Prediction of Software Anomalies using Time Series Analysis – A recent study

Ashima Gupta, Biju R. Mohan, Shanil Sharma, Ritesh Agarwal & Kavya K.

Department of Information and Technology, National Institute of Technology, Karnataka

Abstract – **Modern society relies on the fault-free operation of complex computing systems. So, requests on software reliability and availability have increased greatly due to the current applications. Software applications executing continuously for a long period of time show a degraded performance and/or an increased occurrence rate of hang/crash failure. Computer system outages are more often due to software faults than hardware faults. Predicting software anomalies (like software aging) caused by resource exhaustion is not an easy task. It is difficult to know a priori the parameters involved with the software aging. Thus the capabilities of Machine Learning (ML) algorithms and the statistical methods can be analysed to predict the system crash due to software aging caused by resource exhaustion. In this paper one of the statistical methods, time series analysis has been studied and a survey of the recent work in prediction of software anomalies using the same has been conducted.**

Keywords – Time Series Analysis, Software aging, Software rejuvenation

I. INTRODUCTION

Today, requests on software reliability and availability have progressed greatly due to the current applicationsand so have malicious attacks and faults. As business depends on information and computing technology, continuous reliability is a universal concern. The server processes in most client-server software systems are intended to run continuously forever except is however very difficult, if not impossible, to design and guarantee. When such software bugs (faults) are encountered during executions, they could lead to transient failures with unpredictable and costly after effects [1].

Availability of business-critical application servers is an issue of paramount importance that has received special attention from the industry and academia in the last decades. The cost of downtime per hour can go from 100,000 for online stores up to 6 million dollars for online brokerage services [11]. It is thus important to avoid the occurrence of unplanned failures, mainly due to the phenomenon of software aging.

A. Software Aging

The phenomenon of "software aging" is one in which the state of the software system degrades with time. The primary causes of this degradation are the exhaustion of operating system resources, data corruption and numerical error accumulation. This may eventually lead to performance degradation of the software or crash/hang failure or both. Some common examples of "software aging" are memory bloating and leaking, unreleased file-locks, data corruption, storage space fragmentation and accumulation of round-of ferrors [2]. The progressive resource consumption over time could be nonlinear, or the degradation trend could change over the time [19]. Software anomalies could be related to the workload, or even the type of the workload [28]. Software anomalies can be due to two or more resources simultaneously involved in the service failure. Thus, making the prediction of software anomalies difficult.

Thus, Software aging may be defined as the continuing performance degradation of software application execution due to accumulation of numerical errors, greedy resource allocation policies, non-safe resource releasing strategies, and file system degradation [3]. This phenomenon is particularly observable in long-running applications such as web and application servers.

Grottke et al [13], classify the aging effects as shown in table 1. Besides the classes of aging effects mentioned in table 1, they have also classified aging effects as volatile and non-volatile effects, where volatile effects can be removed by re-initialization of the system or process affected and non-volatile aging effects still exist after reinitializing of the system/process. They

also classify aging indicators into system wide and application specific.

TABLE 1

CLASSES OF AGING EFFECTS

Basic class	Extension	Examples
Resource leakage	(1) OS-specific	- Unreleased
	(2) App-specific	• *Memory* (1, 2)
		• *File handlers* (1)
		• *Sockets* (1)
		- Unterminated
		• *Processes* (1)
		• *Threads* (1, 2)
Fragmentation	(1) OS-specific	- Phys. memory (1)
	(2) App-specific	- File system (1)
		- Database files (2)
Numerical error	(1) OS-specific	- Round-off (1, 2)
accrual	(2) App-specific	
Data corruption	(1) OS-specific	- File system (1)
accrual	(2) App-specific	- Database files (2)

Software aging is likely to be found in any type of software that has some complexity, but it is particularly troublesome in long-running applications. It is not only a problem for desktop operating systems: it has been observed in telecommunication systems, Web servers, enterprise clusters, OLTP systems, and spacecraft systems. This problem has even been reported in military systems with severe consequences such as loss of lives.

