Journal of Theoretical and Applied Information Technology
31st July 2019. Vol.97. No 14

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RISK MITIGATION FOR ANTI SOFTWARE AGEING – A SYSTEMATIC LITERATURE REVIEW

[1]THAMARATUL IZZAH AZMAN, [2]NORAINI CHE PA, [3]ROZI NOR HAIZAN NOR, [4]YUSMADI YAH JUSOH

[1,2,3,4]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia.

E-mail: [1]thamaratulizzah@gmail.com, [2]norainip@upm.edu.my, [3]rozinor@upm.edu.my, [4]yusmadi@upm.edu.my

ABSTRACT

Software normally undergoes an inevitable ageing process that occurs because of the software performance degradation and the failures of the software to execute. Usually, maintenance process is implemented to eliminate and fix the emergence of errors triggering software failures. The phenomenon also can be slow down and prevented by considering risks exposure through risk mitigation. To achieve anti software ageing, IT organizations need to employ relevant and effective approach to counteract ageing occurrences, thus ensuring optimum software quality and performance from degradation to fit with current and future environmental and technological changes. For this reason, it is significant to ascertain and understand the concepts of anti software ageing and available approaches addressing the issues to identify existing studies limitation and research gaps. This paper conduct a systematic literature review on the studies related to risk mitigation for anti software ageing by following a standard systematic literature review guidelines proposed by Kitchenham (2009). It aims to review the existing current state of research on risk mitigation for anti software ageing. A total 106 studies related to the topic from various journals and conference proceedings has been reviewed. The research in this topic is still active throughout the decade. We noticed that majority of the proposed solution are focusing on implementing software rejuvenation approach to delay ageing occurrences. Throughout findings, technical risk is the most addressed and discussed risk in the literature. The results of this review are significance as a reference on the current trend of research in anti software ageing. Although software rejuvenation are mainly proposed to handle software ageing issues, there are still no clear solutions and guidelines modeled by past researchers to effectively mitigate risks to software ageing especially during software maintenance process and addressing the risks from external risk. Thus, this paper had achieved its aim to review current state of existing research related to risk mitigation for anti software ageing.

Keywords: *Anti Software Ageing, Software Performance Degradation, Risk Mitigation, Maintenance Process, Systematic Literature Review*

1. INTRODUCTION

Software is a collection of instructions in a computer program that can aid the users to perform tasks on a computer. After software installation and continuous use of it, software needs to be maintained to fix the contingency of software errors. In software development lifecycle (SDLC), software maintenance is the phase where in the event of software errors and failures, actions will be taken to fix errors as well as modifying and updating the software to meet new demands in business process to improve its performance. Most often, the accumulation of these software errors and failures causes the degradation of software performance and leading software ageing to happen. In software development life cycle, software maintenance is the phase where in the events of software failure, action will be taken to fix the errors as well as modifying and updating the software to improve its performance and thus preventing the occurrences of software ageing.

Software ageing is a phenomenon where a software degradation occurs, generally in term of the software performance, function and its quality, due to the exhaustion of operating systems resources, fragmentation and accumulation of numerical error [1,2]. Failure of the software to adapt with technological and environmental changes also will cause the software to age and getting old. Moreover, software ageing phenomenon could be noticed through a

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

continuously long running software execution, where eventually it degrades condition of the software, which consequently resulting in software crash [1]. On the contrary, anti software ageing is referred as an action to prevent and delay software ageing occurrences. Normally, software rejuvenation is a process proposed by most researchers, to counteract the software ageing by classifying the ageing factors and implementing the reverse action to the software system [2]. Software ageing phenomenon can be delayed using risk mitigation approach to solve and treat the risks. Specifically, risk is the possibility of something unpleasant to happen that may cause negative impact. By identifying the causes and factors of software ageing, the potential risks causing the software to degrade could be determined. Although software ageing is inevitable, mitigation approach at the very least could assist in slowing down and delaying the software ageing processes for a better performance and function of software to sustain its lifecycle.

Thus, this paper aims to review the existing current state of research on risk mitigation for anti software ageing. This paper provides an updated current state of the research in the software ageing field on software ageing detection and forecasting analysis, proposed approach as well as discussion on the risks associated to deficiency and failure which gives rise to software ageing which is different from previous work in the area. The structure of this paper is organized as follows: Section 2 presents the materials and method applied in this study, Section 3 summarizes the results on the findings, Section 4 discusses on the results of the findings on risk mitigation for anti software ageing, Section 5 address on the limitations of the study conducted and finally Section 6 is concluded with a conclusion and further research expectation.

2. METHOD

The study will follow the systematic literature review guideline that has been proposed by Kitchenham [3]. This section discusses on the materials and method involved in collecting, analyzing and evaluating existing studies or work related to the topics. We utilized a software program, Mendeley, as for gathering, managing and referencing the journal articles and conference papers. The SLR guideline that has been proposed consists of three main phases; planning the review, conducting the review and reporting the review.

Planning the review: Initial planning of the review is needed to find the motivation to conduct the review and drive the research. This phase involved with identifying the need of the review, build and specify the research questions, and developing as well as evaluating the review protocol. The need may arises from the requirement of the researchers to evaluate and summarize previous studies related to their topics [3].

Research questions need to be formulated and specified, as it is one of the important parts in SLR to drive the entire SLR methodology [3]. Significantly, it is also needed to gain more understanding and overview on the trend, issues and practices of the selected study. The research questions for this study are as follows:

RQ1: How much research has been done to achieve anti software ageing?
RQ2: What are the approaches to detect software ageing?
RQ3: What are the existing proposed approaches to achieve anti software ageing?
RQ4: What are the risks to software ageing?

Aim of research questions

RQ1: This research question aims to provide number of research that has been done on anti software ageing. A clear definition and understanding of anti software ageing and its whole purposes will be discussed. Studies related to anti software ageing will be reviewed to further investigate the aim and purpose of anti software ageing. In addition, it also helps to provide possible areas of study of software ageing.
RQ2: This research question aim to identify what type of analysis is mostly used to detect ageing occurrences in software by past researchers.
RQ3: This research question is directed to provide an overview of the proposed approaches by past researchers to achieve anti software ageing.
RQ4: This research question aim to identify and categorize risks related to software ageing.
Further discussion on this paper will answer these following research questions and help to direct the study in this topic to identify and summarize the issues, challenges and the limitations of the existing research in this area.

Inclusion and exclusion criteria: Developing and evaluating the protocol is the critical part in SLR [3]. It is adapted from the research questions that has been built and specified earlier. One of the components in protocol review is the study selection criteria, consists of inclusion and exclusion criteria, to determine which study should be included and excluded in the review. The criteria

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

to select the primary studies are shown as in Table 1.

Table 1: Inclusion and exclusion criteria

No.	Inclusion	Exclusion
1	Study in the area of risk mitigation for anti software ageing	Does not related to mitigating risks to achieve anti software ageing
2	Submitted to or/and published in a conference or journal articles or books between 2008 to 2018	Informal surveys, books
3	Content are written in English and can be understand	Studies are not structurally developed and not described in detailed

Search terms: Defining and selecting a suitable search term to select primary study is very important in SLR. This search term can be derived from the research question that has been developed. It helps to guide the researchers to find and search for the relevant studies related to the study disciplined. Search term helps to minimize the time to search for the relevant papers related to the study. The search term are also combined using Boolean AND/OR search string for advanced search. The search term that were used are:

- Anti software ageing
- Software ageing
- Software ageing risk
- Software performance degradation
- Risk mitigation
- Risk management
- Risk mitigation model
- Software maintenance
- Software failures

Data sources: Four electronic database that has been selected and used to perform this study which are IEEExplore, Science Direct, SpringerLink and Google Scholar that includes conference proceedings and journal articles.

Search in digital libraries.

In this study, we had chosen four selected digital libraries that are IEEExplore, Science Direct, SpringerLink and Google Scholar. All of the four selected digital libraries and its url are shown as in Table 2. The relevant studies include titles and keywords analysis and comprehensive reading of abstract. We also dismissed duplicates papers that are found in each databases. Particularly, we only selected papers published in journals articles and conference proceedings to focus on only research studies and locating on more important materials.

Table 2 Selected databases and its URL

Database / Source	URL
IEEExplore	ieeexplore.ieee.org/Xplore/home.jsp
Science Direct	https://www.sciencedirect.com/
SpringerLink	https://link.springer.com/
Google Scholar	https://scholar.google.com/

Data collection

The data collected from each of the papers includes attributes as follows:

1. Paper Titles.
2. Paper Author(s).
3. Year of paper's publication.
4. The journal or proceedings in which the paper is published in.
5. Aim of the study and its problem statement.
6. The proposed solution for the problem stated.
7. Type of risks addressed in the study.

Data Analysis

The data collected will be tabulated and discussed to answer the proposed research questions:
RQ1: The number of papers discussing anti software ageing will be reviewed. Area of focus in software ageing field also will be discussed.
RQ2: Type of analysis is used by past researchers to detect ageing occurrences in software will be categorized.
RQ3: The proposed approach or solution of to achieve anti software ageing will be classified.
RQ4: The identification of risks to software ageing will be categorized. A brief explanation of each risk will be discussed.

Conducting the review: This phase involved with carrying out the procedures to identify and select the existing study using the criteria and protocol that has been developed in the previous phase. It consists of steps such as identification of the research, retrieving the studies from the data sources by using the search term as described. The relevant studies or paper related to risk mitigation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and software ageing disciplined are retrieved from the four selected electronic databases that provide The selected papers are then narrowed down into selecting papers that were published from the year of 2008 to 2018 to gather relevant and newer studies on the topic. Before conducting the paper selection process, we generated the search string term as steps followed:

Step 1: Identify the keywords from the area of research. The keywords are derived from the search term described.

Step 2: Define the generic search expressions as shown in Table 3. [Example of expression: A_1 OR A_2 OR A_3OR AND B_1OR B_2OR B_3OR B_4 AND C_1]

Step 3: Perform the generic search expression using the string in the selected digital libraries.

Table 3 Domain and Keywords

Domain	Key	Keywords/String
Anti Software Ageing	A	A_1 : Anti software ageing
		A_2 : Software ageing
		A_3 : Software performance degradation
		A_4 : Software failures
Risk Mitigation	B	B_1 : Risk
		B_2 : Risk mitigation
		B_3 : Risk management
		B_4 : Risk mitigation model
Software Maintenance	C	C_1 : Software maintenance

The generic search expression is used for paper selection process to conduct systematic literature review. The process is performed as shown in Figure 1. Overall, the phases to select suitable conference proceedings and journal articles.

papers that fits criterion defined involved seven phases. In advance, we also looked at the references of the selected papers and search for its publications in the database that are related to the topic. This is done to obtain more published papers that might not be available in our selected databases or are missed.

