.2021	10.6880	8.3040	6.2499	7.1640
96	12.7831	9.6814	12.6137	15.0614	16.0111
97	13.1585	11.0842	13.2161	16.2962	12.3861
98	12.753	12.240	13.6036	13.6828	12.5033
99	10.3533	12.9568	13.1763	12.7557	11.4137
100	12.4897	13.6137	10.6746	9.2267	7.7609
MAEIT	0	0.1316	0.1734	0.2354	0.2578
MAE	0	0.0171	0.0260	0.0539	0.0464

$$MAE = \frac{1}{n}\sum_{i=1}^{n} abs\frac{T_i - \overline{T_i}}{T_i} \quad (10)$$

Normal root mean square error

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(T_i - \overline{T_i})^2}{\sum_{i=1}^{n} T_i^2}} \quad (11)$$

where, T_i is the ith actual failure time and T_i is prediction time (Table 1). Similarly, mean absolute error of interval time

$$MAEIT = \frac{1}{n}\sum_{i=1}^{n} abs\frac{d_i - \overline{d_i}}{d_i} \quad (12)$$

where, n is the number of prediction periods, d_i is actual value of period i (actual interval time between the ith failure and (i+1)th failure) and $\overline{d_i}$ is prediction value (Table 2).

CONCLUSION

Reliability is one of the most important qualities of software and the important problem of the software reliability model is to calculate and predict the next failure time in advance. This study analyzes the empirical failure data, discovers the fractal relationship between the cumulative time of software failure and the accumulative number of software failure and proposes the software reliability method using fractals to predict the next software failure time which almost fit the actual failure time. The analyses of empirical data prove our method effective in comparison with other methods. We will research the mechanism behind fractals further in future research.

ACKNOWLEDGMENTS

The study described in this research was supported by a grant from the National Science Foundation of China (NSFC) (60671033) and the Doctoral Foundation of the Ministry of Education of China (2006061405).

APPENDIX

Appendix 1: The Musa's data set 1 of failure time series and from left to right the time in each cell denotes the cumulative time of the ith software failure, i = 1, 2, ...

3	33	146	227	342
351	353	444	556	571
709	759	836	860	968
1056	1726	1846	1872	1986
2311	2366	2608	2676	3098
3278	3288	4434	5034	5049
5085	5089	5089	5097	5324

Appendix 1: Continued

5389	5565	5623	6080	6380
6477	6740	7192	7447	7644
7837	7843	7922	8738	10089
10237	10258	10491	10625	10982
11175	11411	11442	11811	12559
12559	12791	13121	13486	14708
15251	15261	15277	15806	16185
16229	16358	17168	17458	17758
18287	18568	18728	19556	20567
21012	21308	23063	24127	25910
26770	27753	28460	28493	29361
30085	32408	35338	36799	37642
37654	37915	39715	40580	42015
42045	42188	42296	42296	45406
46653	47596	48296	49171	49416
50145	52042	52489	52875	53321
53443	54433	55381	56463	56485
56560	57042	62551	62651	62661
63732	64103	64893	71043	74364
75409	76057	81542	82702	84566
88682				

Appendix 2: The Musa's data set 2 of software failure time series, and from left to right time in each cell denotes interval time between the ith failure and the (i+1)th failure, i = 1, 2, …….

320	1439	9000	2880
5700	21800	26800	113540
112137	660	2700	28493
2173	7263	10865	4230
8460	14805	11844	5361
6553	6499	3124	51323
17010	1890	5400	62313
24826	26335	363	13989
15058	32377	41632	4160
82040	13189	3426	5833
640	640	2880	110
22080	60654	52163	12546
784	10193	7841	31365
24313	298890	1280	22099
19150	2611	39170	55794
42632	267600	87074	149606
14400	34560	39600	334395
296015	177395	214622	156400
166800	10800	267000	

Appendix 3: The Musa's data set 3 of software failure time series, and from left to right time in each cell denotes interval time between the ith failure and the (i+1)th failure, i = 1, 2, …… .

