Evergreen Software Preservation: The Conceptual Framework of Anti-Ageing Model

Jamaiah H. Yahaya[1], Aziz Deraman[2], Zuriani Hayati Abdullah[3]

[1,3] Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Selangor, Malaysia
[2] School of Informatics & Applied Mathematics, Universiti Malaysia Terengganu, Kuala Terengganu, Terengganu, Malaysia

jhy@ukm.edu.my, a.d@umt.edu.my, zha.ukm@gmail.com

Abstract. The symptom of degradation in term of quality is observed as the indicator of ageing phenomenon in software system. In human and living creators, ageing is an inescapable manifestation for every living creature on earth. In human being, this phenomenon of delaying the ageing process is normally known as anti-ageing. We try to understand and learn the process of ageing in software through understanding the human ageing process. Unlike human ageing, software ageing can be delayed by identifying factors that influence the ageing. Ageing in software is occurring when the software is degraded in term of its quality, user's satisfaction and dynamic. Previous studies indicated that software ageing factors possibly will be classified into some categorization such as cost, technology, human, functionality and environment. Our past experiences in software quality and certification motivate us to the development of software anti-ageing model and its related areas which are the ageing factors and rejuvenation index. This paper presents the background issues in software ageing which includes software quality and certification, and focus further on the conceptual framework of software anti-ageing model and preliminary formulation of anti-ageing model.

Keywords: software anti-ageing, software ageing, software quality, evergreen software, software certification

1 Introduction

Software ageing in computer science discipline has been introduced earlier when researchers investigated the ageing in software system such as in operating system. Smooth performance degradation has been also called software ageing and is a consequence of the exhaustion of system resources, such as system memory or kernel structures, the accumulation of round off errors, database deadlocks, and the contention for a pool of limited software resources. It also refers to accumulation of errors during the software execution, which are ultimately results in crash or hanging failure [2]. Degradation of software performance is characterized by the software age. Thus, since then, many efforts have been devoted to characterize and mitigate the software ageing phenomenon, that is, the accumulation of errors occurring in long-running operational software. As a result, a significant body of knowledge has been established and an international community of researchers in the area of Software Aging and Rejuvenation (SAR) [1]. Former research by Parnas identified two types of software ageing in application software, which are caused by the results of the changes that have been made and the failure of the software or software to adapt to dynamic environment [3].

In current fast growing technology demand, software engineers are becoming a technology savvy in order to cope with the rapid changes in the environment. Failure to adapt with dynamic changes will results the relevance and vital of software getting lesser to its environment which is called a phenomenon of getting old and age. The characteristics of a software which initially must be built with the capability of modifiable and scalable, thus will give flexibility and enable it to stay young and relevant in their operating environment [7][8][16]. The process of delaying the ageing in software is called anti-ageing process or rejuvenation. It can be done by detecting and classifying the ageing factors that may cause the ageing and implement the reverse action to convey the anti-ageing process.

This paper starts with the research background in software ageing, software quality and certification. Later, a discussion on conceptual framework and the initial formulation of anti-ageing model will be deliberated, and concludes with a conclusion.

2 Background Works

Previous studies indicated that the relevance of the software throughout its life span depended on the quality of the software [19]. So, it is believed that software ageing is closely related to the quality and certification of the software in the specific operating environment [16]. In other words, software can be prevented from ageing and stays young with the assurance of good quality throughout its life cycle. The assurance of good quality can be achieved through certification process [19]. The following section discusses issues and state-of-the-art in software quality and certification.

2.1 Software Quality and Certification

The awareness of the software quality has been increased in most industrial sectors including software sector. Quality by definition is a subjective concept because quality is in the eyes of the beholder. Different people see quality in different views and perspectives. One way to view quality is through user's perspective which to assess product or services and relates it in customer's satisfaction level [5]. On the other hand, software quality in technical perspective can also be measured by three categories of measurements which are: internal measures, external measures and quality in use measures [6]. Internal measures is the process of evaluating on static measures of intermediate product, external measures evaluate on the behavior of the code while the quality in used evaluates the basic set of quality in used characteristics which may affect the software in certain operating environment.

