Asia-Pacific Journal of Information Technology and Multimedia 14(2) 2025: 453 – 462
https://doi.org/10.17576/apjitm-2025-1402-07

Software Product Evaluation and Ageing Index Model: An Instrument Development

Model Penilaian Dan Pengindeksan Penuaan Produk Perisian : Pembangunan Instrumen

Zuriani Hayati Abdullah[1], Aziz Deraman[2], Zulkefli Mansor[3], Jamaiah Yahaya[3]*

[1] Center for Digital Innovations, COE of Immersive Experience*
Faculty of Computing & Informatics, Multimedia University, 63100 Cyberjaya, Selangor, Malaysia
[2]Faculty of Computer Science and Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Nerus, Terengganu, Malaysia
[3] Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor,Malaysia

**Corresponding author: zuriani.hayati@mmu.edu.my*

Received 18 September 2024
Accepted 24 July 2025, Available online 15 December 2025

ABSTRACT

Software ageing refers to the decline in performance experienced by a software product over time. This decline can render software obsolete and lead to issues for users and organizations. Similar to humans, ageing is an unavoidable natural process for software. However, by recognizing and addressing the factors contributing to ageing, its progression can be mitigated. This research delves into the external factors influencing software ageing, particularly focusing on software quality dimensions. Previous research indicates a strong correlation between software ageing and its quality. Consistent quality monitoring is essential for preserving relevance, slowing down ageing, and ensuring the software remains effective and up-to-date. Software ageing evaluation aims to assist software owners in assessing and monitoring the ongoing relevance of their application software within specific environments. Hence, an instrument to evaluate the software product is crucial to be developed. The instrument can be used by software product owners to monitor their software product state. This paper discusses on three phases of instrument development for software product evaluation and ageing index which are the development of measurement requirements, development of the instrument construct and items, and instrument validation. This paper also presents the result of a preliminary study that has been conducted among software practitioners.

Keywords**:** Software product ageing, Anti-ageing software, Software ageing factors, Software quality, Performance degradation

ABSTRAK

Penuaan perisian merujuk kepada penurunan prestasi yang dialami oleh produk perisian dari semasa ke semasa. Penurunan ini boleh menyebabkan perisian usang atau lapuk dan membawa kepada isu untuk pengguna dan organisasi. Sama seperti manusia, penuaan adalah proses semula jadi yang tidak dapat dielakkan. Walau bagaimanapun, dengan mengenali dan menangani faktor-faktor yang menyumbang kepada penuaan, proses penuaan boleh dikurangkan. Kajian ini mengkaji faktor luaran yang mempengaruhi penuaan perisian berteraskan dimensi kualiti perisian. Kajian terdahulu menunjukkan korelasi yang kuat antara penuaan perisian dan kualiti perisian. Pemantauan kualiti yang konsisten adalah penting untuk mengekalkan kerelevanan, melambatkan penuaan, dan memastikan perisian efektif dan terkini. Penilaian penuaan perisian bertujuan untuk membantu pemilik perisian dalam menilai dan memantau kerelevanan perisian aplikasi mereka dalam persekitaran tertentu. Oleh itu, instrumen untuk menilai produk perisian adalah penting untuk dibangunkan agar boleh digunakan oleh pemilik produk perisian untuk memantau keadaan produk perisian mereka. Kertas kerja ini membincangkan tiga fasa pembangunan instrumen untuk penilaian produk perisian dan indeks penuaan iaitu pembangunan keperluan pengukuran, pembangunan konstruk dan item instrumen, dan pengesahan instrumen. Kertas kerja ini juga membentangkan hasil kajian awal yang telah dijalankan dikalangan pengamal perisian

Kata kunci**:** Penuaan produk perisian, anti-penuaan perisian, faktor penuaan perisian, kualiti perisian, penurunan prestasi.

