2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops

An Advanced Methodology for Measuring and Characterizing Software Aging

Pengfei Zheng*, Qingguo Xu*, Yong Qi*
*School of Electronic and Information Engineering
*Xi'an Jiaotong University
No.28, Xianning West Road, Xi'an, Shanxi, China
{p.f.zheng.phd, yishi.2011} @stu.xjtu.edu.cn

Abstract—**Software systems continuously running over a long time may suffer gradual performance degradation or failure rate increasing. This phenomenon, known as 'Software Aging', has become a great challenge to dependability-critical software systems. Researchers have made remarkable achievements in predicting resource exhaustion time and designing optimal rejuvenation plan. However, limited works focus on measuring aging and characterizing aging progress. And currently the only widely used tool is Sen's slope estimator. We pinpoint some drawbacks of this approach: (1) a not unified estimator which needs periodicity test and period length inference in advance (2) an oversimplified linear description of aging progress (3) with no ability to distinguish between abrupt change and the "aging-like" gradual degradation. We design an enumerative Hodrick-Prescott filter to overcome all these shortcomings. And we also propose a new metric AS based on the nonlinear trend estimated by Hodrick-Prescott filter to dynamically measure severity of aging. Our approach and metric are validated on real aging time series collected from a VOD (video-on-demand) server. The results shows our approach improve the Sen's slope estimator a lot.**

Keywords-Software Aging; Hodrick-Prescott filter; Sen's Slope Estimator; Measurement ; Aging progress

I. INTRODUCTION

Software aging was initially reported in multi-user telecommunication systems [1]. When aging exists, the communication platform will enter a "**Failure-Prone**" state that it continues to lose ongoing calls, even not with peak workloads. Recently aging phenomena have been reported or verified on almost all types of software, such as Java Virtual Machine , Apache Web Server [4] [5], AT&T billing systems, SOAP servers, online transaction processing servers, Linux Operating systems [6], cluster systems and military systems. A considerable portion of failures occurring in these systems are not abrupt system halt but "smooth" degradation over long periods. For example, the response time of an E-Commerce server may increase from time to time and finally become unacceptably long or infinite (no response). Another software aging example is related to memory leak in some systems. Memory leak still happens even though GC (Garbage Collection) technique is adopted. Accumulation of memory leaks will diminish system performance severely and deplete all amount of available memory in the worst case. When the available memory is exhausted, any attempt to allocate more memory will fail. The system may terminate itself, or generate a segmentation fault in this situation.

Choosing a reliable system is one of the most important considerations for commercial and industrial clients. This reduces the direct costs of reacting to downtime as well as the costs associated with productivity. Software aging poses great threats and challenges. As complexity of software grows day by day, though systematic testing and validation methods become more and more sophisticated and efficient, aging related bugs can still escape the extensive testing stage and be activated under some irreproducible or uncertain conditions while the system is operational.

Some acknowledged causes of software aging are memory leaks, unreleased file-locks, corrupted data, and round-off error. The best method to counteract aging is software rejuvenation, during which the running system will be scheduled to restart periodically. After the system is restarted, accumulated erroneous conditions or corrupted data are completely swept away. Consequently, the system can recover to a normally running state by rejuvenation. Recently, virtualization technique is also developed to improve software rejuvenation.

Currently, there are still not an elaborately designed metric to quantitatively measure the severity of software aging, except a metric **"the Sen's slope estimator"** widely used by many researchers[3][4][5]. In fact "the Sen's Slope estimator" provides an estimation of magnitude of the "trend" in a time series composed of performance metrics (e.g. response time, throughput rate) or resource usage indicators (e.g. available memory, CPU utilization). The reason why using "the Sen's Slope" to measure software aging is:

Recent research shows that, the primary symptoms of software aging are gradual system performance degradation, gradual system resource exhaustion, or growing failure rate. All of these can be generally manifested by a descending or ascending trend of time series indicating performance (or resource usage). We call these trends the **"aging trend"** to characterize aging phenomena. Sen assumes the trend of a time series to be always a linear trend, and proposes a statistic to estimate the slope. So the Sen's slope estimator is widely accepted by researchers, which measures the strength of the aging trend. And the estimated slope is a metric to measure the severity of aging (the larger the slope is, the more severely aging acts).