The selected papers are then to be used to answer the defined research questions. After a full paper reading, we only selected 106 papers for data extraction and analysis to conduct a review report.

Reporting the review: A comprehensive study will be done through full reading of the selected papers and the report of the review will be discussed in section III. The report of the review is reported accordingly by each of the research questions formulated.

3. RESULTS

In this section, we summarize the findings and results of our studies. A total of 106 studies has been gathered and reviewed from four digital libraries. Each of the relevant papers found are classified accordingly to its year of publications as in Figure 2. We also classify the reviewed papers into two types of publication; journal articles and conference proceedings. Table 4 provides an overview of total number of papers published through journal and conference proceedings along with its percentage.

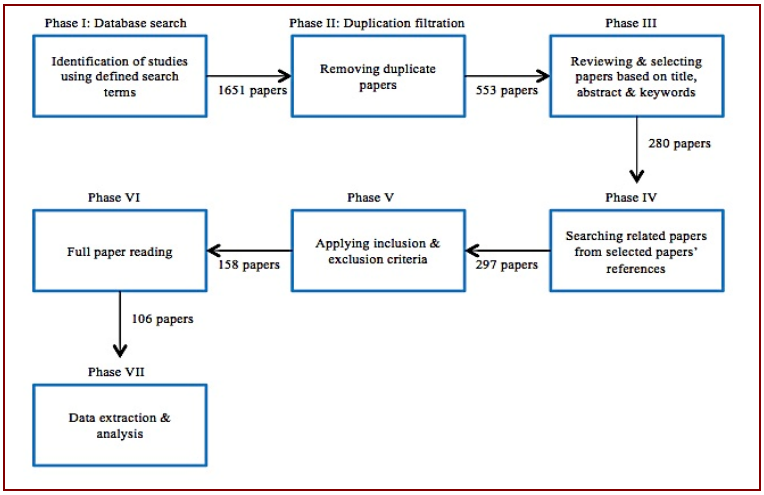

Figure 1 Paper Selection Process

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

To answer research question RQ1, we found out that 106 studies has been done on anti software ageing. The definition and broad explanation of anti software ageing are briefly explained in section 4.1. Thorough reviews of each paper leads to the summarization and categorization of software ageing area of focus into two categories; software ageing detection and software ageing mitigation as shown in Table 5.

To answer research question RQ2, ageing detection approaches are divided into three type of analysis. The papers are then classified accordingly to each analysis used to detect ageing in software with its percentage as tabulated in Table 6. We also provide description and explanation on the analysis used by the researchers in section 4.2. Moreover, we also summarize a detailed view of all information on approaches as well as the ageing indicator used to measure ageing in measurement-based approach for research question RQ2 as shown in Appendix Table 4.2A.

On the other hand, Table 7 provides an overview of seven classes of proposed software ageing mitigation approach by the studies to answer RQ3. Brief discussion on the proposed mitigation approaches is explained in section 4.3.

The risks to software ageing also had been identified and discussed to answer RQ4. Further discussion on the identified risks is explained in section 4.4. Table 8 categorizes previous studies with the risk addressed and discussed by the authors. We also identified those studies that addressed more than one risks.

Lastly, Table 9 in Appendix provides a detailed view of information from reviewed papers contributing to answer research questions RQ1-RQ4.

4. DISCUSSION

4.1 RQ1: How much research has been done to achieve anti software ageing?

Overall, we had identified and gathered 106 studies from four selected digital libraries. Figure 2 illustrates the graph of publications of the reviewed studies by 1-year interval. From the graph, the data shows the trend of research on the topic is active and on going every year, except in the year 2014. However, the research shows an increase trend the year after and even more active in year 2017. The rising trend could be a sign of increasing interest to this field among researchers. Citation for each paper is included in the reference section.

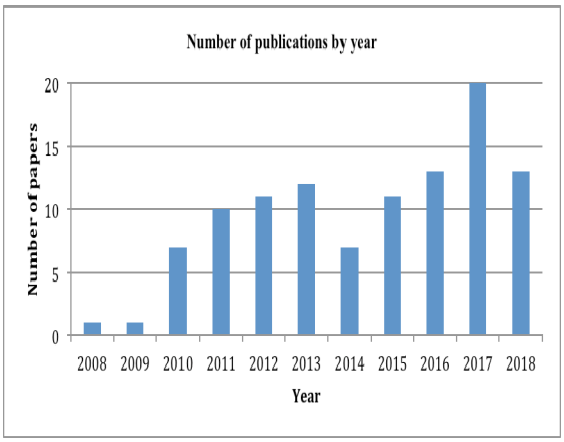

Figure 2 Number of publications by year 1-year interval

From 106 papers, 45 (42.5%) of the publications are articles published in variety of journals publication, some of which published in high reputations such as IEEE, ACM and Elsevier. The articles cover different areas of studies ranging from Software Engineering, Computer Science, Information Systems and etc. Meanwhile, 61 (57.5%) of the papers are published in various international conference proceedings, which are mostly from IEEE conferences and ACM. Table 5 shows the classification of papers into journal articles and conference proceedings. Overall, the publications of reviewed studies are diverse across various journals and proceedings. In particular, the classification is important to help researchers to view and differentiate the number of studies published in each class in an organized and structural manner. From Table 5, we noticed that publication of reviewed papers is published slightly more through conference proceedings than through journals. Nevertheless, conference proceedings may process a faster publication than journal articles, however, journal articles is highly significant as it provides a more in-depth and detailed research than conference proceedings.

Table 4 Type of publications

Classification	References	No. of studies	Percentage (%)
Journal Articles	[2] [4-47]	45	42.5%
Conference Proceedings	[48-108]	61	57.5%
Total		**106**	

Anti software ageing is a process to counteract, delay or slowing down the ageing process in software by identifying the factors and causes of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

software ageing [2,27]. It helps to prevent the software from degrading in terms of its performance and quality [47]. In other words, anti software ageing restrain the chances of software errors and failures to sustain the survivability of software thus prolonging its lifecycle. A good quality of software helps in preserving the software from ageing [47]. Software that maintain its high quality and possess adaptability feature towards its environmental changes enable the software to stay young and relevant [2,19,104].

To achieve anti software ageing, the concept of software ageing must be fully understood. Software ageing is described as the accumulation of errors during software execution, resulting in a crash or failures of the software [1,2]. In particular, software ageing is not a phenomenon where it will immediately cause the system to fail, but rather it will cause software performance to degrade up to the point where the software will crash and hangs [3]. Software ageing phenomenon also known to be associated with the interruption of software maintenance activities that consequently lessen the software quality [5]. In fact, software also may suffer from ageing faster whenever it is untended [5]. Software also may undergo ageing process due to slower response time which resulting to slower software performance. However, by updating the software to a new version, ageing could be overcome [23]. Early studies of software ageing stated that there are two categories of software ageing; first, failure of the product owners to make alterations to the product to meet the changing needs, and second is the results of the changes that are made [4,47]. Generally, to understand the phenomenon, the factors, causes and risks of software ageing are crucial to be identified to achieve anti software ageing.

Past researchers [2,47] suggested two factors leading to software ageing; internal and external factors. The internal factors includes memory bloating, residual disability, memory leak, unreleased files lock, debug failure, quality degradation and drop, and increase complexity meanwhile the external factors includes technology challenge (hardware and software), maintenance cost, competition, evolution requirement, dynamic environment, business stability and consistency, and human [2,19,47]. Internal factors are the factors that are closely related to the function of the system software meanwhile the external factors are associated with the environment of the software. Few past researchers studies analyses software ageing in term of its external factors, dealing with the practitioners and users to assess and measure the software ageing [19,44,97].

Other researchers focuses on the software internal factors to measure software ageing, particularly dealing with aging-related bugs, data corruption fragmentation, memory leaks, failure rates and numerical error accumulation [6,46,51,52,54,55]. Previous studies in software ageing and rejuvenation also focus on the internal factors of the software, specifically on the operating systems, its hardware and software [51,53,55].

Previous researchers [5,19] suggested few main factors of software ageing that has been identified and verified, and categorized them into four factors, which are functional, human, environment and product profile. Functional factors are the factor from the software product itself which user used to interact. This factor is related to the functionalities of the system that may caused the software to age such as memory bloating and leak, errors and bugs and residual defects. Second factor highlighted by the researchers is human which are closely related to the users that used the software or system, for example, employee, end user and consumer. The researchers [5,19] suggest that inexperience and lack of training of people in handling and managing the software may also causes software ageing to happen. Meanwhile, environmental factors comprises of the environment within or outside the system itself, for example, accessories, environmental and technological changes. Inadaptability of software towards its environmental and technological changes may leads to the complexity of the software itself to be updated and modified thus leading to ageing in a software. Lastly, the product profile factor involves factors in relation with the product of the software itself. For example, date of acquisition of the product, date of purchasing the software and age index of the software. The software ageing factors has been identified and verified through empirical studies conducted among software practitioners by the researchers [5].

Software ageing also can be measured using data collected from a system about it resources usage. Internal measure is a measurement based on the internal attributes such as static measures of product, for example, its functionality and interoperability [2]. This measurement is very extensive as it involves with statistical measurement of the software product itself that deals with the software ageing. Some of the measurements are very expensive and hard to apply in assessing software ageing. However, this

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

measurement provides precise statistical measurement to detect software ageing internally.

Table 5 gives an overview on which areas of software ageing are more focused into. In brief, 106 studies are classified into two main areas of focus; software ageing detection and mitigation approach.

Table 5 Area of focus

Area of focus	References	No. of studies	Percentage (%)
Ageing detection	[5,8,12,18, 21,25,30,34, 50-54,57,63, 64,66,77-79, 82,84,85,87, 88,93,94, 100-102, 105-108]	34	32.1%
Mitigation approach	[2,7,9,10,11, 13-15,17,19, 20,23,24,28, 29,31,33, 35-49,56, 58-62,67-69,72-76,81, 90-92,95-99,103,104]	57	53.8%
Combination	[4,6,16,22,26,27,32,55, 65,70,71,80, 83,86,89]	15	14.2%
Total		**106**	

From Table 5, we distinguish 34 (32.1%) studies are focusing on detecting ageing in software. Meanwhile, 57 (53.8%) studies are discussing on delaying software ageing through various ageing mitigation approaches, discussed in section 4.2. On the other hand, 15 (14.2%) studies are discussing on both ageing detection and mitigation. The diverse approaches of ageing mitigation for anti software ageing are discussed in section 4.3.