INTRODUCTION

Software ageing is a phenomenon characterized by a progressive decline in the performance and quality of a software product. This concept was first introduced by David L. Parnas in 1994. Parnas likened programs to humans, suggesting that ageing is inevitable. However, he argued that by understanding the causes and contributing factors of software ageing, steps can be taken to mitigate its effects. In his article, Parnas emphasized that identifying the causes of ageing can help reduce its impact, such as damage and functional failures, and prepare software owners for the eventual obsolescence and irrelevance of their software (Parnas 1994).

According to Controneo et al., research on Software Ageing and Rejuvenation (SAR) has developed along two distinct trajectories: the perspective of software engineering and software dependability (Cotroneo et al. 2011). Most studies have concentrated on the software dependability perspective, emphasizing internal factors of software ageing such as programming errors, software bugs, memory leaks, resource usage inefficiencies, data corruption, and data inconsistency (Avritzer et al. 2020; Z. H. Abdullah et al. 2019; Cotroneo et al. 2014).

From the software engineering perspective, ageing occurs due to the failure of the software owner to update or adapt the software product to meet current needs, or due to complex changes made to the software that degrade its quality and performance (Yakovyna and Uhrynovskyi 2020;Yahaya and Deraman 2012).This study aims to develop instruments for evaluating software products and creating an ageing index model, with a particular emphasis on the external factors of software ageing. By focusing on these external factors, the research seeks to provide a more comprehensive understanding of how to maintain and improve software performance and longevity.

RELATED WORK

The phenomenon of software ageing was initially explored by researchers in the early 1990s. Subsequent investigations revealed that the ageing of software in computer systems could lead to various issues, such as data corruption, fragmentation of data files, and several memory-related problems (Z. H. Abdullah et al. 2019; Cotroneo et al. 2014). These include ageing-related bugs, accumulation of undetermined threads, system failures, memory bloating and leakage, memory shortages, memory overruns, residual defects, and unreleased file locks (Avritzer et al. 2020; Abdullah et al. 2015).

Recently, the study of software ageing has expanded beyond traditional computer software to encompass mobile applications, such as those on the Android platform (Yakovyna and Uhrynovskyi 2020; Xiang et al. 2020). Unlike computers, mobile devices often operate for extended periods without rebooting or shutting down, exacerbating software ageing issues. This poses a significant challenge to optimizing user experience and satisfaction on mobile devices. Furthermore, the impact of software ageing is no longer confined to mobile and computer software but has extended to cloud computing and the Internet of Things (IoT) environments (Liu and Meng 2019). This underscores the critical importance of further research in the field of software ageing.

The phenomenon of software ageing significantly impacts software reliability, as well as the safety of individuals' lives and property. Historical instances of software failures illustrate the severe consequences of software faults, which can result in inconvenience, poor business decisions, and even fatalities. A notable example is the Therac-25 incident, where a software-controlled radiation treatment device used for cancer patients malfunctioned. Between 1985 and 1987, six patients at four medical facilities received excessive doses of radiation due to faults in the Therac-25 equipment. Investigations into the Therac-25 incident identified multiple factors contributing to these injuries and fatalities (Baase 2018).

The accumulation of software errors and bugs is a key symptom of software ageing (Cotroneo et al. 2016). Lessons learned from the Therac-25 incidents highlight the importance of thorough testing before releasing any software product, particularly when reusing existing programs or source code. Developers must conduct adequate levels of testing, ensure comprehensive documentation, and pay careful attention to potential safety risks. It is crucial to verify that the software interacts correctly with devices to function as intended, ensure compatibility between software and hardware, as well as between application software and the operating system, and design user-friendly interfaces (Baase 2018). These measures help prevent post-deployment issues, produce reliable and safe software, and extend the software's lifecycle.