For the last forty years, several software quality models have been developed and among them are the well-known such as McCall model (1977), Boehm model (1978), FURPS model (1987), ISO9126 (1991), Dromey model (1996), Systematic Quality model (2003), PQF model (2007), and SQuaRE (2011). These models demonstrate quality characteristics of a software in term of efficiency, maintainability, usability, reliability, functionality and portability [7][13]. In recent years, human aspect is a new element of software quality measurement that has been included in PQF model which are not introduced in earlier models. Measuring software product quality by reckoning the human aspect that relates to the user's perspective and expectations are recognized and become a challenge today [7][17].

The term certification in general is the process of verifying a property value associated with something, and providing a certificate to be used as proof of validity. Software certification is the extended quality process intended to ensure and guarantee the quality standard of a software based on certain quality benchmark and accordance to the country standard. Results from certification will provide a valuable recognition on the quality of the software organization which can support the buoyancy and trustworthiness of the organization.

In Korea, there is a certification program which call good software to ensure that all software products comes in Korean's industry will be tested and verified according to good software standard[20]. K-model can be applied to small and medium sized business or project for measuring the quality values of the underlying processes. Our research group has developed a certification model based on end product quality approach named as SCM-prod model and a tool, SoCfeS, support tool for certification process [7][19]. This model has been tested successfully in the several business environments in Malaysia. The results from the case studies reported a significant satisfaction of software's developer, manager, and stakeholder, who feel more confidence in using this model for assessing and certifying software product. The certification exercises can be repeated several times during the life cycle of the products and therefore continuous quality is maintained and will delay the ageing process of software. We believe the certification exercises are useful to ensure quality and to maintain sustainable quality and preserve evergreen of the software operating in the environment. Evergreen is defined in general as having an enduring and lasting freshness, success, or popularity. In this scope of research evergreen software can be defined as the enduring and everlasting freshness, success and popularity of the software in its' operating environment.

2.2 Software Ageing and Anti-Ageing Phenomenon

Formerly, software ageing is referred to a phenomenon in long-running software system that shows an increasing of failure rate in which the occurrence of a progressive degradation in software performance and may lead to undesirable hangs and crashes [4]. The accumulation of software errors and failure to perform as user intended such as hang or crush also considered as ageing process [8-9]. The characteristics were described and identified as follows: memory bloating/leaks, shared- memory-pool latching, unreleased file locks, accumulation of un-terminated threads, file-space fragmentation, data corruption/round off accrual, thread stacks bloating and overruns[2][8][10]. Most of these studies focused on the ageing factors of software systems.

Software ageing can also be understood as similar to biological system of human being [3][11]. By using two examples such as human ageing and software life cycle, applications software can be implied, and view as a category of human evolution. This analogy is appropriate because it creates certain realization about the software. First, the application software exists inside a given environment. Furthermore, much like their biological evolution where they progress and adapt to their environment and later they grow old. Finally, the life cycle is a series of stages which a living thing passes from the beginning of its life until its death. So similarly in software life cycle, we can imply that software system and applications eventually die [12] after certain time in their environment.

However, the causes of software ageing are different from the biological organism such as humans. The human gets older when the time passes by which can be measured by number of years. Contrary with the nature and human, software will not subject to weakness or physical deterioration, thus it will not get old along with time [12]. Based on our initial investigation done in Malaysia we discover that software may experience ageing as early as approximately two to four years after being used. Therefore, we cannot determine the age of the software by numbers and years. It can be claimed that it does not matter how long the application software has been used as long as the software in the good quality and dynamic with environment changes, the software will stay young and healthy [16]. Even though in some circumstances, software ageing is inevitable but by understanding the factors of software ageing, in some ways may help to prevent the occurrence of ageing earlier. Those factors will be explained in more detail the next section.

