LETTER

The Software Reliability Model Using Hybrid Model of Fractals and ARIMA

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

SUMMARY The software reliability is the ability of the software to perform its required function under stated conditions for a stated period of time. In this paper, a hybrid methodology that combines both ARIMA and fractal models is proposed to take advantage of unique strength of ARIMA and fractal in linear and nonlinear modeling. Based on the experiments performed on the software reliability data obtained from literatures, it is observed that our method is effective through comparison with other methods and a new idea for the research of the software failure mechanism is presented.

key words: *software reliability forecasting, fractal, ARIMA models, software failures*

1. Introduction

Software reliability is going to become a highly visible and important field. Therefore, software reliability forecasting is a problem of increasing importance for many critical applications and failure analysis is an important part of the research of software reliability. An underlying assumption of these models is that software failures occur randomly in time. It was mainly treated as random and statistical problem.

Recently, along with the development of prediction theory, support vector machines and artificial neural networks have been applied in forecasting software reliability [1], [2]. Some forecasting techniques have been developed, each one with its particular advantages and disadvantages compared to other approaches. This motivates the study of hybrid model combining different techniques and their respective strengths. Different forecasting models can complement each other in capturing patterns of data sets, and both theatrical and empirical studies have concluded that a combination of forecast outperforms individual forecasting models [3], [4], [7]. Terui and Dijk [4] presented a linear and nonlinear time series model for forecasting the US monthly employment rate and production indices. Their results demonstrated that the combined forecasts out-performed the individual forecasts.

Time series data often contain both linear and nonlinear patterns. The ARIMA (autoregressive integrated moving average processes) models are the most general class of models for forecasting a time series which can be stationarized by transformations such as differencing and logging. When modeling linear and stationary time series, the researcher frequently chooses ARIMA models because of their high performance and robustness. Fractal model has good performance in nonlinear patterns which brought good effect [5]. A hybrid ARIMA and fractal model is capable of exploiting the their strengths respectively.

Manuscript received June 4, 2010.

Manuscript revised June 29, 2010.

[†]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.3116

2. Hybrid Model in Prediction of Software Failure

2.1 Fractal Model

The term fractal, which means broken or irregular fragments. Fractals are mathematical or natural objects that are made of parts similar to the whole in certain ways. It belongs to geometrical category. Time series also follow the laws of fractal geometry. According to [5] self-similarity exists in time series and we may investigate the relationship between software failures and fractal. Cao and Zhu have applied it to forecasting software failures and provided software prediction model based on fractal. Please see [5] for a detailed exposition.

2.2 Run Test

The runs test can be used to decide if a data set is from a stationary random process. A run is defined as a series of positive values or negative values. The number of positive or negative values is the length of the run. In a stationary random data set, the probability that the *ith* value Y_i is larger or smaller than the mean value follows a binomial distribution, which forms the basis of the runs test. The first step in the runs test is to compute the sequential differences $Y_i - \overline{Y}$. Positive values indicate the difference values greater than or Equal to 0 and negative values indicate the difference values less than 0. Let N_1 = (the number of positive values) and N_2 = (the number of negative values). $N = N_1 + N_2$ and U_N represents the total number of runs. We adopt hypothesis test to decide if the time series is stationary.

Hypothesize: H_0 : $\{y_t, t = 1, 2, \ldots, N\}$ is stationary random series. When the significance level is 5% and $N_1 \leq 15$ and $N_2 \leq 15$, given the upper limit r_U and the lower limit r_L and if $U_N \geq r_U$ or $U_N \leq r_L$, we reject H_0. Otherwise H_0 is accepted.

If $N_1 > 15$ or $N_2 > 15$, U_N follows the normal distribution $N(\mu, \sigma^2)$, where $\mu = \frac{2N_1N_2}{N} + 1$ and $\sigma = \frac{2N_1N_2(2N_1N_2-1)}{N^2(N-1)}^{\frac{1}{2}}$. Let $Z = \frac{U_N-\mu}{\sigma}$ and N follows $N(0, 1)$. At the 5% significance level, when $|Z| \leq 1.96$ the initial hypothesis is accepted.