978-0-7695-4928-6/12 $26.00 © 2012 IEEE
DOI 10.1109/ISSREW.2012.81

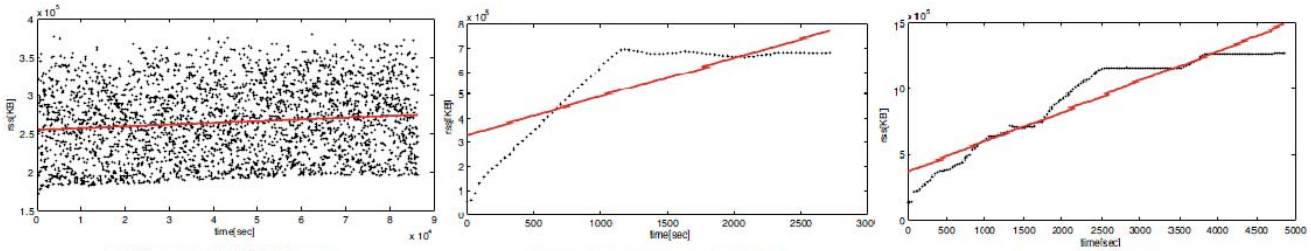


Figure 1. An example of Sen's slope estimator used in [5]

In addition, the linear trend estimated by the Sen's slope estimator is also used to roughly describe the aging progress. For example, in [5] the author performed a series of long-running aging tests on an Apache Server, a James Mail Server (a Java based mail server) and a release of CARDAMOM (a middleware used by air traffic control). Real time memory consumption of these three systems was collected (scatter points in Fig. 1). Statistical trend test was utilized to testify a trend of memory exhaustion. The detected aging trends were then estimated by Sen's slope estimator. They were manifested by red straight lines in Fig. 1, which can approximately characterize the aging progress related to memory depletion.

While in practical applications, Sen's slope estimator still has some intrinsic drawbacks. We summarize its three main limits in measuring and characterizing software aging:

1) A hybrid but not unified estimator : Depending on whether the time series is periodic (or cyclical), Sen's slope estimator takes different estimating approaches, respectively. For a time series without periodic components embedded, **standard Sen's slope estimator** is used. But for periodic time series, another estimator "**seasonal Sen's slope estimator**" proposed by Hirsch [10] is used. These two estimators are **collectively called Sen's slope estimator**. In fact, the Sen's slope estimator is such a hybrid estimator (See Fig. 2) that causes a big trouble.

Because we have no idea about the periodicity of a time series, an automatic procedure must be introduced to test whether the time series is periodic or not. Moreover, another procedure should also be introduced to infer its period length if it is periodic. These two essential preliminary procedures make the Sen's slope estimator difficult to apply in normal situation, because there is no methods can handle these perfectly without human knowledge. Here are some detail reasons:

- Periodicity test for a time series is still an imperfect and developing research target which many researchers still devote them to.
  Although there are some methods provide exact test based on least squares estimation, they are not robust. Even worse, they often present misleading results if the original noise assumptions do not follow Gaussian distribution.

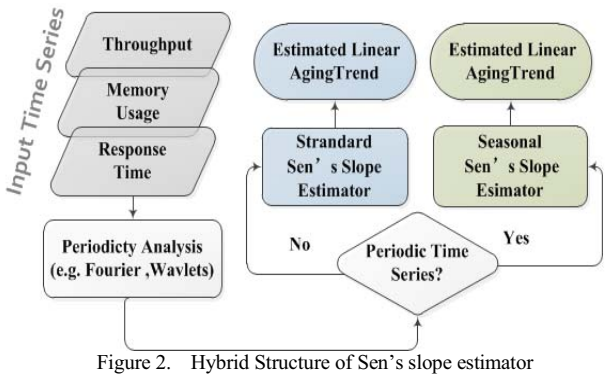


Figure 2. Hybrid Structure of Sen's slope estimator

- To infer period length, Fourier Transform and Wavelets are the most popular tools. Fourier Transform is based on an assumption that any stationary time series may be approximately as superposition of sines and cosines at different periods (or frequencies). But for a significantly non-stationary time series, which are the most cases, Fourier analysis cannot cope. Wavelets can handle non-stationary time series, but choosing the appropriate wavelets basis (e.g. Haar wavelets, Daubachies wavelets) and determining the optimal decomposition level are intricate and also need domain knowledge.