Previous studies have identified two main categories of software ageing factors: internal and external. Internal factors are related to software dependability and indicate a degradation in software performance, functionality, and quality(Z. Abdullahet al. 2014; Zainal Abidin et al. 2017). This degradation is a result of resource exhaustion in operating systems, fragmentation, and the accumulation of numerical errors during program execution. External factors contributing to software ageing are associated with the software engineering perspective, where the phenomenon occurs due to the lack of updates or upgrades by the program owner or due to modifications made to the software. Frequent changes to the software can consequently degrade its performance and quality (Z. H. Abdullah et al. 2019; Yahayaet al. 2016). There is a paucity of research examining the effects of software ageing from a software engineering perspective. This gap exists because the majority of earlier studies have concentrated on ageing from a technical and performance-based standpoint, primarily considering software dependability. Consequently, these studies have predominantly addressed internal factors such as programming errors, software bugs, memory leaks, resource usage inefficiencies, data corruption, and data inconsistency.

Therefore, this study aims to shift the focus towards the external factors of software ageing from a software engineering perspective, particularly emphasizing the software quality dimension. By exploring how factors such as inadequate updates, failure to meet current user needs, and the impact of complex modifications affect software quality and performance, this research seeks to provide a more holistic understanding of software ageing. The ultimate goal is to develop methodologies and tools that can help mitigate the effects of software ageing, thereby enhancing the longevity and reliability of software products

INSTRUMENT DEVELOPMENT METHODOLOGY

This section will discuss the methodology of instrument development. The instrument has gone through three phases or stages in which:

1. Analysis of background study and literature review
2. Development of questionnaire instruments constructs and items.
3. Questionnaire instrument validation.

ANALYSIS OF BACKGROUND STUDY AND LITERATURE REVIEW.

The instrument development process begins with a comprehensive analysis of the background study and a thorough literature review on the body of knowledge surrounding Software Ageing and Rejuvenation (SAR). In this initial phase, factors contributing to software ageing are identified, and existing software ageing models are examined and compared. This comparative analysis aims to determine the suitability of various measurement indices for inclusion in the proposed model. By systematically evaluating the strengths and limitations of current models, the study seeks to develop a robust and effective instrument that accurately measures software ageing and supports the formulation of strategies to mitigate its impact. This foundational work is critical to ensuring that the proposed model is both theoretically sound and practically applicable.

DEVELOPMENT OF INSTRUMENTS CONSTRUCTS AND ITEMS

The questionnaire instrument used in this study comprises nine distinct sections, labeled as Sections A through H, each corresponding to one of eight constructs or factors: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability. Within each section, items are specifically tailored to assess the nuances of the respective construct. Section I encompasses inquiries concerning respondent demographics, including organizational type, length of service, job designation, and other pertinent details.

Table 1 provides a comprehensive breakdown of each section, delineating the constructs addressed and the corresponding number of items featured in the questionnaire. It is imperative to note that the formulation of all constructs and items is grounded in a thorough analysis of background studies and an extensive review of appropriate literature. This ensures that the questionnaire is meticulously designed to capture essential facets relevant to the study's objectives, thereby enhancing its validity and reliability.

TABLE 1. Questionnaire instrument design

Section	Construct	Items	Source
A	Functional suitability	A1: Functional completeness	ISO25010:2011 ISO25030:2019
		A2: Functional correctness	ISO25010:2011 ISO25030:2019 Mc Call
		A3: Functional appropriateness	ISO9126 ISO25010:2011
		A4: Functional compliance	ISO 9126 ISO25010:2011
B	Performance efficiency	B1: Time behaviour	ISO9126 ISO25010:2011
		B2: Resource utilization	ISO25010:2011
		B3: Capacity	Boehm FURPS
C	Compatibility	C1: Co-existence	ISO 9126 ISO25010:2011
		C2: Interoperability	ISO 9126 ISO 25010:2011
		C3: Dynamic technology	(Zainal et al. 2017)
		C4: Dynamic environment	(Yahaya and Deraman 2012) (Parnas 1994)
D	Usability	D1: Appropriateness recognizability	ISO25010:2011
		D2: Learnability	ISO9126 ISO25010:2011
		D3: Operability	ISO9126 ISO25010:2011
		D4: User error protection	ISO25010:2011 (Webber and Carl Gutwin 2018)
		D5: User interface aesthetics	ISO25010:2011 Boehm FURPS
		D6: Accessibility	ISO25010:2011 Boehm
		D7: Attractiveness	ISO9126 ISO25010:2011
E	Reliability	E1: Maturity	ISO9126 ISO25010:2011
		E2: Availability	ISO25010:2011 Boehm FURPS
		E3: Fault tolerance	ISO9126 ISO25010:2011 Mc Call
		E4: Recoverability	ISO9126