Hence, we could see the current state of area of focus in software ageing field where research on proposing mitigation approach for software ageing are quite numerous than research on ageing detection and the combination of both studies (ageing detection and mitigation). Ageing detection involves software system observation and monitoring to locate potential unusual behavior within software. Precise prediction is crucial to assure software system reliability [101]. Moreover, forecasting and detection ageing in software as well includes predicting time to ageing failure, which mainly past researchers tries to improve one another studies by obtaining accurate time to ageing failure to perform rejuvenation. Ageing detection involves applying analysis that includes measurement-based analysis, model-based analysis and hybrid analysis (integration of both measurement- and model-based analysis). Further explanation on the three analyses for ageing detection will be discussed in the next section. Meanwhile, mitigation approach involves with introducing and suggesting best suitable solutions to delay software ageing phenomenon. There are various approach proposed by past researchers to prevent software ageing that includes software rejuvenation, application of maintenance activities, partial computation offloading, software life extension, recovery actions and etc. that will be discussed in section 4.3.

As observed, attention has been given more thorough to delay ageing occurrences rather than detecting the causes of ageing itself. This might be because locating aging-related bugs incurring ageing manifestation within the software system might be difficult even after software system failure [15]. Furthermore, ageing forecasting and prediction also is argued to be a critical area of study since it involves crucial monitoring of the state of software system using certain instrument. Thus, this requires greater effort for successful ageing observation. Moreover, studies that combine both ageing detection and mitigation are still scarce. Whilst studies on ageing detection necessitate great effort, studies that reconcile both areas might acquire a greater effort as it involves both precise observation of ageing manifestation and effective ageing delaying process. However, this could open ways and ideas future studies/research to develop a more effective mechanism and approach for ageing detection and mitigation.

4.2 RQ2: What are the approaches to detect software ageing?

Ageing detection involves with predicting and forecasting ageing events in software/system using specific approach. Overall, there are 34 studies discussed on ageing detection only meanwhile 15 studies discussed on both ageing detection and mitigation. Hence, this concludes that there are overall 50 studies discussed on ageing detection in software. In particular, literature study revealed that there are three major type of approach used by past researchers to detect ageing; measurement-based, model-based and hybrid ageing analysis.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Through review, we noticed 37 (75.5%) out of 49 studies are discussing ageing detection using measurement-based analysis, seven (14.3%) studies uses model-based analysis, four (8.2%) studies detecting ageing using hybrid analysis, meanwhile one (2%) study proposes a new heuristic approach. Table 6 shows the classification of the studies with the type of analysis.

Table 6 Type of Analysis for Ageing Detection

Type of Analysis	References	No. of studies	Percentage (%)
Measurement-based	[4,6,8,12,16, 18,25-27,30, 51-55,64,65, 71,77-80,82, 84,86-89,93, 94,100,102, 105-108]	37	75.5%
Model-based	[5,21,32,50, 57,66,83]	7	14.3%
Hybrid	[22,63,70, 101]	4	8.2%
Others	[34]	1	2.0%
Total		**49**	

From the obtained findings above, measurement-based analysis is the most preferred analysis to detect software ageing by past researchers. It involves with statistical approach by monitoring the software/system behavior directly using aging indicators to predict and forecast aging phenomenon. Aging indicators act as markers to identify the presence of ageing in software [51]. Examples of aging indicators that are widely used as a mean to detect ageing in software/system are resource usage (memory consumption and CPU utilization), response time, memory-related (memory leak and memory fragmentation) and performance indicator. Some studies not only measures ageing using one indicator but also a combination of several ageing indicators as well. In particular, measurement-based analysis uses several approaches to forecast ageing such as time-series approach and machine learning approach. Time-series approach is an approach based on the trend estimation of selected ageing indicators. The trend test also is used to reject or accept the hypothesis of no trend in the data for example Mann-Kendall test used in [84,94] and applying technique to estimate the trend such as linear regression or multiple linear regressions as used in [6,55,78,79,87,108]. Meanwhile, machine learning approach involves with adopting algorithm to identify trend and categorize the software/system into robust state or failure prone state [7,65,105,107]. This approach applies non-linear or non-parametric regression to diverse software/system variables and examines the remaining errors between the predicted and actual software/system values [87].

Model-based analysis involves with adopting stochastic processes to model the aging phenomenon. Seven (14.3%) studies used model-based analysis in their research to detect ageing occurrences. This type of analysis assumes software/system failure and repair time distribution to determine rejuvenation schedule to maximize the software/system variables while minimizing costs and its management [50,66]. Researchers adopted several models to forecast ageing in software such as semi-Markov model [39,83], the Gaussian Mixture Model [21], square quality model [50], ageing index [5,57] and typical polynomial mathematical model [66] to characterize the ageing state of software/system.

Meanwhile, another four papers applied the hybrid-analysis in their studies. Hybrid analysis is a combination of both measurement- and model-based approach. It adopts a stochastic process in model-based approach to describe ageing phenomenon and determining the model parameter by using measurement approach. In [22], the combination of Markov model and time-series techniques is used for ageing detection. In [63], a continuous time-hidden Markov Model (CTHMM) is proposed. The researchers extend the hidden Markov Model (HMM) with a continuous time domain. In [70], a hybrid approach is adopted to analyze software ageing in android. Trend estimation technique is applied which uses memory availability as aging indicator and later uses Markov model to conduct predictive analysis.

On the other hand, only one study in [34] made comparison and exploits the differences between software version as mean to predict and forecast ageing. The ageing is detected by runtime comparisons of different development versions of the same software and analyzing the differences in runtime traces of chosen metric (memory depletion). It utilizes the information gathered via the differentiation between software versions, which is a different type of analysis from general analysis used by other researchers.

Thus, we could see the current states of ageing analysis are slightly more favoured to measurement-based approach than model-based and hybrid approaches. This might be because measurement-based approach uses statistical analysis to observe and monitor data directly to ascertain time window to conduct rejuvenation. The benefit of measurement-based is that it allows data

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

to be directly from the actual system and able to incorporate changes in dynamic environment [15]. Another advantage of this type of analysis is that it produces accurate prediction of ageing occurrences [16]. Furthermore, it allows evaluation of current and future state of software system assembling of parameters at certain extent time [105]. However, this analysis is unable to assess and evaluate persisting dependability attributes such as availability as measurement-based analysis utilizes ageing indicators associated to exhaustion of resources [44]. It is also argued in [106] that most researchers only utilize only one ageing indicator to describe the data in measurement-based analysis, whilst multiple ageing indicators are yet to be exploit.

Meanwhile, model-based analysis is used to analyze and evaluate software system behavior by means of software system availability and performance. It is also seeks to determine the optimal time to perform rejuvenation. It establishes mathematical model and forecast ageing based on historical data [58]. Model-based analysis does not make use of any metrics (ageing indicators as used in measurement-based analysis) upon determining possible ageing state of the software system as it uses dependability attributes for the analysis. Significantly, model-based analysis is easy to be applied across different software systems as it based on the simplification of the software system [44]. The accuracy of this approach however relies on the presumption and estimation made for the model. Another drawbacks include poor flexibility, unable to be validated easily in practice, incapable of making decisions in real time and unsuitable to be utilized for software ageing issues during production phase [105].

On the other hand, hybrid analysis reconciles both approach by the integration of measurement data and stochastic model to illustrate the phenomenon. It utilizes field data to support analytical models to forecast ageing phenomenon. This type of analysis leverages both measurement- and model-based values and offers more effective results of analysis. Based on the review conducted, despite its significant merit, studies on hybrid analysis are yet to be explored.

In brief, ageing analysis involves forecasting and prediction of potential events/caused of software ageing occurrences, determining plausible time to ageing failure and is conjointly associated with software rejuvenation approach to tackle and deal with software ageing occurrences as it as well involves ascertaining best time to perform rejuvenation. Precise time to ageing is crucial to commence countermeasures to delay 0ageing occurrences [4].

An overview of the studies with its type of analysis and approach used in each analysis as well as its ageing indicator contributed for this research question RQ2 is presented in Appendix Table 4.2A.

4.3 RQ3: What are the proposed mitigation approaches to achieve anti software ageing?

To answer RQ3, approach involves with implementing method or techniques to prevent software ageing occurrences such as failures and performance degradation is shown as in Table 7. Through review, there are 56 studies that focused on ageing mitigation only and 16 studies on the combination of both detection and mitigation. This summarize that there are overall 72 studies suggested mitigation approaches to achieve anti software ageing. The proposed approaches are many and varied such as software rejuvenation, maintenance activities, software life extension, partial computation offloading, software regeneration strategies, recovery actions.

We also had identified eleven risk mitigation models that incorporates various risk activities to minimize risks in software leading to failure.

Table 7 Proposed approaches

Approaches	References	No. of studies	Percentage (%)
Software rejuvenation	[4,6,7,10,13, 15-17,20,22,23, 26,27,29,31-33, 35-37,43-47,49, 55,56,58,59,65, 67-69,70-72, 74-76,80,81,83, 86,89,90,92,96, 98,103,104]	50	69.4%
Maintenance activities	[2,19,24,60,68,97]	6	8.3%
Software life extension	[14,91]	2	2.8%
Partial computation offloading	[73]	1	1.4%
Software regeneration strategies	[95]	1	1.4%
Recovery actions	[28]	1	1.4%

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Mitigation model	[9,11,38,39,40,41,42,48,61,62,99]	11	15.3%
Total		**72**	

Software rejuvenation

Thorugh our review, software rejuvenation is an approach that are mostly emphasized by past researchers by having 51 (83.6%) out of 61 studies. It is an action or technique of proactive faults that designs the system for periodic reboots [4,57]. In particular, software rejuvenation is the concept where an application will be restarted periodically and preemptively at a clean internal state after every rejuvenation interval [2]. It performs several operations such as cleaning up file systems, disposing buffer queues, resetting internal kernel tables as well as garbage cleaning [4]. From findings, software rejuvenation approach is divided into main focus; internal factors and external factors.

Software rejuvenation approach that focuses on rejuvenating internal ageing factors such as memory-related, operating system and resources consumption is divided into two categories of rejuvenation scheduling; time-based and inspection-based. Time-based rejuvenation is a form rejuvenation activities performed at a pre-determined time intervals [33]. It is an open-loop control technique [44]. This type of rejuvenation is used in [7,29,35,43,47,70,80,90].

On the other hand, inspection-based is a form of rejuvenation triggered based on measuring the attributes of aging effects whenever it crosses certain fixed limits [33]. It is a closed-loop control technique [44]. Inspection-based rejuvenation is used in [17,23,27,32,33,36,37,44,65,67,71].

Furthermore, another type of rejuvenation includes threshold-based, where it is an action that monitor the aging effects using threshold value, triggering the rejuvenation whenever it exceeds certain pre-fixed threshold value [33]. It is found in [10,13,31,49,55,58,59,68,75,86].