In a word, Sen's hybrid slope estimator for measuring aging trend is difficult to apply in actual situations. We should avoid the periodicity test and period length inference. The best way to settle this is to provide a unified trend estimator, which can estimate the aging trend in a unified procedure, no matter whether the time series are periodic or not.

2) Oversimplified Discription of Aging Progress :
Standard Sen's slope estimator and Seasonal Sen's slope estimaor share a common rationale to perform aging trend estimation. They all assume the estimated aging trend to be a linear trend. In most cases, the real aging trend is not a linear trend. It can be complex like quadratic, logarithmic or sigmoid trend, and in most cases it's hardly to be structured. So the aging trend estimated by Sen's slope estimator is oversimplified and unable to characterize the aging process in detail. It can't tell us: (1) When does aging start to show effect? (2) When do system resources begin to be abnormally encroached owing to aging-related bugs? (3) During which time period are aging

related bugs activated frequently and which time period scarcely?

Firstly, in this paper we introduce an approach using Hodrick-Prescott filter in an enumerative way with a Cauchy type stopping criterion to estimate the aging trend. This approach overcomes the drawbacks of the Sen's slope estimator with some improvements:

- A unified estimator, estimating aging trend in a unified computing procedure no matter whether or not the time series shows periodic pattern.
- Ability to estimate nonlinear aging trend, which characterizes the aging progress more accurately, and provides more detail information of aging.
- Ability to distinguish between abrupt change and aging trend, which is usually a smooth degradation.

Secondly, we propose a metric **Aging Severity (AS in abbreviation)** which measures the severity of aging. AS has 2 advantages that the slope (estimated by Sen's estimator) don't have:

- It is a composite metric which takes into consideration 2 fundamental features of software aging, the degradation amount (represented by **R**) and the degradation persistence (represented by **G**). While, the Sen's slope estimator only measures the degradation amount, with no consideration of the gradualness or smoothness of degradation.
- It is a dynamic metric, which reflects the development of aging dynamically.

The rest of paper is organized as follows: In Section II, we survey the recent literature about software aging, whereas in Section III we describe the methodology of our proposed method. In Section IV, we detail our measurement of software aging. Finally in Section V, we conclude the paper.

II. Related Works

Currently, software aging analysis can be divided into two categories: the model based and the measurement based. Commonly, the analytic models include Markov process models [2], semi-Markov process models, semi-Markov reward process models, hidden-Markov process models, stochastic petri net models and so on. The basic idea of model based methods is to construct a stochastic process describing a few states of the target system and solve the model to determine the optimal rejuvenation interval. Besides, multi-granularity and multiple-action rejuvenation has also been developed. In this situation an optimal rejuvenation strategy is often deduced and the only goal is to minimize costs due to outage of aging failure.

Measurement based approaches primarily focus on applying statistical and machine learning methods on datasets collected through monitoring or profiling tools. Performance metrics or resource usage indicators are often collected periodically to get insight into the operational conditions of the monitored system. The Sen's slope estimator, the Mann-Kendall test [3] [4], and the Seasonal Kendall test [4] are all statistical approaches to analyze aging. For machine learning approaches, when the system is running normally (or abnormally), data can be collected as positive samples (or negative samples) to train models such as ARMA models [4], linear regression models, M5P models [8], or MSET models. Then the models can be utilized to forecast resource usage or performance in the near future. Often a threshold (indicating performance baseline or upper bound of resource consumption) is compared to the predicted values to decide whether rejuvenation should be scheduled to mitigate aging effect.