All of this makes the prediction for optimal time of software rejuvenation unavoidable.

B. Software Rejuvenation

Since aging leads to transient failures in software systems, some methods can be employed pro-actively to prevent degradation or crashes. This involves occasionally stopping the running software, "cleaning" its internal state and restarting it. Such a technique is known as "software rejuvenation". This counteracts the aging phenomenon in a pro-active manner by removing the accumulated error conditions and freeing up operating system resources. Garbage collection, flushing operating system kernel tables and reinitializing internal data structures are some examples by which the internal state of the software can be cleaned [4]. This technique of software rejuvenation incurs cost in the form of downtime, lost transactions, performance etc. Thus, it is important to find optimal times for rejuvenation.

Approaches used to study software rejuvenation can be divided into two categories [5]:

i) Model based studies make assumptions of the mechanism of software aging and construct mathematical models to describe the process of software aging.

ii) Measurement-based studies monitor certain metrics of resource usage, based on the collected data of resource usage, these studies can assess the status of the software system, and predict possible impending failures due to resource exhaustion.

The software rejuvenation approaches can be classified as: Purely Time based Software Rejuvenation Policy (PTSRP) and Purely Prediction based Software Rejuvenation Policy (PPSRP) [6].

i) In the PTSRP, rejuvenation is executed at a fixed interval from the start (or restart) of the software. By building an analytical model, the optimal interval can be determined to achieve the maximum the availability and minimum the downtime cost. This approach is easy to implement in software systems.

ii) In the PPSRP, the time to rejuvenate is predicted based on the collection and statistical analysis of system data. A monitoring program is applied to continuously collect the data about the operating system resource usage, and analysing these data to estimate system degradation level. Rejuvenation is attempted only when the software enters into the "failure probable" state.

Optimal scheduling of software rejuvenation is important to reduce the system downtime. For this, it is necessary that the system crash time is predicted. Machine Learning (ML) algorithms can be used to predict the system crash due to software aging caused by resource exhaustion. Several other models such as Markov decision processes and stochastic Petri Nets have been developed to deal with the optimal scheduling of software rejuvenation. Time Series Analysis is another technique that has been used to predict the future system state.

Since software aging has been observed in the past in systems such as Cisco catalyst switch, Apache web server and Patriot missile defence system [7] etc., it is important to rejuvenate these applications in order to prevent financial loss and in some cases loss of lives.

The growing complexity of software systems is resulting in an increasing number of software faults.

According to the literature, software faults are becoming one of the main sources of unplanned system outages, and have an important impact on company benefits and image. For this reason, a lot of techniques (such as clustering, fail-over techniques, or server redundancy) have been proposed to avoid software failures, and yet they still happen. Many software failures are those due to the software aging phenomena. Thus a prediction model is required which predicts the time of software crash with acceptable prediction accuracy against complex scenarios with small training data sets.

II. ANALYSIS OF PAST WORK

Since 1995, year of publication of the seminal work of Huang et al., much effort has been devoted to characterize and mitigate the *Software Aging* phenomenon [8], that is, the accumulation of errors occurring in long-running operational software systems that leads to progressive resource depletion, performance degradation, and eventually to the hang or crash of the software system. As a result, a significant body of knowledge has been established and an international community of researchers in the area of Software Aging and Rejuvenation (SAR) has grown.