Meanwhile, prediction-based rejuvenation is a predictive approach to estimate the time to failure or time to resources exhaustion caused by the software aging [33], found in [4,6,16,46,56,81,83,98]. Another type of approach used by researches in [45,92,103] is condition-based approach where a certain type of event or command that will trigger software rejuvenation activities to start, in response to the state. Moreover, this type of rejuvenation is combined with time-based rejuvenation used in [96]. The study derived the condition for the optimal rejuvenation time under opportunity time-triggered rejuvenation policy. On the other hand, another combination of rejuvenation policy used are time-and workload-based combination used in [15], time-and prediction-based used in [22,69,72] and time-and threshold-based used in [26,74].

On the other hand, researchers in [20,150] focuses on software rejuvenation strategies that are handles external factors of software aging such as environmental and technological changes, human errors, change in business requirement and hardware aging. In brief, this type of software rejuvenation tackles aging problem in software by mitigating aging-related failures in the design and maintenance stage [20].

Maintenance activities

Maintenance activities are proposed by five (8.2%) studies to handle the occurrences of software ageing in software. Software maintenance is the dynamic behavior of a programming software or system, whenever they are maintained and involved in any modification throughout their life cycle, where mostly, software ageing occurs because of the failure of the software or system to cope with the current changes in its environment [19]. Past researchers emphasized on the importance of software maintenance in a software as software that do not adapt to its environment changes and inflexible may cause software ageing to occur [2,19,60]. In [24], maintenance policies by using corrective and preventive maintenance are used to improve the efficiency of the system. Maintenance is performed to improve the reliability and availability of the system by optimizing the components and procedures [24]. Researchers in [60] measure the complexity of software in software evolution and maintenance and emphasized that the complexity of the software makes it difficult for the software to be changed. Researchers [19] proposed a prevention technique to counteract software ageing by using software maintenance activities/processes that are preventive, perfective, corrective and adaptive maintenance. The researchers also emphasized on the importance of software maintenance activities as it helps to assist practitioners in delaying ageing processes and preserve the software from degrading [19]. The anti-ageing metrics includes software performance, software usefulness, software failure, business demand, changes in the environment, changes in the technology and expertise. Each metrics are suggested with software maintenance prevention technique to help prevent software ageing to happen.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Software life extension

Software life extension is a technique of prolonging the lifetime of software execution whenever the software failure that leads to software ageing occurs [91]. It is proposed in two (3.3%) studies. Software life extension is a preventive maintenance technique that could be implemented to delay the occurrences of software ageing temporarily [14,91]. There are two methods to extend the software lifetime that are dynamic resource allocation by restoring resources that are depleted due to software ageing and workload control by lessening the workloads of software that are ageing [14,91]. Researchers in [14] implemented software life extension by increasing the duration of software execution and used hybrid approach, a combination of software life extension with rejuvenation strategies that shows intended result in delaying software ageing. Meanwhile, researchers in [91] allocates extra memory to the virtual machine which has depleting memory consumption and shows the effectiveness of software life extension technique compared to software rejuvenation.

Software regeneration strategies

Researchers in [95] proposed software regeneration strategy to handle software ageing. The regeneration strategy is implemented in component-level, composed of two parts, first, by determining the degree of software aging states in each component by threshold interval and using the results, regeneration strategy is implemented. The regeneration strategy includes component recovery using serial mode, changing the state of aging component back to normal state.

Recovery actions

Researchers in [28] proposed a recovery actions model to handle software failures caused by Mandelbugs. The model aim to lessen unplanned downtime in business-critical applications using proposed recovery actions taken by IT operations staff. In particular, system failures are detected either by automatic-detection using enterprise system management tools or manual-detection. The problem is then diagnosed and recovery actions are taken using four approaches; restart, reboot, reconfigure and hot-fix (involves with minor change in the source code or changes to the software system such as OS, run-time and library code). The model handles four type of bugs; restart-maskable Mandelbugs, reboot-maskable Mandelbugs, reconf-maskable Mandelbugs and Bohrbugs.

Partial computation offloading

Other researchers in [73] proposed partial computation offloading performed in application-level. The proposed solution is a using application-partitioning algorithm where it divides the application into local and remote parts. This algorithm automatically determines which parts of application tasks to be processed in cloud server meanwhile another parts stays on mobile device to ensure high performance results [73].

Overall, there are eleven studies [9,11,38, 39,40,41,42,48,61,62,99] that proposed mitigation model which involved with processes such as risk identification, risk assessment, risk decision, risk avoidance, risk treatment, risk monitoring, risk evaluation, risk control and risk measurement. Risk mitigation concentrates on determining strategic and tactical approaches in minimizing the effects of identified risks in a system [9]. It helps practitioners or management to understand and control the risk that might occur in a software or system [6]. Risk mitigation usually involves basic activities such as identification of risks, risk treatment, risk decision and risk monitoring. Other activities such as risk evaluation, risk assessment, risk avoidance, risk reducing, risk control and risk measurement can also be included in risk mitigation processes [41,42]. However, most models focuses on mitigating risks during software development to ensure delivering quality software to meet users expectation and current technological future demand to avoid software failures, only one study in [11] mitigate software function and faults after software has been implemented. Thus, this creates the gap in the study in the development of risk model to tackle software ageing especially during software maintenance since there is lack of risk mitigation approach during the phase. These models could be used as for guidelines in the development of future risk mitigation model for anti software ageing during software maintenance.

Through review, software rejuvenation technique is still leading as a dominant way to mitigate ageing occurrences in software as shown in Table 7. It is worth noting that most of software rejuvenation techniques proposed from the studies tackles software ageing issues internally by using factors as such of memory-related, resource

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

consumption and availability, and aging-related bugs. At the same time, another available approaches also handle software ageing by addressing software ageing caused by software internal factor specifically software technical update and malfunction. For this reason, we also could see the trend by applying mathematical and analytical measures to address software ageing phenomenon internally, which disregard another factor leading to software ageing that might not arises from the deficiencies and errors of the software system function itself. In particular, only studies in [2,19,20,97,104,150] addressed software ageing by proposing software rejuvenation actions and maintenance activities using external factors such as human, environmental and technological changes, from the viewpoint of hardware and software environment.

Each studies developed different strategies to conduct software rejuvenation in various software environment in order to effectively delaying software ageing occurrences. As argued in [12], prior work focused more on software rejuvenation to deal with software ageing. In spite of the fact that software rejuvenation has the capability to tackle software ageing effectively, however, it is a proactive approach employed on running software system without eliminating and fixing potential bugs and errors in the code [12]. Thus, these errors and bugs can still manifests in the future leading to software degradation or crash. Moreover, one of the drawbacks of this approach is it increases system downtime and acquires costs to the loss of business and due to software unavailability [23]. Contrarily, other approach attempts to address software ageing by prolonging and extending the lifetime of software lifecycle. This is done through controlling of resources and workloads by allocation additional resources to the software to increase its lifecycle [91]. Despite its attempt to add the lifespan of the software, the approach seemingly does not directly fixing ageing manifestation within the software. Another approach by maintaining the software by performing maintenance activities however has potential to increase the complexity and declining the structural architecture and design of the software after frequent maintenance although it addresses both internal and external caused of software ageing. Some other approach also suggested on dealing with ageing by component- and application-level, however, it involves great effort to disintegrate the modules of components and applications to forecast which modules are infected by ageing. Meanwhile, recovery actions proposed by past researchers only deals with aging-related bugs, which disregard external caused of ageing. From the analysis, one of the shortfall from existing proposed approach for anti software ageing is to deal and tackle software ageing by means of external settings that may contributes to software ageing phenomenon.

4.4 RQ4: What are the risks to software ageing?

From the findings, risks addressed and discussed emerges based on the factor software ageing are grouped into three general types of risk; technical risk, organizational risk and business risk. Technical risk is the most discussed risk from the literature study with overall 86 (81.1%) out of 106 studies, followed by 12 (11.3%) studies discussing on all of the risks to software ageing and four (3.8%) studies discussed on the combination of technical and business risks. Meanwhile, there are only one (0.9%) study discussed on business risk, one (0.9%) study on the combination of both organizational and business risks, and another one (0.9%) study on both technical and organizational risks. However, we noticed that there are none of the studies that discusses on only organizational risk. Table 8 shows the classification of the studies and type of risks addressed.

Technical risk emerges from functional failures of software and unexpected behaviors of the system which affecting the software performance [5,30,49,81]. It also involves with the quality of the software produced [39]. It refers to a set of internal failures of software such as memory-related [6,18,25,33,34,51,52,63,70,78,82,83,88,90,93,102], internal resource unavailability or consumption [4,13,14,16,26,46,73,74,89,91], response time [15,68,92,101], number of jobs queue [84], low processing rate [31], throughput loss [94,95] and aging-related bugs [8,12,28,32,67,77,100]. Hence, in particular, technical risk most often related to the internal factors subjecting to software ageing which makes the software being liable to failures and error-prone thus leading to ageing occurrences.

Operational risk is the risk events that occur in results from the inadequacy from people (human factors) [2,5,19]. It most often related to the loss associated with people having lack of experience and capabilities in the specific area of expert [9]. It also emerges from the events of management failures in handling software [2] and scarce resources within the organization [2]. Organizational risk may arise by the issues of mistrust and failure to communicate among staff members in the organization [9,19].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Business risk in general affected the growth and development of software product [39]. It incorporates external factors of software ageing such as environmental [2,5,41] and technological [2,5,19] factors [39]. Business risk also may occur from the evolution and changes in business requirement [5,19,20]. Consequently, it may affect business stability and consistency in delivering a good quality of software [19].

Table 8 Risks to software ageing

Type of risks	References	No. of papers	Percentage (%)
Technical	[4,6-8,10-18,21-23, 25-37,43-47,49,51-53, 55,56,58,59, 63-96,98, 100-103, 105-108]	86	81.1%
Operational	-	0	-
Business	[61]	1	0.9%
Technical & Operational	[24]	1	0.9%
Operational & Business	[38]	1	0.9%
Technical & Business	[39,48,62,99]	4	3.8%
All	[2,5,9,19,20, 40,41,50,57, 60,97,104]	12	11.3%
Not specified	[42]	1	0.9%
Total		**106**	

In brief, the scope of risks associated is estimated to increase impact stemming from the deficiencies or failures that gives rise to software ageing occurrences. From the review, we could observe the current state of risk discussed by past researchers is more to technical risks than business and operational risks. This might be because technical risks could be minimized internally using direct mitigation strategies implemented onto the software system for risks minimization, whilst, operational and business risks requires greater effort from people and management as the likelihood of risks occurrences cannot be seen directly as technical risks. This concludes that risks from external viewpoint has been paid little attention compared to risks emerges from internal viewpoint. There are studies that discussed on all of the risks collectively, however, the studies focused on the three risks are still scarce and insufficient. Future work could put more on effort to develop approach that assist in addressing all of the risks and possibly locate other potential risks associated with software ageing. This could help to minimize the likelihood of risk impact to assist in delaying software ageing occurrences from internal and external prospect entirely.