III. Improved Aging Trend Estimation

A. Introduction to Hodrick-Prescott Filter

As mentioned in Section I, the hybrid structure of Sen's slope estimator makes it impractical to estimate aging trend, because it is often too complex and intricate to truly detect periodicity and accurately infer period length. An adaptive trend extraction method Hodrick-Prescott filter is introduced in this section. It can estimate the aging trend in a unified procedure, no matter whether or not periodic component is in the time series.

In addition, the trend estimated by Hodrick-Prescott filter is unstructured (with no structure of trend assumed) and data-driven. Therefore, when the raw time series shows a nonlinear pattern, Hodrick-Prescott filter can extract the latent nonlinear trend with cyclical fluctuations and random noises filtered out.

The Hodrick–Prescott filter is a time series decomposition tool proposed by Robert J. Hodrick and Nobel Prize winner Edward C. Prescott [9]. It uses a commonly accepted additive model for time series decomposition:

$$Y_t = T_t + C_t,\ t = 1,2,3,\cdots,T \quad (1)$$

$\{Y_t\}$ denotes a time series $\{Y_1, Y_2, \ldots Y_T\}$ and Y_t can be decomposed as a sum of trend component $\{T_t\}$ and a residual component $\{C_t\}$. $\{Ct\}$ is usually composed of periodic components and irregular noises. Hodrick-Prescott filter treats periodic component and irregular component both as short-term, compared to trend component (represents long term movement). Hence Hodrick-Prescott filter can extract the trend component not matter whether the periodic component is present or not.

The trend component $\{Tt\}$ is the optimal solution of the following programming problem:

$$Min\left\{\sum_{t=1}^{T} {C_t}^2 + \lambda \sum_{t=1}^{T} [(T_t - T_{t-1}) - (T_{t-1} - T_{t-2})]^2\right\} \quad (2)$$

In fact the trend component extracted by Hodrick-Prescott filter is a smooth representation of raw time series. The parameter λ is the "smooth parameter" which penalizes variability in the trend component series .The larger the value of λ is, the smoother the trend component series is. In the extreme case when λ is infinite, T will become {0, 0…0} (extreme smooth with no variability). Because we don't know how large λ should be (but should not be too big), so we design an enumerative filtering procedure, with a small λ initialized and increased by a small step 0.001, until the "smoothness" grows extremely subtle. The "smoothness" of every step is measured by **CV (Coefficient of Variance)** and **w**e provide a Cauchy type stopping criterion:

$$\frac{\left|\frac{Mean(T(i+1))}{Std(T(i+1))}-\frac{Mean(T(i))}{Std(T(i))}\right|}{\left|\frac{Mean(T(i))}{Std(T(i))}\right|} < Cauchy_{stop} \quad (3)$$

T(i) represents the trend component in the i-th loop, and $Cauchy_{stop}$ is set to 0.0005. This is a common value for Cauchy type stopping criterion. The initial value of λ is computing using Schlicht's approach [11].

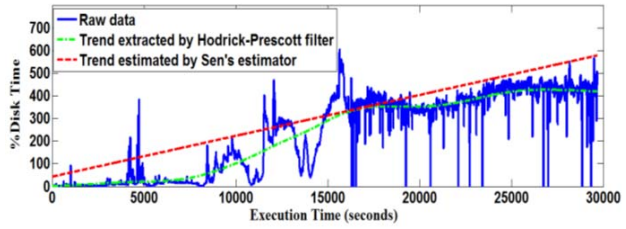
B. Nonlinear Aging Trend Estimation

In this section, we compare the aging trend estimated by our enumerative Hodrick-Prescott filter and the Sen's slope estimator. The raw time series were collected on a "Helix Sever" testbed built in our experiment. In Section IV, we will introduce the data collection process. Here we present 3 examples. They are so typical that we can evaluate the two kinds of aging trend estimator on