5.7683	9.5743	9.1050	7.9655	8.6482	9.9887
10.1962	11.6399	11.6275	6.4922	7.9010	10.2679
7.6839	8.8905	9.2933	8.3499	9.0431	9.6027
9.3736	8.5869	8.7877	8.7794	8.0469	10.8459
8.7416	7.5443	8.5941	11.0399	10.1196	10.1786
5.8944	9.546	9.6197	10.3852	10.6301	8.3333
11.315	9.4871	8.1391	8.6713	6.4615	6.4615
7.6955	4.7005	10.0024	11.0129	10.8621	9.4372
6.6644	9.2294	8.9671	10.3534	10.0998	12.6078
7.1546	10.0033	9.8601	7.8675	10.5757	10.9294
10.6604	12.4972	11.3745	11.9158	9.5750	10.4504
10.5866	12.7201	12.5982	12.0859	12.2766	11.9602
12.0246	9.2873	12.4950	14.5569	13.3279	8.9464
14.7824	14.8969	12.1399	9.7981	12.0907	13.0977
13.368	12.7206	14.1920	11.3704	12.2021	12.2793
11.3667	11.3923	14.4113	8.3333	8.0709	12.2021
12.7831	13.1585	12.7530	10.3533	12.4897	

Appendix 4: The NTDS data set of software failure time series and from left to right the time in each cell denotes the cumulative time of the ith software failure, i = 1, 2, …….

9	21	32	36	43
45	50	58	63	70
71	77	78	87	91
92	92	95	98	104
105	116	149	156	247
249	250	337	384	396
405	540	798	814	849

REFERENCES

Abdullah, S., M.D. Ibrahim, A. Zaharim and Z.M. Nopiah, 2008. Statistical analysis of a nonstationary fatigue data using the ARIMA approach. WSEAS Trans. Math., 7: 59-66.

Durga, B.S., Sobha, T. Rani and R.S. Bapi, 2008. Feature selection using correlation fractal dimension: Issues and applications in binary classification problems. J. Applied Soft Comput., 8: 555-563.

El-Aroui, M.A. and J.L. Soler, 1996. A Bayes nonparametric framework for software-reliability analysis. IEEE Trans. Reliab., 45: 652-660.

Jelinsky, Z. and P. Moranda, 1972. Software Reliability Research. In: Statistica Computer Performance EvaZuation, Freiberger, W. (Ed.). Academic Press, New York, ISBN: 981-02-0640-2, pp: 465-484.

Kanmani, S., V.R. Uthariaraj, V. Sankaranarayanan and P. Thambidurai, 2007. Object-oriented software failure fault prediction using neural networks. J. Inform. Softw. Technol., 49: 483-492.

Khoshgoftaar, T.M. and R.M. Szabo, 1995. Investigating ARIMA models of software system quality. Softw. Qual. J., 4: 33-48.

Lyu, M.R., 1996. Handbook of Software Reliability Engineering. McGraw-Hill and IEEE Computer Society, New York, ISBN: 0-07-039400-8.

Musa, J.D., A. Iannino and K. Okumoto, 1987. Software Reliability-Measurement, Prediction, Applications. McGraw-Hill Book Company, New York, ISBN: 0-07-044093-X.

Pham, H. and X. Zhang, 2003. NHPP software reliability and cost models with testing coverage. Eur. J. Opert. Res., 145: 443-454.

Qin, S.K., W.N. Qian and A.Y. Zhou, 2006. Fractal based algorithms for burst detection over data streams. J. Softw., 17: 1969-1979.

Singpurwalla, N.D. and R. Soyer, 1985. Assessing (software) reliability growth using a random coefficient autoregressive process and its ramifications. IEEE Trans. Softw. Eng., SE-11: 1456-1464.

Su, Y.S. and C.Y. Huang, 2006. Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. J. Inform. Softw. Technol., 80: 606-615.