IEICE TRANS. INF. & SYST., VOL.E93–D, NO.2 FEBRUARY 2010

LETTER

The Software Reliability Model Based on Fractals

Yong CAO[†a)], ***Student Member and*** **Qingxin ZHU**[†], ***Nonmember***

SUMMARY Fractals are mathematical or natural objects that are made of parts similar to the whole in certain ways. In this paper a software reliability forecasting method of software failure is proposed based on predictability of fractal time series. The empirical failure data (three data sets of Musa's) are used to demonstrate the performance of the reliability prediction. Compared with other methods, our method is effective.

key words: *reliability, fractals, prediction, software failure*

1. 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 requirements. The important problem of the software reliability models is to calculate and predict the next failure time in advance. It was mainly treated as random and statistical problem. The Jelinski-Moranda model is based on time measurement and maximum likeness estimation. The Kalman filter is an efficient recursive filter that estimates the state of a linear dynamic system from a series of noisy measurements. The ARIMA models are the most general class of models for forecasting a time series which can be stationarized by transformations such as differencing and logging. Recently Bayesian networks, recurrent neural networks, and support vector machine are applied in time series analysis, which afford good results.

The term fractal, which 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 belongs to geometrical category. A fractal has a self-similar structure that occurs at different scales. Fractal objects, by definition, contain infinite detail, i.e., they contain the same degree of detail in each part as is contained in the entire object, no matter how many times its sections are enlarged. For example, a small branch of a tree looks like the whole tree due to the existence of branching structures. Fractal objects are self-similar under some changes in scale, either strictly or statistically.

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

$$y = Cx^k \tag{1}$$

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

$$\log(y) = k\log(x) + \log C \tag{2}$$

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

$$Y = kX + C \tag{3}$$

The equation $f(x) = C, x^k$ has a property that relative scale change $f(sx)/f(x)$ is independent of x. In this sense, $f(x)$ is scale invariant or lacks a characteristic scale. Consequently, $f(x)$ can be related to fractal because of its scale invariance. The k is called Fractal Dimension.

For Strictly Self-Similar Fractal, two scalar variables x and y fit power law strictly described as Eq. (2); for Statistically Self-Similar Fractal, 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).

Fractals have been applied to analyze some random events like earthquakes, random walks and Brownian motions etc. Hughes et al. [1] has discovered that there exists interrelationship between random events and fractal. Software failures are also random events and the key of studying fractal is self-similarity. If self-similarity exists in time series then we may investigate the relationship between software failures and fractal.

The outline of this paper is the following: Section 2 presents sliding-window fractal forecasting model of software failure and validates the model through analyzing the empirical failure data. Section 3 concludes this paper and describes the future research.

2. The Software Failure Prediction Based on Fractals

Fractals can be characterized 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 $N(\varepsilon)$ which is the number of subsections the earthquake happen. Then we change the time section ε to obtain a new $N(\varepsilon)$ and we repeat the above steps to obtain a series of $\varepsilon - N(\varepsilon)$ pairs. We regard a $\varepsilon - N(\varepsilon)$ pair of the series as

Manuscript received September 15, 2009.
Manuscript revised October 15, 2009.

[†]The authors are with University of Electronic Science and Technology of China, China.
a) E-mail: cn_caoyong@126.com

DOI: 10.1587/transinf.E93.D.376

a point in log – log coordinates and draw a $\log\varepsilon - \log N(\varepsilon)$ graph to analyze the data. This method is inapplicable to analyzing the data of software failure time because there will make a lot of waste if we adopt "scale variation" to analyze the data of software failure. According to Eq. (2), 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:

$$\log(T_i) = \frac{1}{k}\log(i) - \frac{1}{k}\log(C) \tag{4}$$

Let $d = \frac{1}{k}$.The formula (4) can be transformed into:

$$\log(T_i) = d\log(i) - d\log(C) \tag{5}$$

$$T_i = st^d \tag{6}$$

where let $s = C^{-d}$. When software failures happen, the time fractal dimension k will be computed.

$$k = \frac{\log(i_m) - \log(i_n)}{\log(t_{i_m}) - \log(t_{i_n})} \tag{7}$$

where i_m, i_n denote the ordinal number of software failure and t_{i_m}, t_{i_n} denote corresponding failure time.