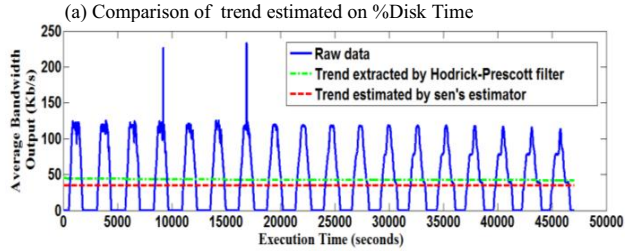
- Periodic and aperiodic time series
- Time series with trend and without trend

In Fig. 3(a) the %Disk Time (ratio of elapsed time when the disk was busy) of Helix sever shows no periodic pattern and it can be divided into 3 phases (a stable phase, a growing phase and another stable phase). The nonlinear trend estimated by Hodrick-Prescott filter successfully captures these details while Sen's trend estimation is not capable to do this.

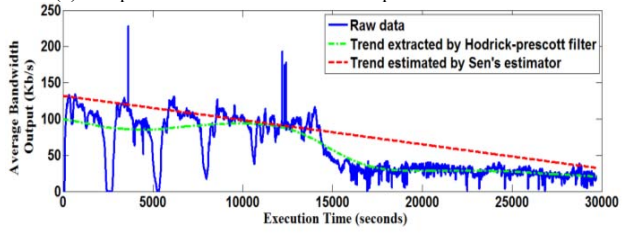
In Fig. 3(b) the Average Output Bandwidth was collected when Helix server ran on light-weight periodic workloads. Apparently the bandwidth is trend free and Hodrick-Prescott filter also accurately extracts a "zero" trend, so as Sen's slope estimator.



(a) Comparison of trend estimated on %Disk Time



(b) Comparison of trend estimated on Output Bandwidth



(c) Comparison of trend estimated on Output Bandwidth

Figure 3. Comparison of trend estimated by Hodrick-Prescott filter and Sen's slope estimator

In Fig. 3(c), bandwidth shows periodic pattern in the early stage, but it gradually decreases afterwards. This time series not only show periodic behavior but also show non-seasonal behavior in the later phase. We can see the Hodrick-Prescott filter is still able to work in this situation. Except these examples we present here, the Hodrick-Prescott filter performs also well on other time series.

According to the comparison, the Hodrick-Prescott filter performs better on nonlinear estimation than the Sen's estimator, which can suggest more information (like aging phase segmentation) about aging process. And whether or not the time series is periodic is of no differences, for the Hodrick-Prescott filter.

C. Distinguishing between Abrupt Change and Gradual Degradaton

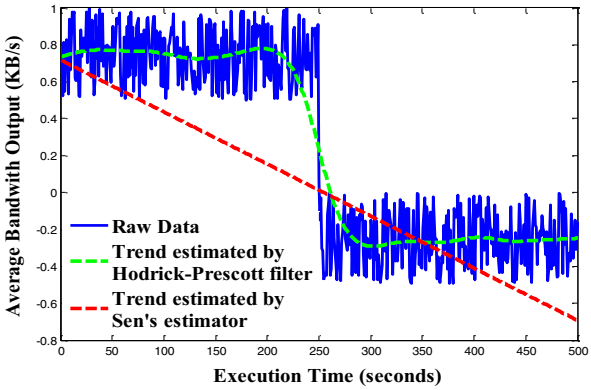


Figure 4. Comparison between Hodric-Prescott filter and Sen's estimator on abrupt change identification

In fact abrupt change is often caused by some transient events and degradation of this type is usually "sharply" (see Fig. 4) but not gradually or smoothly. Aging, however, is an accumulative degradation process, so the trend estimator should be able to differentiate abrupt change from aging trend. We apply both the Hodrick-Prescott filter and the Sen's slope estimator to an artificial series with an abrupt change in the middle. We can see that the Sen's estimator interpret this process as a sustained downward trend, while the Hodrick-Prescott filter estimates a trend which keeps stable in most times and only degrades only in a short range. The Hodrick-Prescott filter can identify abrupt change, which is an improvement to the Sen's slope estimator.