			ISO25010:2011
			FURPS
F	Security	F1: Confidentiality	ISO25010:2011
		F2: Integrity	ISO25010:2011
			PQF
			Boehm
			Mc Call
		F3: Non-repudiation	ISO25010:2011
		F4: Accountability	ISO25010:2011
			Boehm
		F5: Authenticity	ISO25010:2011
G	Maintainability	G1: Modularity	ISO25010:2011
			Mc Call
		G2: Reusability	ISO25030:2019
			ISO25010:2011
			Dromey
			Mc Call
		G3: Analysability	ISO9126
			ISO25010:2011
		G4: Modifiability	ISO25030:2019
			ISO25010:2011
			ISO9126
			Boehm
		G5: Testability	ISO25010:2011
H	Portability	H1: Adaptability	ISO25020:2016
			ISO25010:2011
			ISO9126
			FURPS
			Boehm
		H2: Installability	ISO25010:2011
			ISO9126
		H3: Replaceability	ISO25010:2011
			ISO9126
I	Demography	(I1 to I7 demography question)	
		Total Items: 42	

INSTRUMENT VALIDATION

The content of the questionnaire underwent rigorous evaluation and validation processes by a panel of three experts, comprising two academics and one industry representative. The collective expertise of the panel affirmed that the questionnaire instrument effectively measured the intended content aspects. Following expert validation, the questionnaire was subjected to further validation through a pilot study. The subsequent section of this paper will delineate the outcomes of the pilot study, elucidating the insights gleaned from this preliminary validation phase.

RESULTS AND DISCUSSION

The primary objective of the pilot study was to gauge respondents' comprehension of the questionnaire items and to evaluate the instrument's reliability, ensuring consistency in the measurements obtained. This pilot investigation entailed a small-scale test conducted with a cohort of sixty respondents, all of whom belonged to the target population group of information technology practitioners. However, the survey garnered responses from only thirty-six participants.

To assess the internal consistency of the items within each construct, Cronbach’s alpha coefficient was employed as a measure of reliability. The results of the pilot study were subsequently evaluated based on the Cronbach's Alpha coefficient, with reference to the Rasch measurement model. According to established criteria, an acceptable range for the Cronbach's Alpha coefficient score falls between 0.61 and 0.99. The reliability values obtained from this assessment are presented in Table 2, as derived from Bond and Fox (2015).

TABLE 2**.** The Cronbach's alpha reliability value

Cronbach's alpha score	**Reliability interpretation**
0.9-1.0	Very good and has a high level of consistency and effectiveness
0.7-0.8	Good and acceptable
0.6-0.7	Acceptable
<0.6	Item needs to be repaired
<0.5	Items need to be removed

Cronbach's alpha coefficient test was conducted on the entire construct. Table 3 and Table 4 show the Cronbach's alpha value for the entire construct is 0.94 which is very good and has a high and effective level of consistency.

TABLE 3. The Cronbach's alpha for entire construct

Cronbach's Alpha	**Cronbach's Alpha Based on Standardized Items**	**Total Items**
.947	.949	35

TABLE 4. The Cronbach's alpha for entire construct

Construct	**Item**	***Corrected Item-Total Correlation***	***Cronbach's Alpha if Item Deleted***	***Final Cronbach's Alpha***	**Summary**
Functional suitability	A1	.742	.604	.758	Keep
	A2	.471	.744		Keep
	A3	.566	.695		Keep
	A4	.485	.754		Keep

