Software Ageing Prevention from Software Maintenance Perspective – A Review

Zuriani Hayati Abdullah[1], Jamaiah H. Yahaya[1], Zulkefli Mansor[1] and Aziz Deraman[2]
[1]*Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia*
[2]*School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu,*
21030 Kuala Nerus, Terengganu, Malaysia.
zha.ukm@gmail.com

***Abstract*—This paper presents the background study of software ageing phenomenon and the prevention techniques to contravene the ageing. The accumulation of software failure and degradation of software performance shows that the software gets old and irrelevant to the users in the environment. Ageing of the software might contribute to the negative impact and affect organisational performance. It is crucial for the organisation to find out prevention actions toward the software ageing phenomenon. This paper discussed in detail how software maintenance activity can be used as ageing prevention for software ageing.**

***Index Terms*—Ageing Prevention; Software Ageing; Software Anti-Ageing; Software Maintenance.**

I. INTRODUCTION

In current globalization era and internet of thing, Software has become one of the vital components for ensuring the sustainability and survivability of companies and organisations in their business environment. Computer system is divided into two types: the system and application software.

Similar to human being, software has its own life span and become old or ageing. This is proven by the existence of software life cycle in every software development process and methodology. Software ageing can be determine as software performance degradation during the execution time. Software ageing has been introduced in the early of 90's by Parnas [1]. We cannot stop the occurance of ageing process but with the understanding of software ageing factors it is somehow can help to delay the ageing phenomenon.

Software ageing is the field of study that was evolved around twenty years ago. Along the time, this study has been explored in various aspects by the academicians or industrial researcher. Software ageing has been an established domain study and falls under software and reliability engineering discipline [2]. Previous studies indicated that by identifying the causes or factors that contributing to the phenomenon of software ageing, we could find the solution or action to delay the ageing process.

The process of preventing and delaying the ageing is called rejuvenation or anti-ageing process. Software anti-ageing process can be applied and implemented by investigating and characterizing the ageing factors that might lead to software ageing and implementing the reverse action to apply the anti-ageing process [3].

This paper starts with the research background and related work in software ageing, software ageing factors and software anti-ageing. Later, a discussion on how software maintenance activities can be used as the solution in order to maintain the relevance of the software in certain environment.

II. BACKGROUND STUDY AND RELATED WORKS

A. Software Ageing Issues

Ageing is an inevitable process that every creatures will go through. This phenomenon is applicable to the software environment as well. By identifying the influential factors and causes of software ageing can help to delay or slow down the process and is referred as anti-ageing or rejuvenation process. Software rejuvenation process is one of the approach for handling failure or faults tolerant on the long running software.

Previous studies investigated issues related to software ageing in computer software. They are mostly related to data corruption, memory bloating or memory leaking, accumulation of undetermined threads or failure, data-files fragmentation, unreleased files lock, memory lack and overruns [4, 5]. Nowadays, software ageing does not only focus on computer software or system software, few researchers investigate the ageing that occur in mobile application such and android and windows mobile application [6-8]. Mobile is unlike a PC which is continuously used for a longer time without rebooted. Therefore, software aging in mobile devices lead to a extensive challenge to the ultimate user experience and satisfaction [7].

Previous researchers do believe that good quality software helps in delaying ageing of the software [9]. Good quality software referred to the technical behavior and end users perspective which can ensure that software fulfills user's satisfaction and expectation and keep maintaining the software to become relevant longer in their operating environment [10, 11].

Besides good quality software, Matias et. al indicated that software maintenance must be carried out and implemented systematically to ensure the optimum quality of software throughout its life cycle. In software engineering, there are four main maintenance activities which are preventive, perfective, adaptive, and corrective. These activities may influence the ageing progress of software. Preventive maintenance can help to slow down the occurrence of failures determinable to this cause [12, 13].

Although software ageing is inevitable, but by understanding the factors that influence to software ageing, it may help to restrain the occurrence of ageing to the software system as well as application software. The next section will discuss about the software ageing factors that have been identified by literature study.