Anti-ageing is a process to delay, prevent, and retard the ageing process from occurs premature. In human biological ageing and anti-ageing addressed by Klatz [14] indicates that anti-ageing factors for human are by practicing a healthy lifestyle, such as avoid eating unhealthy food, avoid drinking alcohol, stop smoking, stay slim, regular exercises and stress reduction management. Software anti-ageing may include prevention and rejuvenation actions such as adaptive, corrective, preventive, and perfective can be the anti-ageing action for application software.

3 The Conceptual Framework

A conceptual framework refers to “a theoretical structure of assumptions, principles, and rules that holds together the ideas comprising a broad concept.” The design of this conceptual framework is based on the scope of software ageing, anti-ageing and software quality for application software.

Previous study has revealed that software ageing is closely related to software product quality. The effective and practical approach for managing ageing of the software is through software quality and certification process [16]. This was discussed in section 2.1. Figure 1 shows the conceptual framework of software anti-ageing that consists of main components: software quality models, software certification models, ageing factors, and the algorithm and formulas. The underlying theories in software quality and certifications were carried out to derive the reliable and supportive ageing factors as shown in Fig.1.

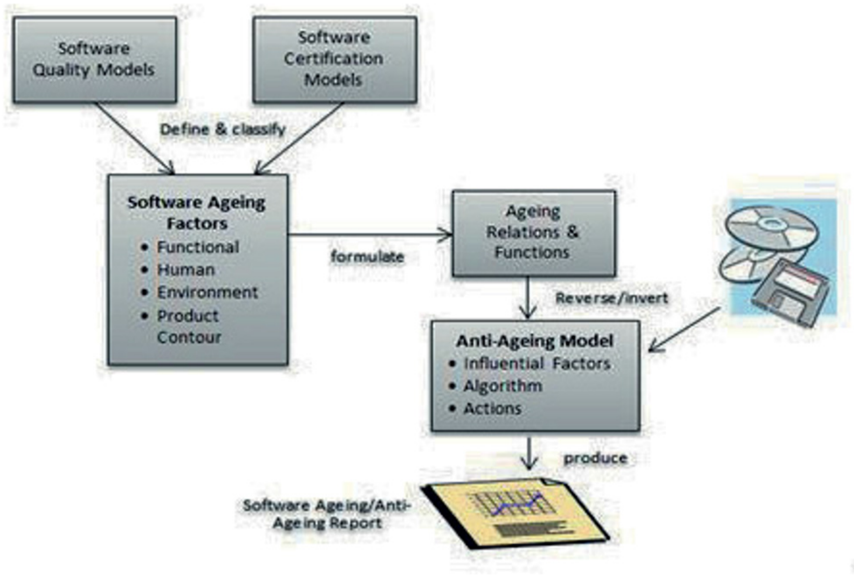

Fig. 1. The conceptual framework of software anti-ageing model

3.1 Software Quality

Software quality is closely related to the occurrence of software ageing. The more good quality of the software the less of software failure or software ageing will occur. Software quality can be measured by a number of variable which can be categorized by external variables and internal variable. The quality attributes of software product need to be known, recognized and classified to absorb and learn the underlying factors associated with ageing and anti-ageing.

3.2 Software Certification

Based on our previous works in software quality and certification indicated that certification practices in the software operating environment is essential to ensure the software stays young and healthy. Software certification models and attributes are essential to be investigated and studied to understand the contributing factors and features of software anti-ageing process and model. This research focuses on application software where application software is a program that is designed to perform specific task for end users. Application software can be used as a business tool that supports and assists the business process thus, it is crucial to study the phenomenon of software ageing and anti-ageing towards application software. It is different from previous works where their focuses are on software systems [1][[2][12][15].

3.3 Software Ageing and Factors

Several questions need to be answered such as what are the indicators of the ageing, how it affects the performances of the software, does it influence the working performance of software practitioners, what are the consequences of software ageing to an organization, software practitioners and the environment, and do software practitioners aware of the ageing phenomenon in their operating environment? Also, how to delay the ageing process of the software?