In this review, we had conducted an updated state of research in this field for the past 10 years ranging from year 2008 to year 2018. From the findings, we had identified and classified past studies area of focus into two scope, which are (1) study on detecting/predicting ageing occurrences and (2) study on proposing approach to mitigate ageing occurrences. We had analyzed and classified the type of analysis used by past researchers to predict and forecast software ageing occurrences into three type of analysis that are model-based, measurement-based and hybrid approach. On the other hand, we also had discovered mitigation approach proposed by past researchers to mitigate the influence of software ageing into several categories; software rejuvenation, maintenance activities, software life extension, software regeneration strategies, recovery actions and partial computation offloading. Moreover, we also had categorized risks based on the factor of software ageing into three types of risk: technical, operational and business risks. This work is different from previous review in the field of anti software ageing as most prior review lack of discussion on risks associated with software ageing phenomenon as they mostly discussed on the causes and type of analysis of software ageing. Furthermore, this review is updated with more current existing studies up to 2018, which contain more additional and fresh different approaches and analysis in the field of software ageing.

From the analysis, existing studies discussed on mitigating software ageing occurrences is highly more than existing studies that focused on forecasting/predicting software ageing occurrences. Nevertheless, each one of the studies tries to improve one another method/approach to accurately predict and forecast software ageing occurrences to detect which internal components of the software causes software ageing to manifest and become apparent. Conversely, the main interest of our review is on mitigation of software ageing occurrences. From the analysis of our findings, we discovered most existing studies proposed software rejuvenation to mitigate ageing influences in software. These create a huge research gap on the proposed approach for anti software ageing. Other existing approach as well has certain limitations and drawbacks need further refinement for effective solution to slow

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

down software ageing manifestation. Furthermore, from the analysis of risks discussed clearly revealed that software ageing occurrences arises by external aspect have been paid a little attention. Hence, this could pave the way for further research to propose anti software ageing approach, which reconciles both ageing internal and external risks/factors mitigation.

5. LIMITATIONS OF THE STUDY

In spite of comprehensive study we had conducted on 106 articles related to risk mitigation for anti software ageing, we can still address few limitations to our study. The study may not been extensive as we limits the papers publications for our study from the year of 2008 to 2018. Few of the studies before the year of 2007 may still be relevant and significant in the area, which are not included in the review. Secondly, our study also may not be thorough. We might neglect and missed few of the relevant papers related to the topic as we only select four databases for the review. Third, the study may be biased due to the researchers' knowledge and understanding on the topic related. Few of relevant studies might possibly be missed during papers abstract filtration and full reading process.

6. CONCLUSION

In this study, we have conducted a systematic literature review of 106 articles related to risk mitigation for anti software ageing. Thus, this paper had achieved its aim to review existing research related to risk mitigation for anti software ageing and discuss the existing research based on the research questions formulated. Knowledge from this review is significant to be used as for further work in the area. From the review, we could see the trend of area of focus in software ageing field are more drawn to ageing mitigation than ageing detection and combination of both (ageing detection and mitigation). This is because it is argued by past researchers that studies involve with detecting and forecasting ageing manifestation in software are more challenging to be done [15]. For this reason, from our review we could see the studies in this area are still limited. Likewise, studies that combine both areas are also still scarce as it incurs considerably greater effort.

From the findings, there are numerous approaches and solutions to address software ageing phenomenon and each one of the approaches drives to the same goal, which is to delay and reduce the occurrences of software ageing for a better software performance and quality. Since it is impossible to eliminate ageing in software, most of researchers developed various and diverse alternatives to prevent its occurrences. Most often the researchers proposed software rejuvenation process as a prevention action to delay the ageing process in software. The techniques used to perform software rejuvenation are many and diverse as researchers are trying to improve and produce best solution and strategies to fit with the current demands in technology. In particular, we could see the trend by applying mathematical and analytical measures to address software ageing. However, there are still lack of approaches that address software ageing from the viewpoint of external software and hardware environment and technological challenges as many of the existing studies focus on tackling software ageing by means of internal factors of ageing. Thus, we could see a huge gap of research on the proposed approach for anti software ageing. On the other hand, few suggested mitigation model, which uses risk management and mitigation activities, only addresses risks during software development process, before implementing the software to market. Only one study proposes mitigation model after the software has been implemented. However, it only addresses risks from internal factors of software.

Moreover, we can see current state of studies addressing and discussing technical risks are huge compared to another type of risks. Studies that discussed on all of the risks however are still scarce. Thus, this creates a gap in the study on the risks associated with software ageing as studies addressing and discussing risks from external point of view such as business and organizational risks are also has been paid a little attention. Future works on this area could consider devoting effort on external risks viewpoint or combination of both internal and external risks prospect for effective risk mitigation to slow down impact of risks to failure leading to software ageing phenomenon.

Therefore, from the review, we could summarize the gap in the literature study by each research questions, hence includes (1) more studies are focused to ageing mitigation rather than ageing detection analysis and combination of both studies, (2) huge gap on the analyses performed in prior work to forecast and detect software ageing phenomenon as most past researchers frequently employed measurement-based analysis compared to model-based and hybrid analysis, (3) numerous studies proposed software rejuvenation as prevention action to delay ageing phenomenon, and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(4) minimal attention has been given to business and operational risks (external prospect) as majority past studies discussed on addressing technical risks (internal prospect).

In conclusion from the obtained findings, it provide strong evidence to encourage further development of new methodology to perform integrated and step-wise risk mitigation technique/approach to ensure the sustainability and survivability of software. New development of mitigation approach might help to improve the ability of existing mitigation approach to cope with current technological demands. Further research direction in this field may build a new in-depth model/framework using integrated risk mitigation activities to address risks from external factors point of view to achieve anti software ageing, different from most of past researchers had suggested and modeled, especially during software maintenance phase. Perhaps it is time for researchers to move their aim and goals in sustaining software life cycle from internal mitigation outlook to a whole nature of the software system that includes external prospect mitigation.

ACKNOWLEDGEMENT

This work was supported by Malaysian Ministry of Education under Putra Graduate Initiative Grant, Universiti Putra Malaysia (GP-IPS/2018/9644600).

REFRENCES:

[1] Li, L., Vaidyanathan, K., & Trivedi, K. S.,An approach for estimation of software aging in a web server in Empirical Software Engineering, Proceedings International Sympsosium 2002, (pp. 91-100). IEEE.

[2] Abdullah, Z.H., Yahaya, J.H. and Deraman, A., The Anti-Ageing Factors for Evergreen Software–a Preliminary Study Sci-Int. Com, 2014, pp.1615-1618.

[3] Kitchenham, B., Pretorius, R., Budgen, D., Brereton, O.P., Turner, M., Niazi, M. and Linkman, S., Systematic literature reviews in software engineering–a tertiary study in Information and software technology **52**(8), 2009, pp.792-805.

[4] Umesh, I.M., Srinivasan, G.N. and Torquato, M., Software Rejuvenation Model for Cloud Computing Platform. International Journal of Applied Engineering Research **12**(19), 2017, pp.8332-8337.

[5] Abidin, Z. N. Z., Yahaya, J. H., & Deraman, A., Software ageing measurement model (SAMM): The conceptual framework in *Electrical Engineering and Informatics (ICEEI), 2015 International Conference on* (pp. 456-461). IEEE.

[6] Cotroneo, D., Orlando, S., Pietrantuono, R., & Russo, S., A measurement-based ageing analysis of the JVM. Software Testing, Verification and Reliability, **23**(3), 2013, 199-239.

[7] Manel, S., Ridha, A., & Alia, M., Optimised Migrate Virtual Machine Rejuvenation. Journal of Computer and Communications, **3**(08), 2015, 33.

[8] Cotroneo, D., Natella, R., & Pietrantuono, R., Predicting aging-related bugs using software complexity metrics. Performance Evaluation, **70**(3), 2013, 163-178.

[9] Shahzad, B., Al-Ohali, Y., & Abdullah, A., Trivial model for mitigation of risks in software development life cycle. International Journal of Physical Sciences, **6**(8), 2011, 2072-2082.

[10] Dang, W., & Zeng, J., Optimization of Software Rejuvenation Policy based on State-Control-Limit.International Journal of Performability Engineering, **14**(2), 2018, 210.

[11] Ahdieh, K., Hashemitaba, N., & Ow, S. H., A novel model for software risk mitigation plan to improve the fault tolerance process. IJITCM: 2012, 1, 38-42.

[12] Qin, F., Zheng, Z., Qiao, Y., & Trivedi, K. S., Studying Aging-Related Bug Prediction Using Cross-Project Models. IEEE Transactions on Reliability, (99), 2018, 1-20.

[13] Fakhrolmobasheri, S., Ataie, E., & Movaghar, A., Modeling and Evaluation of Power-Aware Software Rejuvenation in Cloud Systems. Algorithms, **11**(10), 2018, 160.

[14] Machida, F., Xiang, J., Tadano, K., & Maeno, Y., Lifetime Extension of Software Execution Subject to Aging. IEEE Transactions on Reliability, **66**(1), 2017, 123-134.

[15] Zheng, J., Okamura, H., Li, L., & Dohi, T., A Comprehensive Evaluation of Software Rejuvenation Policies for Transaction Systems With Markovian Arrivals. IEEE Transactions on Reliability, **66**(4), 2017, 1157-1177.

[16] Ficco, M., Pietrantuono, R., & Russo, S., Aging-related performance anomalies in the apache storm stream processing system. Future Generation Computer Systems, 2017.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[17] de Melo, M. D. E. T., Araujo, J., Umesh, I. M., & Maciel, P. R. M., SWARE: An approach to support software aging and rejuvenation experiments. Journal on Advances in Theoretical and Applied Informatics, **3**(1), 2017, 31-38.

[18] Umesh, I. M., Srinivasan, G. N., & Torquato, M., Software Aging Forecasting Using Time Series Model. Indonesian Journal of Electrical Engineering and Computer Science, **7**(3), 2017, 839-845.

[19] Abdullah, Z. H., Yahaya, J. H., Mansor, Z., & Deraman, A., Software Ageing Prevention from Software Maintenance Perspective–A Review. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), **9**(3-4), 2017, 93-96.