2.3 ARIMA Model

Introduced by Box and Jenkins, the ARIMA model has been one of the most popular approaches to forecasting. In an ARIMA model, the future value of a variable is supposed to be a linear combination of past values and past errors, expressed as follows:

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} - \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \quad (1)$$

where y_t is the actual value and ε_t is the random error at time t; ϕ_i and θ_j are the coefficients; p and q are integers that are often referred to as autoregressive and moving average polynomials, respectively.

The autoregressive part AR(p) models a time series as a linear function of p previous observations in order to predict the current one. The moving average part MA(q) determines the moving average of the series with a time window of size q. Finally, the ARIMA process (p, d, q) is based on a series that has been differentiated d times, with p autoregressive terms and q moving average terms.

2.4 The Hybrid Model

The software failures can not easily be captured. Therefore, a hybrid strategy that has both linear and nonlinear modeling abilities is a good alternative for forecasting software failures. Both the ARIMA and the fractal models have different capabilities to capture data characteristics in linear and nonlinear domains, so the hybrid model proposed in this investigation is composed of the ARIMA component and the fractal component. Thus, the hybrid model can model linear and nonlinear patterns with improved overall forecasting performance. We compute the logarithms of the time series to make the curve of series smooth and stationary. The hybrid model (E_t) composed of linear and nonlinear components can be represented as follows:

$$ln(E_t) = ln(A_t) + ln(F_t) \quad (2)$$

where A_t is the linear part and F_t is the nonlinear part of the hybrid model. Both A_t and F_t are estimated from the data set. $\hat{F}_t$ is the forecasting value of F_t and we estimate it through fractal model. Let ε_t represent the residual at time t as obtained from the fractal model. Then

$$\varepsilon_t = ln(E_t) - ln(\hat{F}_t) \quad (3)$$

The residuals are modeled by the ARIMA and can be represented as follows:

$$\varepsilon_t = f(\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-n}) + \Delta_t \quad (4)$$

where f is a linear function modeled by the ARIMA and Δ_t is the random error. Therefore, the combined forecast is

$$\hat{E}_t = (\hat{F}_t)exp(\hat{\varepsilon}_t) = (\hat{F}_t)(\hat{A}_t) \quad (5)$$

where $\hat{\varepsilon}_t$ is the forecasting value of Eq. (4) and $\hat{A}_t$ is the forecasting value of A_t.

In this investigation, at first we adopt fractal model to forecast failure time, and for the sake of accuracy we compute the difference of the logarithm of the forecasting time minus the logarithm of actual time to obtain a series of differences. We apply run-test method to the difference series to check the stationary of the series. Basically the ARIMA model has three phases: model identification, parameter estimation, and diagnostic checking. We adopt autocorrelation and partial autocorrelation analysis to select an appropriate model from AR(p), MA(p), ARMA(p, q) and ARIMA(p, d, q). After the model is selected we use BIC (Bayesian Information Criterion) to determine the order of the model. Please see [6] for a detailed exposition.

Algorithm 1:

Initialization: Suppose the size of slide window $m, l = 1$, the size of training set n, A is a array of corresponding failure time of the *ith* failure, D is a array of the error series, F is a array of forecasting value of fractal model and E is a array of forecasting value of hybrid model;

for $i = l$ to $m + l - 1$ {

$B(i) = log(A(i))$;
$C(i) = log(i)$;}
for $i = m + l - 1$ to n{
(1) According to Eq. (5) in literature [5] and method of linear regression, compute the slope of linear regression in the slide window $b = d = \frac{1}{k}$ and constant $a = log(s) = -dlog(C)$;
(2) Make a prediction Prediction $\hat{F}_{t_i}$ of next point out of the slide window using the above a and b;
(3) Add $\hat{F}_{t_i}$ to F;
(4) Add the practical failure time t_i of the next point to A;
(5) Compute the error $ln(t_i) - ln(\hat{F}_{t_i})$;
(6) Add the error to D;
$l++$;
$B(m + l - 1) = log(A(m + l - 1))$;
$C(m + l - 1) = log(m + l - 1)$;}