IV. Measuring Software Aging

A. A Composite Metric Aging Severity

In the last section, we have shown that the Sen's slope estimator can't identify abrupt changes. The Sen's slope estimator has this disadvantage because it only measures the degradation amount, without incorporating another significant feature of aging-which is "gradualness". We propose a composite metric ***AS* (Aging Severity)** combining these two factors together. See ***R*** represents degradation amount and ***G*** represents the degradation "gradualness", then AS can be computed:

$$AS = R^{w1} \cdot G^{w2}, \quad (0 < R < 1, 0 < G < 1, w1 + w2 = 1) \quad (4)$$

w1 and *w2* are weights assigned to the two factors. The bigger the weights, the more decisive it is to AS. We consider "gradualness" of degradation to be more important, so we set *w1*=0.4, and *w2*=0.6. We give some explanation on computing *R* and *G* respectively. They are both computed using the trend estimate by Hodrick-Prescott filter, with all the short-term fluctuations eliminated.

***For computing G**: G* represents the degradation graduallness. In a time series, the times series is relatively stable in some phases, while degrades severly in other phases. The longer the stable phases, the smaller *G* is. The rationale for computing *G* is that we evenly split the estimated trend into many segments and compare (using Mann-Whitney test) neighboring segments to see whether the later segment has a degraded median, compared with the former segment. If we have *M* segments after splitting, and *K* pairs of neighboring segments are tested a degradation, then the gradualness under *M* segments is *K/(M-1)* , we express this as:

$$G_M = \frac{K}{M-1} \qquad (5)$$

However, we don't know the optimal number of partitions, so we compute *G* under multiple value of *M* and use the median as the final *G*. This is a multi-scale computation for *G*, which is more objectively.

***For computing R**: R* denotes the magnitude of performance degradation. We compute it as the ratio of median degradation, between the first half and the second half of the estimated trend.

The algorithm for computing AS is shown in **Algorithms 1**. All notions are listed below:

- ***P***: A time series indicating resource usage or performance, *P*= {P_1, P_2, ..., P_T}
- ***AS (P, t)*** represents aging severity of time series *P*, measured on {P_1, P_2, ..., P_T}
- ***G (P, t)*** represents degradation gradualness of time series P, measured on {P_1, P_2, ..., P_T}
- ***R (P, t)*** represents magnitude of performance degradation of time series *P*, measured on {P_1, P_2, ..., P_T}

B. Using AS to Dynamically Measure Software Aging

1) Experiments: The setup is composed of a VOD server (based on Helix streaming server) and 3 client machines connected via a LAN. Specific hardware configuration is as follow: Intel Xeon 8 core 1.6GHZ (Processor) / 4GB (Memory) /1Gbps (Network Adapter)/ for VOD server and Intel Pentium4 2.59GHZ / 1GB/ 100Mbps for each client machine.

Helix server supports a great many streaming media formats such as .rm, .ra, .rmvb, .mov and .mpeg. During our experiments, we deployed 160 .rmvb files on Helix server. These files were the popular movies in the recent years within our campus. We requested all the movies and the times that they were played from the Wang An VOD Center (a website in our university). We couldn't access sever logs so we implemented a workload generator - Clients Simulator, adopting RTP/RTSP protocol, to generate user processes concurrently. The simulated users were created and removed on the client side periodically to simulate the diurnal pattern of real workloads. Each user requests media files on Helix server according to a probability distribution. The probability density of each file was the proportion of times that they were played. Server capacity test was executed as a preliminary work. And Clients Simulator will generate no more than 200 clients at a time in case of overloads. This experiment was indeed far from an exact replication of actual environment and access pattern of the real VOD platform in our campus, but it still had some advantages:

- The data and the accessing frequency (probability) of the data were real.
- We carried out our experiments with no faults injected. The aging phenomenon spotted on our VOD server was real and meaningful.

We collected 64 performance and resource usage indicators all together. We carried out the experiment 3 times with different configurations and aging was spotted 2 times. The aged Helix server didn't terminate but entered an inactive state where averagely output bandwidth was only 20KB/s, against almost 100KB/s when the system started. Memory leaks also took place and totally 9 indicators showed significant degrading trend. We will only show some typical indicators as examples because the limited space.