B. Software Ageing Effects

This section presents the effects of software aging based on literature findings [2]. Table 1 shows the effects of software ageing in various aspects.

Table 1
Software Ageing Effects

Effects	Explanation
Operating levels Effects	Ageing effects on the operating system at resource level: • Non-released memory • Round-off and data file fragmentations • Debug errors
Application Levels Effects	Ageing effects on the application level: • Non-released memory • Non-terminated threads • Round-off errors • Data files corruption
User Satisfaction	Ageing effets on users satisfaction: • According to study of human computer interaction the long response time more than 15 seconds can frustrate users. • Frustrated user will results the ignorance of the software and soon to be abandoned and old.
Delayed work and schedule	Ageing effects on work and schedule: • Slow time responsiveness on running application • System down time results in delayed work and running out of schedule
Economic Effects	Ageing effects on economics: • Lots of money has lost because of aging and system down. • Lots of money needed in order to acquire new software

Table 1 shows the effects of software ageing in five aspects which are operating level, application level, user satisfaction, delayed work and schedule, and economic. Operating levels and application levels affect the operating system at resource level, which are causally non-released memory or any other problem that related to memory consumption such as memory bloated or memory leak. It also cause round-off errors, data files fragmentation, data corruption and also debugs error.

Another three aspects affect users, organization performances and also economics. Software ageing defined as slow or degradation of software performance, it might influence the user satisfaction. The unsatisfied users tend to ignore the software and later on will results the software to be abandoned, no longer relevant and thus gets old and retires early.

When the existing software gets old and no longer relavant to the environment, the organization or users need to invest more money to acquire new software, which will results the economic factor of the organization.

It is crucial to prevent software ageing because it is not only affects software system, but also affects user and the universe in general. There was an incident that happened about twenty six years ago, which is caused by the software failure. A report of the General Accounting office which entitled Patriot Missile Defence: Software Problem Led to System Failure at Dhahran, Saudi Arabia reported on the cause of the failure to trace an Iraqi Scud missile and it struck the American army barracks and killed about twenty eight soldiers and around hundred people were injured [14].

Based on the report, the incident caused by an imprecise calculation of the time since boot dued to computer arithmetic errors and resulted the software failure. Hence, the past incidents have motivated us to study further the characteristics of software ageing, the factors that might contribute to the occurance of ageing phenomenon and how to prevent and delay the ageing process.

C. Software Ageing Factors

Literature study has summarized that software ageing factors can be classified into two types: external and internal factors. Table 2 shows the external and internal software ageing factors

Table 2
External And Internal Software Ageing Factors

External Factors	Internal Factors
Dynamic environment	Debug Failure
Function enhancement and change	Residual Defect
Business stability and consistency	Memory leak
Evolution requirement	Memory bloating
Maintenance Cost	Unreleased files lock
Human	Round off error accumulation

The internal factors are mostly associated with system software and its ageing phenomenon while the external factors are mostly related to software ageing for application software. Our research group focuses our exploration on the external factors of software ageing [3], [15-18]. We had conducted empirical studies to verify ageing factors and distributed them to the software practioiners in private and government sectors in Malaysia.

The empirical study revealed the three main factors that contributed to software ageing. Table 3 shows the ageing factors and metrics that can be used to measure the software ageing level. Based on the previous studies and shown in the table, the external factors can be classified into three main classes which are 1: Functional 2: Human 3: Environment. Each of these factors is then broken down into metrics which can be measured and transformed into numerical scales. Table 3 shows part of metrics that we have identified earlier in our previous research work [19].

Table 3
Software Ageing Factors and Metrics

Factors	Metrics
Functional	Software performance
	Usefulness
	Support service
	Time responsiveness
	Software failure
Environment	Support service
	Business Demand
	Changes in business process
	Environment change
	Technology change (hardware & software)
Human	Training
	Support service
	Popularity and technology
	Change in top management
	Expertise

Next section will discuss on how software maintenance activity can be applied in order to counteract with ageing and help to prevent and delay the ageing process.