Performance efficiency	B1	.359	.497		Consider rephrase/ remove
	B2	.503	.250	.569	Consider rephrase/ remove
	B3	.288	.599		Consider rephrase/ remove
Compatibility	C1	.293	.677		Keep
	C2	.526	.509	.650	Keep
	C3	.553	.512		Keep
	C4	.391	.609		Keep
Usability	D1	.640	.888		Keep
	D2	.753	.876		Keep
	D3	.728	.878		Keep
	D4	.806	.869	.896	Keep
	D5	.710	.881		Keep
	D6	.684	.883		Keep
	D7	.613	.894		Keep
Reliability	E1	.753	.778		Keep
	E2	.678	.804	.844	Keep
	E3	.696	.797		Keep
	E4	.622	.832		Keep
Security	F1	.625	.854		Keep
	F2	.651	.851		Keep
	F3	.681	.841	.867	Keep
	F4	.794	.814		Keep
	F5	.711	.834		Keep
Maintainability	G1	.560	.769		Keep
	G2	.513	.784		Keep
	G3	.744	.712	.800	Keep
	G4	.626	.748		Keep
	G5	.485	.790		Keep
Portability	H1	.572	.787		Keep
	H2	.681	.681	.792	Keep
	H3	.688	.660		Keep

After running the Cronbach's alpha coefficient test for the entire construct as illustrated in Table 3, the same test was also conducted according to the eight constructs with their respective items as in the questionnaire. This is intended to test the value of a strong reliability index between constructs and items. Referring to Table 4, Cronbach's alpha values for all constructs and items are greater than 0.65. Only Cronbach's alpha value on the performance efficiency construct scored <0.6 which is 0.56. Refer to Table 2, if the value of Cronbach's alpha is less than 0.6, the following items should be repaired or can be removed. Then, if we refer to the corrected item correlation value in Table 4 it can be seen that the total score is greater than 0.3. That is means all items can be accepted and remain used as items in the actual study.

CONCLUSION

Effective monitoring of software is imperative to uphold its quality and alignment with user needs, particularly in dynamic environments where software undergoes rapid changes. In such contexts, the ongoing relevance of software must be diligently measured and sustained. Establishing and adhering to a standard mechanism is essential to ensure both the quality and relevance of software products.

To address this need, the present study proposes the development of a standardized measurement model aimed at assisting software owners in monitoring the performance and ageing trajectory of their software products. This paper specifically focuses on the creation of the Software Ageing Measurement Instrument, which will serve as a fundamental component within the software ageing measurement model. In future endeavors, the study aims to advance the formulation of measurement techniques and algorithms, facilitating the computation of relevance scores as an integral aspect of the proposed model. These forthcoming efforts will contribute to the enhancement of software monitoring practices, thereby promoting sustained software quality and relevance in dynamic environments.

ACKNOWLEDGEMENT

We extend our gratitude to Universiti Malaysia Terengganu for supporting and enhancing inter-university research collaboration.

REFERENCES

Abdullah, Zh, Jh Yahaya, and Aziz Deraman. 2014. “The Anti-Ageing Factors for Evergreen Software–a Preliminary Study.” *Sci-Int.Com* 2014 (October): 1615–18. http://www.sci-int.com/pdf/796966055PID 106--Zuriani Hayati Abdullah--1615-1618--.pdf.

Abdullah, Zuriani Hayati, Jamaiah Yahaya, and Aziz Deraman. 2015. “Towards Anti-Ageing Model for the Evergreen Software System.” *Proceedings - 5th International Conference on Electrical Engineering and Informatics: Bridging the Knowledge between Academic, Industry, and Community, ICEEI 2015*, 388–93. https://doi.org/10.1109/ICEEI.2015.7352532.

Abdullah, Zuriani Hayati, Jamaiah Yahaya, Siti Rohana Ahmad Ibrahim, Sazrol Fadzli, and Aziz Deraman. 2019. “The Implementation of Software Anti-Ageing Model towards Green and Sustainable Products.” *International Journal of Advanced Computer Science and Applications* 10 (5): 42–50. https://doi.org/10.14569/ijacsa.2019.0100507.