Systematic literature study on software ageing is carried out to identify factors that contribute to the ageing process which also may be used to formulate the anti-ageing factors and model. Aziz et al [18] identifies four main classification of ageing factors: functional, human, environment and product profile. Each of these factors is broken down into metrics and measures that can support in term of measuring the ageing status and values based on quantitative measures.

3.4 Formulation of the Anti-Ageing Model

A survey done in Malaysia with 50 respondents from information technology experts and practitioners discovered that 64% of them do not know anything about software ageing or never heard anything about this topic. The study also showed that only 30% of the respondents have experienced in software ageing related to the software they used, 20% of them never experienced while 50% were uncertainty. This findings reveal that the ageing and anti-ageing issues in software domain is still new but shows a consideration importance to the respondents.

In general mathematical definition, a "relation" is a relationship between sets of information. In this scope of study software ageing can be established as a set of several identified factors. Based on our recent findings 18], the ageing factors of the application software have been identified and may be formulated, as in:

$$\text{Software Ageing (SAS)} \sim f(\text{human, environment, functional, product}) \tag{1}$$

$$\text{SAS} \sim f\,(\text{h, e, f, p}) \tag{2}$$

$$\text{y} = \text{k.x} \tag{3}$$

$$\text{SAS} = \text{k.}f\,(\text{h, e, f, p}) \tag{4}$$

The above relations or equations (1) and (2) are derived from the ageing factors where software ageing score (SAS) can be derived given the values of factor human (h), environment (e), functional (f) and product (p). Each factors associated with ageing mentioned above are broken down into measurable metrics which can be assigned and transformed into numerical values. Thus, we say that given a value of all the factors, we know the computed value of SAS. In this case, SAS is the ageing score and can be mapped and transformed into an ageing status of specific software.

The ageing relation (equation 2) is converted to a function (a well-behaved relation) which means that given a value, we know exactly where to go, given an x we get only and exactly one y (see equation 3). Thus, The SAS function shown in (4) indicates that there is a constant value of k to normalize the linear equation of ageing function. The k value is unknown yet at this stage and still under investigation by our research group. There is a possibility that the k value varies for h, e, f and p. This will be an interesting investigation to come up with the possible k through an empirical study.

Therefore, to develop the anti-ageing model, the relationships between ageing factors are relevant and been explored further. As we understand anti-ageing means the reverse action or inversion of ageing. Thus, we possibly will formulate the anti-ageing model and equation as the reverse or inversion function or negation operator of ageing as shown in (5).

$$\text{Anti-Ageing-Score (AAS)} = k.f\ (h, e, f, p) \quad (5)$$

; Where k is the constant to represent the negation operator.

The formulation discussed in this paper is a preliminary discovery and finding of this research project. Further works need to be carried out in detail to formulate and normalize the relationships and verify the relevance of the formula as well as the practicality of the measurements in real environment.

4 Conclusion

The initial works related to software ageing and anti-ageing in application software have been presented in this paper. The symptom of degradation in term of software quality is observed as the indicator of ageing factors in application software. The ageing phenomenon was observed and experienced through series of software certification practices carried out by this research group. Recently, we have identified software ageing associated factors and they may be considered as the influential factors of ageing. The classifications of factor are defined as functional, human, environment and product profile. Further study and exploration are needed to confirm the correlation between factors and formulate the anti-ageing equations and model. The anti-ageing model proposed in this paper is the preliminary work and it is valuable to prevent ageing and to preserve the software young and evergreen in the environment.

Acknowledgements

This research is funded by the Fundamental Research Grant Scheme, Malaysia Ministry of Higher Education.