III. SOFTWARE MAINTENANCE

Software maintenance as specified in IEEE Standard 1219

as: "The modification of a software product after delivery in order to correct defects, to improve performance or other attributes, or to adapt the product to a modified environment." Maintenance can be differed as the act of keeping an entity in an existing state of efficiency, validity, to preserve from failure, decline or degrades. Software maintenance is a modification of a software product after delivery in order to improve software performance, correct software faults and errors, to adapt the product to a modified and dynamics environment [20]. Software maintenance sustains the software product throughout its life cycle which is from the development to the operations state.

A. Software Maintenance Activity

Software Maintenance contains of four main activities which are preventive, perfective, corrective and adaptive maintenance as describe below [21] :

i. *Preventive maintenance*: modification of a software product after delivery to correct and detect latent faults in the software before they become operational faults.
ii. *Perfective maintenance*: modification of a software after product delivery to provide enhancements for users, improvement of program documentation, and recoding to improve software performance and maintainability.
iii. *Corrective maintenance*: reactive modification of a software product performed after delivery to correct discovered problems(repairs).
iv. *Adaptive maintenance*: modification of a software product performed after delivery to keep a software product usable in a changed or changing environment. For example, the operating system might be upgraded and some changes to the software may be necessary. Basically, this applied to the software which is faced the environment changes.

B. Related Work on Preventing Software Ageing

Previous study revealed that by identifying and responding or acting towards ageing factors could helped in preventing or delaying the ageing process in software evolution. However, this paper focuses on how software maintenance activities can be used as one of the techniques in order to prevent and delay the ageing process. Our research group has constructed a conceptual model which contains of software ageing factors and instrument to measure the software ageing level [19]. Figure 1 shows the proposed conceptual model that can be used as guidelines to software ageing prevention.

As shown in the model, software ageing can be levelled into three stages which are very old, semi-old and young. It shows that after the level of software ageing has been measured, the suggestion for anti-ageing action or software ageing prevention can be applied to the semi-old or very old software. We applied the four main activities of software maintenance as a prevention towards software ageing phenomenon.

Nevertheless, this study was the preliminary and initial study of software anti-ageing. Further work is needed in order to discover the most applicable techniques in order to deal with software ageing. Next section will discuss on how software maintenance activities can be used as a prevention to the software ageing phenomenon. Table 4 shown in details our suggestion for anti-ageing action as shown in the conceptual model of software ageing in Figure 1.