[20] Mahmud, H., A Simple Software Rejuvenation Framework Based on Model Driven Development. UHD Journal of Science and Technology, **1**(2), 2017, 37-45.

[21] ZHAI, Y. Z., LI, Q. Y., & YOU, H. C., Software Health Measurement Method Based on Aging-related Bugs. DEStech Transactions on Computer Science and Engineering, (cmsam), 2017. Conference*

[22] Araujo, J., Oliveira, F., Matos, R. D. S., Torquato, M., Ferreira, J., & Maciel, P. R. M., Software Aging Issues in Streaming Video Player. JSW, **11**(6), 2016, 554-568.

[23] Meng, H., Liu, J., & Hei, X, Modeling and optimizing periodically inspected software rejuvenation policy based on geometric sequences. Reliability Engineering & System Safety, **133**, 2015, 184-191.

[24] Kumar, G., & Kaushik, M., Maintenance policies for improving the availability of a software-hardware system. In Reliability, Maintainability and Safety (ICRMS), 2016 11th International Conference, 2016, pp. 1-5, IEEE.

[25] Yan, Y., & Guo, P., A practice guide of software aging prediction in a web server based on machine learning. China Communications, **13**(6), 2016, 225-235.

[26] Meng, H., Hei, X., Zhang, J., Liu, J., & Sui, L., Software aging and rejuvenation in a j2ee application server. Quality and Reliability Engineering International, **32**(1), 2016, 89-97.

[27] Umesh, I. M., & Srinivasan, G. N., Optimum Software Aging Prediction and Rejuvenation Model for Virtualized Environment. Indonesian Journal of Electrical Engineering and Computer Science, **3**(3), 2016 572-578.

[28] Grottke, M., Kim, D. S., Mansharamani, R., Nambiar, M., Natella, R., & Trivedi, K. S., Recovery from software failures caused by mandelbugs. IEEE Transactions on Reliability, **65**(1), 2016, 70-87.

[29] Meng, H., Hei, X., Li, Y., Du, Y., & Xie, G., A Rejuvenation Model for Software System under Normal Attack. In Trustcom/BigDataSE/ISPA, **1,** 2015, pp. 1160-1164. IEEE. Conference*

[30] Chen, P., Qi, Y., Li, X., Hou, D., & Lyu, M. R. T., ARF-Predictor: Effective prediction of aging-related failure Using Entropy. IEEE Transactions on Dependable and Secure Computing, **15**(4), 2018, 675-693.

[31] Okamura, H., & Dohi, T., Dynamic software rejuvenation policies in a transaction-based system under Markovian arrival processes. Performance Evaluation, **70**(3), 2013, 197-211.

[32] Grottke, M., & Schleich, B., How does testing affect the availability of aging software systems?. Performance Evaluation, **70**(3), 2013, 179-196.

[33] Alonso, J., Matias, R., Vicente, E., Maria, A., & Trivedi, K. S., A comparative experimental study of software rejuvenation overhead. Performance Evaluation, 70(3), 2013, 231-250.

[34] Langner, F., & Andrzejak, A., Detecting software aging in a cloud computingframework by comparing development versions. In Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium, 2013, pp. 896-899. IEEE. *conference

[35] Dohi, T., Zheng, J., Okamura, H., & Trivedi, K. S., Optimal periodic software rejuvenation policies based on interval reliability criteria. Reliability Engineering & System Safety, **180**, 2018, 463-475.

[36] Avritzer, A., Cole, R. G., & Weyuker, E. J., Methods and opportunities for rejuvenation in aging distributed software systems. Journal of Systems and Software, **83**(9), 2010, 1568-1578.

[37] Zhao, J., Wang, Y., Ning, G., Trivedi, K. S., Matias Jr, R., & Cai, K. Y., A comprehensive approach to optimal software rejuvenation. Performance Evaluation, **70**(11), 2013, 917-933

[38] Firdose, S., & Rao, L. M., 3LRM-3 Layer Risk Mitigation Modelling of ICT Software Development Projects. International Journal of Electrical and Computer Engineering, **6**(1), 2016, 349.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[39] Singh, B., Sharma, K. D., & Chandra, S., A new model for software risk management. International Journal of Computer Technology and Applications, **3**(3), 2012, 953-956.

[40] López, C., & Salmeron, J. L., Monitoring software maintenance project risks. Procedia Technology, **5**, 2012, 363-368.

[41] Chowdhury, A. A. M., & Arefeen, S., Software risk management: importance and practices. IJCIT, ISSN, 2011, 2078-5828.

[42] Elzamly, A., & Hussin, B., An Enhancement Of Framework Software Risk Management Methodology For Successful Software Development. Journal of Theoretical & Applied Information Technology, 62(2), 2014.

[43] Bruneo, D., Distefano, S., Longo, F., Puliafito, A., & Scarpa, M., Workload-based software rejuvenation in cloud systems. IEEE Transactions on Computers, **62**(6),2013,1072-1085.

[44] Ning, G., Zhao, J., Lou, Y., Alonso, J., Matias, R., Trivedi, K. S., ... & Cai, K. Y., Optimization of two-granularity software rejuvenation policy based on the Markov regenerative process. IEEE Transactions on Reliability, 65(4), 2016, 1630-1646.

[45] Machida, F., & Miyoshi, N., Analysis of an optimal stopping problem for software rejuvenation in a deteriorating job processing system. Reliability Engineering & System Safety, **168**, 2017, 128-135.

[46] Salfner, F., & Wolter, K., Analysis of service availability for time-triggered rejuvenation policies. Journal of Systems and Software, **83**(9), 2010, 1579-1590.

[47] Kulkarni, P., Software Rejuvenation and Workload Distribution in Virtualized System. International Journal of Innovative Research in Computer and Communication Engineering, **3**(6), 2015, 5966-5973.

[48] Prabha, S., & Ujjawal, R. L., Software Risk Evaluation and Assessment using Hybrid Approach. Proceedings published in International Journal of Computer Applications, 2011.

[49] Meng, H., Wang, Y., Wang, H., Wang, F., & Liu, J., Optimal control method for runtime system maintenance. In Control And Decision Conference (CCDC), 2017 29th Chinese (pp. 3272-3275). IEEE.

[50] Bombardieri, M., & Fontana, F. A., Software aging assessment through a specialization of the SQuaRE quality model. Software Quality, 2009. WOSQ'09. ICSE Workshop, 2009 pp. 33-38. IEEE.

[51] Matias, R., Beicker, I., Leitão, B., & Maciel, P. R., Measuring software aging effects through OS kernel instrumentation. In Software Aging and Rejuvenation (WoSAR), 2010 IEEE Second International Workshop, 2010, pp. 1-6. IEEE.

[52] Macêdo, A., Ferreira, T. B., & Matias, R., The mechanics of memory-related software aging. In Software Aging and Rejuvenation (WoSAR), 2010 IEEE Second International Workshop, 2010 pp. 1-5. IEEE.

[53] Cotroneo, D., Natella, R., Pietrantuono, R., & Russo, S., Software aging analysis of the Linux operating system. In Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium, 2010, pp. 71-80. IEEE.

[54] Zheng, P., Xu, Q., & Qi, Y., An advanced methodology for measuring and characterizing software aging. In Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium, 2012, pp. 253-258. IEEE.

[55] Araujo, J., Matos, R., Maciel, P., Matias, R., & Beicker, I., Experimental evaluation of software aging effects on the eucalyptus cloud computing infrastructure. In Proceedings of the Middleware 2011 Industry Track Workshop, 2011, p. 4. ACM.

[56] Rahme, J., & Xu, H., Preventive maintenance for cloud-based software systems subject to non-constant failure rates. In 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 2017, pp. 1-6. IEEE.

[57] Yahaya, J. H., Abidin, Z. N. Z., Ali, N. M., & Deraman, A., Software ageing measurement and classification using Goal Question Metric (GQM) approach. In Science and Information Conference (SAI), 2013, pp. 160-165.

[58] Fang, Y., Yin, B. B., Ning, G., Zheng, Z., & Cai, K. Y., A Rejuvenation Strategy of Two-Granularity Software Based on Adaptive Control. In Dependable Computing (PRDC), 2017 IEEE 22nd Pacific Rim International Symposium, 2017, pp. 104-109. IEEE.

[59] Liu, J., Zhou, J., & Buyya, R., Software rejuvenation based fault tolerance scheme for

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

cloud applications. In Cloud Computing (CLOUD), 2015 IEEE 8th International Conference, 2015, pp. 1115-1118. IEEE.

[60] Yahaya, J. H., Deraman, A., & Abdullah, Z, H., Evergreen software preservation: The anti ageing model. Proceedings of the International Conference on Internet of things and Cloud Computing, 2016, p.51. ACM.

[61] Narania, S., Eshahawi, T., Gindy, N., Tang, Y. K., Stoyanov, S., Ridout, S., & Bailey, C., Risk mitigation framework for a robust design process. In Electronics System-Integration Technology Conference, 2008. ESTC 2008. 2nd (pp. 1075-1080). IEEE.

[62] Khatavakhotan, A. S., & Ow, S. H., An innovative model for optimizing software risk mitigation plan: A case study. In Modelling Symposium (AMS), 2012 Sixth Asia, 2012 pp. 220-224. IEEE.

[63] Okamura, H., Zheng, J., & Dohi, T., Statistical Framework on Software Aging Modeling with Continuous-Time Hidden Markov Model. In Reliable Distributed Systems (SRDS), 2017 IEEE 36th Symposium, 2017, pp. 114-123. IEEE.

[64] Zhao, J. F., Modeling of Software Aging Based on Non-stationary Time Series. In Information System and Artificial Intelligence (ISAI), 2016 International Conference, 2016, pp. 176-180. IEEE.

[65] Umesh, I. M., & Srinivasan, G. N., Dynamic software aging detection-based fault tolerant software rejuvenation model for virtualized environment. In Proceedings of the International Conference on Data Engineering and Communication Technology, 2017, pp. 779-787. Springer, Singapore.

[66] Kula, R. G., German, D. M., Ishio, T., Ouni, A., & Inoue, K., An exploratory study on library aging by monitoring client usage in a software ecosystem. In Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference, 2017. pp. 407-411. IEEE.

[67] Torquato, M., Maciel, P., Araujo, J., & Umesh, I. M., An approach to investigate aging symptoms and rejuvenation effectiveness on software systems. In Information Systems and Technologies (CISTI), 2017 12th Iberian Conference, 2017, pp. 1-6. IEEE.

[68] Sukhwani, H., Matias Jr, R., Trivedi, K. S., & Rindos, A., Monitoring and Mitigating Software Aging on IBM Cloud Controller System. In 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2017, pp. 266-272. IEEE.