Most SAR work focuses on predicting the *time-to-aging-failure* and on optimal scheduling of software rejuvenation strategies. Two main categories can be identified among these works, that is, *model-based analyses* and *measurement-based analyses*. In model-based analyses, a mathematical model of the system is considered, that include states in which the system is correctly working, states in which the system is failure-prone, and states in which software rejuvenation is taking place. Several kinds of model have been considered for this purpose, such as Markov Decision Processes and Stochastic Petri Nets [9], [4]. These studies typically aim at identifying the optimal time for applying software rejuvenation strategies in order to maximize availability or performability in the long term. Measurement-based analyses are instead based on data collected from a system about resource usage (e.g., free physical memory and used swap space) and performance (e.g., response throughput and latency) [8]. These data are processed using algorithms for time series analysis and machine learning, in order to identify resource exhaustion / performance degradation trends. One of the aims of these studies is to provide a support to on-line planning of software rejuvenation, in order to predict aging failures in the short term and to adapt rejuvenation to the actual workload and resource consumption of the system. Another aim of measurement-based analysis is to provide empirical data about the software aging phenomenon; for instance, this data could be used to populate mathematical models. We also came across *hybrid analyses*, in which the approaches of measurement- and model-based approaches are combined, e.g., by populating models with actual measurements and to validate the effectiveness of models with respect to real aging failures.

We came across another technique in which the merits of the two policies, namely Purely Time based software rejuvenation policy and Purely Prediction based Software rejuvenation policy, were combined to form a new rejuvenation policy, Time and Prediction based Software Rejuvenation Policy (TPSRP). In this policy, time based rejuvenation policy was performed from the system's start or restart, during which prediction based policy was also employed [6].

In another paper [10], the idea of preparing and training a dataset using different machine-learning classification algorithms was presented. Then, in run-time, some time-series analysis (ARMA, Holt-Winters) were used to forecast in advance the trends in some crucial parameters.

Cotroneo et al. [12] claim that it is important to define an approach for isolating the contribution of intermediate layers, such as middleware, virtual machines, and, more in general, third-party Off The Shelf (OTS) items, to aging phenomena. They concluded that aging phenomena in the JVM manifested as i) memory depletion due to the activity of the Just In Time compiler and the garbage collector, and ii) throughput loss which is mainly due the interface between the JVM and the underlying OS. OS resource usage data were collected together with OS workload parameters such as context switch rate and system call invocation rate; a small number of representative workload states were then derived through statistical clustering techniques in their work. For each cluster, a trend detection test was also performed, thus understanding whether a trend exists or not in the samples. To identify the workload parameters which were more relevant to the aging dynamics and relationships between applied workload and aging trends principal component analysis and partial linear regression are adopted.

Wang et al. [14], discuss in their paper the network traffic flow data predict and network anomaly detection. Network traffic prediction was done using ARMA(1,1) model and network anomaly detection using the exponential smoothing model.

In [15], the authors investigate the software aging patterns of a real VOD system. Non-parametric statistical methods were applied to detect software agingand linear regression models to estimate the aging

trend. An artificial neural network model was used to analyse the extracted data series of systematic parameters and to predict software aging of the VOD system. Principal component analysis (PCA) was used to reduce the dimensionality of input variables of ANN after which the sensitivity analysis was made.Mann-Kendall method was used to test the null hypothesis that the time series doesn't exhibit a trend. Linear regression model was used to fit the time series of system resource data. In the experiment, 30 parameters of memory, 15 parameter of CPU and 27 parameters of application server were collected. The input parameters were reduced to 40 by eliminating parameters that were constant and PCA was adopted to reduce them further. The sampling interval was one minute. The data was stored in a user-specified file, and the system dynamic parameters for 131 hours were extracted for the file to estimate and predict the aging of VOD system.

In [16], the authors identify two aging indicators in their analysis of Linux OS, namely the memory consumption and the system call latency. The analysis of experimental data was split in three steps: i) trend detection using Mann-Kendall test, ii) workload-aging correlation analysis using PCA, and iii) analysis of potential aging sources. The experiment was carried on for 4 days. Multiple linear regression was adopted to identify the relationships between aging trends and workload parameters of each subsystem. The experimental procedure revealed that the Linux OS suffers from software aging phenomena, manifested as statistically significant memory consumption trends; instead, there were no statistically significant effect on system call latency.

Silva et al. [17] demonstrated that some SOAP routers are very prone to software aging due to memory leaks.