Avritzer, Alberto, Domenico Cotroneo, Yennun Huang, and Kishor Trivedi. 2020. “Chapter 1: Software Aging and Rejuvenation: A Genesis - Extended Abstract.” *Proceedings - 2020 IEEE 31st International Symposium on Software Reliability Engineering Workshops, ISSREW 2020*, October, 319–20. https://doi.org/10.1109/ISSREW51248.2020.00091.

Baase, Sara. 2018. *A Gift of Fire*. *Pearson*. Vol. 5th Editio.

Bond, T. G, and C. M. Fox. 2015. *Applying the Rasch Model: Fundamental Measurement in the Human Sciences*. 3rd Ed. Mahwah, NJ: L. Erlbaum.

Cotroneo, Domenico, Antonio Ken Iannillo, Roberto Natella, Roberto Pietrantuono, and Stefano Russo. 2016. “The Software Aging and Rejuvenation Repository: Http://Openscience.Us/Repo/Software-Aging/.” *2015 IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2015*, 108–13. https://doi.org/10.1109/ISSREW.2015.7392054.

Domenico Cotroneo, Roberto Natella, Roberto Pietrantuono, and Stefano Russo. 2011. “Software Aging and Rejuvenation: Where We Are and Where We Are Going.” *2011 IEEE Third International Workshop on Software Aging and Rejuvenation*, no. 30 (November), 1–6. https://doi.org/10.1109/WoSAR.2011.15.

Domenico Cotroneo, Roberto Natella, Roberto Pietrantuono, and Stefano Russo. 2014. “A Survey of Software Aging and Rejuvenation Studies.” *ACM Journal on Emerging Technologies in Computing Systems* 10 (1): 1–34. http://dl.acm.org/citation.cfm?doid=2543749.2539117.

Liu, Jing, and Lingze Meng. 2019. “Integrating Artificial Bee Colony Algorithm and BP

Neural Network for Software Aging Prediction in IoT Environment.” *IEEE Access* 7:32941–48. https://doi.org/10.1109/ACCESS.2019.2903081.

Parnas, David Lorge. 1994. “Software Aging Invited.” *ICSE ’94 Proceedings of the 16th International Conference on Software Engineering*, 279–87.

Webber, Sarah, and Carl Gutwin. 2018. “Designing for User Error: A Survey of Error Prevention and Mitigation Techniques.” In *CHI Conference on Human Factors in Computing Systems*.

Xiang, Jianwen, Caisheng Weng, Dongdong Zhao, Artur Andrzejak, Shengwu Xiong, Lin Li, and Jing Tian. 2020. “Software Aging and Rejuvenation in Android: New Models and Metrics.” *Software Quality Journal* 28 (1). https://doi.org/10.1007/s11219-019-09475-0.

Yahaya, Jamaiah H, and Aziz Deraman. 2012. “Towards the Anti-Ageing Model for Application Software.” *Proceedings of the World Congress on Engineering,WCE2012* II.

Yahaya, Jamaiah H, Aziz Deraman, and Zuriani Hayati Abdullah. 2016. “Evergreen Software Preservation : The Anti-Ageing Model.” *ICC ’16 Proceedings of the International Conference on Internet of Things and Cloud Computing,Cambridge, United Kingdom — March 22 - 23, 2016*, 1–6.

Yakovyna, Vitaliy, and Bohdan Uhrynovskyi. 2020. “User-Perceived Response Metrics in Android OS for Software Aging Detection.” *International Scientific and Technical Conference on Computer Sciences and Information Technologies* 1:436–39. https://doi.org/10.1109/CSIT49958.2020.9322031.

Zainal Abidin, Zaiha Nadiah, Jamaiah Yahaya, and Aziz Deraman. 2017. “Software Ageing Measurement Model (SAMM): An Instrument Development.” *Journal of Telecommunication, Electronic and Computer Engineering* 9 (3-5 Special Issue): 51–54.