At first, we compute double log coordinates of the cumulate time of software failure and the accumulative number of software failure $\log t - \log i$ in Musa's data set 1, namely Table A·1 (The results see Fig. 1) [2]. The slope k (fractal dimension) of each beeline connects point pairs $(3, 20)$, $(3, 21)$, $(3, 22)$, …, $(3, 80)$ is between $0.65 - 0.03$ and $0.65 + 0.03$. All points are almost in a beeline, which implies there exist good self-similarity in time series and fractal relationship between the cumulate time of software failure and the accumulative number of software failure.

After software failure happens, maintenance personnel will repair software system, correct mistakes and software reliability will be changed. The fractal dimension k will change and d will also change.

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 $X(n) = [x(n), x(n-1), \ldots, x(n-(N-1))]^T$ of past samples, we arrive at the nonlinear prediction approach:

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

In this way, We adopt method of sliding-window to compute. The algorithm will be described as follow:

Algorithm 1:

Initialization: Suppose the size of sliding-window m, $l = 1$ and A is a array of *ith* failure time;

for $i = l$ to $m + l - 1$ $\{B(i) = \log(A(i))$;/*the logarithm of actual failure time in the sliding-window.*/ $C(i) = \log(i)$;/*the logarithm of failure number in the sliding-window.*/$\}$
Repeat

(1) According to Eq. (5) and method of linear regression, compute the slope of linear regression in the sliding-window $b = d = \frac{1}{k}$ and constant $a = \log(s) = -d\log(C)$;

(2) Make a prediction of next point out of the sliding-window using the above a and b.

(3) Add the actual failure time of the next point to A;

$l++$; /*the sliding-window move backwards.*/

$B(m + l - 1) = \log(A(m + l - 1))$;

$C(m + l - 1) = \log(m + l - 1)$;

Until test over
End

The forecasting algorithm and a one-step-ahead forecasting policy are applied in three examples. The software failure data are obtained from Musa's data set 1, 2, and 3 (Table A·1, Table A·2 and Table A·3) [2], [3]. The performance of the proposed model is compared with the normal distribution [3], Kalman filter [3], adaptive Kalman filter [3], and ARIMA [4], [5] forecasting methods. The experimental results are shown in Fig. 2, Fig. 3, Table 1 and Table A·4. In the investigation, the values of Mean Absolute Error $MAE = \frac{1}{n}\sum_{i=1}^{n} abs\frac{T_i - \overline{T_i}}{T_i}$, Normal Root Mean Square Error $NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(T_i - \overline{T_i})^2}{\sum_{i=1}^{n} T_i^2}}$, where T_i is the *ith* actual failure time

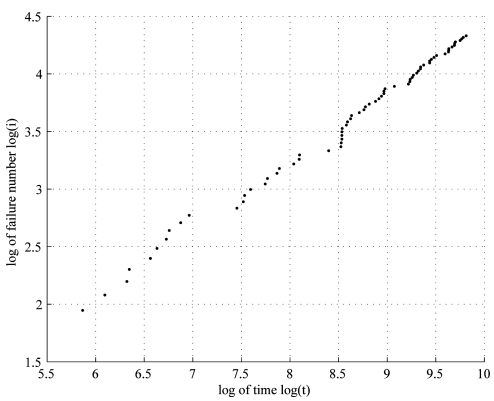

Fig. 1 The double logarithmic $\log t$ -$\log i$ of Musa's failure data set 1.

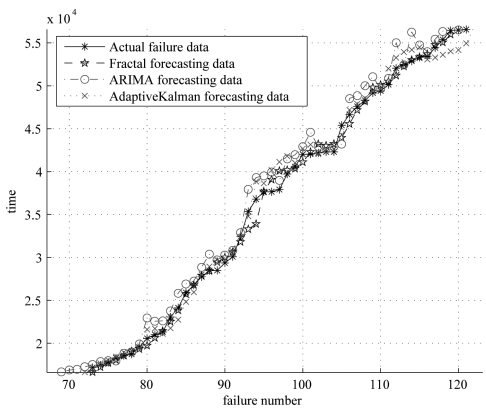

Fig. 2 Forecasting results of different models of Musa's data set 1 (sliding-window size $m = 5$).

and $\overline{T_i}$ is forecasting time (Table 1). Similarly, Mean Absolute Error of Interval Time $MAEIT = \frac{1}{n}\sum_{i=1}^{n}\frac{abs(p_i-\overline{p_i})}{p_i}$, where n is the number of forecasting periods, p_i is actual value of period i (actual interval time between the $(i-1)th$ failure and ith failure) and $\overline{p_i}$ is forecasting value (Table A·4).