Repeat

Repeat steps (1), (2), (3), (4), (5), (6) to compute the nonlinear part of prediction $\hat{F}_t$ of point $m + l - 1$ using fractal model;
Determine the stationarity of the error series D;
Identify the model of ARIMA;
Determine the order of model;
Compute the linear part of prediction $\hat{A}_t$ using ARIMA model and error series D;
Make a total prediction $\hat{E}_t = (\hat{A}_t)(\hat{F}_t)$ and add $\hat{E}_t$ to E;

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

Table 1 The NTDS set of software failure time series, and from left to right the time in each cell denotes the cumulate time of the ith software failure, $i = 1, 2, \ldots$.

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

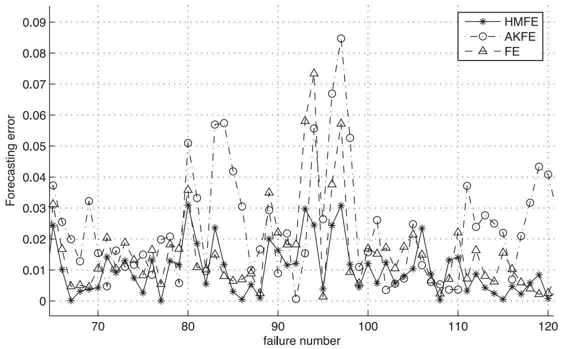

Fig. 1 Prediction errors of different models of Musa's data set 1. HMFE stands for hybrid model forecasting errors, AKFE stands for adaptive Kalman filter model forecasting errors, and FE stands for fractal model forecasting errors. Sliding-window size m = 5 and ARIMA(0, 0, 1).

Until test over
End

3. Experiments

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 (Table A.1 in literature [5]), Musa's data set 2 (Table A.2 in [5]), Musa's data set 3 (Table A.3 in [5]) and NTDS data set (Table 1). The performance of the proposed model is compared with normal distribution [5], Kalman filter [5], adaptive Kalman filter [5], ARIMA [5], and the SVMs model with simulated annealing algorithms (SVMSA) [7] forecasting methods. The experimental results are shown in Fig. 1, Fig. 2, Fig. 3, Table 2 and Table 3. In Fig. 1 85% of the forecasting errors using hybrid model are less than 2%; in Fig. 2 85% of the forecasting errors using hybrid model are less than 4%; in Fig. 3 80% of the forecasting error using hybrid model are less than 6%. They are all better than ARIMA, Adaptive Kalman filter and SVMSA etc. Obviously our method is effective. In the investigation, the values of Mean Absolute Error $MAE = \frac{1}{n}\sum_{i=1}^{n} abs\frac{T_i - \overline{T_i}}{T_i}$ and 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 and $\overline{T_i}$ is prediction time (Table 2). Mean Absolute Error of Interval Time $MAEIT = \frac{1}{n}\sum_{i=1}^{n}\frac{abs(d_i - \overline{d_i})}{d_i}$, where n is the number of prediction periods, d_i is actual value of period i and $\overline{d_i}$ is prediction value (Table 3).

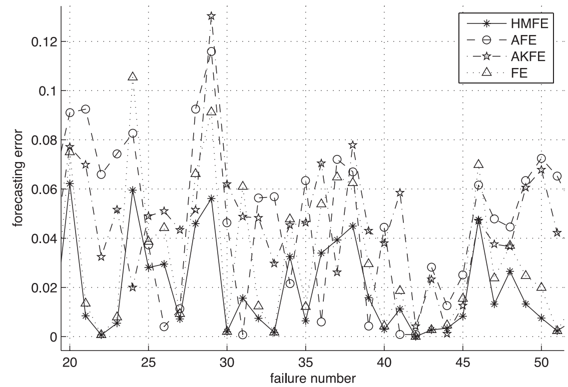

Fig. 2 Prediction errors of different models of Musa's data set 2. HMFE stands for hybrid model forecasting errors, AKFE stands for adaptive Kalman filter model forecasting errors, AFE stands for ARIMA model forecasting errors and FE stands for fractal model forecasting errors. Sliding-window size m = 3 and ARIMA(1, 0, 1).