In [18], Apache was deployed in two different modes,"compact mode" and "default mode". In the compact mode, not all modules are loaded. While in the default mode, all modules will be used on the server. To expedite the aging of Apache, two parameters that are related to the accumulation of the effects of software errors were adjusted: Max Request Per Child and MaxSpareServers. The first parameter was set to zero which means the accumulation of runtime errors will accumulate all through each experiment. The second parameter, MaxSpareServers, was also set to zero, so that no child processes will be killed during runtime. Apart from that, the connection rate was set to a high value to speed up the experiments. The reply rate, error rate and response time with respect to different connection rates were recorded. The data of response time was collected on the clients using Httperf. In Experiment I, Apache was running in compact mode. The experiment ran for about 420 hours before the server crashed. In the result, the response time of Apache increased from the 50th hour. The swapping period of memory and buffer usage also began to increase after the 50th hour, and the cache usage began to decrease at the same time.

In Experiment II, Apache was running in default mode. In the experiment, the server only ran for about 81 hours and then crashed. In the result, the response time of Apache increased from the 63th hour. The buffer usage began to increase and the cache usage began to decrease at the same time

Some repeatable phenomena were observed, i.e. the buffer usage increases abruptly when software aging occurs and the cache usage decreases correspondingly.

In [19], SNMP-Based Distributed Resource Monitoring Tool was used for data collection. More than 100 parameters were monitored at regular intervals (10 min) for more than a month. The system workload was characterized by obtaining a number of variables pertaining to CPU activity and file system I/O. "realMemoryFree" was identified as the more important or critical resource. It was validated that work load affects the way various system resources are depleted.

A methodology for detection and estimation of aging in the UNIX operating system has been discussed in [20]. SNMP based, distributed monitoring tool was used to collect operating system resource usage and system activity data at regular intervals, from networked UNIX workstations. Statistical trend detection techniques were applied to this data to detect/validate the existence of aging. Estimated time to exhaustion was calculated using well known slope estimation techniques. Analysis of trend detection and estimation was aimed towards validation of software aging and estimation of time to exhaustion of various operating system resources any of which will result in a failure.

The prior data of software performance parameters are treated as time series in [21]. The forecast method combines wavelet multiresolution decomposition and neural networks. The next sample of the original time series is predicted by another neural network. The experimental setup consisted of a server running Apache and Tomcat , a client connected via an Ethernet local network. In the experiment, the interval for extracting information about the performance parameters was ten minutes. In the client, httperf was used to generate requests to the server. Two representative parameters had been analysed, free Physical Memory and usedSwapSpace pertaining to system resources.

Software aging was measured in the Apache web server system and time-series analysis methods were employed for detection of software aging and estimation of the system resource exhaustion in [22]. Experimental platform was built which consisted of an Apache web server system, a Linux system status monitoring tool and a webserver workload generator. The performance of Apache was expressed in terms of reply rate and response time. The response time of the webserver becomes larger and the used swap space increases with time.

In [10], the forecasting is supported by a combination between machine leaning (ML) classification algorithms and parameters estimations provided by time-series analysis.In run-time the parameters collected by the *Monitoring* module are fitted by different time-series analysis, and estimated by up to *N* epochs ahead. Based-on the estimated values, the ML classification algorithms are enquired to classify the degree of performance anomaly that is expected to be verified. Naive Bayes classifier, two decision trees (J48 and LMT) and a neural network model (MLP) were used. For time-series analysis: ARIMA (Autoregressive Integrated Moving Average) and Holt-Winters (Triple Exponential Smoothing) were used. ARIMA and Holt-Winters are both adaptive and can model trends and seasonality. The set of estimated parameters was then submitted to the classifier that verifies if such combination may result in absence (”GREEN”), symptom (”YELLOW”) or strong indication (”RED”) of a performance anomaly.To evaluate the effectiveness of the selected techniques in the prediction of performance anomalies two synthetic aging scenarios were induced: a memory leak and CPU contention.

In [23], a multiplicative HW was usedfor studying the test platform, network traffic was monitored for 30 days. Monitoring was done with the network test access point (nTAP). In this study, it was attempted to test the suitability of the test platform for aberrant behaviour detection.