We adopt different sliding-window sizes to compute forecasting values. When the exponential d of fractal model is steady, namely hardly change, the greater the size of the sliding-window the better the linearity and the predictability becomes better. After maintenance personnel fix the software system, d may change sharply. At this time the less size of sliding-window will keep up with the change of the exponential d. When sliding-window size m = 5, 3, and 9, fractal model provides the smallest *MAE* and *NRMSE* values in Musa set 1, 2, and 3 respectively. Good self-similarity is the key of fractal forecasting accuracy. We can make better prediction through the similarity between parts and whole. We have discovered the good self-similarity of time series in Musa set 1. The self-similarities are also discovered in Musa set 2 and 3, which lead to the better goodness-of-fit than other methods (Table 1 and Table A·4). The exponential d of fractal model can be used to evaluate and characterize software quality. As the maintenance personnel continue to improve software system, d increases. Software system will operate correctly for longer time, namely the software quality becomes better. Therefore positive correlation exists between d and software quality.

3. Conclusion

This paper proposes the software reliability model based on fractals to forecasting the next software failure time which almost fit the actual failure time. Studying the empirical data (three data sets of Musa's) and comparison with other models validate our method. Our model is different from existing forecasting approaches for it is based on geometrical notion and method. We will research the mechanism behind fractals further in future work.

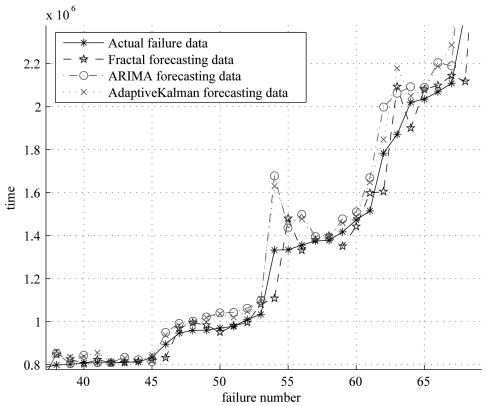

Fig. 3 Forecasting results of different models of Musa's data set 2 (sliding-window size m = 3).

Table 1 Forecasting results of different models of Musa's data set 1 and 2.

	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

References

[1] B.D. Hughes, E.W. Montroll, and M.F. Shlesinger, "Fractal random walks," J. Statistical Physics, vol.28, no.1, May 1982.
[2] J.D. Musa, Software Reliability Data, technical report available from Data Analysis Center for Software, Rome Air Development Center, New York, 1979.
[3] N.D. Singpurwalla and R. Soyer, "Assessing (software) reliability growth using a random coefficient autoregressive process and its ramifications," IEEE Trans. Softw. Eng., vol.SE-11, no.12, pp.1456–1464, 1985.
[4] T.M. Khoshgoftaar and R.M. Szabo, "Investigating ARIMA models of software system quality," Softw. Qual. J., vol.4, no.1, pp.33–48, March 1995.
[5] L. Zhang, Study of the Time Series Forecasting Algorthim and System Realization Based on the ARIMA Model, Master's thesis, Jiangsu University, July 2008.

Appendix

Table A· 1 The Musa's data set 1 of software failure time series, and from left to right the time in each cell denotes the cumulate time of the ith software failure, $i = 1, 2, \ldots$ [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	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

Table A· 2 The Musa's data set 2 of software failure time series, and from left to right time in each cell denotes the interval between the $(i-1)th$ failure and the ith failure, $i = 1, 2, \ldots$ [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		

Table A· 3 The Musa's data set 3 of software failure time series, and from left to right time in each cell denotes the interval between the $(i-1)th$ failure and the ith failure, $i = 1, 2, \ldots$ [2].

5.7683	9.5743	9.105	7.9655	8.6482	9.9887
10.1962	11.6399	11.6275	6.4922	7.901	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	9.575	10.4504
10.5866	12.7201	12.5982	12.0859	12.2766	11.9602
12.0246	9.2873	12.495	14.5569	13.3279	8.9464
14.7824	14.8969	12.1399	9.7981	12.0907	13.0977
13.368	12.7206	14.192	11.3704	12.2021	12.2793
11.3667	11.3923	14.4113	8.3333	8.0709	12.2021
12.7831	13.1585	12.753	10.3533	12.4897	

Table A· 4 Forecasting results of different models of Musa's data set 3. Model I stands for the normal distribution, Model II stands for the Kalman filter and Model III stands for the adaptive Kalman filter [2], [3].

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