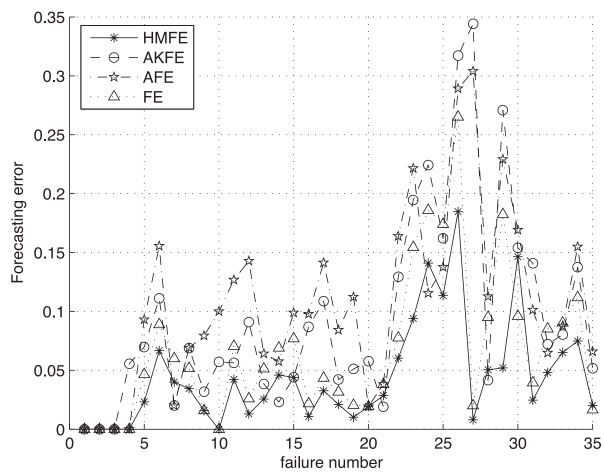

Fig. 3 Prediction errors of different models of NTDS data set. HMFE stands for hybrid model forecasting errors, AKFE stands for adaptive Kalman filter model forecasting errors, AFE stands for ARIMA model forecasting errors, and FE stands for fractal model forecasting errors. Sliding-window size m = 3 and ARIMA(1, 1, 1).

Table 2 Prediction results of different models of Musa's data set 1, 2 and NTDS. AK stands for adaptive Kalman filter, HM stands for Hybrid Model.

	Error	HM	Fractal	ARIMA	AK
Musa 1	MAE	0.0181	0.0271	0.0432	0.0425
	NRMSE	0.0255	0.0312	0.0493	0.0481
Musa 2	MAE	0.0437	0.0574	0.0718	0.0635
	NRMSE	0.0528	0.0645	0.0824	0.0702
NTDS	MAE	0.0492	0.0625	0.1019	0.0947
	NRMSE	0.0741	0.0939	0.1349	0.1203

4. Conclusion

This paper proposed a model of obtaining more accurate predictions by combining fractal and ARIMA. A comparison is made between this approach and other methods for predicting the failure time using actual data sets from

Table 3 Prediction results of different models of Musa's data set 3. HM stands for Hybrid Model, ND stands for normal distribution and AK stands for adaptive Kalman filter [5], and SVMSA stands for hybrid model of support vector machines model with simulated annealing algorithms [7] (Sliding-window size m = 9 and ARIMA(1, 0, 1)).