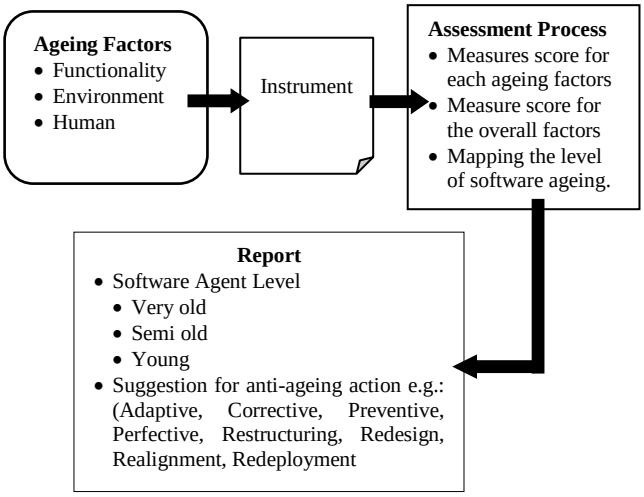

Figure 1: Conceptual Model of Software Ageing

Table 4
Software Maintenance activities and ageing metrics

Metrics	Maintenance activities
Software performance -degradation of performance -time responsiveness	*Perfective* • By monitoring the memory usage • Improving the quality aspects of the software product
Usefulness	*Perfectives* • Improve and enhance the software function to ensure the up-to-date service/functions are available
Software failure	*Corrective* • Correct the fault and error accordingly and systematically. • Improve the change request and process
Business Demand -Changes in business process	*Adaptive* • Improve and enhance the software function/services to ensure the demands in business processes are maintained and achieved.
Environment change	*Corrective & Adaptive* • Easy maintenance for environment change • Easy maintenance for business change
Technology change (hardware & software)	Adaptive • Improve and enhance the software for meeting new/current hardware or software demand and compatibility.
Expertise	Training • Awareness to the staff for quality assurance • Train staff for software quality practices and implementation. • Focus on training for staff in software maintenance and related.

C. Software Maintenance versus Software Ageing

Software maintenance sustains the software product throughout its life cycle from the development to the operational state. Trivedi et.al. [22] focused on systems behaviour and preventive maintenance in transaction related to ageing. Similar to Matias et al [12] they focused on preventive maintenance in order to counteract the software ageing phenomenon. Table 4 specifies how to apply software maintenance to the metrics that we have been identified in previous section [3, 19].

Software maintenance action is recommended to ensure the software stays young and healthy, relevant and fulfil users' need and expectation in the dynamic environment. The suggested actions are shown and described in Table 4. Each

metrics can be assigned to the maintenance activities in order to prevent and delay the ageing of the measured or evaluated software. The mapping of metrics and maintenance activities are derived based on finding from case studies conducted as described in [19]. Further and comprehensive work are needed in order to develop relevant method on how to prevent the ageing phenomenon in application software.

IV. FUTURE WORK

The software ageing has been highlighted and recognized as a major cause of performance degradation and accumulative of failure in the operational phase of the software product as explained in this paper. Future work will focus on :

i. Empirical study on current state of software ageing issues in Malaysia,
ii. Identifies anti-ageing factors for application software,
iii. Proposed anti-ageing model for application software, and
iv. validation and verification of anti-ageing model through expert review and case study.

V. CONCLUSION

This paper has presented background works on software ageing and discussed the ageing effects from various aspects and dimensions. It also described the software ageing factors and their associated metrics. This study extends further by summarizing the software maintenance activities as anti-ageing guidelines and actions in order to counteract with the ageing phenomenon . With the guideline, organization or the stakeholders could decide actions to be taken to overcome the ageing occurrence and minimize the state. This will enhance the software operating status and extend their usage in their environment. For future works, it is recommended to enhance the model by extensive anti-ageing factors and metrics, and apply in a wider scope of application software.

ACKNOWLEDGEMENT

This research project was funded partly by the Fundamental Research Grant Scheme, Malaysia Ministry of Higher Education which was granted under Universiti Kebangsaan Malaysia (FRGS/1/2012/SG05/UKM/02/10) and the Universiti Malaysia Terengganu research grant under FRGS vot 59297.

REFERENCES

[1] D. L. Parnas, "Software aging invited," in *ICSE '94 Proc. 16th Int. Conf. Softw. Eng.*, 1994, pp. 279–287.

[2] S. Ahamad, "Study of software aging issues and prevention solutions," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 8, pp. 307–313, 2016.

[3] J. H. Yahaya, A. Deraman, and Z. H. Abdullah, "Evergreen software preservation : The anti-ageing model," in *ICC '16 Proc. Int. Conf. Internet things Cloud Comput. United Kingdom*, 2016, pp. 1–6.

[4] M. Grottke, R. M. Jr, and K. S. Trivedi, "The fundamentals of software aging," in *Proc. 1st Int. Workshop on Software Aging and Rejuvenation/ © IEEE 19th International Symposium on Software Reliability Engineering*, 2008, pp. 1–6.

[5] S. Russo and N. Federico, "The dual nature of software aging twenty years of software aging research," in *2014 IEEE Int. Symp. on Software Reliability Engineering Workshops*, 2014, pp.431-432.