[69] Jiang, L., Peng, X., & Xu, G., Time and prediction based software rejuvenation policy. In Information Technology and Computer Science (ITCS), 2010 Second International Conference, 2010 pp. 114-117. IEEE

[70] Weng, C., Xiang, J., Xiong, S., Zhao, D., & Yang, C., Analysis of Software Aging in Android. In Software Reliability Engineering Workshops (ISSREW), 2016 IEEE International Symposium, 2016, pp. 78-83. IEEE.

[71] Cotroneo, D., Fucci, F., Iannillo, A. K., Natella, R., & Pietrantuono, R., Software aging analysis of the android mobile os. In Software Reliability Engineering (ISSRE), 2016 IEEE 27th International Symposium on (pp. 478-489). IEEE.

[72] Rezaei, A., & Sharifi, M., Rejuvenating high available virtualized systems. In Availability, Reliability, and Security, 2010. ARES'10 International Conference on (pp. 289-294). IEEE.

[73] Wu, H., & Wolter, K., Software aging in mobile devices: Partial computation offloading as a solution. In Software Reliability Engineering Workshops (ISSREW), 2015 IEEE International Symposium on (pp. 125-131). IEEE.

[74] Meng, H., Zhang, X., Zhu, L., Wang, L., & Yang, Z., Optimizing software rejuvenation policy based on CDM for cloud system. In Industrial Electronics and Applications (ICIEA), 2017 12th IEEE Conference on (pp. 1850-1854). IEEE.

[75] Guo, C., Wu, H., Hua, X., Lautner, D., & Ren, S., Use two-level rejuvenation to combat software aging and maximize average resource performance. In High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conference on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on (pp. 1160-1165). IEEE.

[76] Huang, B., & Qin, Y., Software Aging Analysis Based on Man–Machine–Environment System Engineering. In Proceedings of the 15th International Conference on Man–Machine–Environment System Engineering, 2015 (pp. 681-687). Springer, Berlin, Heidelberg.

[77] Qin, F., Zheng, Z., Bai, C., Qiao, Y., Zhang, Z., & Chen, C., Cross-Project Aging Related Bug

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Prediction. In Software Quality, Reliability and Security (QRS), IEEE International Conference, 2015, pp. 43-48. IEEE.

[78] Yan, Y., Guo, P., & Liu, L. (2014, November). A practice of forecasting software aging in an IIS web server using SVM. In Software Reliability Engineering Workshops (ISSREW), IEEE International Symposium, 2014, pp. 443-448. IEEE.

[79] Mohan, B. R., & Reddy, G. R. M., Software aging trend analysis of server virtualized system. In Information Networking (ICOIN), International Conference , 2014, pp. 260-263. IEEE.

[80] Wang, Q., & Wolter, K., Detection and Analysis of Performance Deterioration in Mobile Offloading System. In Software Reliability Engineering Workshops (ISSREW), IEEE International Symposium 2014, pp. 420-425. IEEE.

[81] Sudhakar, C., Shah, I., & Ramesh, T., Software rejuvenation in cloud systems using neural networks. In Parallel, Distributed and Grid Computing (PDGC), International Conference, 2014, pp. 230-233. IEEE.

[82] Matias, R., Andrzejak, A., Machida, F., Elias, D., & Trivedi, K., A systematic differential analysis for fast and robust detection of software aging. In Reliable Distributed Systems (SRDS), IEEE 33rd International Symposium, 2014, pp. 311-320. IEEE.

[83] Zhao, J., Jin, Y., Trivedi, K. S., & Matias Jr, R., Injecting memory leaks to accelerate software failures. In Software Reliability Engineering (ISSRE), 2011 IEEE 22nd International Symposium, 2011, pp. 260-269. IEEE.

[84] Cotroneo, D., Frattini, F., Natella, R., & Pietrantuono, R., Performance degradation analysis of a supercomputer. In Software Reliability Engineering Workshops (ISSREW), 2013 IEEE International Symposium, 2013, pp. 263-268. IEEE.

[85] Machida, F., Andrzejak, A., Matias, R., & Vicente, E., On the effectiveness of Mann-Kendall test for detection of software aging. In Software Reliability Engineering Workshops (ISSREW), IEEE International Symposium, 2013, pp. 269-274. IEEE.

[86] Araujo, J., Matos, R., Maciel, P., Vieira, F., Matias, R., & Trivedi, K. S., Software rejuvenation in eucalyptus cloud computing infrastructure: A method based on time series forecasting and multiple thresholds. In Software Aging and Rejuvenation (WoSAR), IEEE Third International Workshop, 2011, pp. 38-43. IEEE.

[87] Li, S., & Yong, Q., Software aging detection based on NARX model. In Web Information Systems and Applications Conference (WISA), 2012 Ninth (pp. 105-110). IEEE.

[88] Matos, R., Araujo, J., Alves, V., & Maciel, P., Experimental evaluation of software aging effects in the eucalyptus elastic block storage. In Systems, Man, and Cybernetics (SMC), IEEE International Conference, 2012, pp. 1103-1108. IEEE.

[89] Cui, L., Li, B., Li, J., Hardy, J., & Liu, L. (2012, December). Software aging in virtualized environments: detection and prediction. In Parallel and Distributed Systems (ICPADS), IEEE 18th International Conference, 2012, pp. 718-719. IEEE.

[90] Yang, T., Bao, J., & Wu, Q., Research of a resource to influence on the software aging and rejuvenation cycle. In Industrial Electronics and Applications (ICIEA), 7th IEEE Conference, 2012, pp. 1349-1351. IEEE.

[91] Machida, F., Xiang, J., Tadano, K., & Maeno, Y., Software life-extension: a new countermeasure to software aging. In Software Reliability Engineering (ISSRE), IEEE 23rd International Symposium, 2012, pp. 131-140. IEEE.

[92] Agepati, R., Gundala, N., & Amari, S. V., Optimal Software rejuvenation policies. In Reliability and Maintainability Symposium (RAMS), Proceedings-Annual, 2013, pp. 1-7. IEEE.

[93] Araujo, J., Matos, R., Maciel, P., & Matias, R., Software aging issues on the eucalyptus cloud computing infrastructure. In Systems, Man, and Cybernetics (SMC), IEEE International Conference, 2011, pp. 1411-1416. IEEE.

[94] Bovenzi, A., Cotroneo, D., Pietrantuono, R., & Russo, S., Workload characterization for software aging analysis. In Software Reliability Engineering (ISSRE), IEEE 22nd International Symposium, 2011, pp. 240-249. IEEE.

[95] Jun, G., Bo, W., Yunsheng, W., Bin, Z., & Jiaojiao, W., Research of the software aging regeneration strategy based on components. In Proceedings of the 2011, International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 19–20, 2011, Melbourne, Australia, pp. 601-608. Springer, Berlin, Heidelberg.

[96] Okamura, H., & Dohi, T., Optimization of opportunity-based software rejuvenation policy.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IEEE 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW), 2012, pp. 283-286. IEEE.

[97] Abdullah, Z. H., Yahaya, J., & Deraman, A., Towards anti-Ageing model for the evergreen software system. In Electrical Engineering and Informatics (ICEEI), International Conference, 2015, pp. 388-393. IEEE.

[98] Alonso, J., Goiri, Í., Guitart, J., Gavalda, R., & Torres, J., Optimal resource allocation in a virtualized software aging platform with software rejuvenation. In Software Reliability Engineering (ISSRE), IEEE 22nd International Symposium, 2011, pp. 250-259. IEEE.

[99] Lahon, M., & Sharma, U., Risk assessment and mitigation approach for architecture evaluation in component based software development. In Computing for Sustainable Global Development (INDIACom),3rd International Conference, 2016, pp. 2801-2804. IEEE.

[100] Kumar, L., & Sureka, A., Feature Selection Techniques to Counter Class Imbalance Problem for Aging Related Bug Prediction: Aging Related Bug Prediction. In Proceedings of the 11th Innovations in Software Engineering Conference, 2018, p. 2. ACM.

[101] Li, J., Qi, Y., & Cai, L., A Hybrid Approach for Predicting Aging-Related Failures of Software Systems. In Service-Oriented System Engineering (SOSE), IEEE Symposium, 2018, pp. 96-105. IEEE.

[102] Qiao, Y., Zheng, Z., & Fang, Y., An empirical study on software aging indicators prediction in Android mobile. In IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2018, pp. 271-277. IEEE.

[103]Xiang, J., Weng, C., Zhao, D., Tian, J., Xiong, S., Li, L., & Andrzejakb, A., A New Software Rejuvenation Model for Android. In IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2018, pp. 293-299. IEEE.

[104] Abidin, Z. N. Z., Yahaya, J. H., Deraman, A., & Abdullah, Z. H., Rejuvenation Action Model for Application Software. In 2018 6th International Conference on Information and Communication Technology (ICoICT), 2018, pp. 38-43. IEEE.

[105] Jia, S., Hou, C., & Wang, J., Software aging analysis and prediction in a web server based on multiple linear regression algorithm. In Communication Software and Networks (ICCSN), IEEE 9th International Conference, 2017, pp. 1452-1456. IEEE.

[106] Huo, S., Zhao, D., Liu, X., Xiang, J., Zhong, Y., & Yu, H., Using machine learning for software aging detection in Android system. In Advanced Computational Intelligence (ICACI), Tenth International Conference, 2018, pp. 741-746. IEEE.

[107]Alonso, J., Belanche Muñoz, L. A., & Avresky, D., Predicting software anomalies using machine learning techniques. In 2011 IEEE International symposium on network computing and applications, NCA 2011: 25-27 August 2011, Cambridge, Massachusetts, US: proceedings (pp. 163-170). IEEE Computer Society Publications.