References

1. Cotroneo, D., Natella, R., Pietrantuono, R. and Russo, S. Software Aging and Rejuvenation: Where We Are and Where We Are Going, 2011 IEEE Third Int. Work. Softw. Aging Rejuvenation, no. 30, pp. 1–6, Nov (2011)

2. Cassidy, K.J., Gross, K.C. and Malekpour, A. Advanced pattern recognition for detection of complex software aging phenomena in online transaction processing servers, Proc. Int. Conf. DependableSyst. Networks, pp. 478–482 (2002)
3. Parnas, D.L. Software Aging Invited. IEEE, pp. 279–287 (1994)
4. Zhao, J., Trivedi, K.S., Wang, Y. and Chen, X. Evaluation of software performance affected by aging. 2010 IEEE Second Int. Work. Softw. Aging Rejuvenation, vol. 3, pp. 1–6, Nov (2010)
5. Jin, H. and Zeng, F. Research on the definition and model of software testing quality, Proc. 2011 9th Int. Conf. Reliab. Maintainab. Saf., pp. 639–644, Jun (2011)
6. Suryn, W., Bourque, P., Abran, A. and Laporte, C. Software product quality practices - quality measurement and evaluation using TL9000 and ISO/IEC 9126. Intermag Eur. 2002 Dig. Tech. Pap.2002 IEEE Int. Magn. Conf., pp. 156–160 (2003)
7. Yahaya, J.H., Deraman, A., Baharom, F. and Hamdan, A.R. Software Certification from Process and Product Perspectives. *IJCSNS International Journal of Computer Science and Network Security*, 9(3), March (2009)
8. Sachin Garg, K. S. T., Aadvan Moorsel, Vaidyanathan, K. .A Methodology for Detection and Estimation of Software Aging. Software Reliability Engineering, Proceedings. The Ninth International Symposium (1998)
9. Grottke, M., Li, L., Vaidyanathan, K.. and Trivedi K. S. Analysis of Software Aging in a Web Server. *IEEE Trans. Reliab*. 55(3) pp. 411–420, Sept (2006)
10. Zheng, P., Xu, Q. and Qi, Y. An Advanced Methodology for Measuring and Characterizing Software Aging. 2012 IEEE 23rd Int. Symp. Softw. Reliab. Eng. Work., pp. 253–258, Nov (2012)
11. Sustainment, S. Geriatric Issues of Aging Software. *The Journal of Defense Software Engineering*. pp 4-7 Dec (2007)
12. Constantinides, C. and Arnaoudova, V. Prolonging the aging of aoftware systems. *Encyclopedia of Information Science and Technology* [Online]. Second Edition (8 Volumes) (2009)
13. ISO/IEC 25010. Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) (2011)
14. Klatz, R. Definition of Anti-Aging Medicine. *Academic Journal Article*, Generations , 25(4), Winter (2002)
15. Hanmer, R. Software rejuvenation. Proceedings of the 17th Conference on Pattern Languages of Programs. ACM (2010)
16. Yahaya, J. H. & Deraman, A. Towards a Study on Software Ageing for Application Software: The Influential Factors. *IJACT: International Journal of Advancements in Computing Technology*, 4(14), pp. 51-59 (2012)
17. Yahaya, J.H., Deraman, A., Hamdan, A.R and Jusoh, Y.Y. User-Perceived Quality Factors for Certification Model of Web-Based System. International Journal of Computer, Information, Mechatronics, Systems Science and Engineering Vol:8 No:5, pp. 576-582 (2014)
18. Deraman, A., Yahaya, J.H., Zainal Abidin, Z.N. and Mohd Ali, N. Software Ageing Measurement Framework Based on GQM Structure. *Journal of Software and Systems Development*, 2014 (2014):1-12 (2014)
19. Yahaya, J. H., Deraman, A. & Hamdan, A. R. Continuosly Ensuring Quality through Software Product Certification: A Case Study. Proceedings of the International Conference on Information Society (i-Society 2010), London, UK, 28-30 June (2010)
20. Hwang, S.M. Quality Metrics for Software Process Certification based on K-model. 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, pp 827-830 (2010).