Data was collected from the machines at intervals of 15 minutes for about 53 days on a UNIX OS in [24]. Classical time series analysis techniques such as linear and periodic dependency analysis, and trend detection and estimation were used. Overall, it was found that the two resources *file table size* and *process tablesize* are not as important as *used swap space* and *realmemory free* since they have a very small slope and high estimated times to failure due to exhaustion.

In [26], they adopted the design of experiment technique (DOE) to characterize the aging phenomenon. The results showed evidence that the 'page type' and 'page size' factors were responsible for over 99% of memory size variation in httpd processes. The objectives were: to verify whether or not the web server was suffering from software aging, and if positive, to determine the aging factors and their degrees of influence on the web server software. They firstly subjected the artificial workload then they estimated the trends in the data sets and finally time series model was applied to the data collected. In this paper, they used linux tool PROCMON to sore the data in ASCII files of different size(5kb,50kb…) and htperf to generate requests with a constant time gap. They measured reply rate, response rate & timeout error with the help of httperf. They collected data for 3.5 weeks with 5 mins gap for response time, free physical memory and used swap space.For Time series analysis they used Autoregressive model(AR) model for used swap space (as it was showing some trend) to estimate future values. And were able to adequately predict the future behavior for a period of more than 1.5 weeks.

In [27], firstly they collected data, and then the largest Lyapunov exponent of time series of average load, and results show that software aging process is chaotic. Monitored parameters were response time, available memory, cache, buffer, average load. The result shows that cache decreases with time which confirms software aging. They used Wolf process to calculate lyapunov exponent and Based on chaotic dynamic theory, the reciprocal of Lyapunov exponent present the longest forecasting time.

III. STEPS IN TIME SERIES ANALYSIS

The steps for forecasting using time series are [29]:

1) The first step in the analysis is usually to plot the observations against to give the time plot and look for trend, seasonal variation, outliers and any changes in structure such as slow changes in variance or sudden discontinuities etc.
2) After that ‘clean’ the data, for example, by adjusting any suspect observations.
3) Decide whether the seasonal variation is non-existent, multiplicative, additive or something else.
4) Decide whether the trend is non-existent, global linear, local linear or non-linear.
5) Fit an appropriate model.
6) Check the adequacy of the fitted model by studying the one-step ahead forecast ahead forecast errors over the period of fit. Modify the model if necessary.
7) Compute forecasts.

IV. COMPARATIVE ANALYSIS OF PREVIOUS WORK

While the papers that we investigated address majority of the problems in software aging, we did find some areas that need to be worked upon. In [12], the effect of varying the mail size has not been studied and the workload applied is constant and rather static. Disadvantages inherent with anomaly detection technique are a part of the technique proposed in [14]. For example, the sudden change in behaviour would be categorised as an anomaly whereas it could be normal behaviour. Abnormal detection is too simple. Besides the possibility of using other time series models could have been explored.Number of concurrent processes was varied in [16] but not the requests sent by each process. Impact under varied load could have been studied.

In [17], the sampling interval of 30 seconds has not been justified. TCP-IP sockets, Java RMI and HTTP-Xml were also studied for 5 hours but no aging was observed. 5 hours is a short duration to decide. The experiments should have been run for a longer duration if no significant aging was observed in 5 hours.

PCA has been used in [15] but the number of parameters finally used has not been mentioned. Like many others, the selection of parameters as well as 60 seconds time interval has not been justified properly in [26]. In [25] configuration of OS was not taken into account. They didn't consider multivariate time series model to find interaction between system resources. In [27] , the selection of Wolf method was not explained.The way for choosing the parameters may not be very effective in [24], as the parameters chosen are based solely on the basis of their increase or decrease in trend but a parameter may affect the aging if increased constantly even if the increase is not that big. Recalculating the smoothing constant has to be more effectively done in [23].

The color obtained every time may not predict correctly each time,and it will be difficult to suggest anything specially on the border line in [10].