2) Dynamic measuring of aging by AS: We compute AS for each time series to examine whether it can resonablly indicate the severity variation of software aging. A reasonble metric should be:

ALGORITHM1: AS COMPUTATION

BEGIN
$T=\{T_1, T_2, \ldots, T_t\}$ is the trend estimated from $P=\{P_1, P_2, \ldots, P_t\}$

LET *sf*=2, *sfmax*=20

WHILE sf<sfmax

  Split *TR* into *sf* segments, Label $S_1, S_2, \ldots, S_{sf}$

  IF *sf*=2 **THEN**
    $H_1=S_1$, $H_2=S_2$;
  END

  IF increase of P means performance degradation **THEN**
    td=1;
  ELSE IF decrease of P means performance degradation **THEN** *td*= -1;
  END

  $G_{sf} = \frac{\sum_{i=1}^{sf-1} sgn[td \times MWU(S_i, S_{i+1})]}{sf-1}$,

  $where\ sgn(x) = \begin{cases} 1, x > 0 \\ 0, x \le 0 \end{cases}$

  sf=*sf*+2;

END

$G(P,t) = \text{median}(G_2, G_4, G_6, G_8, G_{10}, \ldots, G_{18}, G_{sfmax})$;

$R(P,t) = \text{abs}\left[\frac{median(H_2) - median(H_1)}{median(H_1)}\right]$;

$AS(P,t) = G_2 \times R(P,t) \times G(P,t)$;

END

- When aging becomes more severe, which means, aging causes more amount of degradation or lasts longer or faster, the metric will become higher.
- When aging mitigate, that is to say, the amount of degradation is reduced or degradation stops or slows down, the metric should become lower.
- When aging doesn't take place, the metric should always keep close to zero.

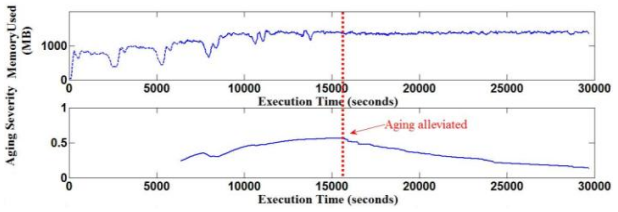


Figure 5.Dynamic aging measurement by AS on memory usage

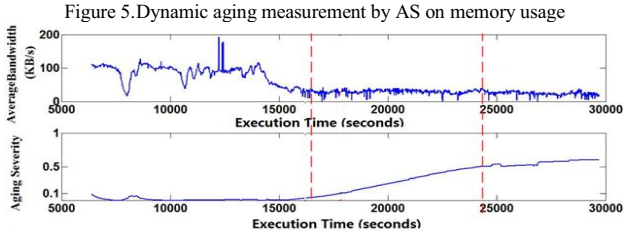


Figure 6. Dynamic aging measurement by AS on bandwidth

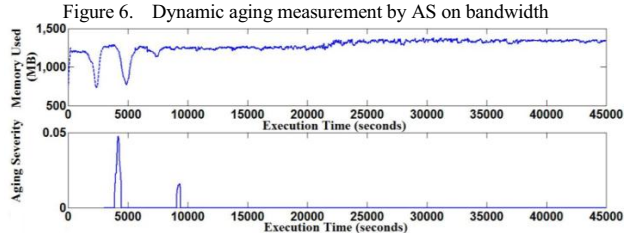


Figure 7.Dynamic aging measurement by AS on memory usage

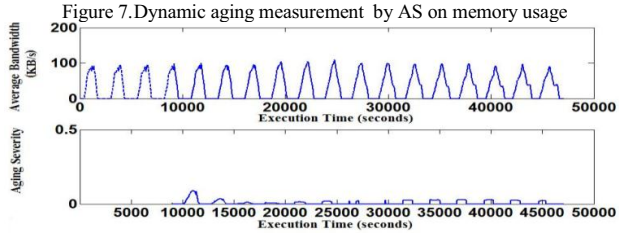


Figure 8. Dynamic aging measurement by AS on Bandwidth

Fig. 5 shows that memory consumption grows much higher after each user period ends (we confirm it is not due to memory caching). As the leaked memory accumulates and lasts, the AS becomes larger and larger. However, when available memory is depleted, the memory consumption stops to grow and AS decreases accordingly.