[108] Du, X., Qi, Y., Lu, H., & He, X., A method for software aging state evaluation. In Advanced Information Management and Service (IMS), 6th International Conference, 2010, pp. 48-53. IEEE.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

APPENDIX

Table 4.2A Summarization of analysis, approach and ageing indicator used

References	Title	Type of Analysis	Approach	Ageing indicator
[4]	Software Rejuvenation Model for Cloud Computing Platform +SR	Measurement-based	Time series	Memory consumption
[5]	Software ageing measurement model (SAMM): The conceptual framework.	Model-based	Measuring software health level	-
[6]	A measurement-based ageing analysis of the JVM + SR	Measurement-based	Linear regression, time-series	Throughput loss, Memory depletion
[8]	Predicting aging-related bugs using software complexity metrics	Measurement-based	Machine learning algorithms	Performance measuring
[12]	Studying Aging-related bugs prediction using cross-project models	Measurement-based	Machine learning	Not specified aging related bugs
[16]	Aging-related performance anomalies in the apache storm stream processing system + SR	Measurement-based	Time series	Perceived performance Resource depletion; memory consumption
[18]	Software Aging Forecasting Using Time Series Model	Measurement-based	Time-series	CPU usage, Memory availability
[21]	Software Health Measurement Method Based on Aging-related Bugs	Model-based	The Gaussian Mixture Model	Memory availability
[22]	Software Aging Issues in Streaming Video Play + SR	Hybrid	Time-series+Markov model	Resource usage
[25]	A practice guide of software aging prediction in a web server based on machine learning	Measurement-based	Machine learning	Memory related
[26]	Software aging and rejuvenation in a j2ee application server +SR	Measurement-based	Accelerated aging test	Resource usage: Memory usage CPU utilization
[27]	Optimum Software Aging Prediction and Rejuvenation Model for Virtualized Environment +SR	Measurement-based	Machine learning,	Resource usage: Memory usage CPU utilization
[30]	ARF-Predictor: Effective prediction of aging-related failure Using Entropy	Measurement-based	Time series, threshold	Performance metrics
[32]	How does testing affect the availability of aging software systems? +SR	Model-based	Semi-markov model	
[34]	Detecting software aging in a cloud computing framework by comparing development version	Others	Comparison, exploits differences between software version	Memory leak and depletion
[50]	Software aging assessment through a specialization of the SQuaRE quality mode	Model-based	Square quality model	
[51]	Measuring software aging effects through OS kernel instrumentation	Measurement-based	OS Kernel Instrumentation Techniques	Memory leaking, fragmentation
[52]	The mechanics of memory-related software aging	Measurement-based	Time series Trend estimation	Memory leak, Memory fragmentation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[53]	Software aging analysis of the Linux operating system	Measurement-based	Time series Trend analysis Workload controlled	Memory consumption, System call latency
[54]	An advanced methodology for measuring and characterizing software aging	Measurement-based	Time series	Memory consumption
[55]	Experimental evaluation of software aging effects on the eucalyptus cloud computing infrastructure +SR	Measurement	Regression based	Memory leak, Memory fragmentation
[57]	Software ageing measurement and classification using Goal Question Metric (GQM) approach	Model-based	Ageing index	
[63]	A Statistical Framework on Software Aging Modeling with Continuous-Time Hidden Markov Model	Hybrid	Continuous-time hidden Markov Model (CTHMM)	Memory usage, response time
[64]	Modeling of Software Aging Based on Non-stationary Time Series	Measurement-based	Time Series	Resource usage: CPU usage Memory consumption
[65]	Dynamic software aging detection-based fault tolerant software rejuvenation model for virtualized environment +SR	Measurement-based	Robust local regression algorithm	Resource usage: CPU load Memory consumption
[66]	An exploratory study on library aging by monitoring client usage in a software ecosystem	Model-based	Typical polynomial mathematical model	Dependability attributes: Reliability
[70]	Analysis of Software Aging in Android +SR	Hybrid	Memory availability as parameter to parametrize Markov model	Memory availability
[71]	Software aging analysis of the android mobile OS +SR	Measurement-based	Time series	Resource consumption; Memory usage Storage usage Tasks performed Garbage collector
[77]	Cross-Project Aging Related Bug Prediction	Measurement-based	Machine learning	Software complexity metrics
[78]	A practice of forecasting software aging in an IIS web server using SVM	Measurement-based	Time series: Linear regression	Resource consumption; Memory
[79]	Software aging trend analysis of server virtualized system	Measurement-based	Time series; Linear regression	Resource usage Performance measuring
[80]	Detection and Analysis of Performance Deterioration in Mobile Offloading System +SR	Measurement-based	Others	Performance deterioration
[82]	A systematic differential analysis for fast and robust detection of software aging	Measurement-based	Time series	Memory leak
[83]	Injecting memory leaks to accelerate software failures	Model-based	Semi-Markov	Memory leak

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[84]	Performance degradation analysis of a supercomputer	Measurement-based	Time series Mann Kendall test	Performance measuring
[85]	On the effectiveness of Mann-Kendall test for detection of software aging	Measurement-based	Time series Mann Kendal test	Memory leak
[86]	Software rejuvenation in eucalyptus cloud computing infrastructure: A method based on time series forecasting and multiple thresholds.	Measurement-based	Time series	Resource utilization
[87]	Software aging detection based on NARX model	Measurement-based	Machine learning Non linear regression	Multiple variables
[88]	Experimental evaluation of software aging effects in the eucalyptus elastic block storage	Measurement-based	Time series	Resource utilization
[89]	Software aging in virtualized environments: detection and prediction	Measurement-based	Time series Trend analysis	Resource usage
[93]	Software aging issues on the eucalyptus cloud computing infrastructure	Measurement-based	Time series	Resource usage: CPU Memory Disk Space
[94]	Workload characterization for software aging analysis	Measurement-based	Workload-dependent analysis Time series; Mann Kendall test	Memory depletion Throughput loss
[100]	Feature Selection Techniques to Counter Class Imbalance Problem for Aging Related Bug Prediction: Aging Related Bug Prediction	Measurement-based	Machine learning	Performance Aging related bugs
[101]	A Hybrid Approach for Predicting Aging-Related Failures of Software Systems	Hybrid	Threshold. Construct a baseline failure model with Weibull distribution, and use runtime performance metrics as covariates	System resource Response time
[102]	An empirical study on software aging indicators prediction in Android mobile	Measurement based	Time series	Memory related
[105]	Software aging analysis and prediction in a web server based on multiple linear regression algorithm	Measurement-based	Machine learning algorithm	Response time Memory utilization CPU utilization
[106]	Using machine learning for software aging detection in Android system	Measurement-based	Machine learning	Page fault
[107]	Predicting software anomalies using machine learning techniques	Measurement-based	Machine learning	Resource consumption
[108]	A method for software aging state evaluation	Measurement-based	Time series Linear regression	Resource consumption

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 9 Summarization of RQ1-RQ4

References	RQ1	RQ2	RQ3	RQ4	Year
[2]	M	-	MA	All	2014
[4]	C	ME	SReju	T	2017
[5]	D	MO	-	All	2015
[6]	C	ME	SReju	T	2013
[7]	M	-	SReju	T	2015
[8]	D	ME	-	T	2013
[9]	M	-	Model	All	2011
[10]	M	-	SReju	T	2018
[11]	D	-	Model	T	2012
[12]	D	ME	-	T	2018
[13]	M	-	SReju	T	2018
[14]	M	-	SLE	T	2017
[15]	M	-	SReju	T	2017
[16]	C	ME	SReju	T	2017
[17]	M	-	SReju	T	2017
[18]	D	ME	-	T	2017
[19]	M	-	MA	All	2017
[20]	M	-	SReju	All	2017
[21]	D	MO	-	T	2017
[22]	C	Hyb	SReju	T	2016
[23]	M	-	SReju	T	2015
[24]	M	-	MA	T,O	2016
[25]	D	ME	-	T	2016
[26]	C	ME	SReju	T	2016
[27]	C	ME	SReju	T	2016
[28]	M	-	RA	T	2016
[29]	M	-	SReju	T	2015
[30]	D	ME	-	T	2018
[31]	M	-	SReju	T	2013
[32]	C	MO	SReju	T	2013
[33]	M	-	SReju	T	2013
[34]	D	Others	-	T	2013
[35]	M	-	SReju	T	2018
[36]	M	-	SReju	T	2010
[37]	M	-	SReju	T	2013
[38]	M	-	Model	O,B	2016
[39]	M	-	Model	T,B	2012
[40]	M	-	Model	All	2012
[41]	M	-	Model	All	2011
[42]	M	-	Model	Not specified	2014
[43]	M	-	SReju	T	2013
[44]	M	-	SReju	T	2016
[45]	M	-	SReju	T	2017
[46]	M	-	SReju	T	2010
[47]	M	-	SReju	T	2015
[48]	M	-	Model	T,B	2011
[49]	M	-	SReju	T	2017
[50]	D	MO	-	All	2009
[51]	D	ME	-	T	2010

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[52]	D	ME	-	T	2010
[53]	D	ME	-	T	2010
[54]	D	ME	-	T	2012
[55]	C	ME	SReju	T	2011
[56]	M	-	SReju	T	2017
[57]	D	MO	-	All	2013
[58]	M	-	SReju	T	2017
[59]	M	-	SReju	T	2015
[60]	M	-	MA	All	2016
[61]	M	-	Model	B	2008
[62]	M	-	Model	T,B	2012
[63]	D	Hyb	-	T	2017
[64]	D	ME	-	T	2016
[65]	C	ME	SReju	T	2017
[66]	D	MO	-	T	2017
[67]	M	-	SReju	T	2017
[68]	M	-	MA	T	2017
[69]	M	-	SReju	T	2010
[70]	C	Hyb	SReju	T	2016
[71]	C	ME	SReju	T	2016
[72]	M	-	SReju	T	2010
[73]	M	-	PCO	T	2015
[74]	M	-	SReju	T	2017
[75]	M	-	SReju	T	2015
[76]	M	-	SReju	T	2015
[77]	D	ME	-	T	2015
[78]	D	ME	-	T	2014
[79]	D	ME	-	T	2014
[80]	C	ME	SReju	T	2014
[81]	M	-	SReju	T	2014
[82]	D	ME	-	T	2014
[83]	C	MO	SReju	T	2011
[84]	D	ME	-	T	2013
[85]	D	ME	-	T	2013
[86]	C	ME	SReju	T	2011
[87]	D	ME	-	T	2012
[88]	D	ME	-	T	2012
[89]	C	ME	SReju	T	2012
[90]	M	-	SReju	T	2012
[91]	M	-	SLE	T	2012
[92]	M	-	SReju	T	2013
[93]	D	ME	-	T	2011
[94]	D	ME	-	T	2011
[95]	M	-	SRS	T	2011
[96]	M	-	SReju	T	2012
[97]	M	-	MA	All	2015
[98]	M	-	SReju	T	2011
[99]	M	-	Model	T,B	2016
[100]	D	ME	-	T	2018
[101]	D	Hyb	-	T	2018
[102]	D	ME	-	T	2018

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[103]	M	-	SReju	T	2018
[104]	M	-	SReju	All	2018
[105]	D	ME	-	T	2017
[106]	D	ME	-	T	2018
[107]	D	ME	-	T	2018
[108]	D	ME	-	T	2018

Footnotes: *M-Mitigation, D-Detection; ME-Measurement-based, MO-Model-based; SReju-Software Rejuvenation, MA-Maintenance Activities, SLE-Software Life Extension, SRS-Software Regeneration Strategies, PCO-Partial Computation Offloading, RA-Recovery Actions; T-Technical Risk, B-Business Risk, O-Organizational Risk*