V. CONCLUSION

It is thus highlighted that a lot of effort has been spent on designing analytical models for rejuvenation time scheduling. These models are becoming more and more refined and comprehensive; however, the works addressing aging by models often lack experimentation on real systems. Most often, models are validated by numerical examples, or by simulation, but to make them actually applicable in operational systems assumptions need to be verified against real system behaviour.

The need for additional experimentation on real systems is also highlighted. The non-negligible percentage of aging symptoms in safety critical systems suggests that: *i)* it is certainly worth to further investigate aging phenomena in safety-critical systems, since although well-tested they show to suffer from aging; *ii)* more attention to aging-related bugs is needed in the design and validation of safety-critical systems, since aging-related bugs can affect reliability requirements that are imposed on long-running systems by safety certification standards [8].

From results, it is clear that much literature is devoted to the optimal schedule determination. A potential direction to further reduce the cost of software rejuvenation regards the development of more efficient techniques for rejuvenating a system. The problem of accounting for the state of the application may be addressed in the future, as well as the implementation of additional mechanisms that try to minimize the downtime by selectively rejuvenating part of the system. The adoption of these rejuvenation strategies among developers could be encouraged by the integration of rejuvenation mechanisms at the OS or middleware level and in programming languages.

VI. REFERENCES

[1] Ohnmar Nhway, "Reliability Modeling and Analysis of ApplicationServers using Stochastic Petri Net Model," in Procs. 7th International Conference onAdvanced Information Management and Service (ICIPM), 2011

[2] S. Garg, A. van Moorsel, K. Vaidyanathan, K. Trivedi. "AMethodology for Detection and Estimation of Software Aging".In Proc. of 9th Intnl. Symposium on Software ReliabilityEngineering, pages 282-292, Paderborn, Germany, November1998.

[3] Rivalino Matias Jr., Kishor S. Trivedi and Paulo R. M. Maciel. "Using Accelerated Life Tests to Estimate Time to Software Aging Failure". In proceedings of 2010 IEEE 21st International Symposium on Software Reliability Engineering

[4] Y. Bao, X. Sun, and K. Trivedi, "A workload-based analysis of software aging, and rejuvenation," Reliability, IEEETransactions on, vol. 54, no. 3, sept. 2005.

[5] Jun Guo, Weiyue Li, Xinya Song, Bin Zhang, Yunsheng Wang. "Software Rejuvenation Strategy Based On Components". In Proceedings of 2010 Second WRI World Congress on Software Engineering

[6] Letian Jiang, Xiangyu Peng, Guozhi Xu. "Time and Prediction based Software RejuvenationPolicy". In proceedings of 2010 Second International Conference on Information Technology and Computer Science.

[7] Michael Grottke, Rivalino Matias Jr., and Kishor S. Trivedi "The Fundamentals of Software Aging". In Proc. 1st International Workshop on Software Aging and Rejuvenation19th International Symposium on Software Reliability Engineering, 2008.

[8] Domenico Cotroneo, Roberto Natella, Roberto Pietrantuono, Stefano Russo. "Software Aging and Rejuvenation: Where we are and where we are going". In proceedings of 2011 Third International Workshop on Software Aging and Rejuvenation.

[9] S. Garg, A. Puliafito, M. Telek, and K. Trivedi, "Analysis of software rejuvenation using markov regenerative stochastic petri net," in Software Reliability Engineering, 1995. Proc.,Sixth Int'l. Symp.

[10] Jo˜ao Paulo Magalh˜aes, Luis Moura Silva." Prediction of Performance Anomalies in Web-Applications Based-on Software Aging Scenarios".2010 IEEE Second International Workshop on Software Aging and Rejuvenation (WoSAR).

[11] Luis Moura Silva, Javier Alonso, and Jordi Torres. "Using Virtualization toImprove Software Rejuvenation". IEEE Transactions on Computers, vol. 58, no. 11, November 2009

[12] Domenico Cotroneo, Salvatore Orlando and Stefano Russo. "Characterizing Aging Phenomena of the Java Virtual Machine". 26th IEEE International Symposium on Reliable Distributed Systems

[13] Michael Grottke, Rivalino Matias Jr., and Kishor S. Trivedi. "The Fundamentals of Software Aging".In Proc. 1st International Workshop on Software Aging and Rejuvenation19th International Symposium on Software Reliability Engineering, 2008.