In Fig. 6, it shows that at first the bandwidth is periodical with no bandwidth loss and AS keeps close to zero accordingly. When bandwidth starts to decline, the AS keeps growing till bandwidth loss mitigate at the end.

In Fig. 7 and Fig. 8, when bandwidth and memory consumption don't show any aging trend, AS keeps to zero except two or three peaks because the time series are not absolutely trend-free in early phase.

We present several examples here, but we checked every time series measured by AS. In most situations, AS can accurately and dynamically describe the aging severity. Moreover, in a few time series, though AS gave reasonable measurement, but it was obviously lagged.

V. CONCLUSIONS

In this paper we pinpoint several limitations of the widely used Sen's slope estimator that is for measuring software aging severity and characterizing aging progress. Its shortcomings can be summarized as: (1) a not unified structure which needs periodicity test and period length inference (2) an oversimplified linear estimation of aging trend (3) unable to distinguish between abrupt change and "aging-like" degradation. We introduce Hodrick-Prescott filter to overcome all these shortcomings. We also propose a composite metric AS based on the estimated nonlinear trend to measure the severity of aging. AS is validated on real aging time series and the results show that it is a reasonable metric.

REFERENCE

[1]. A. Avritzer, E. J. Weyuker, "Estimating the software reliability of smoothly degrading systems," 5th International Symposium on Software Reliability Engineering, IEEE Press, Nov 1994, pp.168-177

[2]. K.S. Trivedi, K. Vaidyanathan and K. Goseva-Popstojanova, "Modeling and analysis of software aging and rejuvenation," Simulation Symposium 33rd Annual(SS 2000), IEEE Press, 2000, pp.270-279

[3]. S. Garg, A. van Moorsel, K. Vaidyanathan and K.S. Trivedi, "A methodology for detection and estimation of software aging," The 9th International Symposium on Software Reliability Engineering, IEEE Press, Nov 1998, pp.283-292

[4]. M. Grottke, L. Lie, K. Vaidyanathan, and K. S. Trivedi, Analysis of software aging in a web server, IEEE Trans. Reliability, vol. 55, no. 3,pp. 411–420, 2006

[5]. A. Bovenzi, D. Cotroneo, R. Pietrantuono, S. Russo, Workload Characterization for Software Aging Analysis. 22st International Symposium on Software Reliability Engineering (ISSRE), 2011 , pp. 240-249

[6]. D. Cotroneo, R. Natella, R. Pietrantuono, S.Russo, "Software Aging analysis of the Linux Operating System," 21st International Symposium on Software Reliability Engineering (ISSRE), 2010 , pp. 71-80

[7]. S. Garg, A. Puliafito M. Telek and K. S. Trivedi, "Analysis of Software Rejuvenation using Markov Regenerative Stochastic Petri Net," In Proc. of the Sixth IEEE Intl. Symp. on Software Reliability Engineering, Toulouse, France, October 1995, pp. 180-187

[8]. J. Alonso, J. Torres, J.L.Berral, R. Gavalda, "Adaptive on-line software aging prediction based on Maching Learning," IEEE/IFIP

[9]. R.Hodrick, E.C.Prescott, "Post-war U.S. business cycles: An Emprirical investigation," Journal of Money, vol. 29,no. 1, 1980

[10]. R.M.Hirsch, J.R.Slack, and R.A.Smith, "Techniques of trend analysis for monthly water quality data," Water Resource Research, vol.18, no.1,1982, pp.107-121

[11]. E. Schlicht, "Estimating the smoothing parameter in the so-called Hodrick-Prescott filter,", Journal of Japan Statistical Association, vol.35, no. 1, 2005, pp. 99-119