NO.	Actual data	HM	ND	AK	SVMSA
45	11.0129	9.5366	10.4177	19.3881	8.6740
46	10.8621	10.0356	11.4835	11.0904	9.2696
47	9.4372	9.9433	11.3140	9.7274	9.6609
48	6.6644	8.7454	9.7977	7.4402	9.4989
49	9.2294	10.1812	6.8764	4.2686	8.7925
50	8.9671	8.2677	9.5838	11.5589	9.0692
51	10.3534	9.5343	9.3004	7.9377	9.0610
52	10.0998	9.6446	10.7603	10.8762	9.3121
53	12.6078	10.2708	10.4852	8.9795	9.7436
54	7.1546	10.9855	13.1355	14.3821	10.0480
55	10.0033	11.3601	7.3952	3.6967	8.9747
56	9.8601	10.4659	10.4011	12.6137	9.8184
57	7.8675	8.7745	10.2433	8.8370	9.5530
58	10.5757	9.5072	8.1421	5.6897	9.2308
59	10.9294	10.2460	10.9974	12.8783	9.9859
60	10.6604	9.8674	11.3641	10.2797	10.2270
61	12.4972	11.7273	11.0736	9.4502	10.3990
62	11.3745	11.8501	13.0074	13.3468	10.8770
63	11.9158	11.6475	11.8162	9.4337	11.1470
64	9.575	11.4602	12.3801	11.3776	11.5190
65	10.4504	10.8593	9.9147	7.0159	10.9720
66	10.5866	11.0874	10.8297	10.3937	10.9530
67	12.7201	10.8477	10.9673	9.8073	11.1490
68	12.5982	11.8802	13.2073	14.0040	11.9540
69	12.0859	12.0526	13.0723	11.4408	12.3210
70	12.2766	12.3860	12.5277	10.6270	12.3980
71	11.9602	12.1034	12.7219	11.4427	12.1320
72	12.0246	12.3699	12.384	10.7036	12.0280
73	9.2873	11.2508	12.4459	11.1151	12.1270
74	12.495	11.4711	9.5809	6.5952	11.9460
75	14.5569	12.5541	12.9403	15.4497	12.4000
76	13.3279	12.8742	15.0996	15.6550	13.0730
77	8.9464	12.3863	13.8044	11.2419	12.9730
78	14.7824	12.3358	9.2257	5.5249	12.7810
79	14.8969	13.8570	15.3558	22.4572	13.5250
80	12.1399	13.5477	15.4692	13.8009	13.7040
81	9.7981	12.5751	12.5737	9.0674	13.2390
82	12.0907	11.6188	10.1218	7.2440	13.3320
83	13.0977	12.8023	12.5184	13.6803	12.9130
84	13.368	13.2046	13.5683	13.0569	13.1180
85	12.7206	12.6442	13.8459	12.5528	13.2160
86	14.192	12.7211	13.1633	11.1382	12.9030
87	11.3704	13.1078	14.6987	14.5875	13.4450
88	12.2021	12.1477	11.7469	8.3918	12.9540
89	12.2793	12.3524	12.6114	12.0519	12.6530
90	11.3667	12.0399	12.6876	11.4003	12.4040
91	11.3923	11.8972	11.7316	9.7078	12.3910
92	14.4113	12.0733	11.7545	10.5370	12.3660
93	8.3333	12.4852	14.9045	16.8667	13.0150
94	8.0709	11.6589	8.5791	4.4545	12.0830
95	12.2021	11.4320	8.3040	7.1640	11.5120
96	12.7831	12.3609	12.6137	16.0111	12.0470
97	13.1585	12.7458	13.2161	12.3861	12.6030
98	12.753	12.6001	13.6036	12.5033	12.7040
99	10.3533	12.3128	13.1763	11.4137	12.4240
100	12.4897	12.7059	10.6746	7.7609	12.1110
MAEIT	0	0.1074	0.1734	0.2578	0.1198

literature. The superior forecasting ability of the proposed model is due to the following two causes. First, good self-similarity exists in software failure time series. Second, good linearity exists in the residual series which actual failure data minus prediction data using fractal. In the future, some other factors which affect the software reliability can be considered in the model to predict software reliability to improve forecasting accuracy.

References

[1] F.E.H. Tay and L. Cao, "Modified support vector machines in financial time series forecasting," Neurocomputing, vol.48, pp.847–861, 2002.

[2] Y.S. Su and C.Y. Huang, "Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models," J. Systems and Software., vol.80, no.4, pp.606–615, April 2006.

[3] M.J. Lawerence, R.H. Edmundson, and M.J. O'Connor, "The accuracy of combining judgemental and stastical forecasts," Manage. Sci., vol.32, pp.1521–1532, 1986.

[4] N. Terui and K.D. Herman, "Combined forecasts from linear and nonlinear time series models," International Journal of Forecasting, vol.18, pp.421–438, 2002.

[5] Y. Cao and Q. Zhu, "The software reliability model based on fractals," IEICE Trans. Inf. & Syst., vol.E93-D, no.2, pp.376–379, Feb. 2010.

[6] R.H. Shumway and D.S. Stoffer, Time Series Analysis and Its Applications: with R Examples, 2nd ed., Springer, 2006.

[7] P.-F. Pai and W.-C. Hong, "Software reliability forecasting by support vector machines with simulated annealing algorithms," J. Systems and Software, vol.79, pp.747–755, 2006.