[14] Guilan Wang, Zhenqi Wang, Xianjin Luo. "Research of Anomaly Detection Based on Time Series". In proc. Of IEEE World Congress on Software Engineering 2009.

[15] Xiaozhi Du, Chongan Xu, Di Hou and Yong Qi. "Software Aging Estimation and Prediction of a Real VOD System Based on PCA and Neural Networks". Proceedings of the 2009 IEEE International Conference on Information and Automation June 22 -25, 2009, Zhuhai/Macau, China

[16] Domenico Cotroneo, Roberto Natella, Roberto Pietrantuono, Stefano Russo. "Software Aging Analysis of the Linux Operating System". In Proc. Of 2010 IEEE 21st International Symposium on Communication, Networking & Broadcasting ;Computing & Processing (Hardware/Software) ; Power, Energy, & Industry Applications

[17] Luis Silva, Henrique Madeira, João Gabriel Silva. "Software Aging and Rejuvenation in a SOAP-based". Fifth IEEE International Symposium on Network Computing and Applications 2006.

[18] Yun-Fei Jia, Lei Zhao and Kai-Yuan Cai. "A Nonlinear Approach to Modeling of Software Aging in a Web Server". In Proc. of 15th Asia-Pacific Software Engineering Conference 2008.

[19] Kalyanaraman Vaidyanathan and Kishor S. Trivedi. "A Comprehensive Model for Software Rejuvenation". IEEE Transactions On Dependable And Secure Computing, April-June 2005.

[20] Sachin Garg, Aad van Moorsel, Kalyanaraman Vaidyanathan, Kishor S. Trivedi. "A Methodology for Detection and Estimation of Software Aging". In Proc. Of The Ninth International Symposium on Computing & Processing 1998.

[21] Jian Xu, Jing You,Kun Zhang. "A Neural-Wavelet based Methodology for Software Aging Forecasting". IEEE International Conference on Systems, Man and Cybernetics2005.

[22] Lei Li, Kalyanaraman Vaidyanathan and Kishor S. Trivedi. "An Approach for Estimation of Software Aging in a Web Server". In Proceedings of the 2002 International Symposium on Empirical Software Engineering.

[23] Jarkko Ekberg, Jorma Ylinen, Pekka Loula. "Network Behaviour Anomaly Detection Using Holt-Winters Algorithm".In Proc. Of 6th International Conference on Internet Technology and Secured Transactions, 11-14 December 2011

[24] Kishor S. Trivedi, Kalyanaraman Vaidyanathan and Katerina Go˜seva-Popstojanova. "Modeling and Analysis of Software Aging and Rejuvenation".

[25] Michael Grottke, Lei Li, Kalyanaraman Vaidyanathan, and Kishor S. Trivedi."Analysis of Software Aging in a Web Server". IEEE Transactions On Reliability, Vol. 55, No. 3, September 2006

[26] Rivalino Matias Jr. and Paulo J. F. Filho. "An Experimental Study on Software Aging and Rejuvenation in Web Servers". Proceedings of the 30th Annual International Computer Software and Applications Conference 2006

[27] Yun-Fei Jia, Xiu-E Chen, and Kai-Yuan Cai. "Chaotic Analysis of Software Aging in Web Server". Proceedings of the Second IEEE International Symposium on Service-Oriented System Engineering 2006

[28] J. Alonso, J. L. Berral, R. Gavald`a, and J. Torres, "Adaptive on-linesoftware aging prediction based on machine learning," in Procs. 40th IEEE/IFIP Intl. Conf. on Dependable Systems and Networks, 2010.

[29] Chris Chatfield,"Forecasting,"in The analysis of Time Series – An Introduction, 6thed., Chapman and Hall, pp. 95-97.

◈◈◈