[6] J. Araujo, V. Alves, D. Oliveira, P. Dias, B. Silva, and P. Maciel, "An investigative approach to software aging in android applications," in *2013 IEEE Int. Conf. Syst. Man, Cybern.*, Oct. 2013, pp. 1229-1234.

[7] H. Wu and K. Wolter, "Software aging in mobile devices: Partial computation offloading as a solution," in *2015 IEEE Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2015*, 2016, pp. 125–131.

[8] Y. Zhao, J. Xiang, S. Xiong, Y. Wu, J. An, S. Wang, and X. Yu, "An experimental study on software aging in android operating system," in *2015 2nd Int. Symp. Dependable Comput. Internet Things*, 2015, pp. 148–150.

[9] J. H. Yahaya, A. Deraman, S. R. A. Ibrahim, and Y. Y. Jusoh, "Software certification modeling : from technical to user centric approach," *Aust. J. Basic Appl. Sci.*, vol. 7, no. 8, pp. 9–18, 2013.

[10] J. Yahaya and F. Baharom, and Aziz Deraman, "User-centred software product certification : Theory and practices," *Int. J. of Digital Society (IJDS)*, vol. 1, no. 4, pp. 281–288, Dec. 2010.

[11] M. Bombardieri and F. A. Fontana, "Software aging assessment through a specialization of the SQuaRE quality model," in *Proc. Int. Conf. Softw. Eng.*, 2009, pp. 33–38.

[12] R. Matias Jr., K. S. Trivedi, and P. R. M. Maciel, "Using accelerated life tests to estimate time to software aging failure," in *2010 IEEE 21st Int. Symp. Softw. Reliab. Eng.*, Nov. 2010, pp. 211–219.

[13] V. Castelli, R. E. Harper, P. Heidelberger, S. W. Hunter, K. S. Trivedi, K. Vaidyanathan, and W. P. Zeggert, "Proactive management of software aging," *IBM J. Res. Dev.*, vol. 45, no. 2, pp. 311–332, Mar. 2001.

[14] D. N. Arnold, "The Patriot Missile Failure," 2000. Avaialable at http://www-users.math.umn.edu/~arnold/disasters/patriot.html

[15] Z. H. Abdullah, and J. Yahaya, "Anti-aging factors for application software - a preliminary study," in *Int. Symp. on Research in Innovation and Sustainability 2014 (ISoRIS '14)*, 2014, pp. 15–16.

[16] J. H. Yahaya, and A. Deraman, "Towards the anti-ageing model for application software," in *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*, pp. 388-393, 2012.

[17] J. H. Yahaya, A. Deraman, and Z. H. Abdullah, "Evergreen software perservation : the conceptual framework of anti-ageing model," in *Information Science and Applications*, K. J. Kim, Ed. 2015, pp. 899-906.

[18] J. H. Yahaya, Z. N. Z. Abidin, and A. Deraman, "Perspective and perception on software ageing: The empirical study," in *10th Int. Conf. Comput. Sci. Educ. ICCSE 2015*, 2015, pp. 365–370.

[19] Z. H. Abdullah, J. Yahaya, and A. Deraman, "Towards anti-Ageing model for the evergreen software system," in *Proc. - 5th Int. Conf. Electr. Eng. Informatics Bridg. Knowl. between Acad. Ind. Community, ICEEI 2015*, 2015, pp. 388–393.

[20] A. A Porter, "Fundamental Laws and Assumptions of Software Maintenance," *Empir. Softw. Eng.*, vol. 2, no. 2, pp. 119–131, 1997.

[21] Z. Nasir and A. Z. Abbasi, "A framework for software maintenance and support phase," in *2010 International Conference on Information and Emerging Technologies*, 2010, pp. 1-6.

[22] K. S. Trivedi, K. Vaidyanathan, and K. Goˇ, "Modeling and Analysis of Software Aging and Rejuvenation." in *Proc. 33rd Annual Simulation Symp. (SS 2000)*, 2000, pp. 270-279.